Multi-camera intelligent space for a robust, fast and easy deployment of proactive robots in complex and dynamic environments

Adrián Canedo Rodríguez

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

DEPARTAMENTO DE ELECTRÓNICA E COMPUTACIÓN

CENTRO SINGULAR DE INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Centro Singular de Investigación en Tecnoloxías da Información

Departamento de Electrónica e Computación

Thesis

MULTI-CAMERA INTELLIGENT SPACE FOR A ROBUST, FAST AND EASY DEPLOYMENT OF PROACTIVE ROBOTS IN COMPLEX AND DYNAMIC ENVIRONMENTS

Author:

Adrián Canedo Rodríguez

PhD supervisors:

Dr. Roberto Iglesias Rodríguez

Dr. Carlos Vázquez Regueiro

June 2015

Dr. Roberto Iglesias Rodríguez, Profesor Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidade de Santiago de Compostela e investigador adscrito al Centro Singular de Investigación en Tecnoloxías da Información (CITIUS).

Dr. Carlos Vázquez Regueiro, Profesor Titular de Universidad del Grupo de Arquitectura de Computadores de la Universidade de A Coruña.

HACEN CONSTAR:

Que la memoria titulada **MULTI-CAMERA INTELLIGENT SPACE FOR A ROBUST, FAST AND EASY DEPLOYMENT OF PROACTIVE ROBOTS IN COMPLEX AND DYNAMIC ENVIRONMENTS** ha sido realizada por **D. Adrián Canedo Rodríguez** bajo nuestra dirección en el Centro Singular de Investigación en Tecnoloxías da Información - Departamento de Electrónica e Computación de la Universidade de Santiago de Compostela, y constituye la Tesis que presenta para optar al título de Doctor.

Junio 2015

Dr. Roberto Iglesias Rodríguez
Director de la tesis

Dr. Carlos Vázquez Regueiro
Director de la tesis

Adrián Canedo Rodríguez
Autor de la tesis

To my parents and sister

All my life I’ve had one dream: to achieve my many goals

Homer J. Simpson

Estuda moito, neno, que o que aprendas ninguén cho vai a poder quitar

María Catoira

Acknowledgments

To the *Centro Singular de Investigación en Tecnoloxías da Información (CITIUS)*, to the *Departamento de Electrónica e Computación*, and to the *Grupo de Sistemas Intelixentes* of the University of Santiago de Compostela, for providing me with the necessary resources for my research. To the Spanish Government, that has provided the funding to carry out this thesis through the research grant BES-2010-040813 FPI-MICINN, and the research projects *Intelligent and distributed control scenario for the fast and easy deployment of robots in diverse environments* (TIN2009-07737) and *Service robots that learn from you and like you* (TIN2012-32262).

To Dr. Roberto Iglesias and Dr. Carlos V. Regueiro, the directors of this thesis. From them, I learned most of the things I know about robotics and artificial intelligence. Most importantly, they taught me what a researcher is supossed to be.

To Dr. Victor Alvarez-Santos and Dr. Cristina Gamallo, my "thesis-mates" over all these years. Thank you for your support, your help, and the passion you put in everything we do together. Thank you for enjoying this process with me.

To Dr. Senen Barro, because he has always given me great advice. To David Santos-Saavedra, Dr. Jose Manuel Rodríguez Ascaríz, Dr. Xose Manuel Pardo, Dr. Manuel Fernandez-Delgado and Dr. Eva Cernadas, because they have always helped me disinterestedly in many matters and they have made very relevant contributions to this thesis. To Jose Manuel Abuin and Ignacio Cabado, because they have developed part of the work presented in this thesis as part of their Final Degree Projects.

To my many work-mates, for making my thesis time a wonderful experience. To my family and friends, for being there.

June 2015

Contents

Resumen

Los robots y los dispositivos robóticos tendrán un gran impacto en muchos mercados existentes y emergentes, entre ellos, el sector de los robots de servicio, tanto profesionales como domésticos. En el futuro, se espera que los robots trabajen con nosotros y que nos ayuden en muchas circunstancias diferentes: serán parte de nuestra vida cotidiana como asistentes, colaborarán en el cuidado de personas mayores, etc. Por este motivo, uno de los retos más relevantes en la robótica actual es la integración de los robots en entornos cotidianos.

Los casos de éxito en robótica autónoma móvil se han restringido hasta ahora a escenarios poco amplios y bien definidos, en los que las condiciones de contorno son conocidas a priori. Por tanto, en estos escenarios el robot puede recurrir a programas de control pre-instalados, diseñados específicamente para el entorno en el que el robot se mueve. Sin embargo, las nuevas aplicaciones de robots de servicios personales y profesionales exigen robots más inteligentes, que deberán operar en ambientes menos restrictivos. Se espera que estos robots sean capaces de realizar tareas cada vez más complejas en entornos cada vez menos estructurados y conocidos, donde cada vez se necesite menos instrucción o supervisión humana. En este sentido, los robots también deberán reconocer personas e interactuar con ellas, en entornos que cambian dinámicamente.

Sin embargo, esto es difícil de lograr con robots autónomos que utilizan sólo la información proporcionada por sus propios sensores (sensores de a bordo). Para hacer frente a esto, la comunidad robótica ha venido explorando la construcción de espacios inteligentes, es decir, espacios donde existen multitud de sensores y dispositivos inteligentes distribuidos por el entorno, que proporcionan información útil al robot. Cada vez más investigadores constatan el potencial innovador de integrar tecnologías robóticas con tecnologías de campos como la computación ubicua, las redes de sensores y la inteligencia ambiental. Esta integración, a menudo denominada "Robótica Ubicua" o "Sistemas Robóticos en Red", proporciona una forma nueva de construir sistemas de robots inteligentes al servicio de las personas.

En esta tesis, exploraremos el uso de estos espacios inteligentes para construir robots que operen en entornos complejos en períodos cortos de tiempo y de forma robusta. Nuestra propuesta consiste en la construcción de un espacio inteligente que permita un despliegue fácil, rápido y robusto de robots en diferentes entornos. Esta solución debe permitir a los robots moverse y operar de manera eficiente en entornos desconocidos, y debe ser escalable con el número de robots y otros elementos. Nuestro espacio inteligente consistirá en una red distribuida de cámaras inteligentes y robots autónomos. Las cámaras detectarán situaciones que podrían requerir la presencia de los robots, les informarán acerca de estas situaciones y también apoyarán su movimiento en el entorno. Los robots, por otra parte, deberán navegar con seguridad hacia las zonas donde se produzcan estas situaciones. Con esta propuesta, los robots no sólo serán capaces de reaccionar a los acontecimientos que se produzcan en su entorno más inmediato, sino a los que se produzcan en cualquier parte del entorno, si éstos son detectados por las cámaras. Como consecuencia, los robots podrán reaccionar a las necesidades de los usuarios, independientemente de dónde estén. Esto hará que los robots sean percibidos como más inteligentes, útiles y con más iniciativa. Además, la red de cámaras apoyará a los robots en sus tareas y enriquecerá sus modelos del entorno. Esto tendrá como resultado un despliegue más rápido y sencillo y un funcionamiento más robusto.

Nuestros robots deben operar en entornos frecuentados por personas, como hospitales o museos. Estos ambientes son relativamente estáticos, en el sentido de que los cambios de diseño no son frecuentes (por ejemplo, nuevas paredes, movimiento de muebles, etc.). Sin embargo, las condiciones de estos entornos son inherentemente dinámicas: siempre habrá gente moviéndose alrededor de los robots, la iluminación podrá cambiar constantemente, etc. Los robots deberán ejecutar sus tareas correctamente en estas condiciones y deberán hacerlo de forma continua. Sin embargo, la inteligencia necesaria para lograr este objetivo no tiene por qué ser propiedad exclusiva de los robots. De hecho, bajo el paradigma de la robótica ubicua, la inteligencia se distribuye entre todos los agentes, cámaras y robots en nuestro caso. El grado de inteligencia que pongamos en cada agente definirá en gran medida el rendimiento del sistema en su conjunto, sus funcionalidades, el comportamiento de cada agente, las relaciones entre ellos, etc. Teniendo esto en cuenta, en esta tesis exploraremos dos alternativas, en cuanto a la forma en que se distribuye la inteligencia en nuestro sistema: inteligencia colectiva e inteligencia centralizada.

Bajo el paradigma de la inteligencia colectiva, la inteligencia se distribuye equitativamente entre todos los agentes. La inteligencia global surge de la interacción entre los agentes individuales y no hay un agente central que soporte la mayor parte de la toma de decisiones. Esto es de alguna manera similar a lo que ocurre en los procesos auto-organizados que se observan en la naturaleza, donde no hay jerarquía ni centralización. En este caso, asumimos que es posible obtener robots que operen en entornos desconocidos a priori cuando su comportamiento surge de la interacción entre un conjunto de agentes independientes (cámaras), que cualquier usuario puede colocar en diferentes lugares del entorno. Estos agentes, inicialmente idénticos, serán capaces de observar el comportamiento de humanos y robots, aprender de forma paralela, adaptarse y especializarse en el control de los robots. En este sentido, nuestras cámaras deben ser capaces de detectar y realizar un seguimiento de los robots y los humanos de forma robusta en condiciones difíciles: cambios de iluminación, gente moviéndose alrededor del robot, etc. Además, deben ser capaces de descubrir a sus cámaras vecinas y de guiar la navegación de los robots a través de rutas de cámaras. Por su parte, los robots sólo deberán seguir las instrucciones de las cámaras y evitar colisiones con obstáculos. Para conseguir esto, en esta tesis se han abordado los siguientes hitos:

1. Se ha diseñado e implementado una red de cámaras que pueden ser desplegadas de forma fácil y rápida en distintos entornos. Nuestras cámaras tienen gran capacidad computacional y pueden alimentarse mediante un enchufe o a través de sus propias baterías, con 4 horas de autonomía. Además, pueden comunicarse de forma inalámbrica entre ellas y con los robots.

2. Se ha desarrollado una arquitectura software que controla la interacción entre todos los agentes del sistema. Esta arquitectura considera la existencia de dos tipos de agentes: agentes-robot y agentes-cámara. La arquitectura propuesta es totalmente distribuida y basada en procesos auto-organizables como los que se observan habitualmente en la naturaleza. Además, nuestra arquitectura es muy escalable, lo que nos permite eliminar e introducir agentes en el sistema sin apenas necesidad de reconfigurarlo.

3. Se ha estudiado la posibilidad de detectar y seguir al robot desde las cámaras utilizando tanto técnicas de reconocimiento de objetos como técnicas basadas en la instalación de marcadores pasivos en el robot (marcadores que no emiten luz). Ninguna de estas técnicas ha arrojado resultados satisfactorios en el contexto de aplicación de esta tesis, por lo que se ha diseñado y desarrollado un algoritmo para la detección y seguimiento

del robot desde las cámaras basado en el uso de marcadores activos. Este algoritmo ha mostrado una gran robustez y puede ser computado en tiempo real. Además, hemos demostrado que nuestro algoritmo puede detectar y seguir múltiples robots, e incluso identificar a cada uno de ellos.

4. Se ha desarrollado un sistema que permite a las cámaras detectar situaciones que requieran la presencia de los robots. Más en concreto, hemos desarrollado algoritmos que detectan: 1) gente saludando a las cámaras, 2) gente parada en ciertas áreas. Ambos algoritmos se han propuesto como ejemplos concretos de eventos de llamada que pueden requerir la presencia de nuestros robots. Sin embargo, nos gustaría destacar que el concepto de evento de llamada puede ajustarse a múltiples aplicaciones robóticas. Por ejemplo, las cámaras podrían detectar charcos de agua para ayudar a un robot limpiador, amenazas potenciales para ayudar a un robot vigilante, etc. Además, la detección de eventos de llamada podría ampliarse a otro tipo de dispositivos: por ejemplo, el usuario podría tener un "tag" o un teléfono inteligente a través del cual podría llamar al robot.

5. Hemos desarrollado un conjunto de algoritmos que permiten al sistema trabajar bajo el paradigma de inteligencia colectiva. Con estos algoritmos, las cámaras son capaces de: 1) detectar a sus cámaras vecinas, 2) construir rutas de cámaras a través de las que el robot puede desplazarse, 3) asignar un evento de llamada a un robot disponible, 4) apoyar la navegación del robot hacia este evento de llamada. Hemos mostrado que nuestras cámaras pueden establecer vínculos de vecindad cuando sus campos de visión se superponen, en base a la detección simultánea de alguno de los robots que se mueven por el entorno.

6. En caso de que no haya superposición de campos de visión de la red de cámaras: 1) nuestros robots pueden construir mapas de navegación entre cámaras vecinas que permitirán el desplazamiento entre sus campos de visión (estos mapas se pueden construir, por ejemplo, utilizando un escáner láser 2D), 2) las cámaras pueden detectar relaciones de vecindad a través de la re-identificación de personas que se muevan por el entorno. Experimentos en condiciones reales han demostrado la capacidad de nuestro sistema para trabajar con robots que no concentran toda la inteligencia, sino que ésta se distribuye entre todos los agentes. Los resultados experimentales han mostrado que nues-

tra propuesta es viable incluso en configuraciones donde el robot tiene una inteligencia muy limitada.

En segundo lugar, bajo el paradigma de la inteligencia centralizada, se le asignará a un tipo de agente mucha más inteligencia que al resto. Por tanto, a este agente se le asignará un mayor peso de cara a la toma de decisiones y coordinación, y su rendimiento tendrá una importancia mayor que el de otros agentes. Para explorar este paradigma, en esta tesis el papel de agente central será asignado al agente robot y la mayor parte de esta inteligencia se dedicará a la tarea de localización y navegación. Más concretamente, nuestros robots implementarán estrategias de localización multi-sensorial, fusionando información de fuentes complementarias a fin de lograr comportamientos robustos en entornos dinámicos y no estructurados. Entre otras fuentes de información, nuestros robots integrarán la información recibida de la red de cámaras. Bajo este paradigma, se han abordado los siguientes hitos:

1. A raíz de un estudio preliminar de las fortalezas y debilidades de diferentes fuentes de información que pueden ser utilizadas para localizar a un robot se ha llegado a la conclusión de que ningún sensor se comporta correctamente en todas las situaciones, pero que la combinación de sensores complementarios incrementa notablemente la robustez de los sistemas de localización.

2. Se ha desarrollado un algoritmo de localización que combina la información proveniente de múltiples sensores. Este algoritmo es capaz de proporcionar una estimación de la posición del robot robusta y precisa, incluso en situaciones donde algoritmos que utilizan un sólo sensor suelen fallar. Nuestro algoritmo puede fusionar información de un número arbitrario de sensores, incluso si no están sincronizados, trabajan a diferente frecuencia, o si algunos de ellos dejan de funcionar.

3. Hemos testado nuestro algoritmo de posicionamiento con múltiples sensores y hemos propuesto modelos de observación para cada uno de ellos.

   a) Escáner láser 2D: hemos propuesto un modelo que tiene en cuenta las diferencias entre la firma láser recibida y la esperada, tanto en su forma global como en cada uno de sus valores concretos.

   b) Brújula: hemos propuesto un modelo que tiene en cuenta las distorsiones magnéticas del entorno, de cara a minimizar su influencia en las lecturas de orientación.

c) Tarjeta receptora WiFi y una tarjeta receptora radio (banda 433MHz): hemos propuesto y analizado tres técnicas diferentes que permiten realizar posicionamiento en base a las potencia de las señales inalámbricas recibidas en cada instante.

d) Red de cámaras externas: hemos propuesto un modelo capaz de estimar la posición del robot a partir de la lista de cámaras que lo detectan en cada instante.

e) Cámara montada en el robot: hemos desarrollado un modelo basado en reconocimiento de escenas que permite estimar la posición más probable del robot a partir de una imagen.

4. Hemos realizado un estudio experimental completo de nuestro algoritmo de localización, tanto en condiciones controladas como en operación real durante demostraciones robóticas con usuarios. Se ha estudiado el comportamiento del algoritmo tanto utilizando cada sensor por separado, como utilizando la mayoría de sus combinaciones. Nuestra principal conclusión ha sido que, con significancia estadística, la fusión de sensores complementarios tiende a incrementar la precisión y robustez de las estimaciones de localización.

5. Hemos diseñado transceptores inalámbricos (motes) y los hemos integrado en nuestro sistema de posicionamiento en interiores. Estos transceptores usan el módulo CC1110, un “system-on-chip” de bajo coste y consumo. Este módulo cuenta con un procesador de frecuencia inferior a 1GHz y se comunica en la banda 433MHz ISM.

6. También hemos estudiado el rendimiento de nuestro algoritmo de posicionamiento cuando los transmisores inalámbricos del entorno son capaces de variar su potencia de transmisión en tiempo real. A través de un estudio experimental, hemos demostrado que esta habilidad tiende a mejorar el rendimiento de un sistema de posicionamiento inalámbrico.

7. Finalmente, hemos diseñado una metodología que permite a cualquier usuario desplegar y calibrar nuestro sistema en un corto periodo de tiempo en diferentes entornos.

Nuestra propuesta es una solución genérica que tiene cabida en muchas aplicaciones de robots de servicio diferentes. En esta tesis, hemos integrado nuestro espacio inteligente con un robot guía de propósito general que hemos desarrollado como ejemplo concreto de aplicación. Este robot está destinado a funcionar en diferentes entornos y eventos sociales, tales como museos, conferencias o demostraciones de robótica. Nuestro robot es capaz de detectar y rastrear a las personas que se encuentran a su alrededor, de seguir a un instructor por todo el entorno, de aprender rutas de interés del instructor y de reproducirlas para los visitantes del evento. Por otra parte, el robot puede interactuar con los usuarios, utilizando técnicas de reconocimiento de gestos y una interfaz de realidad aumentada. Este robot se ha mostrado y se ha testado en múltiples ocasiones tanto en centros de investigación como en otros entornos, como museos, escuelas, institutos y universidades. Durante estas demostraciones, nuestro robot ha sido utilizado con éxito por personas de diferentes edades y entornos.

Una tesis no es un trabajo cerrado, sino que debería abrir líneas de trabajo que puedan dar lugar a resultados interesantes en el futuro. En este sentido, creemos que esta tesis deja espacio para mejorar y para abrir nuevas líneas de investigación, como por ejemplo:

1. Durante la tesis se muestra que bajo el paradigma de inteligencia colectiva, el robot podría comportarse de forma inestable si las cámaras fallasen. Por otra parte, bajo el paradigma de la inteligencia centralizada, se muestra que el sistema completo depende de la correcta construcción de un mapa del entorno durante la fase de despliegue. Si este mapa no pudiese ser construido, el sistema no sería capaz de operar correctamente. En nuestra opinión, sería interesante explorar una solución intermedia. Por una parte, las cámaras podrían ayudar al robot durante la etapa de despliegue, de cara a explorar el entorno y calibrar todos los sistemas automáticamente. Después, el robot podría basar sus acciones en sus propias estimaciones de posición. En caso de que el algoritmo de posicionamiento fallase, las cámaras podrían apoyar la navegación del robot.

2. Sería de gran utilidad mejorar la detección de eventos de interés, para que se pudiesen detectar un número mayor de situaciones. La comunidad de Visión por Computador ha desarrollado decenas de algoritmos encaminados a la caracterización y detección de comportamientos de personas. Este tipo de técnicas podrían ser utilizadas para inferir si se precisa o no la asistencia del robot en situaciones más diversas que las planteadas en esta tesis.

3. En este sentido, consideramos también que la introducción de teléfonos inteligentes en el sistema sería muy interesante. Los teléfonos pueden ser localizados en el entorno y los usuarios podrían utilizarlos para solicitar la asistencia del robot. Además, el sistema podría rastrear a los usuarios para determinar su comportamiento e incluso generar nuevos eventos de llamada.

4. El modelo de localización basado en cámaras externas estima la posición aproximada del robot teniendo únicamente en cuenta la lista de cámaras que lo detectan en cada instante. Sería interesante desarrollar un modelo más exacto que tuviese en cuenta la posición del robot en la imagen y su correspondencia con el entorno físico.

5. La presencia de personas alrededor del robot distorsiona en gran medida sus medidas sensoriales (p. ej. las personas alrededor del robot ocluyen la visión del escáner láser, provocando distorsiones significativas en sus medidas). Consideramos que la inclusión de sensores robustos a estas situaciones puede mejorar el rendimiento de nuestro robot. En este sentido, en el futuro incorporaremos a nuestro robot una cámara omnidireccional que apunte al techo y detecte puntos de referencia en él (p. ej. luces).

6. Creemos que sería recomendable incluir en nuestro robot sensores y técnicas de percepción 3D de cara a evitar situaciones peligrosas (p. ej. caída por escaleras) y lograr una navegación más robusta y segura. Además, también sería interesante integrar esta percepción 3D en el proceso de localización.

7. Durante la tesis se ha demostrado que el uso de potencias de transmisión variables mejora significativamente el rendimiento de los sistemas de posicionamiento inalámbricos. Esto abre la posibilidad de que un agente robot pueda cambiar o influir en el funcionamiento de algunos o todos los agentes del entorno inteligente para mejorar su propio comportamiento y ser más eficaz. Por ejemplo, el robot sería capaz de modificar la potencia de transmisión de los emisores inalámbricos, de cara a descartar hipótesis de localización de forma proactiva (localización activa).

8. Consideramos de gran interés el uso del chip radio que hemos diseñado para comunicación entre los elementos del sistema. Este chip presenta un consumo menor que otras soluciones para comunicación (p. ej. WiFi) y aporta una gran flexibilidad en cuanto a su configuración.

9. Para finalizar, continuaremos mejorando el comportamiento y las funcionalidades de nuestro robot guía de propósito general.

Las arquitecturas, comportamientos y algoritmos que han sido propuestos en esta tesis van más allá del ámbito de la robótica. En los años venideros, se espera que los robots se integren en entornos como hospitales, museos u oficinas. Esta nueva generación de robots deberá ser capaz de navegar de forma autónoma y de operar de forma robusta de condiciones poco favorables: cambios de iluminación, gente moviéndose alrededor del robot, etc. Además, los robots deberán ser capaces de detectar humanos independientemente de donde éstos se encuentren, interaccionar con ellos, seguirlos y cooperar, tal y como hacen nuestros robots. Por otra parte, en la actualidad se observa una tendencia a poblar los entornos con dispositivos conectados con capacidad de procesamiento y percepción, como cámaras, balizas Bluetooth y todo tipo de sensores. Por todas estas razones, creemos que lo que ha sido propuesto dentro del marco de esta tesis puede formar parte de la base de futuras arquitecturas robóticas y comportamientos que puede que incluso sean comercializables en los próximos años.

Finalmente, nos gustaría destacar que esta tesis ha arrojado múltiples resultados de gran relevancia. En primer lugar, el contenido que se recoge en esta tesis ha dado lugar a 7 publicaciones en revistas indexadas en JCR y 1 publicación en revista con otros índices de calidad. Además, esta tesis también ha dado lugar a 6 publicaciones en congresos internacionales y 4 publicaciones en congresos nacionales. Finalmente, nos gustaría destacar que el conocimiento adquirido durante esta tesis es la base de la empresa Situm Technologies, una spin-off de la Universidade de Santiago de Compostela de la que el autor de esta tesis es socio fundador. Situm desarrolla y comercializa tecnologías de localización en interiores para telÃ©fonos inteligentes. La tecnología de Situm se basa en la fusión inteligente de todos los sensores del telÃ©fono inteligente (WiFi, BLE, magnetómetro) y en la estimación precisa del desplazamiento del usuario utilizando el acelerómetro y giróscopo de estos dispositivos. A pesar de haber sido fundada en 2014, Situm ya cuenta con un equipo de más de 10 personas, ha sido galardonada con diversos premios, ha recibido apoyo financiero por parte de un fondo de capital riesgo y ha conseguido facturar más de 100.000 € hasta la fecha.

Abstract

One of the current challenges in robotics is the integration of robots in everyday environments. Successes in applications of autonomous mobile robotics have been so far restricted to well defined, fairly narrow application scenarios in which boundary conditions and exceptions are largely known a priori. Nevertheless, personal and professional service robot applications demand more intelligent robots because they must perform complex tasks in decreasingly well-structured and known environments, where less human instruction or supervision should be needed over time.

However, it is difficult to achieve this with stand-alone robots that use only the information provided by their own sensors (on-board sensors). In this thesis, we will explore the use of intelligent spaces (i.e. spaces where many sensors and intelligent devices are distributed and which provide information to the robot), to get robots operating in complex environments in a short period of time. Our proposal is to build an intelligent space that allows an easy, fast, and robust deployment of robots in different environments. This solution must allow robots to move and operate efficiently in unknown environments, and it must be scalable to the number of robots and other elements.

Our intelligent space will consist of a distributed network of intelligent cameras and autonomous robots. The cameras will detect situations that might require the presence of the robots, inform them about these situations, and also support their movement in the environment. The robots, on the other hand, will navigate safely within this space towards the areas where these situations happen. With this proposal, our robots are not only able to react to events that occur in their surroundings, but to events that occur anywhere. As a consequence, the robots can react to the needs of the users regardless of where the users are. This will look as if our robots are more intelligent, useful, and have more initiative. In addition, the network of cameras will support the robots on their tasks, and enrich their environment models. This will result on a faster, easier and more robust robot deployment and operation.

In this thesis, we will explore two alternatives, regarding how the intelligence is distributed among the agents: collective intelligence and centralised intelligence. Under the collective intelligence paradigm, intelligence is fairly distributed among robots and cameras. Global intelligence arises from the interaction among individual agents, and there is not a central agent that handles most decision making. This is somehow similar to self-organization processes that are usually observed in nature, where there is no hierarchy nor centralisation. In this case, we assume that it is possible to get robots operating in a priori unknown environments when their behaviour emerges from the interaction amongst an ensemble of independent agents (cameras), that any user can place in different locations of the environment. These agents, initially identical, will be able to observe human and robot behaviour, learn in parallel, adapt and specialize in the control of the robots. To this extent, our cameras will be able to detect and track robots and humans robustly, to discover their camera neighbours, and to guide the robot navigation through routes of these cameras. Meanwhile, the robots must only follow the instructions of the cameras and negotiate obstacles in order to avoid collisions. In order to accomplish this, we have achieved the following milestones:

1. We have designed and implemented a network of cameras that can be deployed in a fast and easy manner in different environments. These cameras can communicate wirelessly among them and with the robots.

2. We have developed a software architecture that controls the interaction amongst all the agents of the system. The architecture is fully distributed and very scalable.

3. We have designed and developed an algorithm for robot detection and tracking based on active markers. Our algorithm has shown to be very robust in real experiments, and can work in real-time with several robots.

4. We have developed a system to detect situations that may require the presence of the robots from the cameras. Specifically, we have developed algorithms for the detection of: 1) people waving at the cameras, 2) people standing in certain areas. Both algorithms have been proposed as specific examples of call events that might require the presence of our robots.

5. We have developed a set of algorithms that allows the system to work under the collective intelligence paradigm. With these algorithms, the cameras were able to: 1) detect their camera neighbours, 2) construct routes of cameras through which the robot can navigate, 3) assign a call event to an available robot, 4) support the robot navigation towards this call event. We have shown that our cameras can establish neighbourhood links when their Fields of View (FOVs) overlap, attending at simultaneous detections of the robot. In other case: 1) the robot can construct maps for navigation between neighbour cameras (e.g. occupancy maps from a 2D laser scanner), 2) the cameras can detect their neighbourhood relationships by re-identifying people walking around. Real world experiments have shown the feasibility of our proposal to work with "naive" robots.

On the other hand, under the centralised intelligence paradigm, one type of agent will be assigned much more intelligence than the rest. Therefore, this agent will make most decision making and coordination, and its performance will have a higher importance than that of other agents. To explore this paradigm, in this thesis, the role of central agent will be played by the robot agent, and most of this intelligence will be devoted to the task of self-localisation and navigation. More concretely, our robots will implement multi-sensor localisation strategies that fuse the information of complementary sources, in order to achieve a robust robot behaviour in dynamic and unstructured environments. Under this paradigm, we have achieved the following milestones:

1. We have performed an experimental study about the strengths and weaknesses of different information sources to be used for the task of robot localisation. The study has shown that no source performs well in every situation, but the combination of complementary sensors may lead to more robust localisation algorithms.

2. We have developed a robot localisation algorithm that combines the information from multiple sensors. This algorithm is able to provide robust and precise localisation estimates even in situations where single-sensor localization techniques usually fail. It can fuse the information of an arbitrary number of sensors, even if they are not synchronised, work at different data rates, or if some of them stop working. We have tested our algorithm with the following sensors: a 2D laser range finder, a magnetic compass, a WiFi reception card, a radio reception card (433 MHz band), the network of external cameras, and a camera mounted in the robot. We have also proposed one or several observation models for each sensor.

3. We have performed a complete experimental study of the localisation algorithm, in both controlled and real conditions during robotics demonstrations in front of users. We have studied the behaviour of the algorithm when using each single sensor and most of their combinations. Our main conclusion was that, with statistical significance, the fusion of complementary sensors tends to increase the precision and robustness of the localisation estimates.

4. We have designed wireless transmitters (motes) and we have integrated them into our indoor positioning algorithm. Our motes use an inexpensive, low-power sub-1-GHz system-on-chip (CC1110) working in the 433-MHz ISM band.

5. We have studied the performance of our positioning algorithm when the wireless transmitters in the environment are able to vary their transmission power. Through an experimental study, we have demonstrated that this ability tends to improve the performance of a wireless positioning system. This opens the door for future improvements in the line of active localisation. Under this paradigm, the robot would be able to modify the transmission power of the transmitters in order to discard localisation hypotheses proactively. In general, this will allow robot-agents to influence the behaviour of other agents of the intelligent environment in order to improve its own behaviour and being more effective.

6. Finally, we have designed a methodology that allows any user to deploy and calibrate this system in a short period of time in different environments.

Our proposal is a generic solution that can be applied to many different service robot applications. In this thesis, we have integrated our intelligent space with a general purpose guide robot that we have developed in the past, as an specific example of application. This robot is aimed to operate in different social environments, such as museums, conferences, or robotics demonstrations in research centres. Our robot is able to detect and track people around him, follow an instructor around the environment, learn routes of interest from the instructor, and reproduce them for the visitors of the event. Moreover, the robot is able to interact with humans using gesture recognition techniques and an augmented reality interface.

CHAPTER 1

INTRODUCTION

1.1 The rise of service robots

There are two kinds of robots: industrial robots and service robots. The ISO 8373:1994 standard defines an industrial robot as an "automatically controlled, re-programmable, multi-purpose manipulator programmable in three or more axes". An industrial robot can be either mobile or fixed, although the most common ones are fixed robotic arms used for tasks such as welding, painting, assembly or packaging. Figure 1.1-(a) shows an example of an industrial robot used to weld. On the other hand, the International Federation of Robotics states that "a service robot is a robot which operates semi or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations". Figures 1.1-(b) and 1.1-(c) show examples of professional and personal service robots, respectively.

Industrial robots operate under pre-defined conditions, and the tasks that they have to perform are usually clearly defined at design stage. On the other hand, service robots operate under non pre-defined conditions, in environments where strict boundary conditions do not exist, such as museums, conferences, or shopping centres. In these places, environmental changes are frequent, people walk around, the robot can move almost anywhere, etc. Therefore, the interaction of the robot with the environment and with the users presents a high degree of uncertainty. In fact, successes in applications of autonomous service robots have so far been restricted to well defined, fairly narrow application scenarios in which boundary conditions and exceptions are largely known a priori, and in which the robot is therefore able to resort to pre-installed control programs specifically designed for the environment where the robot moves.

(a) Industrial robot (b) Professional service robot (c) Personal service robot

Figure 1.1: Examples of the different types of robots: industrial, a welding robot from ABB; professional, a pick and place agricultural robot from Harvest Automation; and personal robots, a vacuum cleaner Roomba from iRobot.

Industrial robotics is an established field that has produced hundreds of successful robot models used in industries all around the world. On the contrary, most service robotics applications are still confined within research centres. In fact, service robots currently represent a tenth of the sales of industrial robots. However, in the following decades, personal service robots are expected to become part of our everyday life, either as assistants, house appliances, collaborating with the care of the elderly, etc. In this regard, the International Federation of Robotics (IFR) has estimated [1] that more than 4 million personal service robots were sold in 2013 (28% growth with respect to 2012), which represents a total market value of 1.7 B$. The same organisation estimates that in the period 2014-2017 more than 31 million personal service robots will be sold, representing a total value of 11 B$ in the whole period. Similarly, the Japanese Ministry of Economy, Trade and Industry and the Industrial Technology Development Organisation has also forecasted a growth of almost 1700% of service robots in Japan: from 0.45B€ to 7.8B€ in the period 2011-2020 (Fig. 1.2). According to the predictions shown in Fig. 1.2, service robots sales are expected to surpass the industrial robots around year 2020.

Nowadays, the robotics community has already developed a first generation of personal service robots that performs limited yet useful tasks for humans in human environments, enabled by progresses in robotics' core fields such as computer vision, navigation, or machine learning. In parallel, hardware elements like processors, memories, sensors, and motors have been continuously improving, while their price has been dropping. This makes roboticists positive about the possibility of building quality robots available to the vast majority of soci-

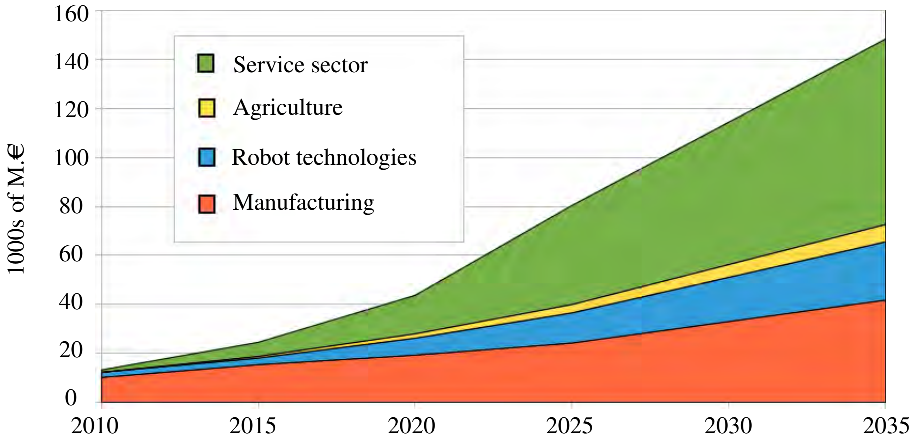


Figure 1.2: Market size estimates for the service robotics sector in Japan. Source: the Japanese Ministry of Economy, Trade and Industry and New Energy and the Industrial Technology Development Organisation (NEDO).

ety. Moreover, since some companies are already investing in business models such as robot renting, getting robots to work in places like museums, conferences, or shopping centres will become more affordable. Even more, according to our experience, more and more research groups are being requested to take their robots to social events (e.g., public demonstrations). In our opinion, all of this reflects the increasing interest of society for robots that assist, educate, or entertain in social spaces.

At this point, it is paramount to start providing affordable solutions to answer to society's demand. There are still a number of challenges that should be overcome in order to allow these robots to reach the mass consumer market. More specifically, the robotics community [2] has identified the following ones:

C1. The development of systems that can perceive the environment and the humans in a more robust and reliable level.

C2. The development of autonomous systems that are capable of learning from the user and the robot's experience.

C3. The development of cognitive and reasoning systems that are able to operate in dynamic and non structured environments.

C4. The development of interfaces that allow and ease the human-robot interaction.

C5. The development of systems that are capable of autonomously managing the robot's energy, exploring new energy sources.

C6. The design of modular architectures for the development and reuse of advances systems.

In this thesis, we will address the challenge C3: the development of systems that are able to operate in dynamic and non structured environments. In addition, we consider that there are two problems that are restraining this first generation of robots to get out of the research centres: (1) the cost of the deployment of robotic systems in unknown environments, and (2) the seamless integration of robotic technologies in the environments where people works and lives. Ideally, the deployment of robots in new environments should be fast and easy, but in practice it requires experts to adapt both the hardware and the software of the robotic systems to the environment. This includes programming "*ad hoc*" controllers, calibrating the robot sensors, gathering knowledge about the environment (e.g., metric maps), etc. This adaptation is not trivial and may require several days of work, making the process inefficient and costly. Instead, we believe that the deployment must be as automatic as possible, prioritizing online adaptation and learning over pre-tuned behaviours, knowledge injection, and manual tuning in general. On the other hand, the seamless integration of the robots in our environments will require robots and intelligent systems that are able to interact with people in a natural way, be easy to use even to non-experts, perceive the whole environment and the events that occur within them, and show initiative to offer the services by anticipating users' needs.

1.2 Intelligent spaces

Ubiquitous computing enhances computer use by making many computers available throughout the physical environment, while making them effectively invisible to the user [3]. This was stated in 1993 by Mark Weiser, father of the term "ubiquitous computing". He envisioned a world were computing would be immersed in the environments where people live and work, allowing interaction in a natural way through gestures and speech. By invisible, he meant that computing tools should allow the users to focus on the tasks, instead of in the tools. As such, computing should be used unnoticed and without the need of technical knowledge about the equipment.

Intelligent spaces were born as a direct consequence of these new ideas. Intelligent spaces are environments equipped with networks of sensors and actuators able to perceive their surrounding world, interact with people, and provide different services to them. To date, several research projects have explored the concept of intelligent spaces both in corporate [4, 5] and domestic environments [6, 7, 8]. Even more, there is a current trend ("Internet of Things") to populate our environments with networked devices with processing and sensing capabilities, such as cameras, Bluetooth beacons, and all sorts of sensors.

Unfortunately, most service robots still carry out all the deliberation and action selection on-board, based only on their own perceptions. This is the case of a great number of the most remarkable personal robots of the last decades, such as Rhino (1995) [9], Minerva (1999) [10], Robox (2003) [11], Tourbot (2005) [12], and Urbano (2008) [13]. These robots are only able to react to events that occur in their surroundings and interact with users that are near them, which is very restrictive. Opposed to this philosophy, a new paradigm called ubiquitous robotics [14] proposes to integrate robots as part of intelligent spaces. Therefore, the intelligence, perception and action components would be distributed amongst a set of networked devices (robots, laptops, smart-phones, sensors...). Within this paradigm, for instance, a robot can perceive users' needs anywhere in the intelligent space, regardless of where the robot is. This will look like if the robot has initiative, and it will improve people's opinion on its role. On the other hand, this ubiquitous space could enrich the robot's models of the environment, and support it on the tasks that it carries out. This will definitelly reduce the dependency of previous knowledge and hard-wired controllers, which will result in a faster robot deployment.

Lee et al. (Hashimoto Labs.) were pioneers in combining intelligent spaces and robotics [15]. They proposed a system of distributed sensors (typically cameras) with processing and computing capabilities. With this system, they were able to support robots' navigation [16, 17] in small spaces (two cameras in less than 30 square meters). Similarly, the MEPHISTO project [18, 19] proposed to build 3D models of the environment from the images of highly-coupled cameras. Then, they utilized these models for path planning and robot navigation. Their experiments were performed in a small space (a building's hall) with four overlapped cameras. Finally, the Electronics Department of the University of Alcalá (Spain) has proposed several approaches for 3D robot localization and tracking, using a single camera [20] or a camera ring [21, 22]. Moreover, they have proposed a navigation control strategy that uses 4 infrared cameras [23], which was tested in a 45 m^2 laboratory with four cameras.

All of these works have demonstrated the feasibility of robot navigation supported by external cameras. Regretfully, all of them have been designed and validated to work in small places with a great number of sensors, which compromises their scalability and cost. Furthermore, they are not focused on the provision of useful services for the users, but mainly on the robot localization and support of its navigation by the cameras. Moreover, they rely on a centralized processing of the information, and wired communications, making the scalability even harder.

A different concept is explored in the PEIS Ecology (Ecology of Physically Embedded Intelligent Systems) [24, 25], which distributes the sensing and actuation capabilities of robots within a device network, such as a domotic home would do. They focus on high level tasks, such as in designing a framework to integrate a great number of heterogeneous devices and functionalities [26] (cooperative work, cooperative perception, cooperative re-configuration upon failure or environment changes...). However, this project does not tackle low level tasks critical for our purposes, such as robot navigation or path planning.

The Japan NRS project and the URUS project went a step further. The NRS project focuses on user-friendly interaction between humans and networked environments. These environments consist of sensors, robots and other devices to perceive the users, interact with them, and offer them different services. They demonstrated the use of their systems in large real field settings, such as science museums [27], shopping malls [28], or train stations [29], during long term exhibitions. In these works, most communications are wired and all processing takes place in a central server, which compromises the systems' scalability. On the other hand, their robots are not fully autonomous, since a human operator controls them in certain situations. Finally, they did not use video-cameras to sense the environment, which is a sensor that can provide rich information of the environments and human activities. On the other hand, the URUS project (Ubiquitous networking Robotics in Urban Settings) [30, 31, 32, 33] proposes a network of robots, cameras and other networked devices to perform different tasks (informative tasks, goods transportation, surveillance, etc.) in wide urban areas (experimental setup of 10,000 m^2). The cameras of the system are able to detects robots and persons and recognise gestures from persons. The robots, on the other hand, are able to navigate autonomously throughout the environment. In this system, the cameras are always in the same fixed positions, most communications are wired, and the control processes (e.g., task allocation) and information processing takes place in a central station, etc. This makes clear that neither the efficiency of the deployment phase nor other characteristics such as the scalability and flexibility to introduce new elements in the system were considered (probably because the system is intended to operate always in the same urban area).

The projects above are the most representative in the context of intelligent spaces and mobile robots. Even more, Sanfeliu et al. [34] described the last three projects (PEIS, NRS and URUS) as "the three major on-going projects in Network Robotic Systems" by the end of 2008, and to the best of our knowledge, this assertion is still valid. We consider that the described works do not target the problem of the efficient robot deployment in unknown environments. Thus, a solution to get robots out of the laboratories within reasonable times and costs is still to be proposed.

1.3 Goal of the thesis

In this thesis, we propose to combine technologies from ubiquitous computing, intelligent spaces, and robotics, in an attempt to provide robots with initiative and get them to work robustly in different environments. Our proposal is to build an intelligent space that allows for an easy, fast, and robust deployment of robots in different environments. This solution must allow robots to move and operate efficiently in unknown environments, and it must be scalable to the number of robots and other elements.

Our intelligent space will consist of a distributed network of intelligent cameras and autonomous robots. The cameras will detect situations that might require the presence of the robots, inform them about these situations, and also support their movement in the environment. The robots, on the other hand, will navigate safely within this space towards the areas where these situations happen. With this proposal, our robots are not only able to react to events that occur in their surroundings, but to events that occur anywhere. As a consequence, the robots can react to the needs of the users regardless of where the users are. This will look as if our robots are more intelligent, useful, and have more initiative. In addition, the network of cameras will support the robots on their tasks, and enrich their environment models. This will result on a faster, easier and more robust robot deployment and operation.

Our proposal is a generic solution that can be applied to many different service robot applications. In this thesis, we have used our intelligent space with a general purpose guide robot that we have developed in the past [35], as an specific example of application. This robot is aimed to operate in different social environments, such as museums, conferences, or robotics demonstrations in research centres. Our robot is able to detect and track people around him, follow an instructor around the environment, learn routes of interest from the instructor, and reproduce them for the visitors of the event. Moreover, the robot is able to interact with humans using gesture recognition techniques and an augmented reality interface.

This thesis is organised as follows:

1. Chapter 2. We provide a general description of our intelligent space, focusing on the hardware construction of cameras and robots, and on the general tasks that they must perform.

2. Chapter 3 . We describe the algorithms used by the cameras to detect the robots and the situations that require their presence.

3. Chapter 4. We describe the distributed software architecture that controls the interaction amongst all the agents of the system, and that ensures that all the tasks are performed adequately. More concretely, we describe how the cameras are able to: 1) detect their camera neighbours, 2) construct routes of cameras through which the robot can navigate, 3) assign a call event to an available robot, 4) support the robot navigation towards these call event. In this chapter, we assume that the robot does not have any self-localisation capability, and only simple navigation abilities. Therefore, all the intelligence is assigned to the cameras.

4. Chapter 5. We describe a robot localisation algorithm that combines the information from multiple sensors. This algorithm is able to provide robust and precise localisation estimates even in situations where single-sensor localization techniques usually fail. We have tried this algorithm with the following sensors: a 2D laser range finder, a magnetic compass, a WiFi reception card, the network of external cameras, and a camera mounted in the robot.

5. Chapter 6. We explore the use of varying transmission powers to increase the performance of a wireless localization system. To this extent, we have designed a robot positioning system based on wireless motes. Our motes use an inexpensive, low-power sub-1-GHz system-on-chip (CC1110) working in the 433-MHz ISM band.

6. Chapter 7. We describe our general purpose guide robot, and its integration with the results of this thesis.

7. Chapter 8. We describe the company Situm Technologies, which provides indoor positioning technologies for smartphones. The author of this thesis is co-founder of Situm Technologies, and part of the know-how acquired during this thesis was transferred to the company.

8. Chapter 9. We describe the main conclusions of this thesis and explore the research lines that this thesis opens for future work.

Chapter 2

Intelligent space for a fast robot deployment

Intelligent spaces are able to perceive the environment, the people, and the events that occur in them. Additionally, they have the ability to perform actions that affect the environment and the people, and to provide communication capabilities among the different elements of the space. For instance, an intelligent space in a building might perceive the temperature in every room, plus the presence of people or the preferences of each person, in order to regulate this variable. Similarly, we believe that an intelligent space for mobile robotics should perceive situations that require the presence of robots and people within the environment. Likewise, the robots will be in charge of performing the core robotic tasks, supported by the intelligent space, that provides information to help them in these duties.

With this in mind, we have designed an intelligent space composed of two main elements (Fig. 2.1): (a) an intelligent control system formed by camera-agents spread out on the environment (CAM1 to CAM7 in the figure), and (b) autonomous robots navigating on it (RA and RB, in the figure). The cameras are able to detect the robot, detect situations that might require the presence of the robots (call events, CE en the figure), inform them about these situations, and also support their navigation. The robots, on the other hand, navigate safely within this space towards the areas where these situations happen. When they arrive, they can develop different tasks.

Note that the concept of call event can be accommodated to a wide range of robotics applications. For instance, we could detect water spills to aid a cleaning robot, potential

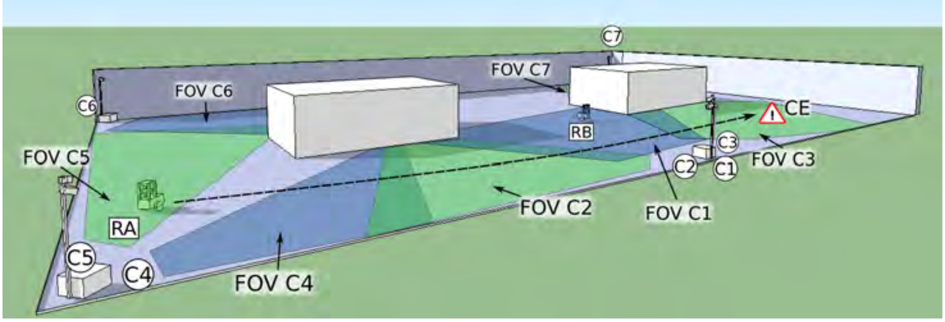


Figure 2.1: Example of operation of our system. We can observe 7 cameras (C1 to C7), their Fields of View (FOV C1 to FOV C7), and 2 robots (RA and RB). Camera C3 is detecting a call event (CE), robot RA is being seen from camera C5 and robot RB from camera C1. We can also see a trajectory which the robot RA can follow to get to CE.

threats to aid a security robot, etc. In the case of a tour-guide robot, call events may be triggered by users asking explicitly for assistance (e.g., by waving in front of a camera). Even more, call events can be activated even if no user triggers them explicitly, for instance taking into account the behaviour of the people in the environment. For instance, the cameras could detect groups of people that are staying still at the entrance of a museum, infer that they may need assistance, and send them the robot. Upon arrival, the robot would offer them information, assistance, etc.

2.1 Camera agents

Each of our camera-agents consists of an aluminium structure like that of Fig. 2.2, which is easy to transport, deploy and pick up. This aluminium structure has two parts: a box and a mast with one or more cameras on top. The box has a polycarbonate cover, and contains a processing unit (e.g. laptop), a WiFi Access Point and four 12V lead-acid batteries. Moreover, each box contains a DC/AC laptop adapter (to power up the laptop using the batteries), and a AC/DC 12A battery charger. The autonomy of the camera agent is 4 hours when using the batteries, and indefinite when the battery charger is connected to a wall plug.

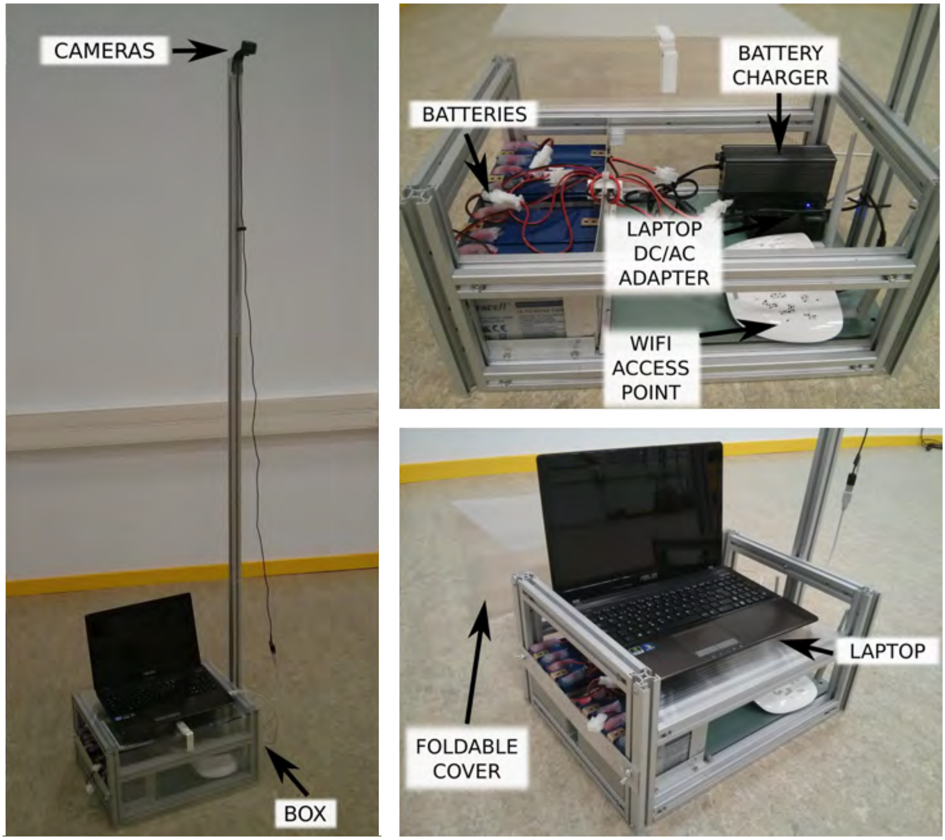


Figure 2.2: Camera agent.

From a functional perspective, the cameras have to carry out four main tasks:

1. Detect and track the robot (Sec. 3). The cameras will need to detect any robot within their Field of View (FOV) and track its position. They will also be required to distinguish among different robots.

2. Establish neighbourhood relationships among them (Sec. 4). The cameras must detect who are their camera neighbours and establish neighbourhood relationships with them.

3. Detect events which require the robots' presence (Sec. 3). These events are called *call events*.

4. Calculate routes of cameras through which the robot can navigate towards where a call event was detected.

5. Support the robot navigation towards where its presence is required.

2.2 Robot agents

We work with Pioneer 3DX robots like the one in Fig. 2.3. These robots are equipped with a processing unit (laptop) and four main sensors: a laser scanner, a magnetic compass, a WiFi network card, and a Microsoft Kinect. In addition, we have placed in our robots a set of colour LED strips, that form patterns that can be recognized from the cameras, as we will explain in Sec. 3. Finally, our robots have a screen, a microphone, and speakers. These elements support the interaction of the robot with the users.

Our robots carry out one main task: navigate safely towards call events detected by the cameras. In addition, we must bear in mind that, as an example application, we will integrate our proposal with a general purpose tour-guide robot that we have developed in the past [35]. This tour-guide robot and the integration with our system is described in Sec. 7. As a brief introduction, this robot has the following abilities:

- Person following. The robot is able to detect and track a human, distinguish him from others, and follow him.

- Interaction with humans. Our robot recognises human's gestures (commands). Depending on the gestures recognised, it executes different behaviours.

- Route recording. An instructor can teach the robot different routes of interest in the event where the robot operates. To this extent, the robot must follow this instructor along the desired route. Moreover, the robot can record voice messages at points of interest within the route.

- Route reproduction. A visitor of the event where the robot operates can request the robot to reproduce the routes and voice messages previously recorded.

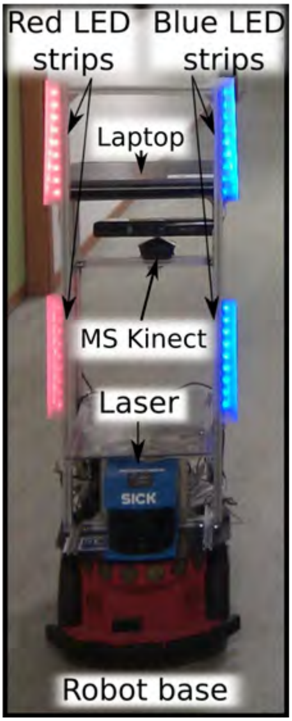


Figure 2.3: Robot agent and its main components: base, sensors, active markers and processing unit.

2.3 Fast and easy deployment

During the last decades, we have seen how the deployment time of non-ubiquitous robots has decreased up to an acceptable point. For instance, Rhino and Minerva originally required 180 and 30 days of installation respectively, while Tourbot and Webfair could be deployed in less than two days [12]. Following this trend, Urbano required less than an hour for a basic installation (map building for localization) [13].

Previous works on ubiquitous robotics have focused on developing intelligent spaces that would work at fixed locations, instead of allowing to use them in different environments (Sec. 1). Therefore, the problem of the easy and fast robot deployment has not been addressed properly in this kind of systems. On the contrary, we have made an special effort to make our system easy and fast to deploy in different environments. All the elements of our system are easy to transport, to mount and, over all, to configure. As we will see, we always prioritise self-configuration over manual tuning. In addition, we have provided all the agents of the system with sufficient computational power to process their sensor information locally, instead of having to send this information to a central server. This allowed us to use only wireless communications, avoiding the cost of deploying a wired network infrastructure. Moreover, all the agents are powered by batteries, which can be hot-swapped. This allows our system to work even if the environment does not have an electric power infrastructure, although obviously this is highly recommended. Finally, the system is scalable and flexible, meaning that the introduction or elimination of elements during operation is easy and fast (no re-design and minimal re-configuration required). Therefore, we can introduce or eliminate cameras and robots without major adjustments, which makes the deployment very incremental.

The use of the system in a new environment goes through two different stages: a "deployment" phase, and an "operation" phase. The "deployment" phase consists on placing all the elements in the environment, and configuring them correctly. We have designed a methodology to achieve this deployment, which consists on the following steps:

1. The user deploys the cameras in the environment.

2. The user moves the robot within the environment (Fig. 2.4). During this process, both the robot and the cameras gather information that is required for the efficient functioning of the system. The user may move the robot either: a) with a joystick or, 2) with

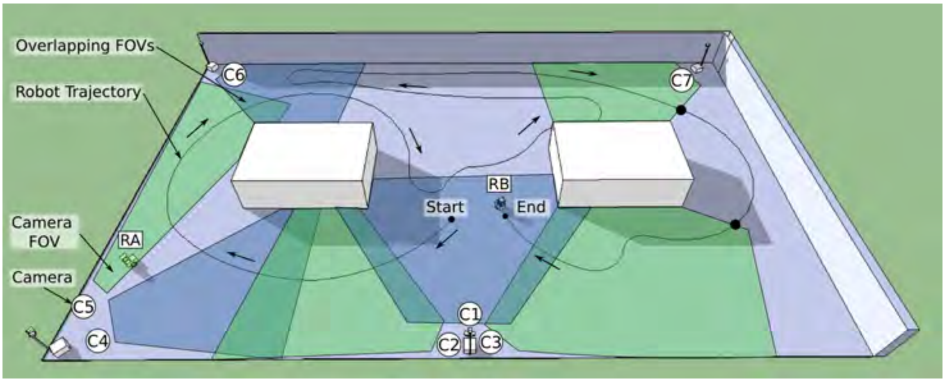


Figure 2.4: Example of deployment of our system. The user drives the robot around the environment to capture all the data needed for calibration.

our person-following behaviour with gesture based interaction and voice feedback (Sec. 7). We have observed in real tests that non-expert users are able to perform this step successfully using any of both methods.

With this methodology, a single user can deploy and configure a system of 5 cameras and 1 robot in an environment of 1100 m^2 in less than 1 hour (Sec. 5). On the other hand, during the "operation" phase is when the system provides the services that are useful for the users.

2.4 Where is the intelligence?

Under the ubiquitous robotics paradigm, the intelligence is distributed amongst a set of agents, which are embodied in networked devices such as robots, smart cameras, smart-phones, etc. The degree of intelligence that we put in each agent will greatly define the functionalities of the system as a whole, the behaviour of each agent, the relationships among them, etc. In this thesis, we will consider two different paradigms:

– Collective intelligence. Under this paradigm, intelligence is fairly distributed among agents. Global intelligence arises from the interaction among individual agents, and there is not a central agent that handles most decision making. This is somehow similar to self-organization processes that are usually observed in nature, where there is no hierarchy nor centralisation. In this scenario, for instance, robots will be able to negotiate obstacles and navigate between specific locations, but they will need the help of the rest of the agents.

– Centralised intelligence. Under this paradigm, one type of agent will be assigned much more intelligence than the rest of the agents. Therefore, this agent will make most of decision making and coordination, and its performance will have a higher importance than that of other agents. To explore this paradigm, in this thesis, the role of central agent will be played by the robot agent.

Note that these are not crisp definitions, as there is a range of possibilities between fully centralised and fully collective intelligence. Moreover, a system may transition between collective and centralised intelligence, or viceversa. For instance, a robot may rely heavily on a camera network that supports its navigation, until it constructs a model of the environment that allows the robot to make its own decision making autonomously. Similarly, an intelligent robot that does all the decision making may help cameras to establish neighbourhood relationships among them. This will increase the intelligence of the cameras and allow the robot to transfer some of its responsibilities to them.

CHAPTER 3

DETECTION OF ROBOTS AND CALL EVENTS

The objective of the camera-agents in an intelligent space is to support the operation of robots and to detect call events (situations that require the presence of the robots). This implies that the cameras will need to carry out two visual recognition tasks: 1) the detection and tracking of the robots, and 2) the detection of call events that happen in the environment. These tasks will be performed by the cameras in any of the two approaches detailed in the previous section (collective intelligence vs. intelligence centralised on the robot).

In this chapter, we will explain the approach that we have followed to accomplish both tasks. First of all, we will describe how to detect robots with object classifiers and passive markers. Then, we will explain an algorithm to detect robots that carry active markers. Finally, we will describe how our cameras can detect people and groups of people that require the presence of the robots, as specific examples of call events.

3.1 Robot detection: an overview

In this section, we will describe the different approaches we have considered in order to detect our robots from the cameras.

3.1.1 Robot detection and tracking with object classifiers

Our first approach was to use standard object recognition techniques, as a generic approach to detect a robot. These techniques aim at finding and identifying objects in an image or video sequence, by using classifiers previously trained with examples of each object class.

We tried two of the most popular techniques. First, we tried a method proposed by Dalal et al. [36], based on an SVM (Support Vector Machine) classifier with HOG features (Histogram of Oriented Gradients). Second, we used the standard Viola-Jones detector [37, 38], based on Haar-like features and cascades of decision-tree classifiers. We carried out several experiments and after analysing the results we decided to discard both techniques, because:

1. The process of training the classifiers was tedious and time consuming. As a test, we have built a small database of approximately 200 positive examples and 2000 negative examples and trained the algorithms with them. The training process took several hours and the results were not very promising in real experiments, due to high rates of false positives and false negatives. This might have happened because the cameras have to detect the robots in different lighting conditions, positions, angles, sizes (very close or very far away from the cameras), etc. We believe that we could get better results with a larger database, but it did not seem reasonable to spend weeks collecting images of the robot and training the classifiers.

2. The Dalal method required too much computation to be processed in real-time (almost a second for each image). On the contrary, the Viola-Jones method could be computed in real time, but the results were not satisfactory: high rate of false positives and false negatives.

3. We wanted to be able to detect several robots and to distinguish them. Since all of our robots are Pioneer 3DX or similar, we would have to put a different markers in each robot, and train the classifiers to distinguish them. Given the poor results obtained when trying to detect a single robot, we did not even try to tackle this task.

These observations have led us to discard the use of these standard object recognition techniques. Instead, we tried to put markers in our robots, to simplify the detection and identification of the robots.

3.1.2 Robot detection and tracking with passive markers

Our second approach to detect the robot from the cameras was to use passive markers. First of all, we put a matte carton marker on top of the robot, like the one shown in Fig. 3.1. Then, the cameras filtered each video image to erase all the colours, except the marker's colour. This resulted on a list of blobs of the same colour as the marker (in our context, a blob is a set of

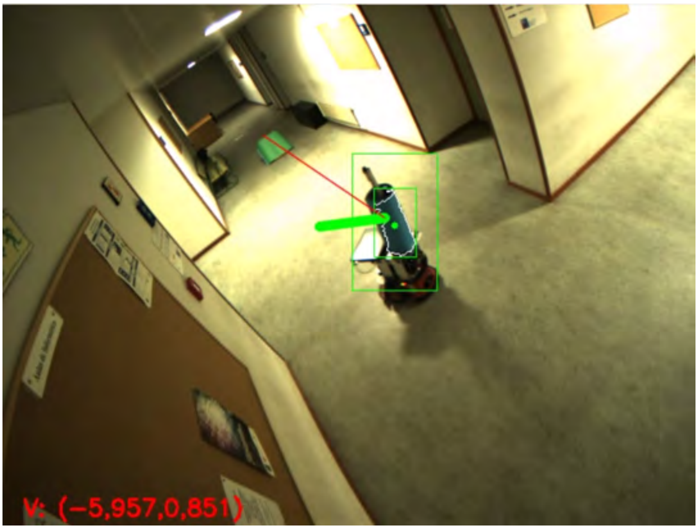


Figure 3.1: Robot detection with a blue passive marker. We represent: 1) the centroid of the robot blob with a green dot, 2) the area where the blob is detected with a small green box, 3) the area where the marker will be searched in next image frame with a big green box, 4) the direction of movement of the robot with a green line, and 5) the direction where the robot should aim (call event) with a red line.

connected pixels of the same colour). From this list, the cameras would select the blob with the closest height-width ratio to the one of the robot. Then, the cameras would just track this blob over time, by searching for it in subsequent image frames. Moreover, by computing the displacement of the blob among consecutive frames, we would be able to detect direction and orientation of the movement of the robot (Fig. 3.1).

With this method, we were able to detect the robot efficiently and the cameras were able to support the robot navigation throughout the environment (see experiments in Secs. 4.1.4 and 4.1.5). However, this method had two main flaws:

1. The colour filtering stage produced a high number of false positives, that sometimes were very difficult to filter out based on height-width ratio rules. Therefore, noise blobs would be from time to time mistaken as a robot. This would introduce errors in the robot navigation. This was not a big issue because we were able to solve this problem by using background subtraction techniques [39], which basically delete everything in the image that is part of the background (e.g. remains static for a long time).

2. The marker appearance, as seen from the cameras, was very dependent on the lighting conditions. Therefore, this required a fine adjustment of the parameters of the colour filters for each camera, which was tedious and inefficient. Even then, when we had different lighting conditions within the same scene, these parameters could only be adjusted to work properly under one of the different conditions (e.g. we could only detect the robot within dark or light areas, but not both within the same scene). As a result, it was common that the camera would lose the robot, which would end up in the robot not being able to navigate properly.

Consequently, we decided to use markers that emit light (active markers) in order to make the robot detection robust against changes in lighting conditions, texture, colour, etc.

3.1.3 Robot detection and tracking with active markers

Provided the results obtained with the two methods mentioned before, we have decided to design a robot detection and tracking system based on light emitting markers of different colours that we mount in our robots (active markers). These markers can form different colour patterns that are easy to recognise from the images. We preferred this solution for several reasons:

1. When tracking rigid objects, using artificial markers tends to provide higher accuracy rates than the use of natural features [40].

2. According to our experience, it is not possible to extract reliable features at the distances that we want our system to work (up to 20 meters), neither considering the robot as an object that must be identified nor using passive markers.

3. We require our system to work in various illumination conditions, which can range from over illuminated spots to semi-dark places. In our experiments, we observed that neither the robot natural features nor passive markers were robust enough to cope with this (similar problems were reported by [41, 42]).

4. The use of markers of different colours enables our cameras to differ among multiple robots, and thus increase the scalability of the system.

5. Using markers relieves us of having to train classification algorithms with images of our robots, which can be a unmanageable if we want to have decent detection rates in different environments, illumination conditions, etc.

6. Overall, the use of markers simplifies the robot detection problem. Therefore, we can design simpler algorithms that can work in real-time.

We weighed up the use of different active marker mechanisms. For instance, Cassinis et al. [41] built an active marker with a circular grid of high intensity red LEDs pointing down towards a mirror that reflects the light upwards to the cameras. The LEDs pulse at a given frequency that the cameras recognize to detect the robot position. The mirror's orientation, controlled by the robot, provides the robot's orientation. Similarly, Kim et al. [43] proposed a LED cube whose sides pulsed at different frequencies, which allowed them to measure the marker's orientation. Although both approaches are robust (especially [41]), we believe that they add unnecessary complexity for our purposes. We also considered the use of IR LEDs [23], but their performance is very bad under natural illumination conditions.

In our case, we decided to build an active marker with two red and two blue LED strips, like the one shown in Fig. 2.3. Our cameras must detect this colour pattern, and measure its position and orientation, in order to retrieve the position and orientation of the robot. This task is carried out by our "detection and tracking" algorithm, described in the next section.

3.2 Detection and tracking algorithm with active markers

Our "detection and tracking" algorithm is shown in Fig. 3.2. First of all, the algorithm performs a colour filtering to search for the colour LEDs in the camera images. Since we are using active markers, the colour filtering is much robust to the illumination conditions than when using passive ones. Then, the algorithm seeks the robot's marker pattern in the colour filtered image, and calculates its position and orientation. We have divided this algorithm in four main stages: (1) blob detection and tracking, (2) robot's marker recognition, (3) robot detection, and (4) robot's pose estimation. In the following sections, we describe each of these stages in detail.

We would like to point out that our algorithm allows us to use more than one robot simultaneously: it is sufficient that each robot carries a marker with a different colour pattern (see the experiment in Sec. 3.2.6). However, for the sake of clarity, we will describe the algorithm

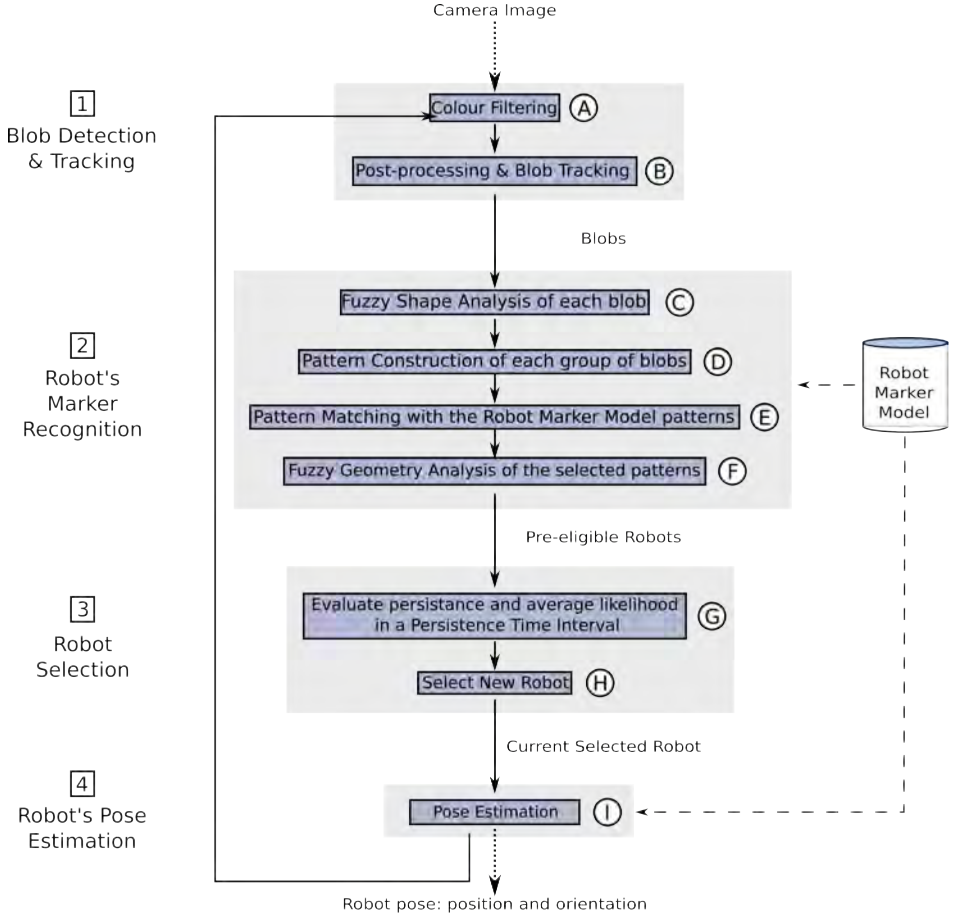
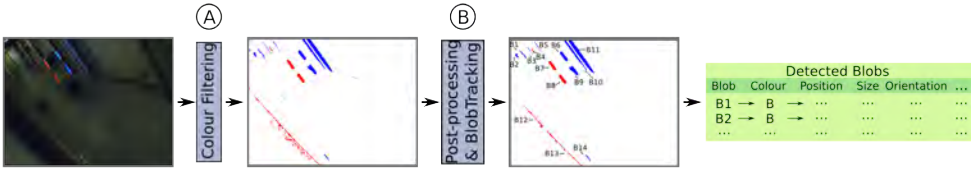

Camera Image
1
Blob Detection & Tracking
Colour Filtering
A
Post-processing & Blob Tracking
B
Blobs
2
Robot's Marker Recognition
Fuzzy Shape Analysis of each blob
C
Pattern Construction of each group of blobs
D
Pattern Matching with the Robot Marker Model patterns
E
Fuzzy Geometry Analysis of the selected patterns
F
Robot Marker Model
Pre-eligible Robots
3
Robot Selection
Evaluate persistance and average likelihood in a Persistence Time Interval
G
Select New Robot
H
Current Selected Robot
4
Robot's Pose Estimation
Pose Estimation
I
Robot pose: position and orientation

Figure 3.2: Robot detection and tracking algorithm. Continuous arrows indicate flow of execution. Dashed arrows indicate influence relationships. Dotted arrows indicate data input/output.

with the example of the detection and tracking of a single robot that carries the marker shown in Fig. 2.3.

3.2.1 Stage 1: blob detection and tracking

Figure 3.3 shows the blob detection and tracking stage of our algorithm. In our context, a blob is a set of connected pixels of the same colour. The first step of this stage (step A, Fig. 3.3) filters out the pixels whose colour does not match up with any of the colours of the LED



Detected Blobs

Blob		Colour		Position	Size	Orientation	...
B1	→	B	→	...	...	...	...
B2	→	B	→	...	...	...	...
...		...		...	...	...	...

Figure 3.3: Stage 1: Blob detection and tracking.

strips (red and blue in this case, as in Fig. 2.3). Two kinds of pixels, and therefore blobs, will pass this filtering process: the robot blobs, and spurious blobs (pixels that match the colours of the LED strips, but are due to artificial lights, reflections of the light coming out from the robot active markers, etc.).

After this filtering, we assign a Kalman Filter to each blob (step B, Fig. 3.3). Thus, we can assign an identity to each blob and track it over consecutive frames (B1, B2, B3, etc., in Fig. 3.3). Basically, each filter will contain information about the position and velocity (movement) of a blob on the image. Therefore, we can "predict" the expected position and velocity of each blob when a new frame comes. With this information, each blob detected in the new frame can be associated with the closest "predicted" blob. This process is repeated for each frame, to keep track of the blobs on the image.

3.2.2 Stage 2: robot's marker recognition

In this stage, represented in Fig. 3.4, we identify the groups of blobs that may correspond to the robot's marker LED strips. For example, in Fig. 3.3, these blobs would be B6, B7, B8, and B9. To do this, we carry out the three stage process described in the next subsections, aimed to calculate the degree of similarity (likelihood) between each group of detected blobs and the robot marker.

Shape analysis of each blob

First of all, we evaluate whether or not the aspect ratio (height-width ratio) of each blob is approximately equal to the aspect ratio of the LED strips of the marker (this value is six in our case). From this evaluation, we assign a likelihood to each individual blob, and only those with a non-zero likelihood will be considered in the next step (*eligible blobs*, step C at Fig. 3.4).

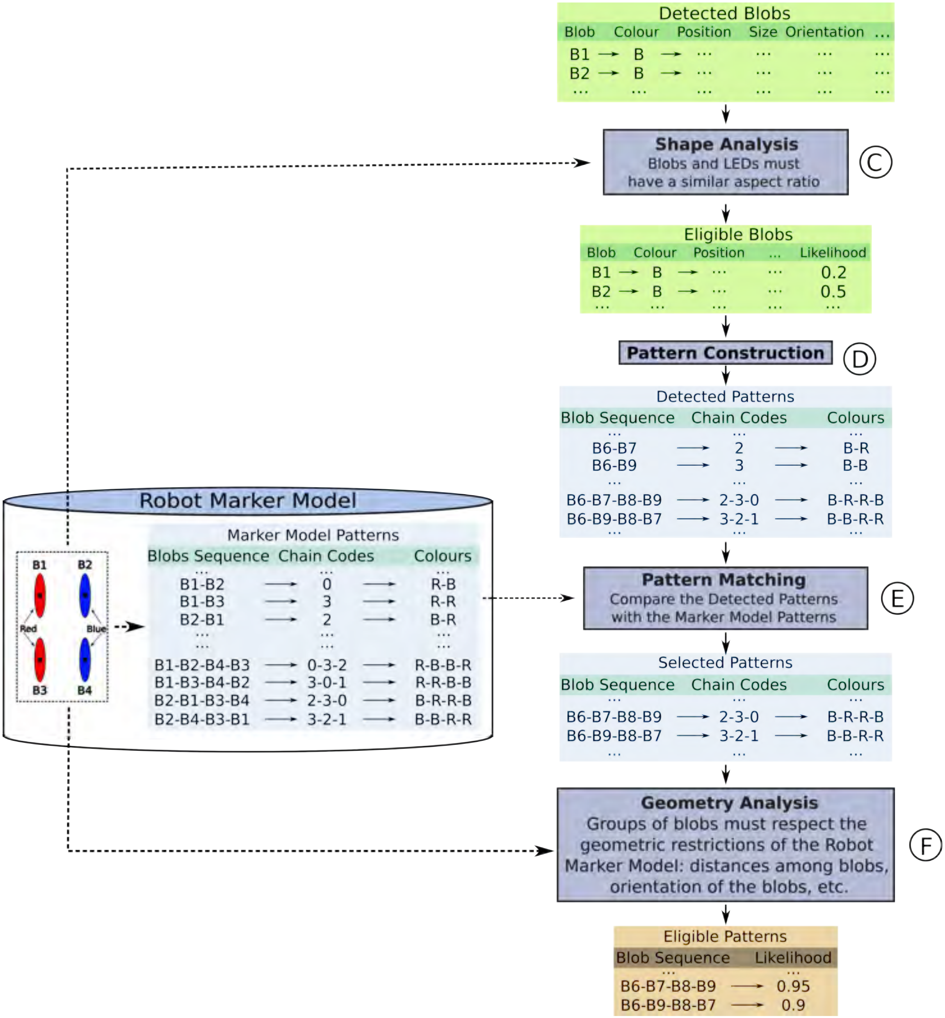

Detected Blobs
Blob Colour Position Size Orientation ...
B1 → B → ... ...
B2 → B → ... ...
Shape Analysis
Blobs and LEDs must have a similar aspect ratio
C
Eligible Blobs
Blob Colour Position ... Likelihood
B1 → B → 0.2
B2 → B → 0.5
Pattern Construction
D
Detected Patterns
Blob Sequence Chain Codes Colours
B6-B7 → 2 → B-R
B6-B9 → 3 → B-B
B6-B7-B8-B9 → 2-3-0 → B-R-R-B
B6-B9-B8-B7 → 3-2-1 → B-B-R-R
Robot Marker Model
B1
B2
Red
Blue
B3
B4
Marker Model Patterns
Blobs Sequence Chain Codes Colours
B1-B2 → 0 → R-B
B1-B3 → 3 → R-R
B2-B1 → 2 → B-R
B1-B2-B4-B3 → 0-3-2 → R-B-B-R
B1-B3-B4-B2 → 3-0-1 → R-R-B-B
B2-B1-B3-B4 → 2-3-0 → B-R-R-B
B2-B4-B3-B1 → 3-2-1 → B-B-R-R
Pattern Matching
Compare the Detected Patterns with the Marker Model Patterns
E
Selected Patterns
Blob Sequence Chain Codes Colours
B6-B7-B8-B9 → 2-3-0 → B-R-R-B
B6-B9-B8-B7 → 3-2-1 → B-B-R-R
Geometry Analysis
Groups of blobs must respect the geometric restrictions of the Robot Marker Model: distances among blobs, orientation of the blobs, etc.
F
Eligible Patterns
Blob Sequence Likelihood
B6-B7-B8-B9 → 0.95
B6-B9-B8-B7 → 0.9

Figure 3.4: Stage 2: Robot's marker recognition.

Pattern construction and matching

LEDs do not have univocal features, thus their blobs cannot be identified independently. Nevertheless, the group of LEDs of the marker forms a known colour pattern, called *marker model patterns*. To detect this pattern in the images, we construct groups of *eligible blobs* and calculate their respective colour patterns (step D, Fig. 3.4). These patterns will be called *detected* *patterns*. Then, we match the *detected patterns* against the *marker model patterns* (step E, Fig. 3.4).

We have chosen a pattern representation that captures: (a) the blobs that form the pattern, (b) their colours, and (c) their geometrical disposition. The identification of the blobs on each pattern is straightforward, because we have already identified each blob in the blob detection and tracking step. The same applies to their colours: the colour filtering step (step 1, Fig. 3.2) has already separated the blobs in red and blue ones. Finally, to codify the geometrical disposition of the blobs of the group, we adapt the concept of chain code [44], well known in the Computer Vision community: if, like in Fig. 3.5-(b), each blob is represented as a point, the geometry of the group of blobs can be represented by the set of sequential displacements that connect an initial blob (point) with the remaining ones. Given the shape of our marker, we allow displacements in the four directions sketched in Fig. 3.5-(a): right, up, left and down, which can be codified as 0, 1, 2, and 3, respectively. Note that we calculate these displacements with respect to the blob coordinates, not to the image coordinates (Fig. 3.5-(a)): this makes our chain codes invariant to rotations lower than ±90°. Clearly, the points of Fig. 3.5-(b) accept several representations, depending on the election of the initial blob and on the rotation sense of the displacements (clockwise or counter-clockwise): 0-3-2, 3-0-1, 1-0-3, 0-1-2, *etc*.

At this point, we are able to construct the pattern of each group of blobs, based on the properties previously described. The *marker model patterns* (Fig. 3.4) will be formed by all the possible combinations of 2, 3, and 4 blobs of the model (B1-B2, B1-B2-B3, B1-B2-

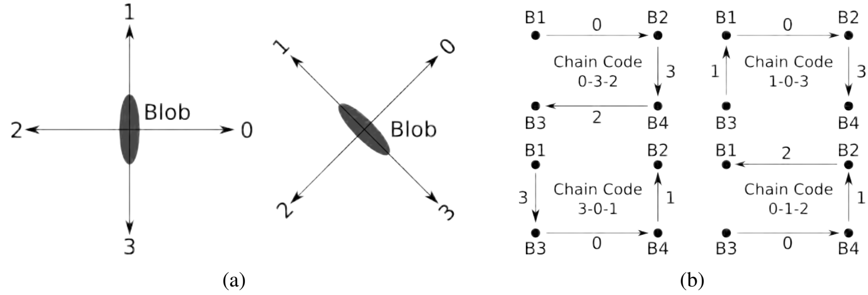


Figure 3.5: Chain code representation.(a) Displacement codes of the chain code. We calculate the displacements with respect to the blob coordinates: invariance on rotations lower than ±90°. (b) Examples of chain codes of a set of four points. Not all possible chain codes are included.

B3-B4, etc.). First of all, we calculate the chain code of each sequence. For example, the sequence B1-B2-B4-B3 would be associated with the chain code 0-3-2. Then, we construct the sequence of colours of the blob sequence. In the same example, since B1 and B3 are red, and B2 and B4 are blue, the colours for B1-B2-B4-B3 will be R-B-B-R. This codification may seem redundant, but we are just codifying all the possible colours and geometrical dispositions of the blobs, in the absence of a mechanism to identify them univocally.

We follow a similar process to construct the *detected patterns* (step D, Fig. 3.4). Then, we match the *detected patterns* with the *marker model patterns* (step E, Fig. 3.4). This results in a list of *selected patterns*, one of which might be the true pattern of the robot.

Geometry analysis of each pattern

Finally, our algorithm measures the geometrical likelihood of the *selected patterns* (step F, Fig. 3.4). This likelihood represents the degree of compliance of each *selected pattern* with the geometrical restrictions of the robot marker model (e.g., the distances among the blobs of the patterns must be similar to the distances among the LED strips of the marker). We compute the geometrical likelihood of each *selected pattern* as the product of the likelihoods of its pairs of blobs: for example, the likelihood of the group of blobs B1-B3-B4-B2 would be the product of the likelihoods B1-B3, B3-B4, and B4-B2. Then, we multiply the obtained likelihood with the likelihood of each individual blob (from the *eligible blobs* list, Fig. 3.4). To calculate the likelihood of each pair of blobs, we measure the degree of compliance with the following criteria:

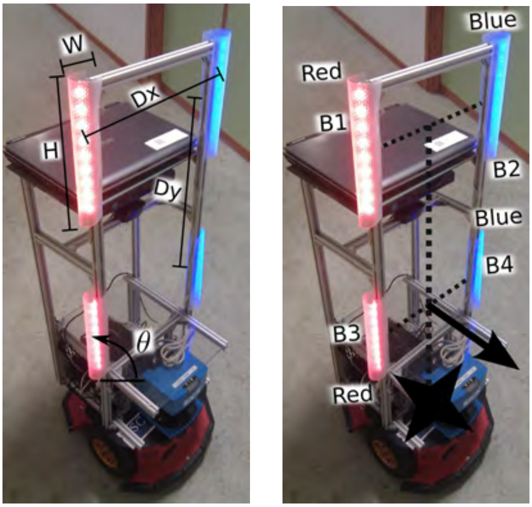
1. The blobs must be similar in height, width, and orientation (represented as H, W and θ in Fig. 3.6-(a), respectively).

2. The blobs must respect certain distances among them.

   a) If the blobs are collinear (like blobs B1 and B3 in Fig. 3.6-(a)), the distance in x (D_x in Fig. 3.6-(a)) must be approximately zero, and the distance in y (D_y in Fig. 3.6-(a)) approximately twice the height of one of the blobs.

   b) If the blobs are not collinear (like blobs B1 and B2 in Fig. 3.6-(a)), D_y must be approximately zero, and D_x less or approximately equal to the height of one of the blobs.

Like in the case of the *chain codes* (Fig. 3.5), all the criteria are calculated with respect to the coordinates of the blob, not with respect to the coordinates of the image (except for the blob's orientation). Hence, this likelihood measurement will be invariant to rotations.



(a) (b)

Figure 3.6: Geometry of active markers. (a) Geometric properties taken into account in the pattern matching process: Blob width W, height H, and angle with respect to the horizontal axis θ, and horizontal (D_x) and vertical (D_y) distance among blobs. (b) Robot position and orientation estimation.

3.2.3 Stage 3: robot detection

So far, our algorithm is able to determine a list of blob groups that might correspond to the marker of the robot in the scene, and the degree of similarity of each group with the marker (likelihood). This list will be called the list of *eligible patterns* (Fig. 3.4). First of all, to decide which one of these patterns corresponds to the robot that carries the marker, our algorithm discards those that do not persist over a minimum time interval (*persistence time interval*). Next, the algorithm calculates the average likelihood of the remaining blob groups over the mentioned interval (step 6, Fig. 3.2).

The *persistence time interval* was introduced to increase the robustness of the algorithm against spurious noises: reflections of the marker or ambient lights, objects moving around, spurious noise that passes the colour filter, etc. While the robot blob groups are usually stable (high average likelihood rates over long time intervals), noise blobs are rather erratic in persistence, shape, and position (possible instantaneous likelihood peaks, but low average likelihood rates in general). On the other hand, this interval softens the impact of momentary drops on the likelihood of the robot blob groups.

At this point, the algorithm is able to decide whether there is a robot present in the camera images (step H, Fig. 3.2). The decision process varies depending on whether the algorithm has previously detected a robot or not. In case that the robot was detected in the previous iteration, the algorithm checks if its previously detected blob group exists in the current list of *eligible patterns*. If that is the case, the algorithm selects the same blob group as in the previous iteration, provided that its likelihood is higher than zero. It may also occur that some of the blobs of this group do not exist anymore. In this case, the algorithm selects the *eligible pattern* with the highest number of blobs in common with the group selected in the previous iteration. In following iterations, the algorithm will seek to extend this group with new blobs, to recover from the loss of the missing blobs. This ensures the stability of the robot tracking and its robustness against temporary occlusions (e.g. people walking by the robot). On the other hand, if the blob group detected in previous iterations disappears completely, or if the robot was not detected in the previous iteration, the algorithm simply selects the *eligible pattern* with the highest number of blobs. In case of a tie among two or more groups, the algorithm selects the one with the highest average likelihood over the *persistence time interval*.

3.2.4 Stage 4: pose estimation

The last stage of the algorithm (step I, Fig. 3.2) calculates the robot pose in the image (position and orientation). The robot position is calculated by projecting the centre of masses of its blobs onto the ground plane (star in Fig. 3.6-(b)). We know the real dimensions of the LED lights and their height with respect to the ground, so the projection is straightforward. With regard to the robot orientation, the algorithm first decides whether the robot is giving its front or its back to the camera, considering the chain code and the colours of its blobs. Then, the orientation direction, represented as an arrow in Fig. 3.6-(b), is calculated from the lines connecting parallel blobs. Although this does not give a high precision, it is more than enough for our purposes.

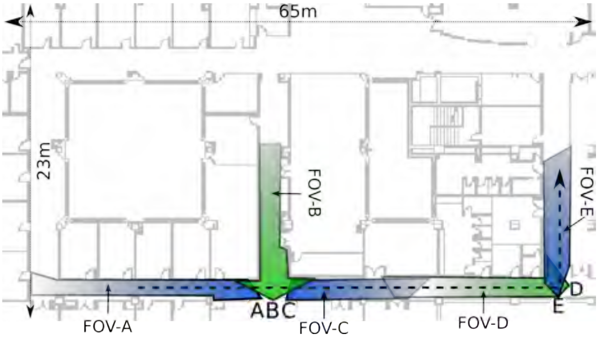


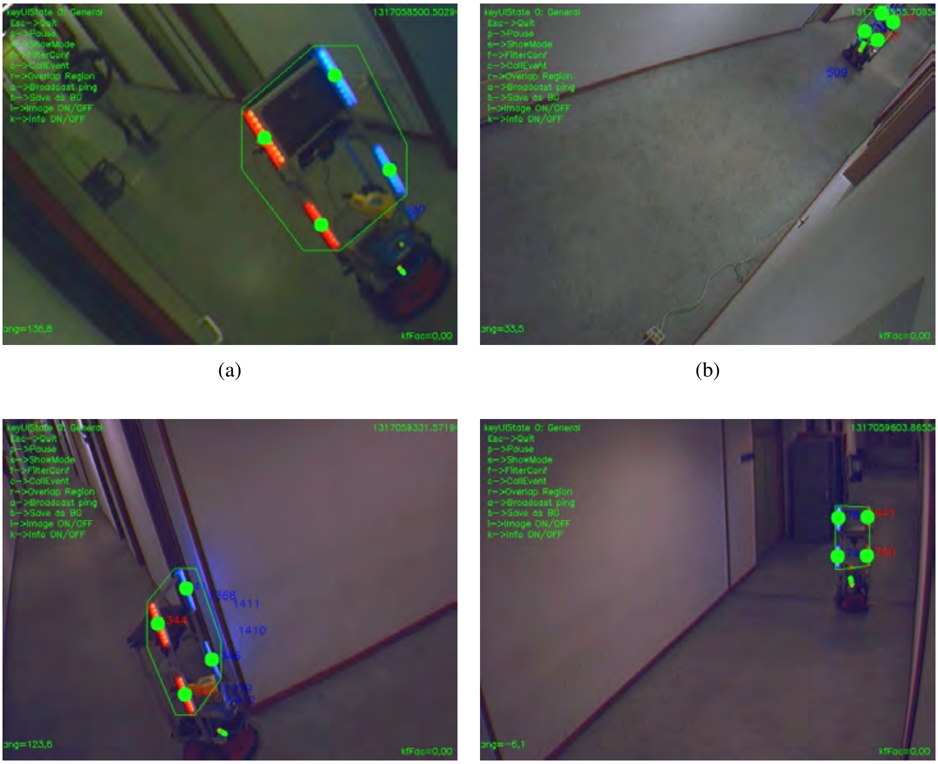
Figure 3.7: Experimental setup of the robot detection and tracking experiment with active markers. We have deployed 5 cameras in the environment (A, B, C, D, E), and we have moved the robot through the dashed trajectory while the cameras were recording.

3.2.5 Experimental results: robot detection and tracking

We have tested our algorithm at the Department of Electronics and Computer Science, at the University of Santiago de Compostela, Spain (Fig. 3.7). The robot used was a Pioneer P3DX equipped with a SICK-LMS200 laser. On the other hand, each camera-agent used either a Unibrain Fire-i camera or a PointGrey Chameleon CMLN-13S2C with an omnidirectional lens. Both models worked at a frame rate of 15 fps and at a resolution of 640 × 480 pixels. The processing units were Intel Core 2 Duo CPUs (P8600@2.4 GHz, T5600@1.83 GHz, or T5600 Mobile P8700@2.53 GHz) with 4 GB RAM. To build the camera software, we used the OpenCV 2.2 library [45].

We have deployed 5 cameras in the experimental environment as shown in Fig. 3.7. Then, we have moved the robot at a maximum speed of 0.5 m/s through a trajectory that crossed the FOVs of the 5 cameras. In this experiment, camera B recorded the robot at a distance up to 4 m (approximately), E up to 15 m, A and C up to 30 m, and D up to 50 m (see Fig. 3.8). With the video recorded by each camera, we have constructed a dataset of 12564 frames. We have executed our algorithm on this dataset with different parameter combinations to test their impact on the performance of our algorithm. Particularly, we have considered variations on:

- Colour Filters. To detect the marker of the robot, we need to detect its colours first (see Sec. 3.2.1). We define each colour that has to be detected as an interval in the HSV colour space. To detect the colours of the marker, we filter out every pixel whose colour is out of the desired intervals. We observed that the illumination conditions affect the



(a) (b)

(c) (d)

Figure 3.8: Example of camera captures from the experiment in Fig. 3.7: (a) camera A, (b) camera C, (c) camera D, (d) camera E. The detected marker is rounded by a convex polygon, and each of its blobs is tagged with a circle.

S and V colour components very strongly, thus their corresponding intervals have to be adjusted for each environment. A careful adjustment is prohibitive if we want to deploy our robots quickly, but a coarse one may let undesired colours pass the filtering stage. For these reasons, we assessed the performance of the algorithm with two colour filters: (1) one with carefully adjusted intervals, and (2) another with coarsely adjusted intervals.

- Persistence Time Interval. This refers to the number of consecutive frames in which the robot marker has to be detected to consider that there is a robot in the FOV of the camera (see Sec. 3.2.3). We analysed two scenarios to measure the impact of this interval: (1)

interval length equal to zero (no interval), (2) interval length equal to 10 frames (0.66 s).

For each frame, our algorithm tells us whether it detected the robot, and if that is the case, its position. To analyse the results, we classified each frame into one of the following five classes (confusion matrix in Table 3.1):

- *True Positive* (TP): if there was a robot in the scene and the camera detected it. The higher the better.
- *False Positive—Robot in Scene* (FP_{RS}): if there was a robot in the scene and the camera detected another element. The lower the better.
- *False Negative* (FN): if there was a robot in the scene and the camera did not detect it. The lower the better.
- *False Positive—No Robot in Scene* (FP_{NRS}): if there was no robot in the scene but the camera detected one. The lower the better.
- *True Negative* (TN): if there was no robot in the scene and the camera did not detect any robot. The higher the better.

We divided the False Positives in two different groups to have a separate measure of the noise caused by the markers' lights reflections (which commonly causes the FP_{RS} rate) from the rest of the noise (which commonly causes the FP_{NRS} rate).

			Is there a robot?	
			Yes	No
Detects	Robot	The detection corresponds with the real robot	TP	FP_{NRS}
		The detection does not correspond with the real robot	FP_{RS}	
	Nothing		FN	TN

Table 3.1: Confusion matrix used to classify the robot detection and tracking algorithm results.

To analyse the results, we have considered two different levels of requirement regarding the maximum distance at which our algorithm must detect the robot: (1) arbitrarily large distances, and (2) distances below 20 m. We are mostly interested in this last case, because at higher distances, the marker is too small to be detected accurately.

Table 3.2 shows the results that we have obtained in this analysis. From these results, we can draw the following conclusions. First of all, Table 3.2 shows that the average FP rates are low, and the average TN rates are high. This ensures that when the algorithm detects a robot, this detection is usually correct. This is very adequate for our needs, because an incorrect robot detection may result in an incorrect information sent to the robot, and thus in a wrong robot navigation. In the same column, we see that the average TP and FN rates are not ideal, but acceptable for our purposes: if a camera fails to detect a robot occasionally, our system will still be able to support the robot navigation, because the acquisition rate of each camera is high with respect to the speed of the robot. In fact, it is enough if the cameras are able to detect and send information to the robot every few seconds.

On the other hand, we observe that the results of the algorithm are the best when we use carefully adjusted colour filters. However, to tune them in every camera at deployment time would be prohibitive. Fortunately, we also observe that we obtain similar results if we use a persistence time interval of 10 frames, even with coarsely adjusted colour filters. Specifically, this improves the FP and TN rates, which are the most important for us. Therefore, we can adjust the colours filters coarsely, which can be done very quickly, and still achieve a great robustness on the detection and tracking of the robot. Finally, as it was expected, the overall results of the algorithm improve when we do not require the algorithm to detect the robot at distances greater than 20 m. The most typical scenario in real deployments is highlighted in boldface.

We have also used the following metrics to measure the classifier performance [46]:

- Positive Predictive Value or Precision (PPV): proportion of frames where the robot is detected and the detection is correct (takes values in the interval [0, 1], the higher the better).

$$PPV = \frac{TP}{TP + FP_{RS} + FP_{NRS}} \tag{3.1}$$

- Negative Predictive Value (NPV): proportion of frames where the robot is not detected and there is no robot (takes values in the interval [0, 1], the higher the better).

	MaxDist = ∞				MaxDist = 20m				
	ColFilt = Coarse		ColFilt = Careful		ColFilt = Coarse		ColFilt = Careful		
	Int = 0	Int = 10	Int = 0	Int = 10	Int = 0	Int = 10	Int = 0	Int = 10	Avg
TP	0.530	0.451	0.609	0.516	0.760	**0.614**	0.872	0.764	0.639
TN	0.960	0.999	1.000	1.000	0.946	**0.999**	1.000	1.000	0.988
FP_{RS}	0.240	0.007	0.018	0.006	0.132	**0.003**	0.028	0.009	0.055
FP_{NRS}	0.042	0.001	0.000	0.000	0.054	**0.001**	0.000	0.000	0.012
FN	0.234	0.543	0.373	0.479	0.108	**0.383**	0.101	0.228	0.306
PPV	0.627	0.983	0.971	0.989	0.725	**0.992**	0.969	0.989	0.906
NPV	0.881	0.769	0.829	0.790	0.962	**0.893**	0.966	0.926	0.877
TPR	0.693	0.454	0.620	0.519	0.876	**0.616**	0.897	0.770	0.680
MCC	0.485	0.632	0.700	0.666	0.713	**0.717**	0.879	0.804	0.699

Table 3.2: Classification results of the detection and tracking algorithm with active markers. We have analysed eight different scenarios by varying: MaxDist (maximum distance at which we require the cameras to detect the robot), ColFilt (whether the colour filters have been coarsely or carefully adjusted), and Int (time persistence interval). The typical configuration that we use on robot control experiments (Sec. 4.1.5) is highlighted in boldface. The acronyms are: *TP* (True Positives), *TN* (True Negatives), *FP-RS* (False Positive—Robot in Scene), *FP-NRS* (False Positive—Not Robot in Scene),*FN* (False Negatives), *PPV*(Positive Predictive Value or Precision), Negative Predictive Value (NPV), *TPR* (True Positive Rate or Recall), *MCC* (Matthews Correlation Coefficient).

$$NPV = \frac{TN}{TN+FN} \tag{3.2}$$

- True Positive Rate or Sensitivity/Recall (TPR): proportion of frames where there is a robot and is correctly detected (takes values in the interval [0, 1], the higher the better).

$$TPR = \frac{TP}{TP+FN} \tag{3.3}$$

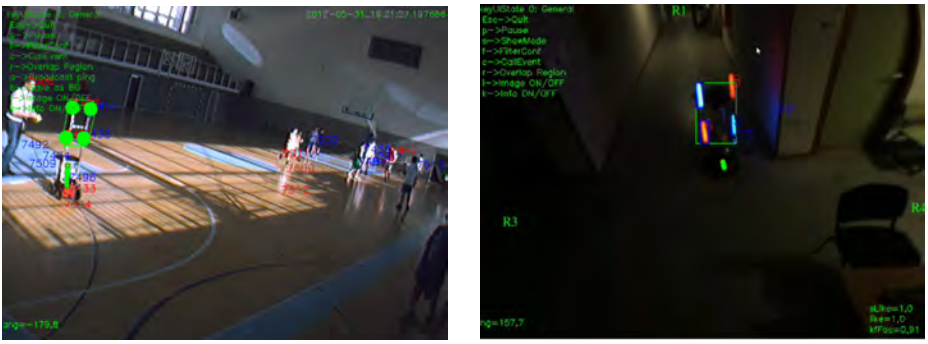
- Matthews Correlation Coefficient (MCC): measures the quality of the classifications (takes values in the interval [-1, 1], with -1 representing an inverse prediction, 0 an average random prediction and +1 a perfect prediction).

$$MCC = \frac{TP \cdot TN - (FP_{RS} + FP_{NRS}) \cdot FN}{(TP + FP_{RS} + FP_{NRS})(TP+FN)(TN + FP_{RS} + FP_{NRS})(TN+FN)} \tag{3.4}$$

For the reasons already mentioned, the robot control problem requires to maximize the PPV rate, and maintain an acceptable NPV rate and TPR value. The results in Table 3.2 confirm that our algorithm is very well suited to this problem. Moreover, the MCC values, which measure the quality of the classifications, confirm the good performance of the proposed classifier.

Summarizing, we conclude that our algorithm is very robust to be used in our system. We have also observed that the use of the persistence time interval ensures the robustness of the algorithm even with coarsely adjusted colour filters. This guarantees that we will be able to deploy our system in different environments in a fast and easy manner.

In addition, we think that it is important to point out that in the real operation of the system we are able to use always the same colour filters (highlighted in boldface in Table 3.2), for all the cameras, and regardless of the illumination conditions. We have tried our algorithm in different environments with satisfactory results: Fig. 3.9 contains two examples of challenging situations where our algorithm is able to perform correctly (a dark corridor during the night, and at a sports pavilion under direct intense sunlight).

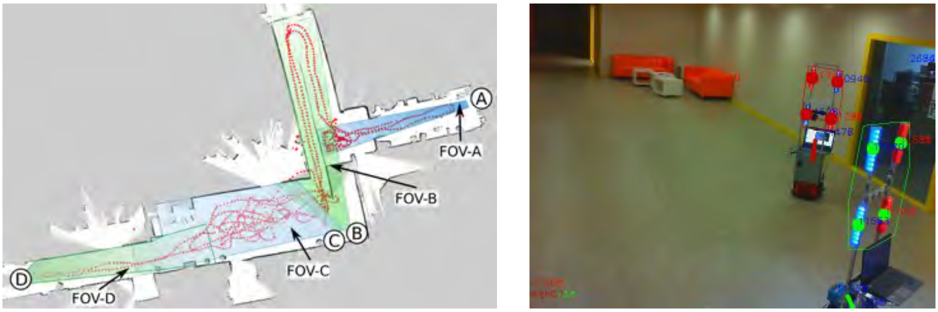


(a) (b)

Figure 3.9: Robot detection under different illumination conditions.

3.2.6 Experimental results: scalability with the number of robots

We have performed an experiment to test our robot detection and tracking algorithm with more than one robot. This experiment was performed at the CITIUS research building (Centro de Investigación en Tecnologías de la Información de la Universidad de Santiago de Compostela), in Santiago de Compostela (Spain). In the experiment, we have deployed four Microsoft HD-5000 cameras working at a frame rate of 15 fps and at a resolution of 640 × 480 pixels. The processing units of the cameras were Intel i7-2670QM CPU @ 2.20 GHz with 4 GB RAM. Then, we moved a Pioneer P3DX and a Pioneer 3AT robot through the environment (each robot carried a different active marker). Figure 3.10-(a) shows the disposition



(a) (b)

Figure 3.10: Deployment of our self-organised multi-agent network with two robots working in the environment. a) FOV of each camera, and trajectory of one of the robots. b) Snapshot of both robots taken from camera C.

of the cameras in the environment and the trajectory followed by one of the robots, and Fig. 3.10-(b) shows both robots seen from one of the cameras.

Table 3.3 shows the results of the experiment. On the one hand, the TP, TN, FP_{RS}, FP_{NRS}, and FN rates are similar to those obtained with one robot (Sec. 3.2.5). On the other hand, the misclassification rate ($MISCL$) indicates that our algorithm almost never confuses one robot with the other. This ensures that increasing the number of robots to be detected does not degrade the performance of the algorithm.

	TP	TN	FP_{RS}	FP_{NRS}	FN	$MISCL$
$Robot1$	0.781	0.995	0.031	0.005	0.18	0.004
$Robot2$	0.821	0.996	0.283	0.004	0.151	0.009

Table 3.3: Classification results of the detection and tracking algorithm with active markers with two robots. $MISCL$ (misclassification rate) is the percentage of frames where the algorithm confuses one robot with the other. The rest of the acronyms have been defined in Sec. 3.2.5.

Finally, Table 3.4 shows the processing time per image of our algorithm using a modern computer . We can extract three main conclusions. First, it takes around 16 ms to detect a robot on each image. This time is very low, taking into account that our cameras provide one image each 66 ms (acquisition rate of 15 fps). Second, the computational time does not increase when executing more than one camera concurrently, as expected of a multi-core processor. Finally, the processing time scales linearly with the number of robots to be detected, each robot increasing this time in 16 ms (e.g., 2 robots require 32 ms). We would like to point out that the current processing time is good enough for our requirements, but there is still a lot of room for improvement: code optimization, parallelization, use of GPU, etc. For example, right now each camera computes the detection of all the robots in the same CPU core, but it could instead compute the detection of each robot in a different core.

	NumRobots = 1			NumRobots = 2		
	NumCams = 1	NumCams = 2		NumCams = 1	NumCams = 2	
	Cam1	Cam1	Cam2	Cam1	Cam1	Cam2
$\mu(ms)$	15.66	16.60	16.31	32.57	32.63	32.57
$\sigma(ms)$	1.37	1.49	1.67	3.27	3.24	3.27

Table 3.4: Execution times of the robot detection and tracking algorithm with active markers. μ is the average processing time per image (in milliseconds), and σ is the standard deviation (in milliseconds). NumRobots refers to the number of different robots that the algorithm tries to detect. NumCams refers to the number of camera-agents executed on a single computer (concurrently).

3.3 Call event detection

In the previous section, we have explained how the cameras are able to detect and track the robot. In this section, we will explain how the cameras can detect situations that require the presence of the robots (call events). With both abilities, our cameras will be able to support the robot navigation within the environment towards where their presence is required.

The concept of call event is generic and can be accommodated to a wide range of robotic applications. In the context application of our work (tour-guide robot), we have considered two simple call events situations, but in other contexts other call events could be implemented. The call events that we will consider are: 1) the detection of a person (or group of persons) standing still in specific areas, and 2) the detection of a person waving at a camera. Our cameras will consider both as situations that require the presence of a robot. In the first case, we will assume that users standing still at the same place for long periods of time might be bored, lost or waiting for someone to offer them information. Therefore the robot should approach them, showing a proactive behaviour and therefore improving peoples' opinion on its role. In the second case, we will assume that users will wave at a camera when they want to receive help or information from the robot. In this case, the robot works on users' demand.

The work described in this section was developed in collaboration with Dr. Pablo Quintia and Ignacio Cabado, and presented as part of their PhD Tesis [47] and Final Degree Project

[48] (respectively). We would like to point out that it is not an objective of this thesis to contribute to the field of person detection and tracking. On the contrary, our aim was just to provide a proof-of-concept of a real application of call events.

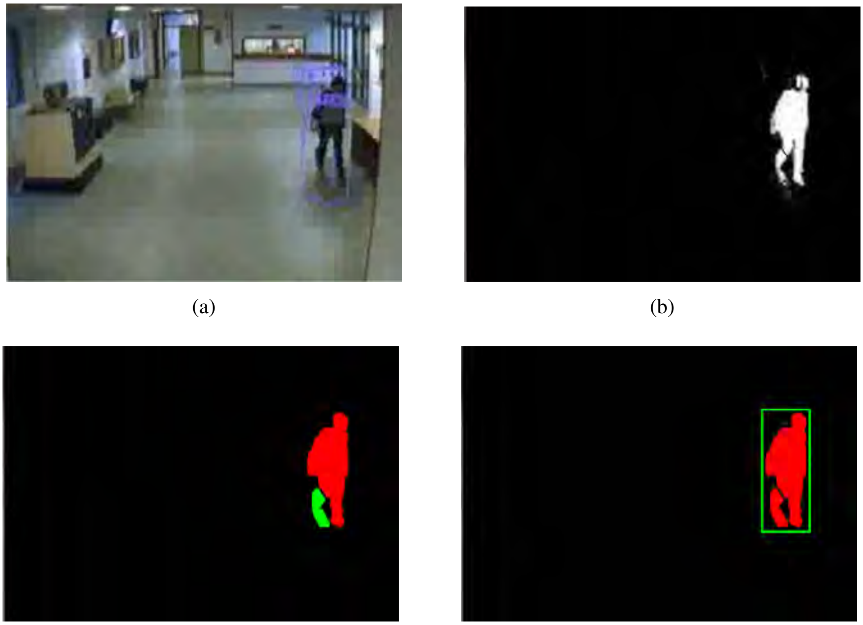
3.3.1 People or groups of people standing still at specific areas

In order to detect people standing still, each camera needs to detect and track the people moving within its FOV. We have relied in simple and fast techniques that can be computed in real-time, therefore we discarded object detection techniques such as those explained in Sec 3.1.1. Instead, we first apply a background subtraction algorithm [39] to the original image (Fig. 3.11-(a)) just consider the objects that do not belong to the background. This gives us a black and white image like the one in Fig. 3.11-(b), with the foreground pixels in white. These pixels can be connected to form blobs (sets of connected pixels of the same colour). Our goal is to decide which blobs are persons, and which blobs are not. First of all, we remove all the blobs that do not have a certain minimum area (typically noise blobs), because some may correspond to shadows, areas where an illumination change occurred, etc. Then, we group together all the blobs that are sufficiently close. This is necessary because individual objects are rarely captured by a single blob. For instance, in Fig. 3.11-(c) the camera might detect a separation within the leg of the person (e.g. due to a lighting reflection). After the joining process, we compute the width, height and aspect ratio (width-height ratio) of each group of blobs, and decide which ones correspond to a person (Fig. 3.11-(d)).

Afterwards, we must track each detected person over consecutive frames. Finally, we extract certain features of the movement of each person over time, such as a history of the positions that he occupied in the image (dots in Fig. 3.12). With this history of positions, we can decide whether or not a certain person has been within a certain area for a long period of time, which would trigger a call to the robots.

We can also detect groups of people standing in certain areas. In this case, the cameras will join together persons that are close to each other to form groups of people. Similarly, any person can be included in an existing group, and groups can even be joined together, following the same distance criteria. Obviously, the same rules apply when splitting groups: if a person or group of persons separate from the rest, the group may be split in two. Figure 3.13 contains several examples of this process.

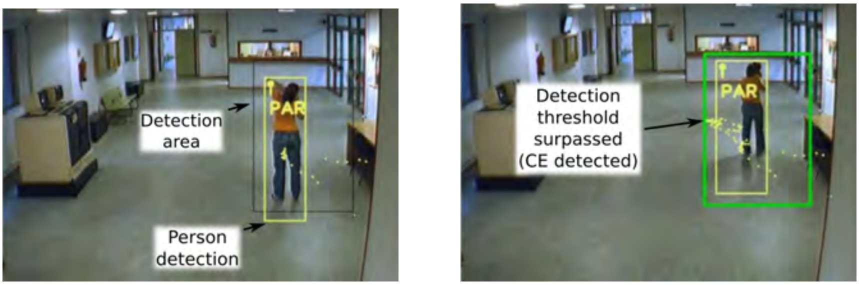
After this, the cameras must track the detected groups over consecutive frames, in a similar way as with the individual persons (based on the current and previous area and position of each



(a) (b)

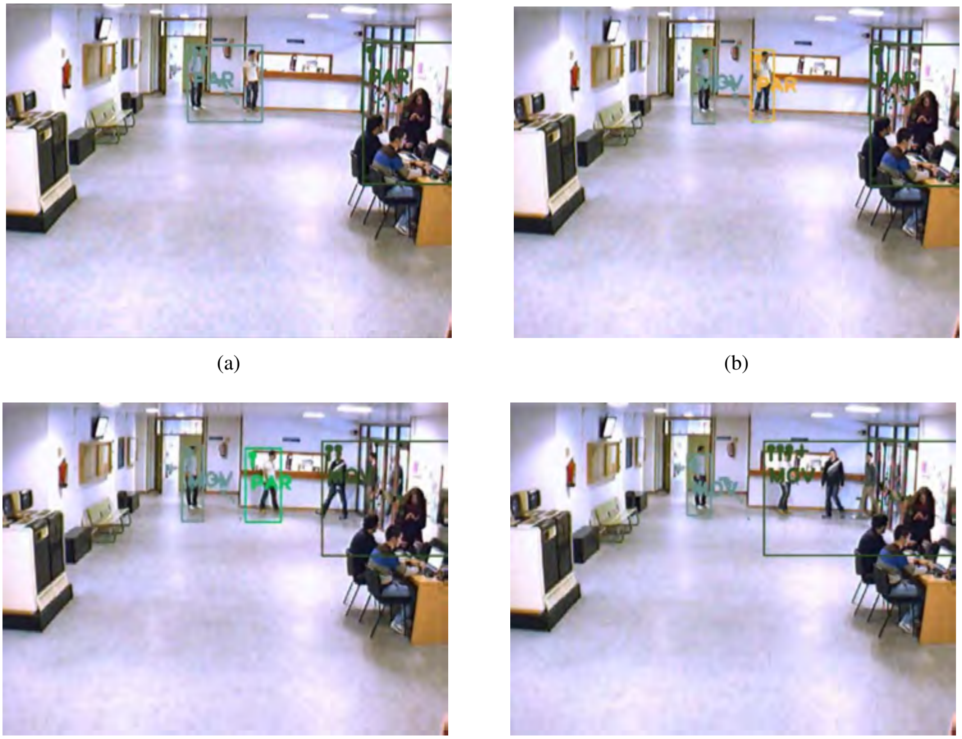
(c) (d)

Figure 3.11: Background subtraction example. a) Original image. b) Binary image, with the background pixels in black and the foreground pixels in white. c) Image where noisy blobs have been erased and two blobs corresponding to the same person that should be joined. d) Person detection after joining the blobs.



(a) (b)

Figure 3.12: Detection of a person standing in a certain area. a) before a certain time threshold, and b) after a certain time threshold (call event triggered).



(a)

(b)

(c)

(d)

Figure 3.13: Group split and join example. a) Two persons are close enough to be part of the same group. b) The separation between both persons grows, therefore the group splits in two. c) A new group of persons enters the scene. d) One person of the initial group is close enough to the new group to join it.

group). Then, the cameras can compute a history of the positions occupied by each group, and detect whether a group has been standing in certain areas for a long period of time. We would like to mention that we were also able to detect the number of persons in each group. This was very useful in order to assign a priority value whenever the cameras called the robot for assistance (larger groups would have larger priorities).

3.3.2 Person waving at a camera

In this case, the cameras look for individual persons, standing still, that are waving at them. First of all, the cameras detect and track each person within their FOV, just like we explained

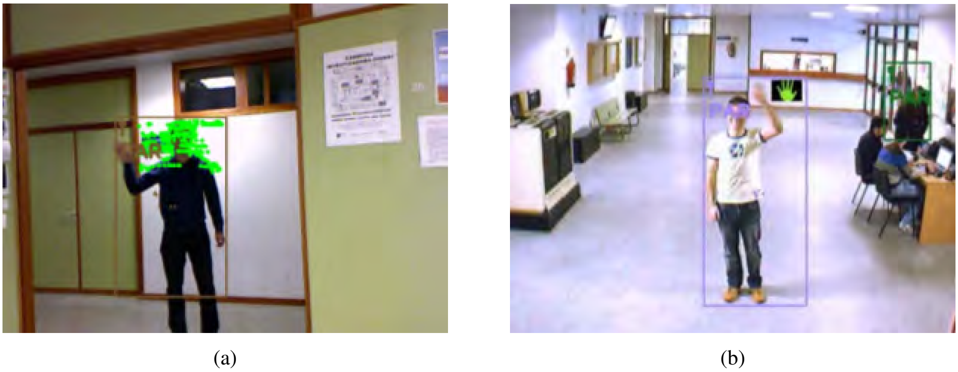

(a) (b)

Figure 3.14: Gesture detection. a) Extraction of characteristic points that will be tracked over time. b) Detection of a waving gesture.

in Sec. 3.3.1. Then, the camera tries to detect waving gestures that might be performed by any person that is not moving (Fig. 3.14-(a)). This is done by computing over several frames the optical flow within the upper half of each persons' bounding box, where the waving gesture is expected. The optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of the object or the camera.

To compute the optical flow, we first extract characteristic points from the upper half of each persons' bounding box using the Shi&Tomassi algorithm [49]. An example of extraction of this points can be seen in Fig. 3.14-(a). Then, these points are tracked over consecutive frames using the Lucas&Kanade algorithm [50]. Therefore, we can store the history of positions of each point, the history of displacements, and other features such as the average speed of each point or of all the points. Taking into account that waving gestures tend to have a certain amplitude, speed and time duration (within an interval), we can detect waving gestures using these features. Figure 3.14-(b) contains an example where one of the cameras detected a waving gesture from an user.

3.4 Discussion

In this section, we have described how our intelligent cameras are able to detect robots and call events. Although this is not a central objective of this thesis, we have achieved adequate detection mechanisms that ensure a correct operation of our system. First of all, we have described an algorithm for robot detection based on active markers that we put on our robots. This algorithm allows us to detect robots robustly and efficiently, and to distinguish among them. Then, we have suggested methods to detect: 1) people waving at the cameras, and 2) people entering certain predefined areas. Both events have been used as examples of call events to which our robots could attend, but we would like to remark that other events can be detected.

Chapter 4

Self-organised and distributed route planning

In Section 2, we have explained that in addition to detecting call events, the network of intelligent cameras had another objective: to support the robot's navigation. Depending on the amount of intelligence assigned to each agent, this support will be more or less relevant. For instance, the cameras could just help the robots to estimate their position in the environment (more intelligence on the robots), or they could direct the whole navigation process (more intelligence on the cameras). In this chapter, the cameras will handle most of the complexity of the navigation process. The intelligence will be fairly distributed among all the cameras ("collective intelligence"), so there will not be a "central" agent guiding the robot. Instead, the coordination among agents will be fully distributed, based on local interactions amongst independent agents and self-organization processes, similar to those usual in biology. This will provide us with a robust and redundant system, that scales well with the size of the environment and the number of cameras and robots. This system will be flexible to the changes in the environment, to the number of agents used, and to their disposition.

To support the robot navigation in this scenario, each camera needs to carry out four main tasks:

1. Discover who are its neighbouring cameras.
2. Detect situations that require the presence of the robots (call events).

3. Calculate routes of cameras through which the robot can navigate to reach these call events.

4. Support the robot navigation towards the next camera on the route (when the robot is within its FOV). This process ends when the robot arrives to the call event.

In this chapter, we will explain the software architecture and the algorithmic processes that allow our cameras and robots to fulfill this tasks. In Sec. 4.1, we will assume that neighbour cameras have at least some degree of overlap among their FOVs (e.g. cameras C4 and C2 in Fig. 2.4). Therefore, the robot can move among them without traversing non covered areas. Then, in Sec. 4.2 we will explore the possibility of having neighbour cameras whose FOVs do not overlap (e.g. cameras C4 and C5 in Fig. 2.4). This implies that the robot will need to traverse non covered areas, therefore it will require a higher degree of intelligence. Finally, we will discuss the results obtained (Sec. 4.4).

4.1 Cameras with overlapping FOVs

The overall software architecture that allows our system to accomplish all these tasks is represented in Fig. 4.1. Internally, each camera or robot-agent consists on a number of concurrent modules that that are able communicate and share data internally. Moreover, both types of agents can exchange messages over an IEEE 802.11 local wireless network via UDP (continuous arrows). We will represent these messages with a tag of the form *X2Y_TYPE*, where *X* is the sender, *Y* the receiver, and *TYPE* the type of the message. For example, *C2R_CALL* would be the message that a camera sends to a robot when calling it to attend an event.

We observe that the camera-agents are composed by four kinds of modules:

– *Vision Module*: responsible for the detection and tracking of robots and call events. Most of its functionality has already been described in Chapter 3.

– *Network Module*: responsible for network communications.

– *Neighborhood Module*: responsible for the detection of neighbourhood relationships with other cameras.

– *Call Modules*: responsible for the assignment of the call events detected to the appropriate robots.

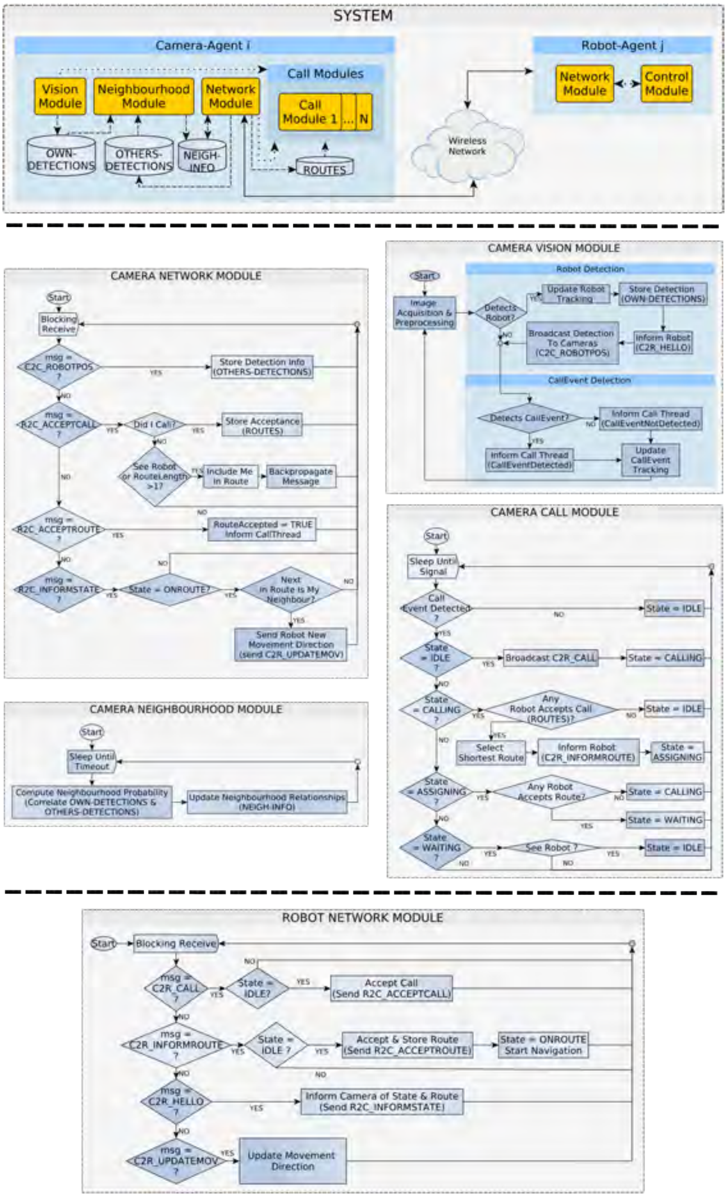


Figure 4.1: Software architecture of our system (cameras with overlapping FOVs).

On the other hand, the software that controls each robot runs just two concurrent modules:

- *Network Module*: responsible of network communications (Fig. 4.1).
- *Control Module*: implements the robot intelligence. In the case of “collective intelligence” (intelligence on the cameras), this module just implements the robot navigation algorithm, using naive controllers that require the support of the cameras. In other case, this module is responsible for the tasks of robot localisation and autonomous navigation.

Note that there might be other different modules depending on the tasks that the robot has to perform. For instance, in the case of a tour-guide robot, some of these models could be: person detection and tracking, person following behaviours, route reproduction, reproduction of visual and auditive information, etc.

In the following sections, we will review the main tasks that our camera-agents need to perform, in order to support the robot navigation towards call events: detection of their neighbour cameras, computation of routes of cameras through which the robot can navigate, and support of the robot movement from a camera to the next camera on the route.

4.1.1 Dynamic neighbourhood detection

The cameras can support the robot navigation by calculating sequences of cameras (routes) through which the robots can move. In order to do so, the cameras must know who are their neighbours. In our system, each camera can determine its neighbourhood relationships automatically. This is done by detecting parts of their FOV that overlap with the FOV of other cameras. These overlapping areas in the FOV will be called Neighbourhood Regions from now on. More concretely, each camera divides its FOV in squared regions, and calculates the Neighbourhood Probability of each region with all the remaining cameras. The Neighbourhood Probability of each region of camera i, with respect to each camera j ($i \neq j$), is the likelihood that this region overlaps with any part of the FOV of camera j.

These likelihoods depend on the simultaneous detection of the presence of the robot among cameras. Essentially, simultaneous detections among cameras increase their Neighbourhood Probability, and non simultaneous detections decrease it. During the “deployment” phase, we move a robot around the environment to force these simultaneous detections. When a camera detects a robot, it stores and broadcasts its position to all the other cameras (*C2C_ROBOTPOS* in Camera Vision Module, Fig. 4.1). Of course, this camera may receive similar messages from others (Camera Network Module, Fig. 4.1), when they detect a robot as well. Periodically, each camera checks whether the detection of the robot is taking place simultaneously with the detection of the same robot but from any other camera (Camera Neighbourhood Module, Fig. 4.1). When two cameras detect a robot simultaneously, they increase the Neighbourhood Probability of the region where the robot was detected. On the contrary, when this detection does not take place simultaneously, they decrease this probability. We compute all the probability updates using a Bayesian binary filter [51].

The process described before runs continuously so that the cameras can update their neighbourhood information at any time: they can establish new neighbourhood relationships (e.g. if we include a new camera), modify them (e.g. if we change the FOV of a camera), or eliminate existing ones (e.g. if we remove a camera). Figure 4.2 represents the evolution of the Neighbourhood Probabilities between two cameras, A and B. Initially (Fig. 4.2-(a)), there were no simultaneous robot detections among them, so the Neighbourhood Probability was zero for all the regions. After a few common robot detections (Fig. 4.2-(b)), both cameras recognised their common regions with probability close to one. Therefore, they established a neighbourhood link, and tagged those regions as Neighbourhood Regions.

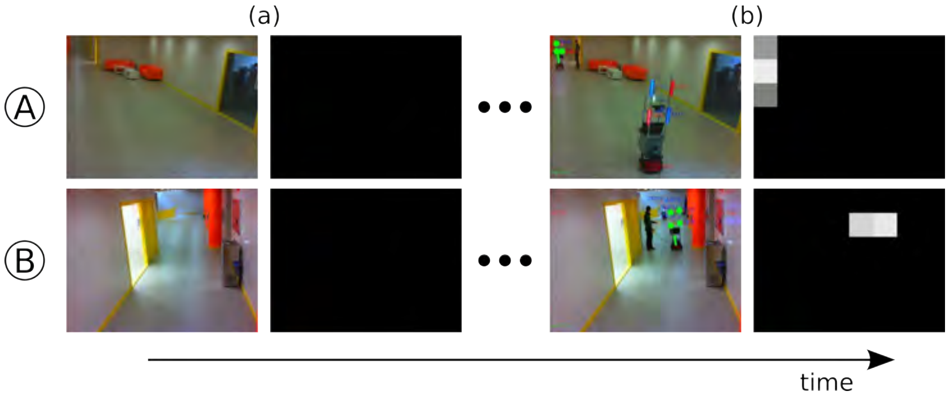


Figure 4.2: Evolution of the Neighbourhood Probability of two cameras A and B through time. Black represents a probability equal to zero, and white a probability equal to one. a) Initially, the cameras assume no neighbourhood relationships. b) Nevertheless, as the robot is moved around and the time elapses, the simultaneous detections allow the establishment of non-null neighbourhood values between both cameras.

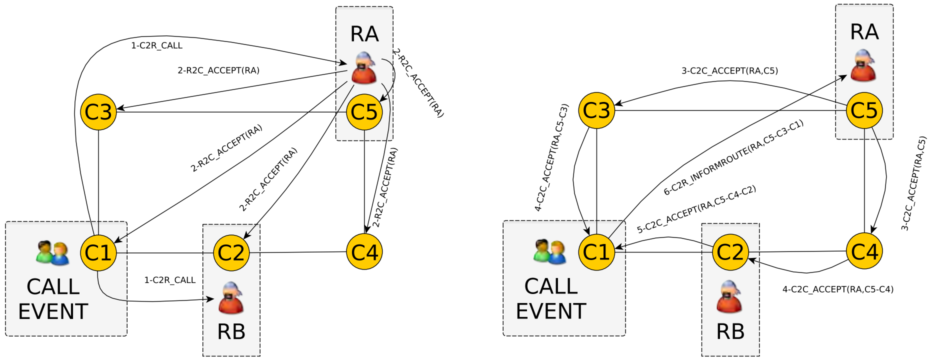
The neighbourhood relationships among the cameras could also be established based on the detection of persons in the environment. This will be explored in Sec. 4.3.

4.1.2 Distributed route planning

A route is an ordered list of cameras through which the robot can navigate. Basically, the robot will go from the FOV of one of the cameras to the FOV of the next camera on the route, without needing metric maps of the environment. These routes will be generated as a result of local interactions amongst the cameras, without the intervention of any central agent. We will explain the route planning process through the example shown in Fig. 4.3.

1. A camera detects a call event (CE) that requires the presence of a robot (Vision Module, Fig. 4.1). Then, a Call Module of this camera broadcasts a call message to all the robots (*C2R_CALL* broadcast, Fig. 4.1). In Fig. 4.3-(a) camera C1 detects a call event, and broadcasts a call to all the robots.

2. Each robot receives this call (Robot Network Module, Fig. 4.1). Those that are free (IDLE state) accept it, and send an acceptance to all the cameras (*C2R_ACCEPTCALL*, Fig. 4.1). In Fig. 4.3-(a), we assume that robot RA is free and broadcasts a call acceptance.



(a) (b)

Figure 4.3: Distributed route planning procedure with overlapping FOVs. Cameras (C1-C5) have established their neighbourhood relationships, forming a network altogether. a) Call event detection and call for robots. b) Back-propagation process for route formation. See Section 4.1.2 for a detailed explanation.

3. The cameras that receive this acceptance (Camera Network Module, Fig 4.1) and are seeing the robot that issued it, start a back-propagation process to create a route. When a camera receives this message, it appends its identity on it and forwards it to its neighbours (except to those through which the message has already passed). For example, in Fig. 4.3-(b), C5 is detecting the robot SA, so it sends both to C3 and C4 a message with the route *C5*. Likewise, C3 forwards to C1 a message with the route *C5-C3*. Similarly, C4 sends to C2 a message with the route *C5-C4*. Finally, C2 sends to C1 a message with the route *C5-C4-C2*.

4. Finally, the camera that has issued the call receives all the back-propagated acceptances with all the feasible routes (Camera Network Module, Fig. 4.1). The active Call Module of this camera (Fig. 4.1) selects the route that minimizes the number of cameras involved, and informs the corresponding robot (*C2R_INFORMROUTE*, Fig. 4.1). Then, it waits until the robot arrives, provided that it accepts the route (Call Module, Fig. 4.1). In the example of Fig. 4.3-(b), camera C1 receives two possible routes (*C5-C3* and *C5-C4-C2*), selects the first one, appends its identity to it, and sends the resulting route (*C5-C3-C1*) to the robot.

It is clear after this description that the route planning does not emerge from a globally coordinated process, but from a self-organized process coming from multiple local interactions among neighbouring agents, which only handle local information. The proposed distributed routing process adapts to any changes without needing any manual tuning, without the intervention of any central entity, and without requiring any cameras to monitor the state of the others. The malfunction of a camera does not collapse the performance of the system, since the calling camera would still receive the remaining routes in the network. For example, in Fig. 4.3, if the camera C3 fails, the camera D1 still receives the route *C5-C4-C2-C1*. If there is not any route (e.g. if cameras C4 and C3 fail in Fig. 4.3), the cameras can ask the user to establish new neighbourhood connections.

4.1.3 Support to robot navigation

Any robot following a route must traverse the FOVs of those cameras involved in it. To this extent, each camera on the route helps the robot to move towards the next Neighbourhood Region, so that the robot reaches the FOV of the next camera in the route. This remote control process is illustrated in Fig. 4.4. First of all, if a camera sees a robot (Vision Module, Fig. 4.1), it informs it about this fact (*C2R_HELLO* in Fig. 4.1). If the robot is following a route, it sends this information to the camera (*C2R_INFORMSTATE* in Robot Network Module, Fig. 4.1). Then, the camera measures the robot's position and orientation, and informs the robot about the direction that it should follow to get to next Neighbourhood Region (*C2R_UPDATEMOV* in Camera Vision Module, Fig. 4.1).

The camera will repeat this process (e.g., each few seconds) until the robot arrives at the next camera, which will continue guiding the robot by the same process, and so on. The control of the robot is purely reactive, updated with each new camera command. Unlike most proposals [17, 52, 41, 18, 31], we do not require our cameras to calculate the robot position in the world coordinates (just its orientation or movement direction), and hence we do not need to calibrate the cameras.

On the other hand, the robot navigation is based on the classic but robust Vector Field Histogram controller [53]: the robot moves towards the goal position commanded by the

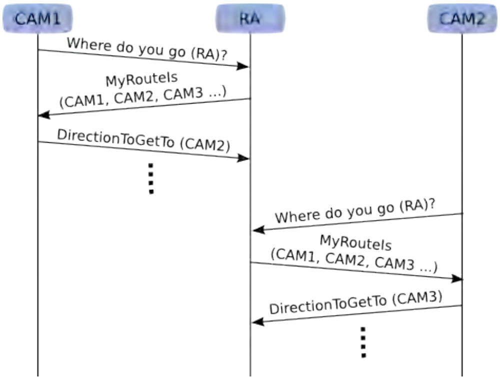
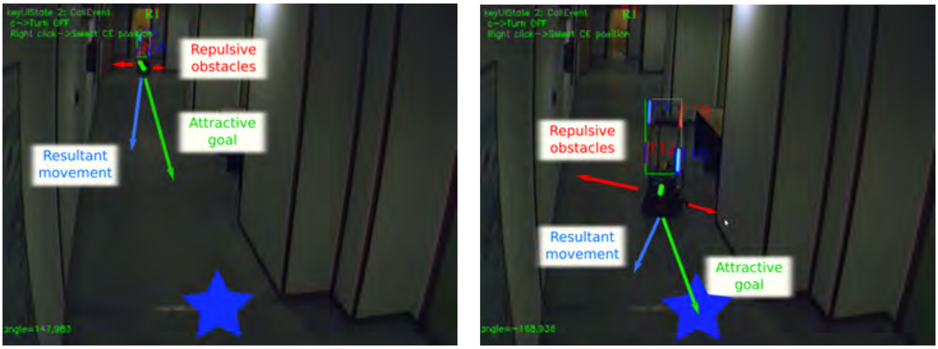


Figure 4.4: Remote control process example. First, Camera 1 (CAM1) detects the Robot A (RA) and asks it about its current route. RA answers with its route, and CAM1 corrects its movement direction towards the next camera on the route (CAM2). This process continues until RA reaches the Field of View of Camera 2 (CAM2), which will start a similar process to steer the robot towards Camera 3, and so on.

camera, which exerts an attractive force on it, while the obstacles detected by the laser scanner exert repulsive forces, so that the robot avoids colliding with them.

Figure 4.5 shows an example of support of robot navigation from a camera. In this example, the robot has to reach the point tagged with a star (next camera on the route). To this extent, the camera detects the position and orientation of the robot, and calculates the movement direction towards the star (green arrow). Meanwhile, the robot calculates the forces that repel it from the obstacles (red arrows). Finally, the resultant movement direction results from a lineal combination of both repulsive and attractive forces.



(a) (b)

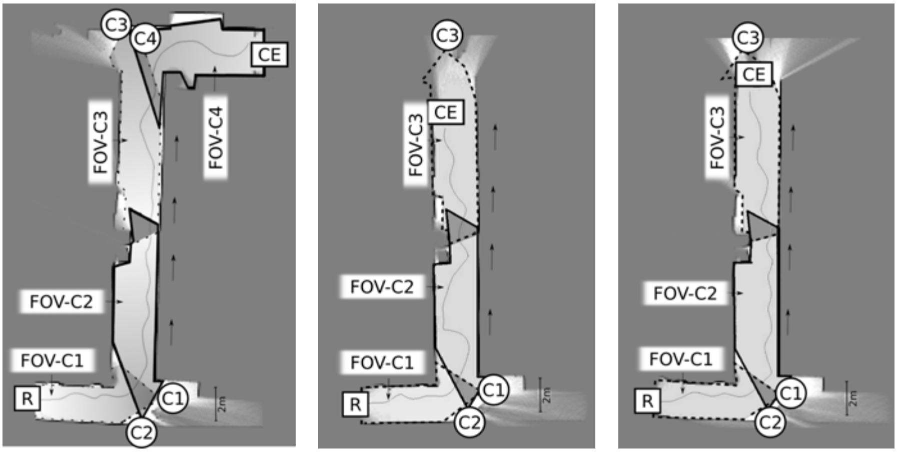
Figure 4.5: Example of support of robot navigation from a camera using a Potential Fields Controller. The star represents the position that the robot has to reach. The red arrows represent repulsive forces exerted from the obstacles in the environment. The green arrow represent the attractive force towards the goal position. The blue arrow represents the resultant force, which indicates the movement direction.

4.1.4 Experimental results: robot with a passive marker

First of all, we have tested our system with a robot carrying a passive marker (see Fig. 3.1). In this case, the camera-agents implemented the robot detection and tracking algorithm based on passive markers described in Sec. 3.1.2. We have tested this configuration at the Department of Electronics & Computer Science, at the University of Santiago de Compostela, Spain. We have used a Pioneer P3DX robot with a SICK-LMS200 laser. On the other hand, we have used either a Unibrain Fire-i camera, or a PointGrey Chameleon CMLN-13S2C camera with a FUJINON FUJIFILM Vari-focal CCTV Lens (omnidirectional). Both models worked at a frame rate of 15 fps and at a resolution of 640 × 480 pixels. The processing units were either a DELL Latitude E550 (Intel(R) Core(TM) 2 Duo P8600 @ 2.4 GHz, 4 GB RAM) or a Toshiba Satellite A100-497 (Intel(R) Core(TM) 2 Duo T5600 @ 1.83 GHz, 4 GB RAM). The software of the robot controller was implemented using the Player(v-3.0.2)-Stage(v.4.0.0), and the software of the cameras used the OpenCV 2.2 library [45]. Finally, messages were passed over an IEEE 802.11g local wireless network via UDP.

The objective of the experiment was to test how the cameras detect a call event, establish neighbourhood relationships among them, form global routes, and support robot navigation to reach the call event. To this extent, we have performed three different tests, represented in Fig. 4.6. In all the tests, we have used a call event simulated by software for simplicity.

In the first test (Fig. 4.6-(a)), the network consisted on a robot-agent (R), four camera-agents (C1, C2, C3, C4) and a call event (CE). During the experiment, C4 sighted the call event (CE), started the robot call process, and triggered the route formation. Upon reception of the route, R started navigating through the network towards CE, supported by C1, C2, C3 and C4 (in this order, from left to right, bottom to top in the Figure), while avoiding the obstacles detected. In the second and in the third tests, the network consisted on three cameras



(a) (b) (c)

Figure 4.6: Trajectories described by the robot in three tests. a) The robot R navigates from the FOV of camera C1, passing through C2, C3 and finally C4, which triggered the call upon detection of the call event CE. b,c) the robot navigates from the FOV of camera C1, passing through C2, towards C3. The map and trajectory of the robot was obtained from the robot's odometry and laser readings using the PMAP SLAM library. In the figure we can also see the approximate positions of the cameras and their FOVs.

(C1, C2, C3) instead of four. The performances during these other two tests were similar to the performance of the first one, described above. The robot's trajectories during these experiments are shown in Fig. 4.6-(b-c). These trajectories and the maps of the scenario were obtained from the robot's odometry and laser logs using the PMAP SLAM library, compatible with Player-Stage. We repeated these tests a few times, obtaining in all of them a satisfactory performance of the robot.

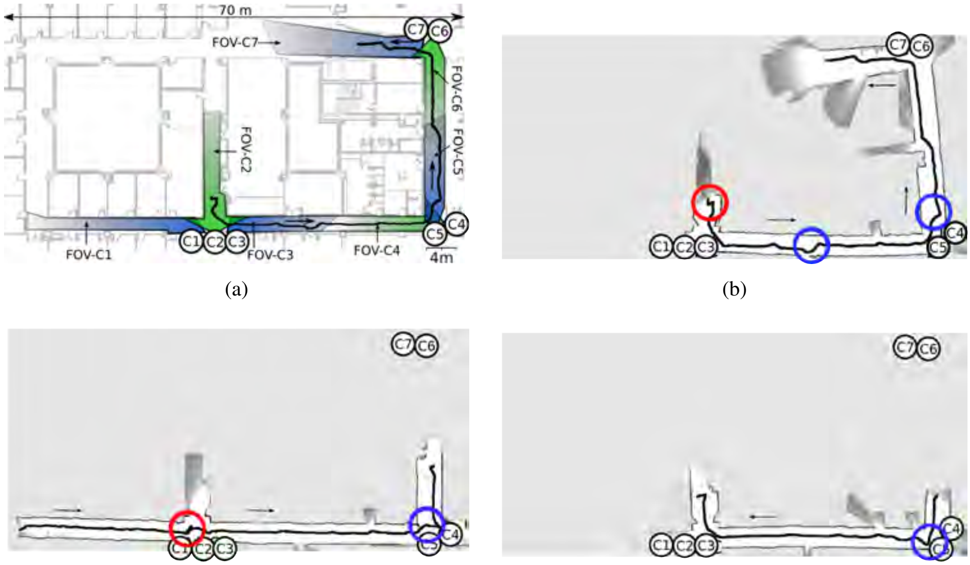
Although the results are satisfactory, there is room for improvement. Particularly, observe that the trajectories are a bit curvy, and therefore inefficient. This was due to the poor performance of the robot detection and tracking algorithm with passive markers (Sec. 3.1.2), and the lack of intelligence of the robot controller.

In this experiment each camera has only two neighbours, nevertheless the FOV of three or more cameras might overlap and therefore the number of neighbouring cameras can be higher than two. Our system would be able to cope with this, because the route that is finally selected to be followed by the robot is always the one that involves the fewest number of cameras.

4.1.5 Experimental results: robot with active markers

In a second experiment, we have tested our system with a robot carrying active markers (see Fig. 3.6). In this case, the camera-agents implemented the robot detection and tracking algorithm based on active markers described in Sec. 3.1.3. We have tested this configuration at the Department of Electronics and Computer Science, at the University of Santiago de Compostela, Spain. In this experiment, we have deployed seven camera-agents (C1 to C7) and a robot-agent (R) (Fig. 4.7-(a)). Each camera covered one corridor, and each had two neighbours at most (their FOVs can be seen both in Figs. 4.7 and 3.8). For instance, cameras C2 and C3 covered two perpendicular corridors, being C1 and C3 the neighbours of C2, and C2 and C4 the neighbours of C3. After the deployment of the system, the cameras recognised their camera neighbours correctly.

Initially, in all the experiments there was only one camera detecting the robot, and one camera detecting a call event. This last camera always called the robot, and the robot accepted the call. Next, the cameras calculated the appropriate route of cameras towards the call event. Finally, the robot moved through this route supported by the cameras. We show three of the most representative robot trajectories from these experiments in Fig. 4.7-(b,c,d). In all the experiments, the robot moved at the maximum possible speed considering its weight (0.5 m/s). As an example, in Fig. 4.8 we show the linear speed of the robot during the trajectory



(a)

(b)

(c)

(d)

Figure 4.7: a) Camera network deployed in the experiments. Each camera is represented by a letter (from C1 to C7), and each FOV is represented by a coloured region. The trajectory represented is the same as in (c). b) Robot trajectory through the route C1-C2-C3-C4-C5 (60 m long). c) Robot trajectory through the route C2-C3-C4-C5-C6-C7 (70 m long). d) Robot trajectory through the route C5-C4-C3-C2 (50 m long). We provide explanations of each trajectory in the text.

in Fig. 4.7-(c). The speed of the robot varies depending upon the environment and the camera commands. We obtained all these data from the robot's odometry and laser logs using the PMAP SLAM library, compatible with Player-Stage. The maps and trajectories were obtained off-line just for visualization purposes: as we have explained before, so far our system does not need maps of the environment.

In the first experiment (Fig. 4.7-(b)) camera C7 sighted the call event, started the robot call process, and triggered the route formation. Upon reception of the calculated route (C2-C3-C4-C5-C6-C7), the robot moved towards the call event position supported by the cameras while avoiding the obstacles detected (from camera C2). The experiment shows that most of the trajectory is regular and most transitions among cameras are smooth. This trajectory was 70 m long.

In the second experiment (Fig. 4.7-(c)) camera C4 sighted the call event, and the robot navigated through the route C1-C2-C3-C4-C5, starting from camera C1. The trajectory was

60 m long, and as in the previous experiment, the result was satisfactory. Fig. 4.8 represents the linear speed of the robot during this trajectory.

In both experiments, the robot movement was counter-clockwise. In the last experiment (Fig. 4.7-(d)), we tested a clockwise movement. In this case, camera C2 detected the call event, and the robot moved towards it through the route C5-C4-C3-C2, starting from camera C5. The trajectory length was 50 m long approximately.

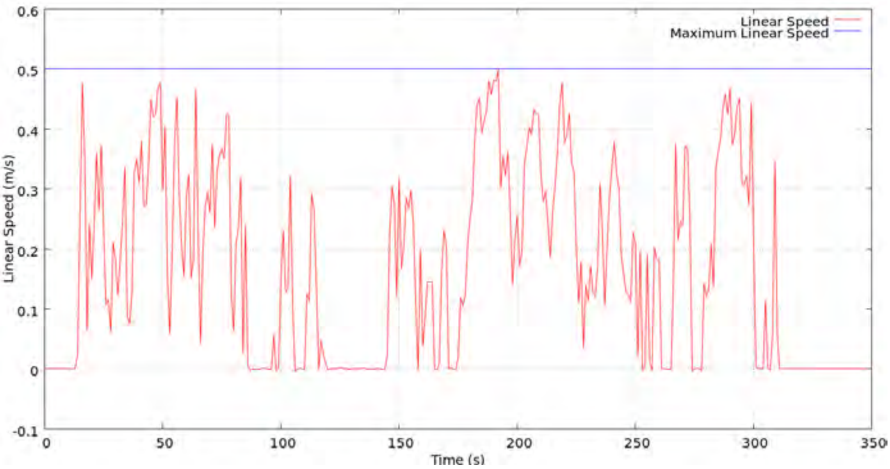


Figure 4.8: Linear speed of the robot during the experiment represented in Fig. 4.7-(c).

We observe that the robot movement is now much smoother than in the previous experiments (Sec. 4.1.4). This is because the robot detection and tracking algorithm with active markers is much more precise and robust than its homologous with passive markers. Nevertheless, we have observed in the trajectories two kinds of anomalies that require further explanation. We have tagged these anomalies in Fig. 4.7-(b,c,d) either with a red or a blue circle, depending on the kind of error. First, red circles represent parts of the trajectory where the cameras estimated incorrectly the orientation of the robot, causing momentary navigation errors. This kind of anomaly is not common, but nevertheless it is clear that our system recovers from it very quickly. Second, blue circles represent anomalies during transitions among cameras. For instance, in the transition from C3 to C4 represented in Fig. 4.7-(b), the robot should have moved straight on, but instead it turned slightly. This happened because the camera C3 provided the robot with an inaccurate orientation command to get to the FOV of camera C5. In general, we have observed that orientation commands are less accurate as the robot gets farther from the camera commanding it (C3 in this case) and closer to the objective area (the FOV of C4, in this case). However, this problem is not critical nor common: as we can see, it did not appear in the experiments in Fig. 4.7(c,d).

On the other hand, the problem observed during the transitions from cameras C5 to C4 and vice versa is more important. During these transitions, there were time lapses when the robot navigated without guidance (blue circles). We observed that the overlap between cameras C4 and C5 was not sufficient, depending on the trajectory followed by the robot. These anomalies are rather harmless, but nevertheless we can correct them by increasing the FOV overlap amongst both cameras. Another option would be to use alternative systems to provide the robots with information to traverse the areas that the cameras do not cover. This would allow us to eliminate the need of overlapping FOV among neighbouring cameras.

We have performed many of these experiments varying the initial position of the robot and of the call event. The system proved to be robust, because the robot always arrived within a circle of 2 m around the call event area. We believe that this is sufficient for most real world applications where our system could be used. For example, this would suffice if the robot is meant to approach groups of people to offer them information, or to navigate towards areas where an anomaly is detected. On the other hand, in all the experiments that we have carried out, the success of the robot was 100%.

4.2 Cameras with non overlapping FOVs

In the previous section, we have assumed that two cameras could only be neighbours if their FOVs overlapped (partially, at least). With this restriction, we need a large amount of cameras to cover relatively small areas. Moreover, the deployment of the cameras is not as easy as we would like. On one hand, the cameras should always be placed in elevated positions pointing in a direction which ensures that their FOVs cover medium distances. On the other hand, cameras on the same mast should be pointed downwards to ensure an overlap between their FOVs, but this limits the coverage to short distances. To achieve a solution that satisfies both requirements, we usually need to orient our cameras very carefully. This process requires expertise and increases the complexity of the deployment. We have also used wide-angle cameras to alleviate this problem, but in turn they add distortion to the images, and they do not work well in medium distances.

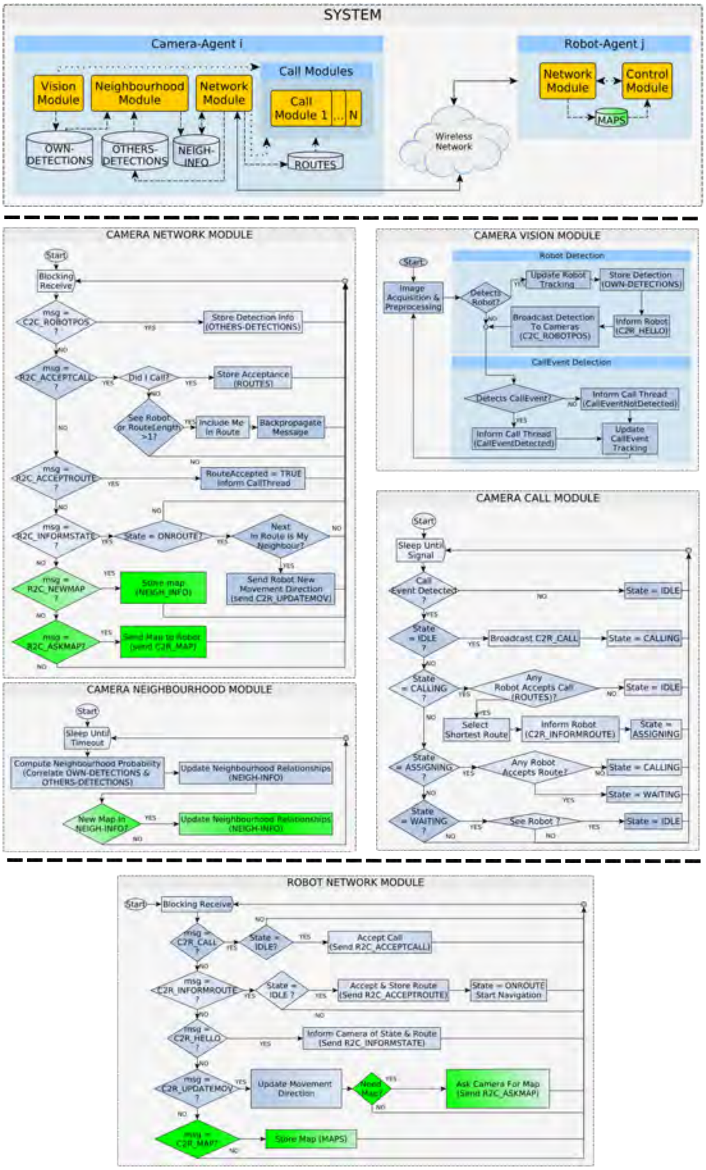


Figure 4.9: Software architecture of our system (cameras with non-overlapping FOVs). We highlight the differences with Fig. 4.1 in green.

It is clear that the FOV overlap assumption is highly restrictive. In this section we will describe several changes we have carried out to mitigate this restriction. The new software architecture is depicted in Fig. 4.9, with the changes with respect to the previous one in green.

4.2.1 Dynamic neighbourhood detection

In order to alleviate the FOV overlap restriction, we will consider that two cameras are neighbours if the robot can move among them without passing through any other camera. This may happen in two cases:

1. Obviously, when their FOVs overlap. For example, cameras C1 and C2 in Fig. 4.10.
2. When their FOVs do not overlap, but there is a passable path among them which does not cross the FOV of any other camera. For example, cameras C4 and C5 in Fig. 4.10. This extends and generalizes the first case.

In order to capture the neighbourhood relationships among the cameras, we move the robot around the environment during the "deployment" phase. The process by which overlapping cameras discover their neighbourhood relationships has been already explained in Sec. 4.1.1. The second case, where two cameras whose FOVs do not overlap can be neighbours, is more challenging. In this case, when the robot is moving within the FOV of a camera, this camera

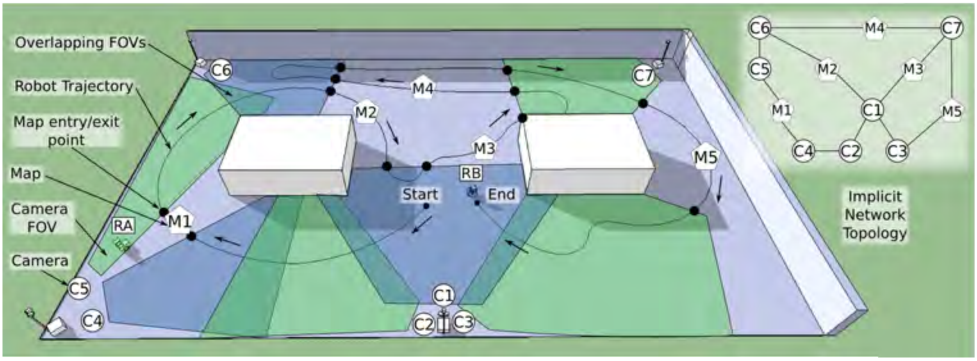


Figure 4.10: Example of our system's deployment. To capture the neighbourhood relationships of the cameras, we move the robot in the environment. The Network Topology is implicit, i.e. the cameras only store information about their local neighbours (overlapping FOVs and/or interconnection maps).

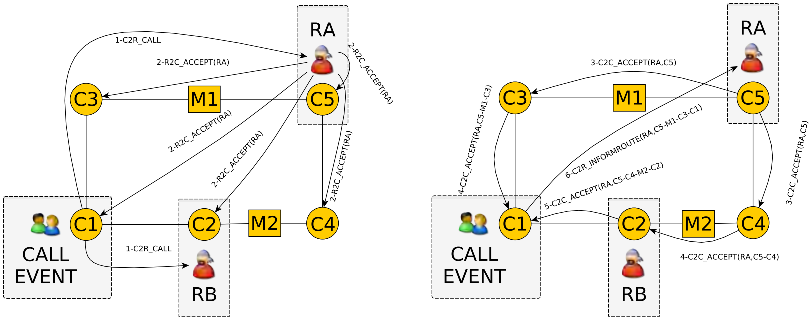
sends periodic messages to it (*C2R_HELLO* in Camera Vision Module, Fig. 4.9). Because of this, it is clear that when the robot is moving and it is not receiving any of these messages, we can assume that it is moving in a non-covered area. Therefore, the robot begins to map this local area, until it reaches the next camera FOV. Then, it informs both cameras (the last camera which saw the robot and the new one that is seeing it) about their neighbourhood relationship, and sends them the map which connects them. When the cameras receive this map (*R2C_NEWMAP*, Camera Network Module in Fig. 4.9), they store it and update their neighbourhood relationships (Camera Neighbourhood Module, Fig. 4.9).

As an example of this process, in Fig. 4.10, when the robot exits the FOV of C4, it maps its trajectory until it reaches the FOV of C5, building the local map M1. Then, it sends this map M1 to C4 and C5, which store it and establish a neighbourhood relationship between them. Instead of a map, we could also use any way to move from one place to another: a robot-behaviour like wall-following, a trajectory (e.g. based on odometry, GPS, or WIFI localization), etc. The environment can now be represented as an implicit topological graph (Implicit Network Topology, Fig. 4.10), where the nodes are either cameras or maps, and the arcs their neighbourhood links. This representation is used just for clarification: no agent keeps global information of any sort.

Again, the process of dynamic neighbourhood detection runs continuously so that the cameras can update their neighbourhood information any time: they can establish new neighbourhood relationships (e.g. if we include a new camera), modify them (e.g. if we change the FOV of a camera), or eliminate existing ones (e.g. if we remove a camera).

4.2.2 Distributed route planning

The distributed route planning process is similar to that explained in Sec. 4.1.2. First of all, a camera detects a call event and broadcasts a call message. This is what happens with the camera C1 in the example shown in Fig. 4.11-(a). Then, the robots that can accept the call broadcast an acceptance message to all the cameras, such as the robot RA does in Fig. 4.11-(a). This acceptance starts a similar backpropagation process to the one described in Sec. 4.1.2. Again, when a camera receives this message, it appends its identity to the message and forwards it to its neighbours (if the message has not passed through them yet). However, in this case if the camera is connected to a neighbour by a map, it will append the identity of the map (to the message forwarded to this particular neighbour). Finally, the camera that has issued the call will receive all the feasible routes, that may or may not contain interconnection maps. For instance, in Fig. 4.11 we observe how the system calculates two feasible routes: *C5-M1-C3-C1* and *C5-C4-M2-C2-C1*. From the two, camera C1 will choose the shortest one (*C5-M1-C3-C1*).



(a) (b)

Figure 4.11: Schematic representation of the distributed route computation in a network of 5 cameras (C1 to C5), 2 maps (M1 and M2) and 1 robot (RA). a) Camera C1 detects a call event and calls for robots. b) The cameras back-propagate the robot acceptance message to form a route. The first element of the message contains the identity of the robot, and the second the route.

4.2.3 Support to robot navigation

As in Sec. 4.1.3, the robot navigates through the route of cameras calculated by the network: each camera supports its movement towards the next camera on the route until it reaches the call event. In case that the robot has to traverse an area amongst two cameras that is not covered by any of them, it will have to navigate using their interconnection map. To this extent, the robot will be headed to its entry point from the camera that is seeing it. Then, the robot will ask the camera for the map (*R2C_ASKMAP* in Robot Network Module, Fig. 4.9), and the camera will send it to it (*C2R_MAP* in Camera Network Module, Fig. 4.9). Finally, the robot will use the map to navigate towards the next camera on the route.

4.2.4 Experimental results with active markers

We have tested this proposal at the Department of Electronics & Computer Science, at the University of Santiago de Compostela, Spain. We have deployed our system using different camera distributions (Figs. 4.12-(a-c), cameras C1, C2, C3 and C4), and one robot (R). Camera C1 shared a mast with C2, and camera C3 another one with C4. In all the tests, C4 was the camera seeing the call event (CE). The system calculated the route of cameras and maps (if any), and supported the robot navigation towards the call event, at an approximate linear speed of $0.4m/s$.

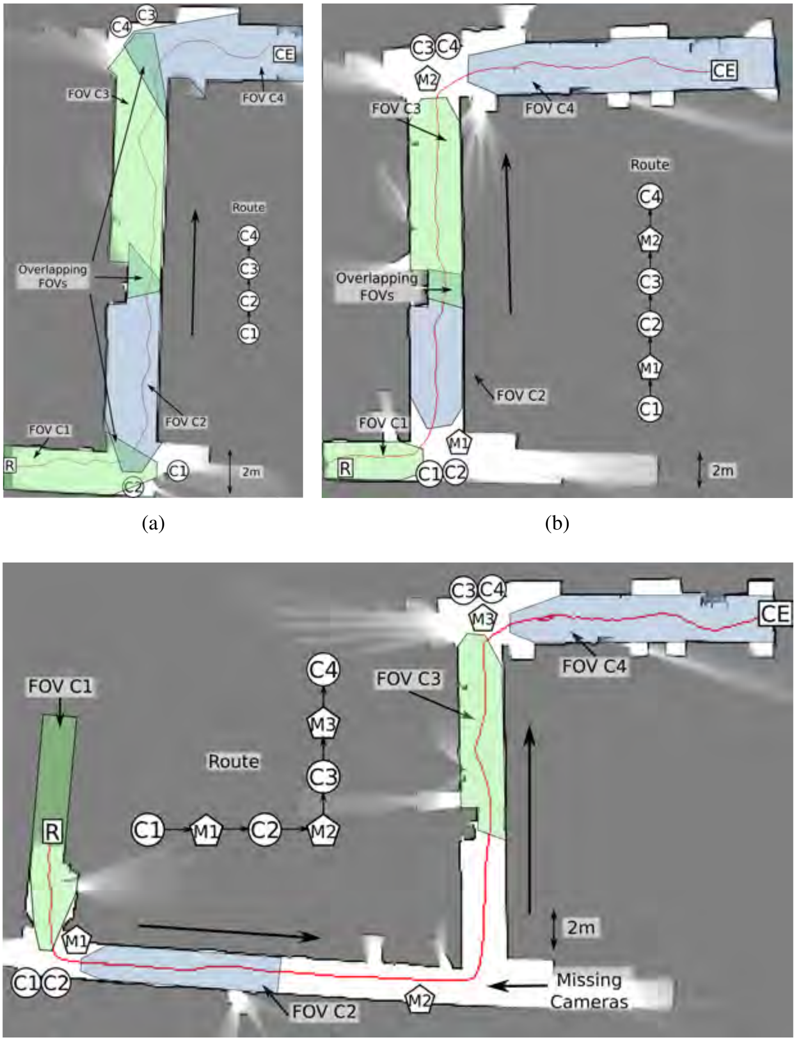
We have made many experiments to test our system, all of them successful, but we have just selected the three most representative to show in this thesis. In the first experiment (Fig. 4.12-(a)), there is some overlap between the FOV of the neighbouring cameras, such as in Sec. 4.1.5. In the second (Fig. 4.12-(b)), there is some overlap between some of the FOVs of neighbouring cameras, but there are also cameras whose FOVs do not overlap. In the third (Fig. 4.12-(c)), there is no overlap between any two neighbouring cameras. The maps used by our robot to navigate among neighbour cameras were metric maps built with the PMAP SLAM library [53]. It is important to be aware of the fact that the trajectories and the maps shown in Fig. 4.12 have been obtained only for visualization purposes, but our system does not use a complete map of the environment.

In the first experiment (Fig. 4.12-(a)), all the cameras' FOVs overlap by pairs (C1 with C2, C2 with C3, C3 with C4). This can be seen in Fig. 4.13, which shows a capture of each camera with their overlapping FOV areas shaded. In this experiment, the robot moves always supported by at least one camera, without needing any interconnection map. The robot's trajectory is smooth, even during the transition between cameras.

With this restriction of overlapping FOVs between neighbour cameras, the area covered is relatively small, and the robot trajectory is short (around 30 meters). We have already discussed that the overlapping FOV restriction has several drawbacks. For instance, it leads to a system that requires too many cameras to cover a wide area. Moreover, it forces the users to orientate the cameras very carefully to satisfy the restriction (e.g. cameras C1, C2, and C3 in Fig. 4.13 had to be rotated), or to use wide-angle cameras (like C4 in Fig. 4.13).

In the second experiment (Fig. 4.12-(b)), only the FOVs of cameras C2 and C3 overlap. During the "deployment" phase, we have moved our robot to create a map between C1 and C2 (M1), and between C3 and C4 (M2). In Fig. 4.14 we show frames captured from the cameras and a representation of the local metric maps.

To get to the call event (CE), the robot alternates between a navigation directly supported by the cameras (like in the previous experiment) and a map based navigation. In this last case, the robot uses Adaptive Monte-Carlo Localization (AMCL) for localization, Wavefront for path planning, and Vector Field Histogram (VFH) for navigation [53]. The robot trajectory



(a)

(b)

(c)

Figure 4.12: Examples of maps and trajectories obtained from three different tests of our system (robot with active markers). a) There is some overlap between the FOVs of all the neighbouring cameras. b) There are neighbouring cameras whose FOVs do not overlap. c) There is no overlap between any two cameras. These maps were not used for robot navigation: they were obtained after the experiments just for visualization purposes.

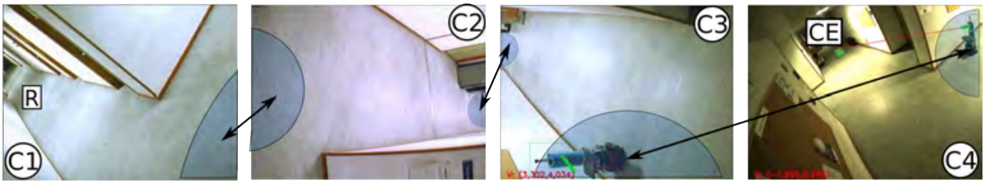


Figure 4.13: Frames captured simultaneously from the cameras on the first experiment (Fig. 4.12-(a)). The overlap between FOVs are shaded in blue, and they are connected with double arrows.

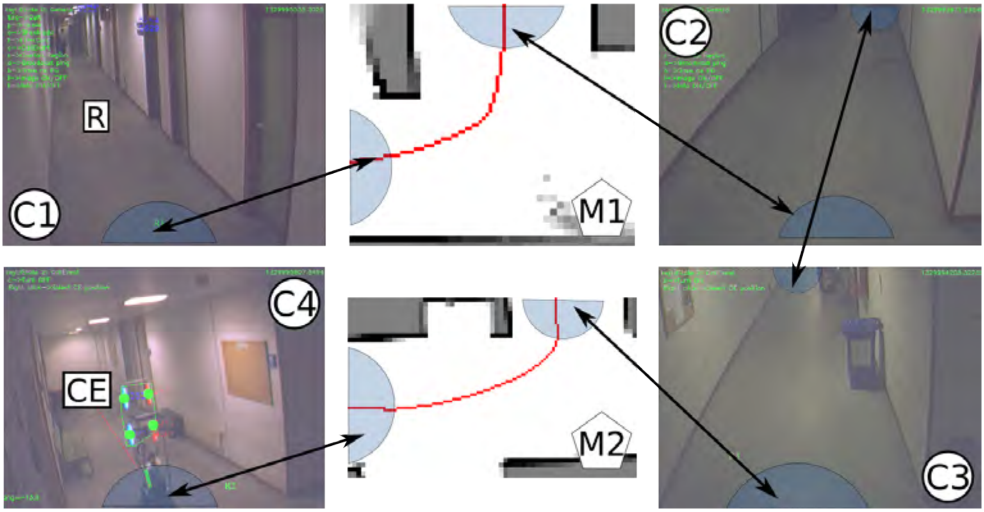


Figure 4.14: Frames captured simultaneously from the cameras on the second experiment (Fig. 4.12-(b)), and representation of the metric maps among them. The overlap between FOVs and the entry/exit areas of the maps are shaded in blue and connected with double arrows.

(Fig. 4.12-(b)) is smooth both when controlled by the camera and when based on a map (and on the transitions). Moreover, as we can see in Figs. 4.12 and 4.14, now we cover wider areas, and the robot is able to move along longer trajectories (around 40 meters in this experiment).

Finally, in the third experiment (Fig. 4.12-(c)) there is no overlap between the FOVs of any cameras. In this experiment, the robot navigates from C1 to C2 using the map M1, from C2 to C3 using M2, and from C3 to C4 using M3. Here, we cover a much wider area with the same number of cameras, and the robot is able to move along a longer trajectory than in the previous experiments (around 80 meters). The robot moved smoothly along the whole trajectory, except when it entered in the FOV of camera C4 (Fig. 4.12-(c)). Here, it moved incorrectly for a moment, because the Vision Module of camera C4 estimated wrongly the orientation of the robot. However, our system is very robust to these situations, therefore in this particular case the robot was able to recover, and it finally got to the CE position without any problem.

Note that the uncovered area mapped by M2 is particularly large. In this area, it is not possible to detect call events from the cameras: generally speaking, the more space we leave uncovered by the cameras, the less real time feedback the robot receives from them. This degrades the robots' ability to detect user's needs out of its immediate surroundings, loosing the benefits from the Ubiquituous Robot approach [14]. Thus, as a rule of thumb, the less uncovered areas, the better our system will work. However, as we demonstrate with this experiment, we also provide the possibility to work with wide non covered areas. This may be beneficial in different situations: when these areas are non interesting for call event detection, when their illumination conditions cause a bad performance of the cameras, when there are restrictions that prohibit their monitorization, etc.

4.3 Dynamic neighbourhood detection from the movement of people in the environment

The work presented in this section has been developed in collaboration with Jose M. Abuin, and has been presented as part of his Final Degree Project [54]. In the previous section, we have proposed that two cameras should be considered neighbours even if their FOVs do not overlap, provided that there is a passable path among them, and this path does not cross the FOV of any other camera. To detect these paths, we proposed to move the robot around and rely on the re-identification of the robot from the cameras. The robot could just map the space between cameras and use it to navigate when no camera is detecting it.

In this section, we will continue exploring this concept. Now, instead of detecting passable paths by relying on the robot, we will try to detect persons that have passed through several cameras, in order to establish neighbourhood links [55, 56]. First of all, each camera will detect and track the persons within its FOV. Then, each camera will extract features of these persons and build a model of each of them. These models will be broadcasted to all the cameras. Therefore, each camera can compare the persons that it is detecting with the models available, in order to re-identify people that has already been detected by other cameras. The number of re-identifications will serve to compute a probability of neighbourhood relationship among the cameras.

We would like to remark that the dynamic neighbourhood detection task will usually be performed during deployment time. Therefore, we will not have severe real-time constraints like in other cases, such as when detecting a robot to support its navigation smoothly. Even more, we could just record all the video sequences during deployment, and perform the neighbourhood detection offline. Consequently, we can use almost any algorithm, regardless of how computationally intensive it is.

4.3.1 Person detection

First of all, the cameras must detect the people within their FOV. Note that in this case we do not have severe real-time constraints, therefore we can use state-of-the-art algorithms. Therefore, we will use the person detector algorithm available in the OpenCV 2.4 library [45]. This detector is based on the extraction of HOG features [36] and classification using a SVM classifier [57] and a sliding window approach.

The output of the person detector is a bounding box surrounding each person (Fig. 4.15). We prefer to be conservative when establishing new links among cameras. Therefore, it is

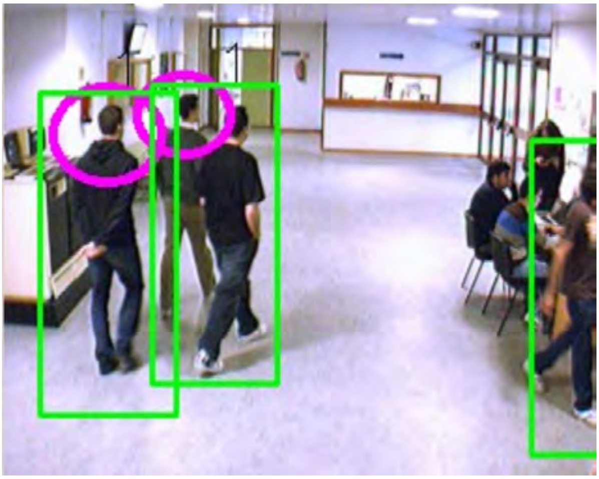


Figure 4.15: Person detection. First, we detect the persons in the scene (green bounding box) and then we refine the results by detecting their heads (purple circle).

very important that false positives remain as low as possible, but we will be tolerant to false negatives. For this reason, we refine the results by trying to detect the head of each person. This will be done with Viola-Jones head detector [37] provided by OpenCV 2.4. This detector finds rectangular regions in a given image that are likely to contain the objects the detector has been trained for (heads in this case). The detection is based on Haar-like features and weak classifiers (decision trees) trained using AdaBoost [58].

4.3.2 Person characterisation

Each person detected has to be described. To this extent, we extract several features from the torso of the person: hue and saturation values of the colour in the HSV space, Center Symmetric Local Binary Pattern (csLBP) features [59], and the partial derivatives in the horizontal and vertical directions [60]. Then, we build a histogram $hist_f$ with each feature f, that will be updated with each new detection of the same person. The set of histograms $H_i = \{hist^i_H, hist^i_S, hist^i_{LBP}, hist^i_{dx}, hist^i_{dy}\}$ will be the model that we will use to re-identify each person i. Therefore, each camera will send periodically the histograms of the persons that have passed through its FOV to the rest of the cameras, along with the timestamps of the detections. The comparison of visual features using histograms has the advantage of being invariant to the size of the human region, and to the pose of the human [61, 62].

4.3.3 Person re-identification

With each new frame, the cameras will detect several persons within their FOVs, and build or update their models. These models can be checked against the models of the persons detected in other cameras. If the model of a person being detected is very similar to the model of a person detected by other camera, we will assume that a re-identification has been produced. To evaluate the similarity among each pair of models H_i and H_j, we will use the average of the Bhattacharyya distance among each pair of histograms:

$$d_{model}(H_i, H_j) = \frac{1}{n_f} \sum_{f=1}^{n_f} \left[\frac{\sum_{k=1}^{b} \sqrt{hist^i_f(k) hist^j_f(k)}}{\frac{1}{b}\sqrt{\sum_{k=1}^{b} hist^i_f(k) \sum_{k=1}^{b} hist^j_f(k)}} \right]^{1/2} \tag{4.1}$$

where f represents a certain feature, n_f the number of features, and b is the number of bins in those histograms. This similarity measurement gives us a number between 0 and 1, where 1 means more similarity.

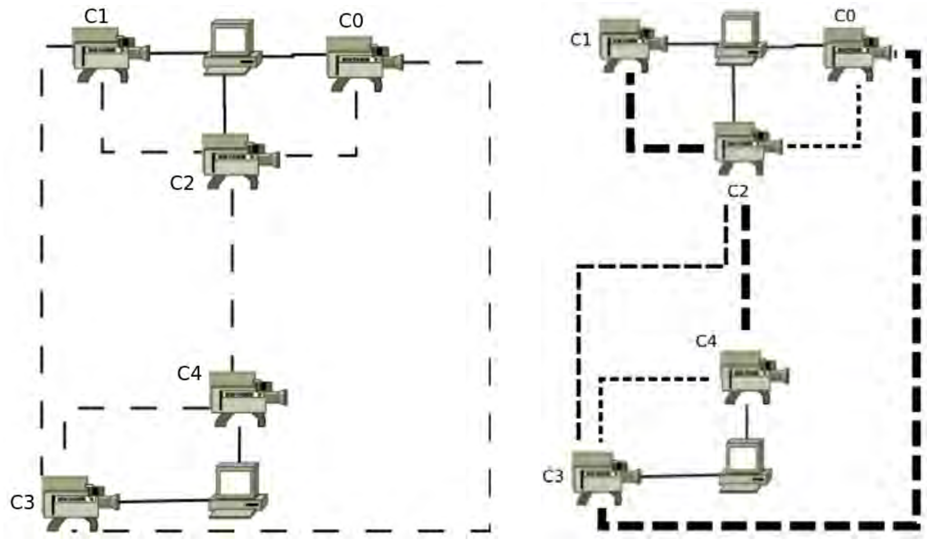
4.3.4 Neighbourhood link establishment

After some time, there will be several persons that will pass through the FOV of the cameras in the network. Therefore, we would expect several re-identifications to occur. Each camera C^i will count the number of re-identifications with each other camera C^j (which will be represented as n_{reid}^{ij}) and communicate it to the rest. Therefore, we can establish a measurement of the probability or likelihood of each neighbourhood link among two cameras C^i and C^j as:

$$p(C^i, C^j) = \frac{max(n_{reid}^{ij}, n_{reid}^{ji})}{\max\limits_{\forall k,l \in C}(n_{reid}^{kl})} \tag{4.2}$$

where $C = \{C^1, \ldots, C^{n^C}\}$ is the whole set of cameras (n^C is the total number of cameras). Figure 4.16-(a) contains an example of a camera network used in one experiment, and Fig. 4.16-(b) the result of the neighbourhood establishment process after 15 minutes. The results depend on the topology of the network, the number of persons that are walking around, and the running time. Nevertheless, we can observe how the process captures the neighbourhood relationships correctly, giving more likelihood to those where the affluence of people is higher. Note that the link between cameras C2 and C3 has been established incorrectly, however with a low likelihood. This happened because some persons that moved from the area covered by camera C2 to the area covered by camera C3 where not detected by camera C4, therefore they did not contribute to the link between C2 and C4, but to that of C2 and C3. In addition, we observed that the neighbourhood link among C1 and C3 was not established. This happened because the number of people that moved among them in the experiment was not sufficient.

This information serves to calculate routes of cameras through which the robot can move. Moreover, we have inferred the average time that a person needs to get from one camera to another. Therefore, the robot can know approximately how long it would take him to navigate among each pair of cameras. This would be very interesting if we wanted our robot to perform automatic exploration of the environment, since it would reduce the inherent uncertainty of this process.



(a) (b)

Figure 4.16: a) Camera network with the real neighbourhood connections as dotted lines. b) Established neighbourhood connections among cameras (the wider the line, the likelier the connection).

4.4 Discussion

In this chapter, we have presented a software architecture and algorithmic procedures to operate our multi-agent system following the collective intelligence paradigm. Our architecture is fully distributed amongst the agents, based on local interactions amongst them and self-organization processes. The system we have proposed is robust and redundant, and scales well with the size of the environment and the number of cameras and robots. Moreover, it is flexible to the changes in the environment, to the number of agents used, and to their disposition.

Under the collective intelligence paradigm, intelligence and decision making is fairly distributed among agents. Therefore, global intelligence arises from individual interactions without the intervention of a central agent. In our case, a great deal of intelligence was assigned to the network of camera-agents: although initially identical, they were able to observe human and robot behaviour, learn in parallel, and specialize in the control of the robots. Meanwhile, the robots only followed the instructions of the cameras and negotiated obstacles in order to avoid collisions. Although this approach has proven to be effective, we must bear in mind that in real operation the cameras may not be completely reliable at all times. For instance, the cameras may fail sporadically due to extreme illumination conditions, or people walking by the robot may occlude their vision. In these cases, the robot behaviour might not be as robust as we expect.

This issue can be mitigated by assigning more intelligence and decision making capability to the robots. In our case, this intelligence will be devoted to perform the localisation and navigation tasks autonomously. On the one hand, this will make our system more robust and less sensitive to the failures of the cameras. On the other hand, these strategies will allow the robot to navigate through areas that are not covered by the cameras very naturally.

Chapter 5

Multisensor localisation

Hitherto, we have described a distributed and self-organised system of cameras and robots that can be deployed in different environments in a fast and easy manner. This system was based on the collective intelligence paradigm, where intelligence is fairly distributed among agents. In our system, the cameras detect situations that require the presence of the robots, and guide them towards the areas where these situations take place. In order to support the robot navigation, the cameras are able to detect their camera neighbours and compute routes of cameras through which the robot can navigate.

We have seen that if the FOVs of two cameras overlap, the robots can move between them just by following commands from the cameras. In addition, robots can move between non-overlapping cameras using local maps among them. However, none of these solutions is excessively robust. For instance, if the cameras fail, the performance of the system may suffer. However, we want to build robots that are able to operate in environments where there is people performing different activities, such as hospitals or museums. In these environments, the conditions are inherently dynamic: there will always be people moving around the robots, the illumination conditions will change depending on the time of the day, etc. Robots must execute their tasks correctly under these conditions, and they must do it on a continuous basis.

In order to achieve this, we will explore solutions based on the centralised intelligence paradigm, that assign most of the intelligence to the robots in order to exploit their capabilities. More concretely, we will enable our robots to determine their position accurately and robustly at all times (with or without the help of the cameras), so our robots will be able to navigate autonomously. To this extent, we will propose a multi-sensor fusion algorithm for mobile robot localisation based on particle filters [63]. We will explore the fusion of information sources both on-board and off-board the robot, such as: a) a 2D laser range finder, a WiFi positioning system designed by us, a camera mounted on the robot, a magnetic compass (information sources on-board of the robot), and b) our network of external cameras (information sources off-board of the robot).

First of all, in Sec. 5.1 we will discuss whether plain robot localisation or Simultaneous Localisation and Mapping (SLAM) should be used in our context. After that, in Sec. 5.2 we will review the most important works regarding mobile robot localisation, giving particular emphasis to multi-sensor fusion techniques. Then, in Sec. 5.3 we will describe a preliminary experimental study that we have performed regarding strategies that can be used in mobile robot localisation. After this study, we will propose our own multi-sensor localisation algorithm (Sec. 5.4), describe how to integrate each sensor in this algorithm (Sec. 5.5), and discuss some of the properties of the algorithm (Sec. 5.6). Finally, in Sec. 5.7 we will describe the main experimental results that we have obtained.

5.1 Robot localisation vs. Simultaneous Localisation And Mapping (SLAM): choosing a functional approach

Mobile robot localisation is the problem of determining the pose (position and orientation) of a robot relative to a map. This problem is one of the most important in mobile robotics, because most robotic tasks require knowledge of the robot pose [63, 64]. In order to work, localisation algorithms need a map of the working environment. This map is usually constructed by the robot, using Simultaneous Localisation and Mapping (SLAM) techniques [63]. Theoretically, these techniques could be used to construct the initial map, to localise the robot during its operation, and to update the map continuously.

However, in this thesis we will opt for a simpler but more functional approach. Again, our robot will work in two stages: deployment and operation. During the deployment stage, we will construct a map of the environment where our robot will work, using standard SLAM techniques. We will assume that we can create this map under controlled circumstances when the place is not crowded with people (e.g. non-working hours), and that this map will not be modified once created. Then, during the operation stage, the robot will use this map to localise itself, and to navigate towards where the users require it.

Of course, we could update the map continuously, but taking into account that the working environment does not change significantly, this adds little value while it consumes scarce computational resources. Moreover, in our experience continuous SLAM may eventually integrate people around the robot as part of the map, which would lead to failures. For instance, in the experiment of Fig. 5.1 our robot was following a group of people, and the Gmapping SLAM [65] algorithm integrated their legs as part of the free environment (dots on the corridors). These situations are still very challenging for state-of-the-art SLAM [66].

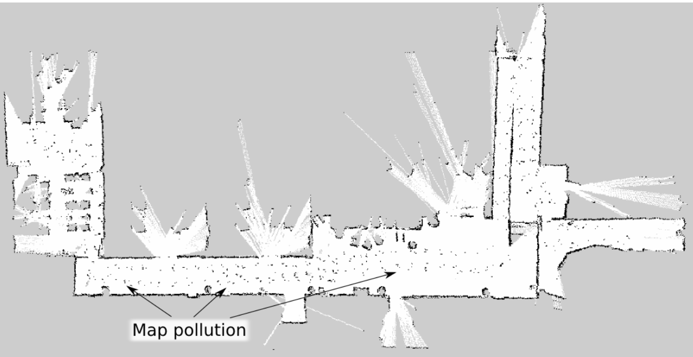


Figure 5.1: Laser occupancy map created with the Gmapping algorithm. The map was polluted due to the presence of people around the robot.

5.2 Multi-sensor fusion for robot localisation: an overview

Most localisation systems rely on the information provided by one sensor, such as sonar sensors [67], 2D lasers [63], 3D lasers [33], cameras mounted on the robot [68], etc. Nevertheless, there is not such a thing as a perfect sensor: each sensor has its limitations, and no sensor is applicable to all situations. Therefore, we can not expect these systems to respond robustly in every single situation that may happen in the real world. Localisation works have tackled this problem mostly by exploring the use of new sensors and by designing increasingly complex algorithms. In a complementary way, we want to explore the use of multi-sensor fusion techniques [69]: we believe that we can increase the robustness and redundancy of localisation systems by combining the information of sensors of different nature, which may fail in different situations.

Conceptually, we can categorise information sources depending on whether they are better at providing global or local estimates of the robot position. Sensors from the first group can provide a rough estimate and ensure that the robot is not completely lost, while the sensors of the second group can help on achieving high accuracy.

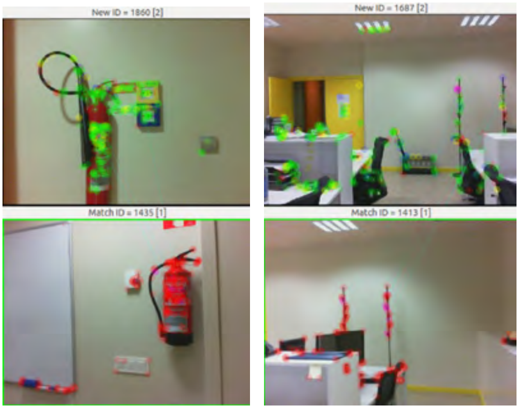
1. Global estimates. Sources such as wireless localisation or external cameras can provide rough global estimates of the robot position. For instance, when a camera detects a robot, the robot knows that it must be within its FOV. Similarly, when the robot receives a certain signal power from three or more wireless transmitters, the area where this reception is possible can be roughly delimited. In both cases it is unlikely to receive the same measurement in a totally different area, therefore the position estimate will be of high confidence (although it might not be highly accurate).

2. Local estimates. Sources such as 2D/3D laser range finders or on-board cameras might not be able to provide global estimates with great confidence at all times. This is because they might provide similar information on different areas. For instance, for a camera all the corridors of a hospital might look alike, therefore distinguishing among them might not be a trivial task. On the other hand, these sources usually can achieve highly accurate estimates, specifically when previous estimates of the robot pose are available.

Simultaneous Localisation and Mapping (SLAM) and mobile robot localisation has been traditionally approached by using one sensor only. A common approach has been to use onboard sensors that capture features of the world from the point of view of the robot. The most typical features are: a) distances to objects provided by sonars [67] or 2D [63] and 3D laser range finders [33], b) images provided by monocular [70] or omnidirectional cameras [68], c) combinations of both provided by RGBD [71] or stereo cameras [72].

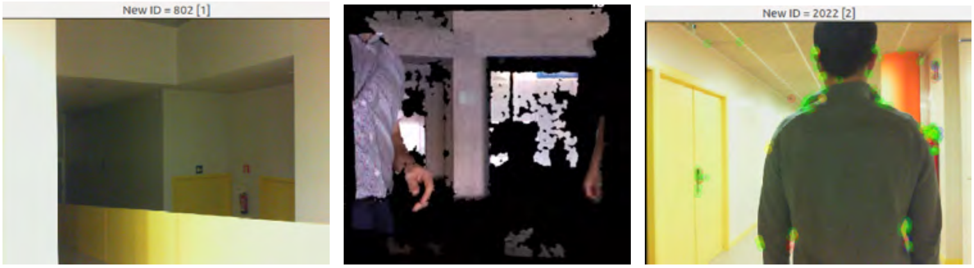
Usually, algorithms using these sensors are able to achieve sub-meter accuracy, but suffer from common problems that impact their robustness. First of all, the sensors may have different perception limitations (e.g. laser range finders might not detect glass walls, and cameras might be affected by illumination changes). On the other hand, they are unable to distinguish amongst dependencies that are similar to each other. For instance, in our experiments 2D laser range finders find it difficult to distinguish among geometrically similar rooms [73].

Vision based techniques such as Visual SLAM [74] rely on the detection and matching of visual cues, but in indoor spaces the distance to the scene is usually small and many identical



(a) (b)

Figure 5.2: Wrong image matching from visual cues using RTAB-MAP SLAM [71]. The robot extracts visual cues from the image provided by the camera (upper image) and detects a match with an image stored during the mapping stage (lower image). a) Miss-match due to the presence of a similar object (extinguisher). b) Miss-match caused by similarities in the furniture (all the offices in this building have the same furniture).



(a) (b) (c)

Figure 5.3: Challenging situations for Visual SLAM. a) No visual cues can be detected in an image (lack of texture). b) An RGBD sensor detects the presence of many areas out of range during a normal robotics demonstration. c) A person blocks the robot vision while the robot is following him.

structures exist. In Fig. 5.2 we show two situations were Visual SLAM failed in our tests due to the reasons mentioned before. In addition, indoor spaces sometimes lack of sufficient texture to extract relevant features (Fig. 5.3-(a)). 3D information provided by RGBD cameras or 3D lasers can be helpful, but RGBD cameras are hard to use in wide open environments due to their limited depth range (Fig. 5.3-(b)) and the cost of 3D lasers might be prohibitive depending on the application.

Finally, the presence of people distorts the measurements of all these sensors significantly. For instance, people walking around the robot can block the range measurements of a laser or the vision of a camera and difficult the localisation (Fig. 5.3-(c)). Additionally, dynamic and crowded environments are still an active area of research in Visual SLAM techniques [74, 68]. For instance, in crowded environments Visual SLAM algorithms might not be able to extract significant visual cues due to the presence of people close to the camera, or even worse, they might extract visual cues from humans and fail [66].

Other works achieve localisation using external elements. For instance, wireless positioning systems use the signal received from Bluetooth beacons or WiFi Access Points to estimate the position of the robot [75, 76, 77, 78, 79, 80]. The accuracy of these systems is lower than the one obtained with the strategies based on range finders or cameras. In compensation, they rarely give estimates far away from the real position, which is common when laser or camera-based systems get lost (e.g. due to environment similarities). It is known that factors such as the presence of people, changes in humidity and even changes on the environment furniture affect wireless signals [81, 82, 83, 84, 85, 86]. However, in these situations these localisation systems usually experiment a drop on accuracy [84], but not to the point of leading to a completely lost robot. Consequently, wireless localisation algorithms could help to provide a coarse estimate of the robot position, while range or camera-based sensors could refine the estimation.

Other authors have relied on sensor networks able to measure directly the position of the robot. For instance, Corke et al. [87] deployed a network of 54 Mica motes and used a flying robot with a GPS to localise the motes on the network. Then, the motes would be used to localise and guide the robot. In addition, Brscic et al. [88] used a network of 49 3D range sensors (time-of-flight and RGBD cameras, and 3D lasers) and were able to track both robots and persons in a 900 m^2 space. This approach can localise the robot with high accuracy, but requires a high number of expensive hardware devices and is not exempt from problems such as occlusions caused by people.

The use of camera sensor networks is more common. Typically, the position and/or field of view of each camera is computed first (this stage is usually known as camera calibration), and then the camera network can localise the robot. Camera calibration can be performed by extracting visual information from moving objects (e.g. persons) or by detecting common features among overlapping fields of view [89]. In addition, mobile nodes [90] and even robots [91, 52, 92] can be used to locate the position of the cameras, and then the camera network can localise the robot accurately. For instance, in [52] Rekleitis et al. were able to calibrate a network of 7 cameras in a 450 m^2 environment. They recognised that additional sensing would be needed in areas not covered by the cameras, and the presence of people was not considered in these works.

Most previous works did not consider the benefits of using several sensors for mobile robot localisation. In contrast, Zhang et al. [93] have presented a sensor fusion strategy for line-based SLAM (Simultaneous Localisation and Mapping) with a monocular camera and a laser range-finder. They demonstrated that the fusion of both sensors improved the results with respect to the use of individual sensors. Biswas et al. [94] analysed the strengths and weaknesses of sensors such as a laser range-finder, a depth camera, and WiFi. They showed that no single sensor is universally superior for localisation, but it is possible to bound the localisation error by selecting different sensors for different areas of the map. Gamallo et al. fused a low cost GPS and visual information using a particle filter for outdoor robot localisation. They demonstrated that both sensors were complementary and that their sensor fusion produced significantly better results than when considering each sensor individually [95]. Quigley et al. [96] reached similar conclusions (sensor fusion is essential) when trying to perform localisation with WiFi and computer vision of a mobile device carried by an user.

These works have only combined the information of on-board sensors, assuming self-contained, stand-alone robots which do all the sensing, deliberation and action selection on board, based only on their own perceptions. However, recent advances in Ubiquituous Robotics [14] and Networked Robots [97] have demonstrated the benefits of coordinating robots and sensor networks. In this regard, Lee et al. [17, 16] suggested systems of distributed sensors (typically cameras) to support robots' navigation in small spaces (e.g. two cameras in less than 30 m^2). Similarly, the MEPHISTO project built 3D models of the environment using highly-coupled cameras, and then used these models for path planning and robot navigation [18]. On the other hand, Fernandez et al. [23] used 4 infrared cameras to locate and guide a robot within a small environment. Likewise, Losada et al. [22] used an external array of cameras for 3D robot localisation and tracking in a $8 \times 8\, m^2$ environment. These works have demonstrated the feasibility of robot localisation using external cameras in small spaces. Regretfully, these systems do not scale well with the size of the environment, due to the high density of sensors required. Moreover, these works did not address the fusion of sensors of different nature (in-board and off-board sensors), lacking some of the benefits of multi-sensor fusion.

The closest works to our philosophy are the Japan NRS project and the URUS project. The NRS project deployed networked systems of robots and environmental sensors in large real field settings, such as science museums [27], and shopping malls [28]. They performed robot localisation using external infrared cameras [27], or external laser range-finders [28], but it is not clear whether they use on-board sensors as well. On the contrary, the URUS project [31] used a particle filter for robot localisation with a 2D and a 3D laser range finder, a compass, and a GPS [33], and they have evaluated this proposal mainly outdoors. They also suggested to include an external network of cameras, but unfortunately they validated this proposal only in simulation. They recognised that the deployment and calibration of the cameras would be very challenging in practice [33].

5.3 Preliminary study: complementary sensors for robot localisation

In this section, we will study different strategies that can be used in mobile robot localisation, using either commercial solutions or open software. Our objective is to gather a first vision of how different sensors or systems can complement each other to increase the precision and robustness of localisation strategies, before implementing our own multi-sensor localisation system. To this extent, we will analyse the use of:

1. 2D laser based probabilistic localisation. A 2D laser range finder provides a set of range measurements on a 2D plane among the robot and nearby objects. In order to infer the position of the robot, this information can be matched against an occupancy map of the environment (a grid where each cell may correspond with free or occupied space). For instance, in Fig. 5.4 it is clear that the robot can only be somewhere in a long and narrow corridor. This information can be easily exploited by probabilistic localisation techniques such as those explained in the previous section to provide estimations of the pose of the robot. More specifically, we will use the Adaptive Monte-Carlo Localisation algorithm (AMCL) [98] (provided by Player/Stage).

2. WiFi localisation. From the wide range of wireless location systems available [77], we have decided to use WIFI location systems. These systems enable our robots to infer

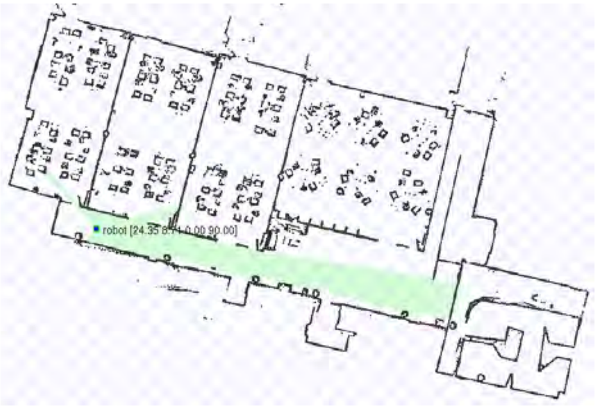


Figure 5.4: Representation of an occupancy map, a robot (blue square) and its 2D laser signature (green area).

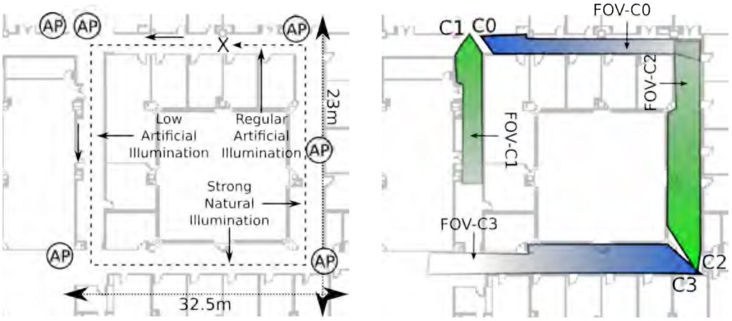
their position (not orientation) by measuring the signal received from WIFI Access Points. More concretely, we will use the Ekahau WIFI localisation system. This is a commercial system that has an accuracy of 1 meter in the best case [77] (allegedly).

3. Compass. A magnetic compass provides the orientation of the robot with respect to the Earth's magnetic north. This can be used as a complement of systems that do not provide orientation information (e.g. WiFi localisation), or to refine the estimation of those who do. In this study, we will use the CMPS10 Devantech compass.

4. Multi-camera network. If we know a correspondence between the FOV of view of a camera and the reference system of the robot, we can use the robot detections by the camera as a localisation source. Therefore, we will use our multi-camera network and a robot with active markers to perform this study.

We have performed several experiments at the Department of Electronics & Computer Science, at the University of Santiago de Compostela (Spain). We have deployed a system of four cameras, one robot (Pioneer P3DX with SICK-LMS100 laser), and 6 WiFi APs. In Fig. 5.5-(a) we show a map of the building, and we provide a brief description of its illumination conditions. Moreover, we show the position of each WiFi AP. Similarly, in Fig. 5.5-(b), we show the position of the cameras (C0 to C3) and their FOVs.

During the system "deployment", we moved the robot along the trajectory represented in Fig. 5.5-(a) as a dotted line: starting from the position tagged as X, and going along the



(a) (b)

Figure 5.5: Representation of the experimental setup for the preliminary robot localisation study. a) Map of the building with a description of its illumination conditions and the positions of the WiFi APs. b) Map of the building with the position of the cameras (C0 to C3) and their FOVs.

dashed arrow. Then, we constructed a map from the laser and odometry information using Simultaneous Localisation And Mapping techniques (SLAM) [51]. After that, we calibrated the Ekahau location system with the recorded data.

We configured each information source to send data at different rates.

- Cameras: 1 time per second at most. Each camera sends information to the robot (via the wireless network) when it detects it.
- AMCL (probabilistic localisation): 1 time per 100 milliseconds.
- Ekahau tags (WIFI location): 1 time per second at most. In practice the robot receives one location reading each 3 to 5 seconds, due to network delays, communication drops, etc.
- Compass: 1 time per 100 milliseconds.

After the "deployment" phase, we performed several tests. In each test, we injected into AMCL the initial pose of the robot, to ensure a proper initial localisation. Then, we moved the robot around the environment at an average speed of $0.4m/s$, following different trajectories along the same corridors shown in Fig. 5.5-(a) (either in clockwise or counter-clockwise directions). Meanwhile, we recorded each detection of the robot from the cameras, the Ekahau location positions, the compass readings, and the succession of poses estimated by the AMCL algorithm.

5.3.1 Robot localisation tests

Figures 5.6 to 5.10 represent five representative robot localisation tests. In each figure, from left to right, we represent: the AMCL positions of the robot, its Ekahau positions, and its positions when each camera detected it (AMCL and Ekahau positions). For the sake of clarity, we down-sampled the AMCL readings by a factor of 10, so we represent 1 AMCL reading per second. Below, we describe each experiment in detail.

In the experiment in Fig. 5.6, we moved the robot clockwise. It is clear that AMCL, Ekahau, and the cameras localised the robot smoothly. It is also clear that Ekahau provides less data than AMCL, since its data rate is lower. In addition, we observe that the periodicity of Ekahau is unstable: for example, the number of Ekahau location readings is significantly lower in region A than in the rest of the experiment. The explanation is simple: the number of communication failures increased there, primarily between the robot tag and the Ekahau server. On the other hand, we observe that Ekahau fails to localise the robot in some regions (e.g. B and C), even if no communication failures were observed. This underlines another Ekahau weakness: its uncertainty, which can go up to 5 meters.

In the experiment in Fig. 5.7, we also moved the robot clockwise. In this experiment, AMCL performed well during the whole experiment. On the other hand, Ekahau failed to localise the robot in region A: this was due to communication failures. Particularly remarkable is the fact that cameras C3 and C2 failed to detect the robot (regions B and C respectively). At that time, bright sunlight was coming from the windows (see Fig. 5.5-(a)), which reduced the apparent intensity of the LED marker as seen from the cameras. As a result, the cameras C2 and C3 were unable to detect the robot. This highlights the importance of working with complementary information sources: in this case, if we only relied in the cameras, the robot would be lost.

In the experiment in Fig. 5.8, the robot movement was counter-clockwise. In this experiment, Ekahau performed poorly: it provided very few location readings in regions A and B, and almost no readings in regions C and D. Looking at the high number of robot detections from the cameras, it is clear that they were able to communicate at all times. On the other hand, we checked that the robot was able to communicate with the Ekahau server as well. Thus, the Ekahau tag alone failed again to communicate its position to the server, or Ekahau failed to provide any estimation. Again, this shows that a single information source alone is not enough to deal with all the possible situations.

In the experiment in Fig. 5.9, the robot movement was counter-clockwise again. We would like to emphasise two facts. First, the fact that AMCL got temporarily lost in region A. This was caused by a small error in the initialisation of AMCL: the initial pose was 0.5 meters away from the robot real position. This shows that AMCL performance depends very strongly on the precision of its initialisation, but it also shows that AMCL is able to recover from this error. Finally, in Fig. 5.9-(d) we can see the big delays suffered by Ekahau. For example, when the camera C0 detects the robot in region C, Ekahau considers that the robot is still at region B: in this case, the robot tag informed the Ekahau server about its position with a delay of 8 to 10 seconds.

The last experiment is represented in Fig. 5.10. Here, we moved the robot clockwise, but we initialised ACML as if the robot were going to be moved counter-clockwise: in other words, the initial AMCL orientation had an error of 180 degrees. As we can see, AMCL localisation was incorrect during the whole test: while the real robot path starts and ends at region A, AMCL estimated that the path started out of the corridors (region B), and ended at region C, 20 meters away from the real end point. On the contrary, Ekahau and the cameras performed reasonably well: both could have helped AMCL to recover from the initial error.

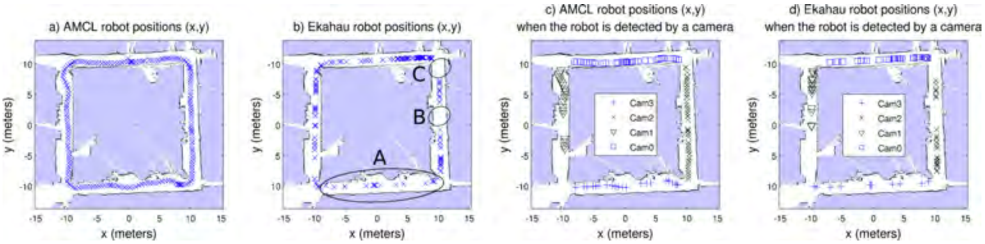


Figure 5.6: Test number 1 in the preliminary study for robot localisation.

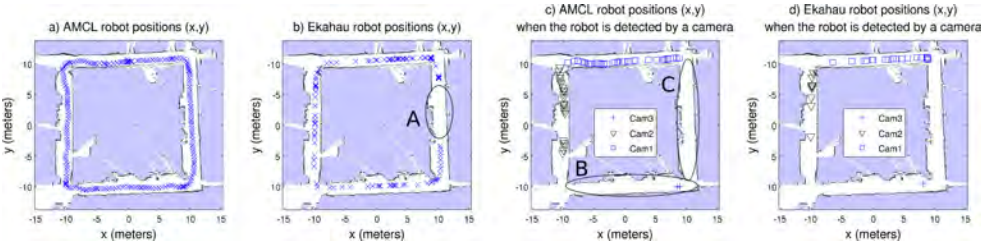


Figure 5.7: Test number 2 in the preliminary study for robot localisation.

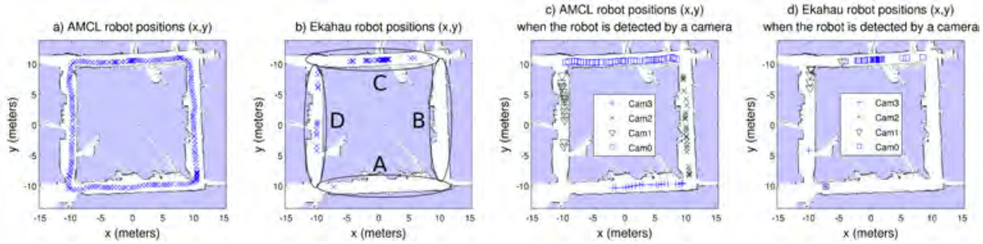


Figure 5.8: Test number 3 in the preliminary study for robot localisation.

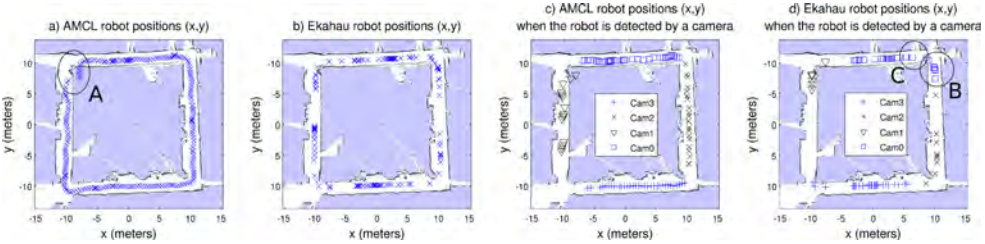


Figure 5.9: Test number 4 in the preliminary study for robot localisation.

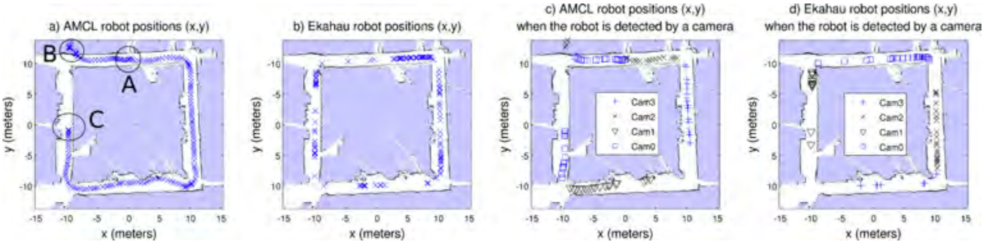


Figure 5.10: Test number 5 in the preliminary study for robot localisation.

Correspondence between map and camera-robot detections

From the Figs. 5.6-(c,d)-5.10-(c,d), we can deduce that it is possible to find a correspondence between the FOV of each camera and the area of the map that it covers. This could be done by matching the camera detections with the AMCL and Ekahau localisation information. On the other hand, the position of each camera can be retrieved by placing an Ekahau location tag in each camera box. We have explored this last option: in Fig. 5.11, we show the successive camera positions provided by Ekahau during one experiment. Instead of obtaining a single position, we obtain a cloud of positions, due to the inherent uncertainty of Ekahau. Besides robot localisation, this information can be used to establish robust neighbourhood relationships among the cameras.

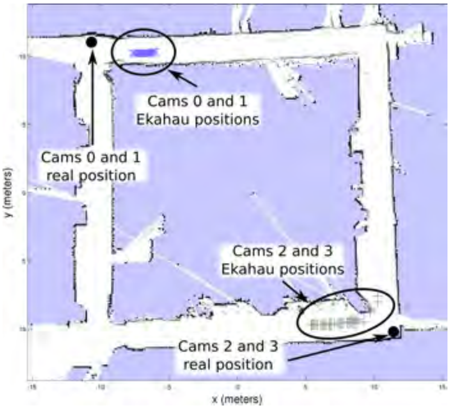


Figure 5.11: Successive positions received from the Ekahau tags attached to the cameras.

Compass

We have also recorded the robot orientation readings provided by the magnetic compass. We represent them in Fig. 5.12, as well as the robot orientation provided by the AMCL algorithm. Figure 5.12-(a) represents a counter-clockwise trajectory (initial orientation is 0°), and Fig. 5.12-(b) represents a clockwise trajectory (initial orientation is 180°).

The robot trajectory was square-like, as in Fig 5.5-(a). Thus, at each corner, the robot should turn 90°, and in the end, the robot should have turned 360° in total. For this reason, in Fig. 5.12-(a) we represented five regions of interest, A, B, C, D, and E, where the average orientation of the robot should be around 0°, 90°, 180°, 270°, and 360°, respectively. Similarly,

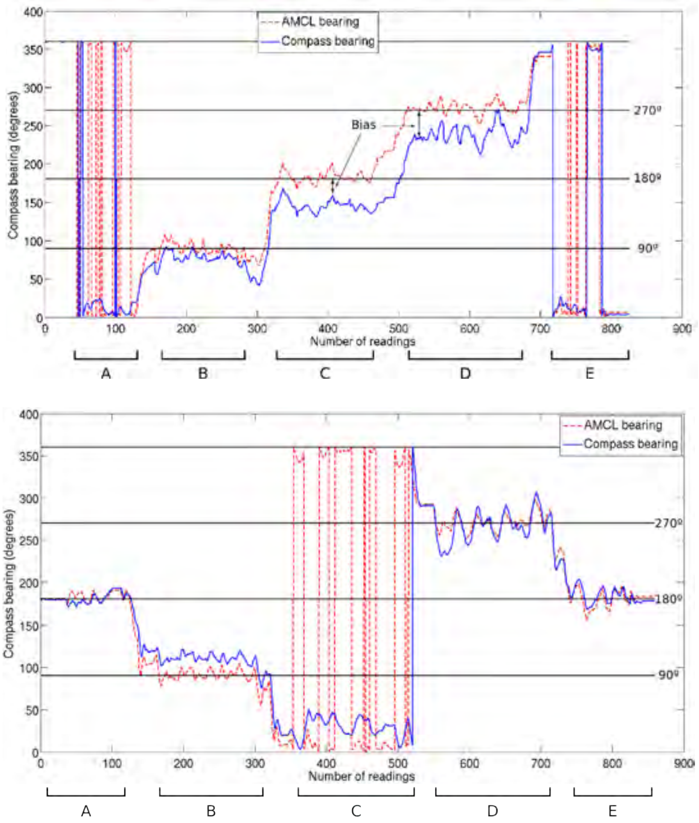


Figure 5.12: Robot orientation provided by AMCL and the magnetic compass in two different tests. The orientation range is $[0°, 360°)$. Therefore, when the orientation is 359° and the robot turns left, the orientation value becomes 0°. Similarly, when the orientation is 0° and the robot turns right, the orientation value becomes 360°. a) The navigation direction was counter-clockwise. b) The navigation direction was clockwise.

in Fig. 5.12-(b), the average orientation of the robot should be around 180°, 270°, 360°, 90°, and 180° for regions A, B, C, D, and E, respectively. In absence of a more reliable ground truth, this can give an insight about the accuracy of the orientation estimations. Note that the robot movement was squiggly (Figs. 5.6 to 5.10), what explains the oscillations on the orientation measurements.

From Fig. 5.12, we draw three conclusions. First, the compass is affected by a constant bias, which causes an accumulative error. This bias is typically caused by the presence of ferrous objects near the compass: in our case, the metal structure of the robot. A simple compass calibration should mitigate this effect significantly. Second, even with this bias effect, the compass error is reasonably low in most cases. This would provide a sufficient accuracy to complement the Ekahau location system, and even to improve AMCL estimations when they are especially erroneous (like in Fig. 5.10-(a)). Third, except for the bias effect, the shape described by both orientation measurements (AMCL and compass) looks alike, which indicates that other ferrous/magnetic elements, such as motors, electric machines, or electric cables, did not affect significantly the performance of the compass. Nevertheless, we are not very confident on this, since in other tests we have observed that objects such as vending machines or data centre rooms cause magnetic interferences that induce important errors.

5.3.2 Conclussions of the preliminary study of robot localisation

From this study, we can draw several conclusions. First of all, we have observed that Ekahau accuracy is not very high, its measurements arrive with big delay, and any communication issue affects it strongly. In fact, we have observed that the precision and robustness of Ekahau is very unsatisfactory, despite of being regarded as one of the most precise nowadays [77, 76]. Moreover, the Ekahau calibration was tedious, time consuming and error prone, because it required us to move all around the environment constantly informing the Ekahau computer of its position. For these reasons, we have decided to develop our own WiFi localisation system.

More importantly, we have observed that no information source is reliable in all the situations. For example, the cameras can not be used in the presence of direct sunlight, and AMCL performance depends heavily on the quality of its initial pose estimate. With respect to the magnetic compass, we have seen how ferrous objects near it may cause measurement drifts. On the other hand, Ekahau has several drawbacks as well, as we have already explained. Table 5.1 summarises how different situations may affect the performance of each sensor.

On the other hand, we have seen the nature of the errors that affect each information source differs from the rest. Moreover, as we summarise in Table 5.2, each sensor has different properties that make it suitable for situations, but not for others. For instance, WiFi and cameras work at low data rates and do not give precise estimates of the robot position. This makes them unable to respond rapidly to changes on the robot pose or the environment, therefore they are not advisable for local localisation (tracking). However, they demonstrated to be very reliable in telling roughly the area were the robot is. Therefore, they can be used for global localisation. On the other hand, laser and compass work at higher data rates and provide very accurate

	Camera	WiFi	Laser-Map	Compass
Oclussion	No	Yes	No	Yes
Out-of-FOV	No	Yes	Yes	Yes
Illumination	No	Yes	Yes	Yes
Communication	No	No	Yes	Yes
EM interferences	Yes	No	Yes	No

Table 5.1: Is each sensor robust to situations such as occlussion, robot out-of-FOV, severe illumination conditions, communication failures and electro magnetic interferences?

	Camera	WiFi	Laser-Map	Compass
Data rate	$\leq 1Hz$	$\leq 1Hz$	$\geq 1Hz$	$\geq 1Hz$
Rough global localisation	Yes	Yes[1]	No	Yes[2]
Precise local localisation	No	No	Yes	Yes[2]

Table 5.2: Summary of relevant properties of each sensor for mobile robot localisation. [1] Position only, [2] orientation only.

measurements, so they can respond rapidly and accurately to changes on the robot pose or the environment. Therefore, they can be used for local localisation (tracking). However, laser can not distinguish areas with similar geometric structure (e.g. corridors), no matter how far they are. On the other hand, the compass does not provide position estimates, although the information it provides can be relevant both to improve the accuracy of the local localisation estimates, and to reduce the uncertainty when the robot is operating in global localisation.

From this considerations, it seems that each source can complement each other to achieve a more robust and redundant system. In the next section, we will propose and validate a localisation methodology based on the probabilistic fusion of several independent sensors.

5.4 Multisensor localisation algorithm

In the previous section, we have performed an experimental study whose main conclusion was that the use of complementary sensors could improve the robustness and precision of localisation systems. In this section, we will describe and validate an algorithm that is able to combine the information of several information sources to solve the task of mobile robot localisation. Our localisation algorithm is aimed to be used in environments such as museums or office environments, where layout changes are not frequent. As we have already explained, we decided that the most functional approach was to build a map of the environment in a deployment stage, and then use this map during the operation stage. Our objective is not to propose yet another localisation algorithm, but to build a localisation system that is robust in real situations (e.g. people moving around the robot) by fusing the information of multiple sensors.

Most of the main works in mobile robot localisation use Bayesian filtering techniques [51]. These techniques maintain a probabilistic model of the state of a system which evolves over time and is periodically observed by a sensor (or sensors). From all the Bayesian filtering approaches, Kalman filters and particle filters are by far the most popular. Kalman filters work well when [63] : a) the robot motion is linear, b) the motion and sensor noises are white, Gaussian and can be modelled accurately, c) there is an explicit and unimodal mapping (observation model) between states and observations, and d) the best estimate of the state is unique (unimodal probability distribution over the state space). Some of these conditions do not hold in the context of robot localisation. For instance, the robot movement is usually non-linear, although this would not be a critical issue, given the low linear and turn acceleration that our robots can achieve. Moreover, Kalman filter variants such as the Extended Kalman Filter or the Unscented Kalman Filter [51] could be used. However, other restrictions are critical in our context. For instance, in practice it may be hard to model with Gaussian distributions the observation models and their noises. Above all, the most important problem is that observation models and therefore state probability distributions are usually multi-modal.

Particle filters [51] are a powerful yet efficient alternative to Kalman filters. Particle filters do not make any of the previous assumptions. First of all, they work with non-linear non-Gaussian systems with multi-modal probability distributions, where there is no explicit mapping between sensor observations and system states. This allows us to maintain several localisation estimates simultaneously, which is very useful when sensor readings are not sufficiently discriminative to estimate an unique position. Moreover, with particle filters the probabilistic mapping between sensor measurements and position candidates can follow any probabilistic distribution. This provides us with a great flexibility when modelling the localisation sensors. Moreover, these filters are very robust even if the system and sensor noises are poorly estimated [63]. Therefore, particle filters will be our choice in order to implement our localisation algorithm.

Our localisation system has to estimate, at every instant t, the pose of the robot (state) $s_t = (x_t, y_t, \theta_t)$ with respect to a map, where (x_t, y_t) represents the position in cartesian coordinates, and θ_t the orientation. Our system estimates this pose based on: 1) perceptual information $Z_t = \{z_t^1, ..., z_t^{n_t^z}\}$ (set of n_t^z sensor measurements available at time t), and 2) control data u_t (the robot movement as provided by odometry encoders). In addition, we will estimate $\Sigma_t(s)$ (the covariance of s_t). In order to accomplish our goal, our system iterates over a two-step process:

1. *Pose probability estimation* (Secs. 5.4.1-5.4.2). This step computes the pose probability distribution over all possible robot poses. This distribution is usually called the *belief distribution*, and it represents the belief that any possible pose refers to the actual current position and orientation of the robot. As the pose of the robot changes over time so does the belief distribution $bel(s_t)$.

2. *Pose estimation* (Sec. 5.4.2). Estimation of the most likely current pose s_t from the pose probability distribution $bel(s_t)$.

5.4.1 Recursive Bayes filtering to estimate the pose probability distribution

In the Bayesian filtering approach to dynamic pose estimation, the belief distribution is the posterior probability density function of the pose based on all the available information, i.e. the set of actions taken by the robot $u_{t:1} = \{u_t, u_{t-1}, ..., u_1\}$, and the set of received sensor measurements $Z_{t:1} = \{Z_t, Z_{t-1}, ..., Z_1\}$ [63, 99].

$$bel(s_t) = p(s_t|Z_{t:1}, u_{t:1}) \tag{5.1}$$

In our case, an estimate of the robot's pose is required every time that the robot performs an action and receives a set of sensor measurements. Computation of Eq. 5.1 requires to store the complete data set, and process it as a batch when new data becomes available. Instead, a recursive filter is a much more convenient solution: the received data can be processed sequentially, and previous data can be discarded. One can derive such filter by using the Bayes rule [63]:

$$bel(s_t) \propto p(Z_t|s_t, Z_{t-1:1}, u_{t:1})p(s_t|Z_{t-1:1}, u_{t:1}) \tag{5.2}$$

and the law of total probability [63]:

$$bel(s_t) \propto p(Z_t|s_t, Z_{t-1:1}, u_{t:1}) \int p(s_t|s_{t-1}, Z_{t-1:1}, u_{t:1})bel(s_{t-1})ds_{t-1} \tag{5.3}$$

The recursive Bayes filter in the context of robot localisation is also known as Markov localisation, because of what is called the Markov assumption [63]. This assumption states that past and future data are independent if one knows the current state (pose). In other words, provided the current state, past states or data are not relevant to future predictions. Therefore, the knowledge of s_{t-1} and u_t is enough to predict s_t:

$$p(s_t|s_{t-1}, Z_{t-1:1}, u_{t:1}) = p(s_t|s_{t-1}, u_t) \tag{5.4}$$

Similarly, the knowledge of s_t is enough to predict Z_t:

$$p(Z_t|s_t, Z_{t-1:1}, u_{t:1}) = p(Z_t|s_t) \tag{5.5}$$

Therefore Eq. 5.3 can be rewritten as:

$$bel(s_t) \propto p(Z_t|s_t) \int p(s_t|s_{t-1}, u_t) bel(s_{t-1}) ds_{t-1} \tag{5.6}$$

This is the general form of the recursive Bayes filter, represented in Fig. 5.13. Initially, the first belief distribution $bel_0(s)$ may be initialised randomly, e.g. following an uniform distribution. Then, the filter performs iteratively in two stages: *prediction* and *update*.

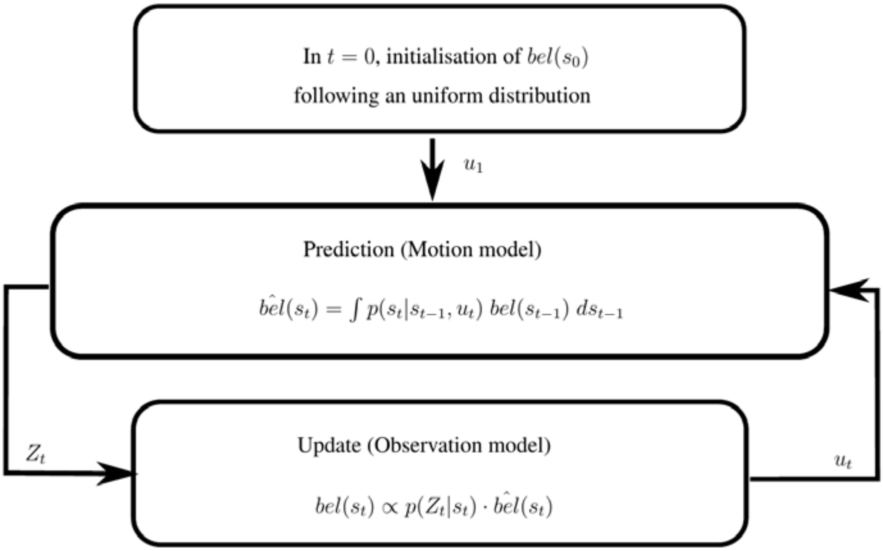

In $t = 0$, initialisation of $bel(s_0)$
following an uniform distribution
u_1
Prediction (Motion model)
$\hat{bel}(s_t) = \int p(s_t|s_{t-1}, u_t)\, bel(s_{t-1})\, ds_{t-1}$
Z_t
Update (Observation model)
$bel(s_t) \propto p(Z_t|s_t) \cdot \hat{bel}(s_t)$
u_t

Figure 5.13: Block diagram of the recursive Bayes filter for mobile robot localisation.

Prediction

The prediction stage predicts a new belief distribution $\hat{bel}(s_t)$ of the pose of the robot from one instant to the next one, considering the current actions taken by the robot u_t (robot movement) and the previous distribution $bel(s_{t-1})$.

$$\hat{bel}(s_t) = \int p(s_t|s_{t-1}, u_t)\; bel(s_{t-1})\; ds_{t-1} \tag{5.7}$$

The term $p(s_t|s_{t-1}, u_t)$ is called the *motion model* of the robot. It describes, from a probabilistic perspective, the evolution of the pose when only the actions taken by the robot are considered. This model depends on the specific kinematics of the robot (i.e. how much its position will evolve considering the angular and linear velocities performed).

Update

The update stage uses the latest sensor measurements Z_t to correct the belief distribution previously predicted $\hat{bel}(s_t)$, producing the true posterior distribution $bel(s_t)$:

$$bel(s_t) \propto p(Z_t|s_t) \cdot \hat{bel}(s_t) \tag{5.8}$$

The function $p(Z_t|s_t)$ is called the *observation model*: it specifies the probability of receiving a certain measurement Z_t provided that the robot is in a certain pose s_t. Therefore, Eq. 5.8 corrects $\hat{bel}(s_t)$ based on the similarity between the sensor data received Z_t and the sensor data expected to be received from each possible pose s_t. Under this approach, the most probable poses will be those with the highest similarity between expected and received sensor data.

The joint probability function $p(Z_t|s_t)$ is hard to estimate in practice, specially if we have non-synchronised sensors that work at different rates. Assuming that the sensors are conditionally independent provided s_t:

$$bel(s_t) \propto \hat{bel}(s_t) \prod_{k=1}^{n_t^z} p(z_t^k|s_t) \tag{5.9}$$

Note that each observation model depends on the specific sensor that it represents (we will explain in Sec. 5.5 how these models are obtained). Moreover, note that at each instant t we only take into account Z_t, the set of n_z^t readings received at that instant. In other words, the algorithm does not require a reading from every sensor at each update. Therefore, the sensors do not need to be synchronised, because the algorithm will integrate their information when it is available. For the same reason, the sensors do not need to have the same data rate. However, in practice it is a good idea to not allow a sensor to provide data much faster than the rest (e.g. 10 times faster), because it would dominate the fusion. Finally, since Eq. 5.9 allows us to have a variable number of sensors, the algorithm will continue working even if some sensors stop providing data. Moreover, we can include new sensors in the fusion process even if the robot is in operation. All these situations are very challenging but our system handles them properly, as we will see in Sec. 5.7. This ensures the robustness and scalability of the fusion methodology.

5.4.2 Implementation of the recursive Bayes filtering with particle filters

The problem of the Bayesian filtering described before is that in practice it is not straightforward to compute $bel_t(s)$. However, there exist several methods to approximate probability density functions (PDFs) as *mixture models*, that is, as the weighted sum of a finite number of simple distributions, usually known as kernels. One popular method is the Sequential Importance Sampling (SIS) algorithm (also known as *particle filter*) [63, 99, 100, 101], which implements the recursive Bayesian filter described above using sequential Monte Carlo simulations.

This method is based on the following idea. Assume that, at time t, we have a set of n^p *samples* from $bel(s_t)$ (usually called *particles*). This sample set will be represented as $P_t = \{P_t^i = (s_t^i, \omega_t^i),\ \forall\, i \in \{1, ..., n^p\}\}$, where s_t^i is the pose of the sample, and ω_t^i a weight associated to the sample (the sum of the weights must add up to 1). In this case, we could approximate $bel(s_t)$ as [100, 101]:

$$bel(s_t) \approx \sum_{i=1}^{n^p} \omega_t^i \cdot \delta(s_t^i - s_t) \tag{5.10}$$

where $\delta(s_t^i - s_t)$ is the Dirac's delta function centred at s_t^i. Note that the more weight the particle has, the more likely its pose is. Unfortunately, we can not easily take samples from the marginal distribution $bel(s_t) = p(s_t|Z_{t:1}, u_{t:1})$ [100]. Instead, we will sample from $p(s_{t:0}|Z_{t:1}, u_{t:1})$ (*target distribution*), and ignore the samples $s_{t-1:0}$. The samples are chosen using the principle of *importance sampling* [100, 101]: since the target distribution is unknown, we will sample from a *proposal distribution* $q(s_{t:0}|Z_{t:1}, u_{t:1})$ defined by us (from which it is easy to take samples) and weight each sample i according to:

$$\omega_t^i = \frac{p(s_{t:0}^i|Z_{t:1}, u_{t:1})}{q(s_{t:0}^i|Z_{t:1}, u_{t:1})} \tag{5.11}$$

Assuming that the following factorisation is possible (Markov assumption):

$$q(s_{t:0}|Z_{t:1}, u_{t:1}) = q(s_t|s_{t-1}, Z_t, u_t)q(s_{t-1:0}|Z_{t-1:1}, u_{t-1:1}) \tag{5.12}$$

it can be shown that we can compute Eq. 5.11 recursively in terms of ω_{t-1}^i [100, 101]:

$$\omega_t^i \propto \frac{p(Z_t|s_t^i)p(s_t^i|s_{t-1}^i, u_t)}{q(s_t^i|s_{t-1}^i, Z_t, u_t)} \omega_{t-1}^i \tag{5.13}$$

Therefore, at time t we can simply sample a new pose for each particle from the proposal distribution $q(s_t|s_{t-1}, Z_t, u_t)$. A convenient choice for this function is the motion model of the robot:

$$q(s_t|s_{t-1}, Z_t, u_t) = p(s_t|s_{t-1}, u_t) \tag{5.14}$$

Consequently:

$$\omega_t^i \propto \omega_{t-1}^i p(Z_t|s_t^i) \tag{5.15}$$

With this in mind, our particle filter performs as follows. First of all, a set of particles P_0 is generated with random poses s_0^i and equal weights ω_0^i. Then, the algorithm iterates over the following steps:

1. Prediction. With each robot displacement u_t, we sample a new pose s_t^i for each particle using the robot motion model:

   $$s_t^i \sim p(s_t|s_{t-1}^i, u_t) \quad \forall\, i \in \{1, ..., n^p\} \tag{5.16}$$

   In practice, this can be interpreted as a displacement of each particle as indicated by u_t, plus some random noise in position and orientation to accommodate the error of the odometry of the robot (see [63] for details and other options).

2. Update. We correct the weight of each particle with Z_t (sensor information available at time t):

   $$\omega_t^i \propto \omega_{t-1}^i \; p(Z_t|s_t^i) = \omega_{t-1}^i \prod_{k=1}^{n_t^z} p(z_t^k|s_t^i) \tag{5.17}$$

   This way, each particle will integrate the past sensor information (using the previous weight) and the fusion of the current sensor information. The weights are normalised to sum up to one.

Resampling

The particle filter algorithm suffers from the *sample depletion* phenomenon [101, 102]: after a while, all but one particle will have negligible weights. The degree of depletion can be defined as the effective number of effective particles [101, 102]:

$$N_{eff} = \frac{1}{\sum_{i=1}^{n^p} (\omega_t^i)^2} \tag{5.18}$$

The maximum number of effective particles is $N_{eff} = n^p$, when all weights are equal ($\omega_t^i = 1/n^p$, lower degree of depletion). The minimum number of particles is $N_{eff} = 1$, when only one particle accumulates all the weight (this is undesirable, highest degree of depletion).

In order to correct the depletion phenomenon, it is common to perform a resampling step when the number of effective particles falls below a certain threshold value (e.g. $\frac{2}{3}n^p$). The resampling step consists on the construction of a new set of particles from the current one. First of all, we take n^{nr} particles from the current set with probability proportional to their weight using the Low Variance Resampling technique [63] (we may repeat particles). Secondly, we generate a variable number n^r of random particles ($n^{nr} + n^r = n^p$). The generation of random particles allows the algorithm to recover when it converges erroneously to a wrong pose. We calculate n^r by keeping a long term average ω_t^l and a short term average ω_t^s of the weight of the particle set:

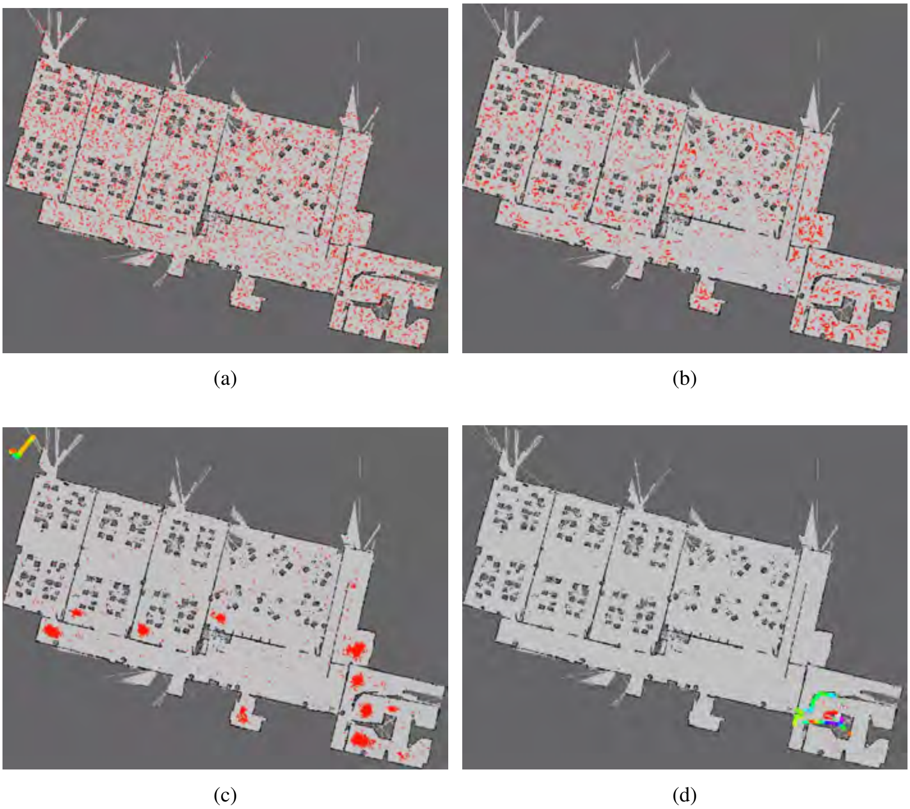
$$\omega_t^l = \omega_{t-1}^l + \alpha^l \cdot \left(\frac{1}{n^p} \sum_{i=1}^{n^p} \omega_t^i - \omega_{t-1}^l \right) \tag{5.19}$$

$$\omega_t^s = \omega_{t-1}^s + \alpha^s \cdot \left(\frac{1}{n^p} \sum_{i=1}^{n^p} \omega_t^i - \omega_{t-1}^s \right) \tag{5.20}$$

$$n^r = \left\lceil n^p \cdot max \left(0, 1 - \frac{\omega_t^s}{\omega_t^l} \right) \right\rceil \tag{5.21}$$

where $0 < \alpha^l << \alpha^s < 1$. The ratio ω_t^s / ω_t^l estimates whether the quality of the particle set is increasing (increasing ratio) or decreasing (decreasing ratio). Therefore, the more this ratio decreases, the more random particles we generate.

The impact of the resampling step can be seen in Fig. 5.14. In Fig. 5.14-(a), the particles were initialised randomly. Then, the robot started to move and to integrate information, going through cycles of *prediction* and *update* steps. This lead to particles with different weights, and therefore to several resampling steps. Figs. 5.14-(b-c) show the evolution of the cloud of particles after the occurrence of succesive resamplings. Observe that the algorithm tends to



(a)

(b)

(c)

(d)

Figure 5.14: Particle filter evolution. a) Initially, all the particles are distributed randomly across the environment. b-c) Soon, particles start to concentrate around the most problable areas (given a number of sensor readings and robot displacements). d) Finally, most particles converge around the most probable pose (bottom-right part of the image).

accumulate particles in the areas where the robot might be with the highest probability, until it converges to the true robot pose (Fig. 5.14-(d)).

Pose estimation

At this point, we have computed $bel(s_t)$, and we need to estimate the most likely pose s_t:

$$s_t = \underset{s}{\operatorname{argmax}}\{bel(s_t)\} \tag{5.22}$$

In our case, we have to estimate this pose from the particle set P_t. There are many other methods to achieve this: a weighted mean over all the particles is one of the most popular [63]. However, in some cases methods based on simple analytics like that one might deal strange results. For instance, when we have two or more bulks of particles with similar weights in different positions, a weighted mean would estimate that the correct position is between these two sets, in the region of lowest probability.

To solve these issues, we propose a clustering based process. The key idea is that most particles will concentrate in few regions (the most likely regions), therefore we should be able to detect pose hypotheses (clusters of particles), and select one of them. To this extent, we perform the following two-step process:

1. Hypothesis generation. First of all, we use agglomerate clustering [103] to group the particles in clusters. Each particle will be assigned to its closest cluster, provided that the distance between the particle and the cluster centroid is lower than a threshold distance in position $dist_{th-xy}$ and in orientation $dist_{th-\theta}$. Otherwise, the particle will create a new cluster. Each cluster i can be interpreted as an hypothesis about the position of the robot. We will represent each hypothesis as $H_t^i = \{\Omega_t^i, \mu_t^i, \Sigma_t^i\}$, where Ω_t^i is the total weight of the particles contained in the cluster i, μ_t^i is the average pose of the particles of the cluster, and Σ_t^i is their covariance matrix. The influence of each particle on these two statistics (mean and covariance) is proportional to its weight.

2. Hypothesis selection. We will select the hypothesis that accumulates more weight, provided that this accumulated weight exceeds certain threshold: $\Omega_t^i > \Omega_{th}$ ($\Omega_{th} \in [0.5, 1]$). Then, the robot pose s_t will be μ_t^i and the covariance $\Sigma_t(s)$ will be Σ_t^i. In the next step, this hypothesis will be chosen again if $\Omega_t^i > 1 - \Omega_{th}$. This increases the stability of the hypothesis selection.

5.5 Sensor observation models

Our system fuses the information from all the sensors using their observation models (Eq. 5.17). In order to train the observation models, we must: 1) capture a number of measurements at different points of the environment (calibration data), and 2) build the training sets. This last step requires us to relate each measurement with the position where it was measured. There exist a number of alternatives to accomplish this. For instance, there are systems that provide the position of the robot at every instant (e.g. external infrared cameras for 3D marker tracking, such as Tracking Tools from Natural Point). However, these systems are extremely expensive and adequate only for small areas. As an alternative, an user may indicate where each measurement took place, but this method has some serious drawbacks: it is tedious, time consuming and error prone. Instead, we follow the procedure depicted in Fig. 5.15 and which can be summarised in the following steps:

1. Collect the calibration data by moving the robot around the environment. The conditions of the environment must favour the quality of the data (e.g. no people moving around the robot).

2. Process the collected data off-line.

    a) Build the occupancy map of the environment: a grid where each cell may correspond with free or occupied space. Figure 5.16 contains an example of an oc-

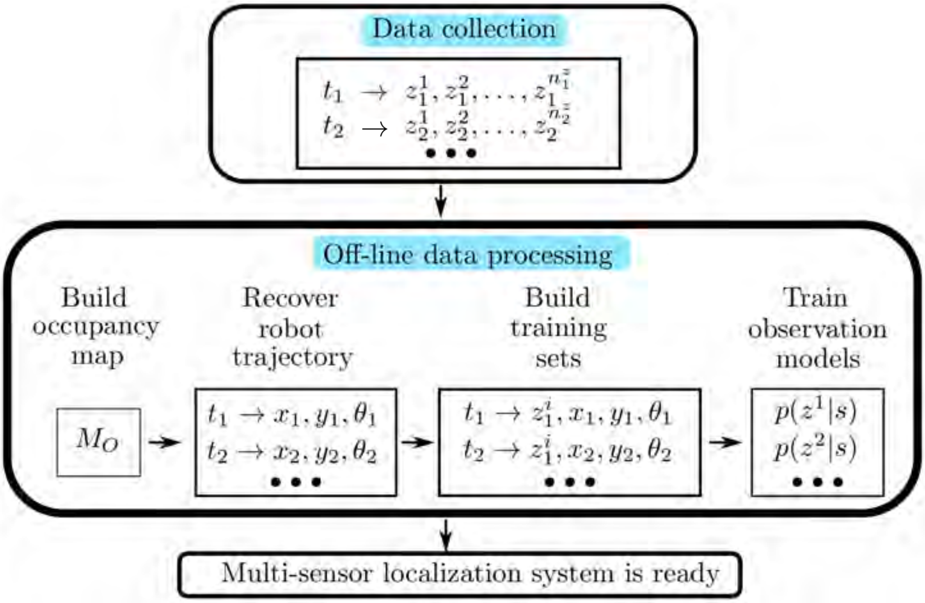

Data collection
$t_1 \rightarrow z_1^1, z_1^2, \ldots, z_1^{n_1^z}$
$t_2 \rightarrow z_2^1, z_2^2, \ldots, z_2^{n_2^z}$
• • •
Off-line data processing
Build occupancy map
Recover robot trajectory
Build training sets
Train observation models
M_O
$t_1 \rightarrow x_1, y_1, \theta_1$
$t_2 \rightarrow x_2, y_2, \theta_2$
• • •
$t_1 \rightarrow z_1^i, x_1, y_1, \theta_1$
$t_2 \rightarrow z_1^i, x_2, y_2, \theta_2$
• • •
$p(z^1|s)$
$p(z^2|s)$
• • •
Multi-sensor localization system is ready

Figure 5.15: Procedure to calibrate the observation models.

cupancy map. To construct this map we use an SLAM algorithm with laser and odometry information.

b) Compute the trajectory followed by the robot and associate each pose with its timestamp (x_t, y_t, θ_t).

c) Build a training set for each sensor. Each training set contains the data captured by a sensor associated with the corresponding pose $(z_t^i, x_t, y_t, \theta_t)$.

d) Train the observation model of each sensor $(p(z^i, s))$.

Our sensor fusion methodology allows us to use a variable number of sensors of different nature (we discuss some practical constraints in Sec. 5.6.3). To demonstrate the performance of this methodology in practice, we will use a 2D laser range-finder, an WiFi receiver, a magnetic compass, an external camera network, and a camera mounted on the robot. Hereafter, we will propose an observation model and a mapping strategy for each sensor. We follow a general methodology that allows us to calibrate the sensors with few data (fast deployment) avoiding overfitting, and that can be easily adapted to other sensors.

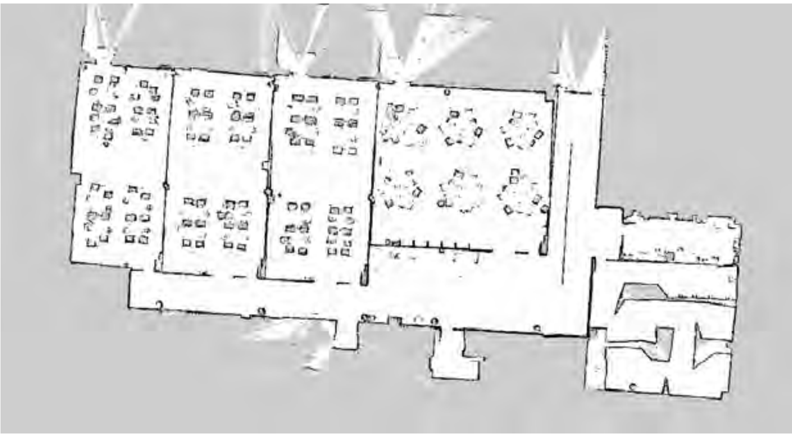


Figure 5.16: Example of a laser occupancy map M_O. White represents free space, black represents occupied space, and grey represents unknown space.

5.5.1 2D laser rangefinder - Observation model

At any time instant t, a 2D laser range finder provides a set $z_t^l = \{l_t^1, ..., l_t^{n^l}\}$ of n^l range measurements among the robot and the nearby objects. All ranges are on the same plane (typically, parallel to the ground), each one pointing on a different angle. This vector is usually called the laser signature. The signatures observed at different poses are usually different: therefore, we can compute the likelihood of a certain pose depending on how much the signature received resembles the one expected from that pose.

To this extent, we must build a map of *expected signatures*. We will refer to this map as the *laser ranges map* M_{LR}. The occupancy map M_O serves as a base to calculate M_{LR}. For each cell, we calculate a vector of distances between the cell and the nearest occupied cell (in several radial directions, with a certain angular resolution). We use M_{LR} to calculate the expected laser signature $\tilde{z}^l(s) = \{\tilde{l}^1(s), ..., \tilde{l}^{n^l}(s)\}$ at each possible robot pose according to the map. With this information, we approximate the laser observation model by the similarity between $\tilde{z}^l(s)$ and z_t^l:

$$p(z_t^l|s) = hd(\tilde{z}^l(s), z_t^l) \cdot td(\tilde{z}^l(s), z_t^l) \tag{5.23}$$

where $hd(\tilde{z}^l(s), z_t^l)$ estimates the shape similarity among both laser scans (measured and expected) using the Hellinger distance [104]:

$$hd(\tilde{z}^l(s), z_t^l) = \left[1 - \frac{\sum_{i=1}^{n^l} \sqrt{\tilde{l}^i(s) \cdot l_t^i}}{\frac{1}{n^l}\sqrt{\sum_{i=1}^{n^l} \tilde{l}^i(s) \sum_{i=1}^{n^l} l_t^i}} \right]^{1/2} \tag{5.24}$$

and $td(\tilde{z}^l(s), z_t^l)$ takes scale into account by calculating the average difference among each pair of range measurements $(\tilde{l}^i(s), l_t^i)$, normalized in the range $[0,1]$ using a triangular function:

$$td(\tilde{z}^l(s), z_t^l) = \left[\frac{1}{n^l} \sum_{i=1}^{n^l} max\left(1 - \frac{|\tilde{l}^i(s) - l_t^i|}{max_{LD}}, 0\right) \right] \tag{5.25}$$

The parameter max_{LD} (maximum laser difference) indicates the maximum allowed difference among each pair of laser ranges.

5.5.2 WiFi - Observation model

There are several technologies that have been used in wireless indoor positioning [75, 76, 77], such as RFID, UWB, Bluetooth, TV, or GSM. However, WiFi positioning has been the most popular alternative by far. WiFi localisation is usually based on fingerprinting [105, 106]. Fingerprinting refers to techniques that: 1) on a calibration stage, collect features or *fingerprints* of the wireless signal and the location where they were measured to build a *radio map*, and 2) on a use stage, estimate the position of the receptor by matching online measurements with the *radio map*. Usually, the RSSI (Received Signal Strength Indication) features are used, which are related to the signal power received from the Access Points.

There are two basic kinds of radio-maps that can be constructed with fingerprinting techniques: model-based and empirical maps [105]. Model-based maps are defined by a set of parameters that specify the characteristics of the environment (e.g. walls, materials, etc.) and/or the characteristics of the signal propagation. These parameters are usually adjusted using calibration data. On the other hand, empirical methods (which tend to achieve better results [107, 108]), work directly with the fingerprints to build radio maps. There are two kinds of empirical maps: deterministic and probabilistic maps [105]. Deterministic maps assign a single value to each position of the map, such as a fingerprint in that position, or an average of the closest fingerprints. The most important drawback of these maps is that a single value can not capture the random nature of wireless signals. As a solution, probabilistic maps characterise the wireless signal at each position using probability distributions [106] (e.g. Gaussian, Log-Normal, Weibull, etc.).

On the other hand, almost any algorithm from the fields of machine learning and estimation could be used as a position estimation method. Usually, estimation methods are divided in two groups [105, 106]: deterministic and probabilistic. Deterministic methods estimate the location of the receptor directly from the value of the measurements received. Techniques such as Artificial Neural Networks [109, 110], Support Vector Machines [73, 111] and Nearest Neighbours and its variants [106, 112, 107] have been used to implement deterministic localisation techniques. On the other hand, probabilistic methods estimate the position of the device as part of a random process. Usually, they integrate the measurements sequentially, and exploit information about the movement of the device or about the topology of the environment. It has been shown that probabilistic techniques tend to have better results than deterministic ones [106]. Probabilistic methods are usually based on Bayesian inference [106], Hidden Markov Models [113], or particle filters [73, 114].

In this section, we will explore the use of empirical probabilistic maps and probabilistic algorithms for wireless positioning. We will capture the data that we will use to build the radio map with a WiFi card on-board the robot. This card is able to receive the signal power from the WiFi APs in the environment. This information can be represented as $z_t^w = \{w_t^1, ..., w_t^{n_t^w}\}$, where n_t^w is the number of APs available at time t, and w_t^i is the power in dBms of the i^{th} AP at time t. Using the data captured at the "data collection" stage (Fig. 5.15), we build a training set for each AP $D_i^w = \{\,(x_t, y_t)\,;\, w_t^i)\}$. Each sample of the training set consists on an output w_t^i (power of the i^{th} AP received from the scan at time t), associated with an input (x_t, y_t) (position where the scan took place).

This "fingerprinting" data set can be used to calibrate the WiFi observation models using several *supervised learning* techniques. Essentially, we want to know the probability of receiving a certain measurement z_t^w when the robot is in any position (x_t, y_t). In this thesis, we will explore different options in order to achieve this goal. In short, these options are:

1. Compute the most probable position $(\hat{x}_t^w, \hat{y}_t^w)$ for a certain measurement. This can be achieved with any regression technique, trained with the measurements as inputs and the positions where they were taken as outputs. Then, we can define the observation model as a function of probability around that position. In our case, we have tested the performance of the ε-*Support Vector Regression* technique.

2. Compute directly the probability of receiving a certain measurement z_t^w from every position (x_t, y_t). This can be achieved in two different ways:

   a) Estimating from training data the expected measurement values $\hat{z}_t^w$ at each position (x_t, y_t). This can be easily achieved with spatial regression techniques, trained with the positions were we have taken measurement as inputs, and the actual measurements as outputs. Then, we can compute the differences between $\hat{z}_t^w$ and z_t^w at each position, to define the observation model as a function of these differences. In this thesis, we have used the *Gaussian Processes Regression* technique (GP regression).

   b) Estimating, for each measurement in the training set, the density function of its occurrences in the whole space. This can be achieved with any density function estimation technique. The training input is a collection of measurements and the positions where they were taken, and the output the estimated density across space. Therefore, the observation model will be defined by these density functions: when a new measurement arrives, its probability will be related to the density of occurrence of that measurement at each position. In our case, we have used a technique based on *Gaussian Mixture Models*.

Computation of the most probable position: ε-Support Vector Regression

The first strategy we have tested to create the WiFi observation model consists on translating the Wi-Fi measurements z_t^w into an intermediate estimate of the robot's position $(\hat{x}_t^w, \hat{y}_t^w)$, and then obtain the measurement model from these estimations. In order to estimate $\hat{x}_t^w$ and $\hat{y}_t^w$, we use two regression functions f_x^w and f_y^w:

$$\hat{x}_t^w = f_x^w(w_t^1, ..., w_t^{n^w}) \tag{5.26}$$

$$\hat{y}_t^w = f_y^w(w_t^1, ..., w_t^{n^w}) \tag{5.27}$$

where n^w is the number of APs in the environment, and w_t^i is the power in dBms of the i^{th} AP at time t. In order to learn f_x^w and f_y^w, we have chosen the ε-Support Vector Regression technique with Gaussian Radial Basis Function kernels (ε-SVR-RBF) [115]. The prediction error of the ε-SVR-RBF can be approximated by a zero mean Laplace distribution [115], and therefore the *observation model* of our Wi-Fi sensor is defined by Eq. 5.28:

$$p(z_t^w|s) = \left[\frac{1}{2\sigma^w(x)} exp\left(-\frac{|\hat{x}^w - x|}{\sigma^w(x)}\right)\right]\left[\frac{1}{2\sigma^w(y)} exp\left(-\frac{|\hat{y}^w - y|}{\sigma^w(y)}\right)\right] \tag{5.28}$$

where $\sigma^w(x)$ and $\sigma^w(y)$ are the noise parameters, which can be estimated from training data.

Computation of expected values at each position: Gaussian Processes Regression

The second strategy that we have analysed to create the WiFi observation model consists on estimating the expected measurement values $\hat{z}_t^w$ at each position (x_t, y_t), and then deriving the observation model. We have mentioned that we could use any spatial regression technique for this purpose. In this thesis, we will used Gaussian Processes regression (GP regression) [116].

GP regression has already been described as a powerful tool to solve regression problems, and has already been used with great success to build probabilistic radio-maps [80, 117]. Particularly, Duvallet et al. described several of their advantages to map the WiFi signal [80]. These advantages apply to the compass and the cameras as well. First, GPs are non-parametric, so they do not require a regression model to fit the data. Second, both linear and non-linear models may emerge from the regression (whichever fits best the data). Third, GPs are continuous, meaning that: a) training points do not have to be gathered at regularly spaced intervals, b) the environment does not need to be discretised, and c) predictions can be generated for any point in the environment. Finally, contrary to other alternatives such as ε-SVR [57], GPs correctly handle uncertainty in both the process and the estimation and naturally provide probabilistic estimations. We would like to add to this list that GPs are specially suited to solve 2D spatial regression problems, because of the use of a kernel function that can model the spatial correlation among nearby points in the environment. In addition, due to their probabilistic nature, GP regression techniques provide probabilistic observation models, therefore they can be integrated naturally with Bayesian inference algorithms [80, 117].

As in any regression problem, GP regression attempts at predicting a continuous output of a system for any arbitrary input d_{in}^*. Gaussian Processes are a *supervised learning* technique, therefore they require a training data set $D = \{(d_{in}; d_{out}) | i = 1, ..., n\}$. This data set consists on n samples in $\mathbb{R}^d$ such that:

$$d_{out} = f(d_{in}) + \varepsilon \tag{5.29}$$

where f is the function that defines the system, and ε is additive Gaussian noise with zero mean and variance σ_n^2. Following a matrix notation, D_{in} is the $d \times n$ matrix containing the set of training inputs, and d_{out} the $1 \times n$ vector containing the set of training outputs.

GPs rely on a covariance function kernel $k(d_{in}^p, d_{in}^q)$ that specifies the correlation among inputs. The idea behind this function is that input points that are close to each other are likely to have similar output values. There are many choices for this kernel [116], but in this thesis we have used the squared exponential kernel [80]:

$$k(d_{in}^p, d_{in}^q) = \sigma_f^2 \exp\left(-\frac{1}{2}(d_{in}^p - d_{in}^q)^t L (d_{in}^p - d_{in}^q)\right) \tag{5.30}$$

where L is a diagonal matrix whose diagonal elements are scale parameters of the kernel for each dimension of the input. This kernel can be represented as a matrix K. Moreover, we will represent as k^* the vector of covariances between an arbitrary input $d^*{}_{in}$ and the training inputs in D_{in}. The covariance among inputs leads to the definition of the covariance among outputs:

$$cov(d_{out}^p, d_{out}^q) = K + \sigma_n^2 I, \tag{5.31}$$

where I is the identity matrix and σ_n^2 is the observation noise.

Gaussian Processes do not compute the function f directly. Instead, they define a distribution of probability over functions that aim at explaining the training data. For any arbitrary input point d_{in}^*, the posterior distribution over these functions will be:

$$p(f(d^*{}_{in})|d^*{}_{in},D) \sim \mathcal{N}(\mu^*,\sigma^{*2}) \tag{5.32}$$

where

$$\mu^* = k^{*t}(K+\sigma_n^2 I)^{-1}d_{out} \tag{5.33}$$

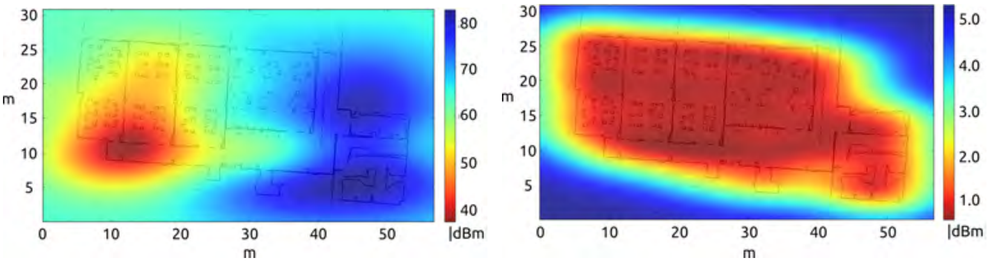
$$\sigma^{*2} = k(d^*{}_{in},d^*{}_{in}) - k^{*t}(K+\sigma_n^2 I)^{-1}k^* \tag{5.34}$$

In other words, given any input $d^*{}_{in}$ and a training set D, GP regression predicts a probabilistic output defined as a normal function centred in μ^* (most probable output) with a typical deviation of σ^*. The typical deviation models the noise in the training data and the uncertainty of the prediction at the same time. All the parameters of the GP regression can be learned from training data by maximizing the log marginal likelihood of the observations conditioned on the parameters [116].

In our case, the regression will compute, for each AP, the functions $\mu_i^w(x,y)$ and $\sigma_i^w(x,y)$. These functions represent the average and the typical deviation of the signal strength of the i^{th} AP across the environment. Figure 5.17 shows a representation of these functions for a sample AP.

These functions will be the base of the observation model of the WiFi. First of all, we will compute the observation model of each AP independently:

$$p(w_t^i|s) \propto \frac{1}{\lambda^w \sigma_i^w(x,y)\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{w_t^i - \mu_i^w(x,y)}{\lambda^w \sigma_i^w(x,y)}\right)^2\right] \tag{5.35}$$



(a) $\mu_i^w(x,y)$

(b) $\sigma_i^w(x,y)$

Figure 5.17: Output of the Gaussian Process regression for a sample WiFi AP. a) The lowest values (red) correspond to the highest power. It is clear that the AP was located near the position (10m, 10m), and that the propagation of the WiFi signal is not isotropic, therefore very hard to model analytically. b) The lowest values (red) correspond to the lowest typical deviation power.

where λ^w is a parameter that scales the typical deviation estimated by the GP regression. Therefore, it modifies the confidence that we have in the sensor model (the greater λ^w, the lower the confidence and the higher the tolerance towards noise). If we assume that the transmitters are conditionally independent given an arbitrary pose s:

$$p(z_t^w|s) \propto \prod_{i=1}^{n_t^w} p(w_t^i|s) \tag{5.36}$$

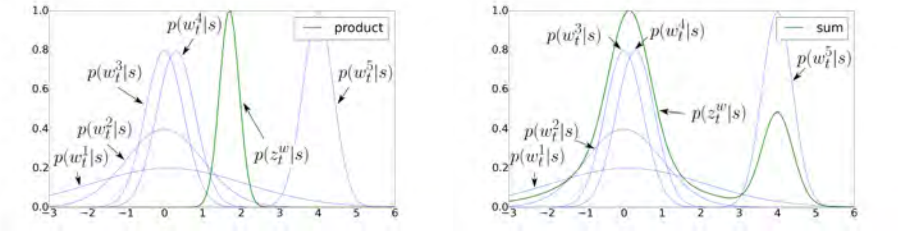
where $p(w_t^i|s)$ represents the probability that the robot receives a signal power w_t^i from the i^{th} transmitter, assuming that the robot is at pose s.

In our experience this approximation is not robust, because it depends too much on the output of each individual function $p(w_t^i|s)$. This is because the combination is a product of probabilities, so even if most functions agree on a certain pose, a single function that has a value near zero around that pose will cause the product to fall close to zero. This is represented in Fig. 5.18-(a), where even if most models $p(w_t^i|s)$ would predict a position around $s \approx 0$ (probably correctly), a single model ($p(w_t^5|s)$) is enough to distort the prediction $p(z_t^w|s)$ so that the maximum falls at a intersection between all the distributions (probably incorrect). This represents an issue because even small noises on the power received from the transmitters can have a big impact on the robustness of the predictions.

We would like to remark that wireless signal power measurements fluctuate due to several factors, such as hardware construction of the receiver, device orientation, human presence, humidity, and changes in the environment such as open/locked doors (see [81, 82, 83, 84, 85, 86]). For instance, in [81] fluctuations of up to ±7.5 dBm were reported in static environments. Similarly, in [82] variations of 3dBm were observed depending on the orientation of the wireless receiver, and the presence of people caused differences of up to 8dBm. These effects are much more drastic in environments were multipath signal propagation does not exist [118]. However, this is not the case of indoor wireless positioning, because in indoor spaces there always exist more than one propagation path from transmitter to receiver. Obviously, these fluctuations have an effect on the positioning error. For instance, in [84] the positioning error increased from $2.10m$ in ideal conditions to $3.06m$ due to changes in humidity, to $3.96m$ due to the presence of people, and to $7.17m$ due to the effect of opens doors.

Therefore, a simple voting scheme can be a much more robust solution:

$$p(z_t^w|s) \propto \sum_{i=1}^{n_t^w} p(w_t^i|s) \tag{5.37}$$



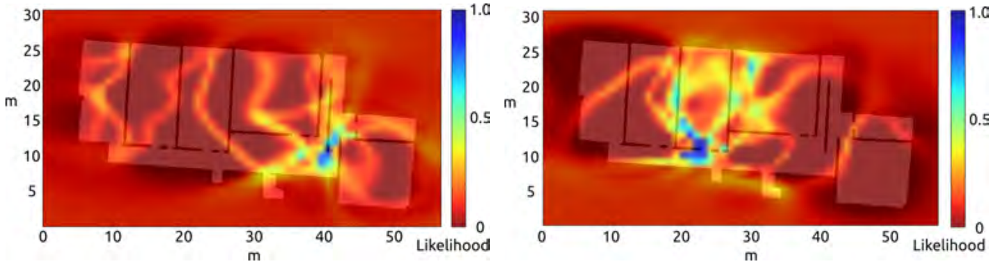
(a)

(b)

Figure 5.18: Likelihood distribution resultant from: a) Eq. 5.36 (product), b) Eq. 5.37 (sum). Each blue line represents an individual function $p(w_t^i|s)$, and the green line the fusion of these functions using Eq. 5.36 or 5.37, respectively.

The underlying idea is that each individual model can contribute to the final prediction, but no model should have such an influence as in Eq. 5.36. Essentially, Eq. 5.37 solves this problem by aggregating the individual probabilities. This is represented in Fig. 5.18-(b), where the maximum value of $p(z_t^w|s)$ is achieved where most $p(w_t^i|s)$ achieve their respective maximum (where most distributions agree), which is probably the correct prediction. The resultant distribution is therefore less affected by individual fluctuations.

Figure 5.19 shows two examples of this likelihood distribution when the robot is at two different positions. Note that in both cases the distribution is multi-modal, but there is usually an area that concentrates most of the likelihood.



(a)

(b)

Figure 5.19: Likelihood distribution provided by the WiFi observation model for two different sensor readings (highest values in blue, lowest values in red).

Computation of density functions: Gaussian Mixture Models

In the previous section, we have estimated the expected measurement values $\hat{z}_t^w$ at each position (x_t, y_t) from training data using GP Regression. Then, we have shown how to compute the observation model taking into account the difference between the received measurements z_t^w and the expected measurements $\hat{z}_t^w$ at each position. In this section, we will estimate directly the probability of receiving a measurement based on the density of occurrence of that measurement at each position. This can be achieved using density estimation techniques, such as Gaussian Mixture Models [119].

A density estimation problem can be defined as follows: given a set $D = \{d_1, ..., d_N\}$ of N data points in n_d dimensions, and a family F of probability density functions on $\mathbb{R}^{n_d}$, find the probability density $f(d)$ that is most likely to have generated those points. Mixture Models try to compute this function $f(d)$ as a weighted sum of K functions $f_k(d)$:

$$f(d) = \sum_{k=1}^{K} \lambda_k f_k(d) \tag{5.38}$$

where the numbers λ_k are called the mixture probabilities, because $\lambda_k \geq 0$ and $\sum_{k=1}^{K} \lambda_k = 1$. In principle, the family F could be formed by completely arbitrary functions, but in practice it is common to use parametric mixture models, where the functions $f_k(d)$ are all from the same parametric family, so the model becomes:

$$f(d) = \sum_{k=1}^{K} \lambda_k f_k(d; \theta_k) \tag{5.39}$$

where θ_k are the parameters of each function $f_k(d)$. A usual choice is to use a family F of Gaussian functions, therefore:

$$f(d) = \sum_{k=1}^{K} \lambda_k \mathcal{N}(d; \mu_k, \Sigma_k) \tag{5.40}$$

The mixture model has a set of parameters $\theta = \{\lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k\}$. The density estimation problem can now be defined as the problem of finding the vector θ of parameters that specifies the model from which the training points are most likely to be drawn. To this extent, we will use the Maximum Likelihood criteria, which defines the likelihood of observing the data as a function of the parameters. Assuming independent data:

$$l(\theta) = \prod_{n=1}^{N} f(d_n; \theta) \tag{5.41}$$

For Mixtures of Gaussians, we have:

$$l(\theta) = \prod_{n=1}^{N} \sum_{k=1}^{K} \lambda_k \mathcal{N}(d; \mu_k, \Sigma_k) \tag{5.42}$$

and the parametric density estimation problem will be solved by finding the vector $\hat{\theta}$ that maximises this likelihood function, that is:

$$\hat{\theta} = \underset{\theta}{argmax} \ l(\theta) \tag{5.43}$$

To achieve this, we can use any maximisation algorithm. Particularly, we have used the Expectation-Maximisation algorithm [119], which works particularly well for this problem. More precisely, we have computed a density function for each WiFi AP, using the training sets D_i^w that we have captured during the deployment stage. In this case, the training inputs will be of the form (x_t, y_t, w_t^i), and the output will be directly the density of occurrence of each data point. Therefore, the observation model of each AP can be defined as:

$$p(w_t^i|s) = \sum_{k=1}^{K} \lambda_k \mathcal{N}(d; \mu_k, \Sigma_k) \tag{5.44}$$

and the complete observation model can be computed by combining the observation models of all the APs using a simple voting scheme:

$$p(z_t^w|s) \propto \sum_{i=1}^{n_t^w} p(w_t^i|s) \tag{5.45}$$

In practice, we did not achieve satisfactory results using this method. We found the following issues:

- The algorithm tends to overfit. This is expectable, because in a context of fast deployment the ratio between the amount of data and the size of the environment is low, therefore the training data will not be very diverse. However, we have observed that this method overfits more than the GP regression approach. We have tried to add synthetic data to increase the noise of the training set. This improved the results, but they were still not satisfactory.
- No spatial interpolation capability. The overfit makes Gaussian functions to have small covariance components. Therefore, in the areas where we do not have training data the model always gives us probability values near to zero (e.g. areas farther away than 1 metre from the nearest training point).

- Penalisation of areas with a low number of training points. The algorithm tries to explain the data “in average”. In practice, this means that the Gaussians will tend to be centred around areas near the centre of the map (where we usually have more training points), but not on the periphery. Therefore, the results on the periphery are usually not good. We could solve this by balancing the data, but since we already have few data points, this does not give good results.

Given these problems, in this thesis we did not use this method in real robot operation.

5.5.3 Magnetic compass - Observation model

A magnetic compass is a low cost sensor available in many robots. It provides the orientation $z_t^c = \theta_t^c$ of the robot with respect to the Earth’s magnetic north. The compass measurements are affected by magnetic fields produced by elements common in public indoor environments, like metallic structures, vending machines, etc. Therefore, the observation model should account for the existance of these distortions. We will explore two ways of modelling these noises:

1. Assuming that the probability of distortion does not vary across the environment (stationary distortion).
2. Assuming that the probabilty of distortion changes depending on where the robot is (non stationary distortion).

Stationary distortion

In this case, we will assume that the measurement noise is Gaussian and independent of the robot position:

$$p(z^c|s) = \frac{1}{\sigma^c\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{\theta_c - \theta}{\sigma^c}\right)^2\right] \tag{5.46}$$

where θ is the orientation component of the robot pose s, and σ^c is a noise parameter of this sensor, which can be estimated experimentally. This assumption is reasonable in absence of strong magnetic interferences, which would require a more complex alternative.

Non-stationary distortion

In this case, we must model the distortion across the whole environment. First of all, we collect a set of k true robot orientations $\theta^T = \{\theta_1^T, ..., \theta_k^T\}$, and we store the corresponding set of compass readings $\theta^c = \{\theta_1^c, ..., \theta_k^c\}$. Then, we compute the differences among both sets:

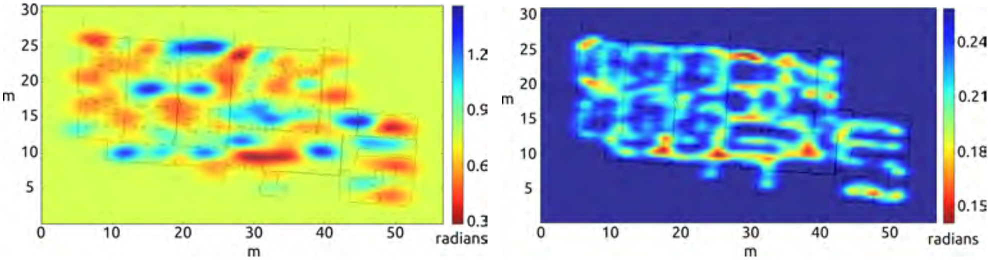
$$e^c = \{e_1^c, ..., e_k^c\} = \{|\theta_1^c - \theta_1^T|, ..., |\theta_k^c - \theta_k^T|\} \tag{5.47}$$

Wherever the spurious magnetic interference is strong, this difference (error) will be high, which indicates that the compass is not reliable. With this information, we build a training set $D^c = \{\,(x_t, y_t)\,;\,e_t^c)\}$. Each sample of the training set consists on a sample output e_t^c (error of the compass at time t, associated with a sample input (x_t, y_t) (position where the error took place). Applying GP regression to this training set, we build $\mu^c(x,y)$ and $\sigma^c(x,y)$, the average and typical deviation of the error across the environment (Fig. 5.20). These functions are the base of the observation model that we propose:

$$p(z_t^c|s) = \frac{1}{\sigma^{ec}(x,y)\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{z_t^c - \theta}{\sigma^{ec}(x,y)}\right)^2\right] \tag{5.48}$$

where θ is the orientation component of the robot pose s, and $\sigma^{ec}(x,y)$ is a noise parameter that we approximate as:

$$\sigma^{ec}(x,y) = \mu^c(x,y) + \lambda^c \sigma^c(x,y) \tag{5.49}$$



(a) $\mu^c(x,y)$ (b) $\sigma^c(x,y)$

Figure 5.20: Average and typical deviation of the compass error in radians across the environment. In the blue areas the average magnetic interference is strong, and in the red areas it is weak.

This way, we take into account the errors that induce the magnetic interferences in different parts of the environment ($\mu^c(x,y)$), together with the typical deviation of that error and the uncertainty of the regression ($\sigma^c(x,y)$) scaled by a factor λ^c.

5.5.4 Camera network - Observation model

Our cameras use a robot detection algorithm based on active markers (LED lights) that we place on our robots [120] (already described in Sec. 3.1.3). This algorithm allows the cameras to detect multiple robots very robustly in crowded environments and in real time. We have seen that this algorithm is very conservative, in the sense that it almost never produces a false positive (false robot detection). This comes at the cost of a lower true positive detection rate [120] (Sec. 3.2.5).

While a camera detects a robot, it sends the robot a periodic message (e.g. every 2 seconds) that contains the identity of the camera. Therefore, the robot can construct vectors $z_t^C = \{C_t^1, \dots, C_t^{n^C}\}$, where n^C is the total number of cameras and C_t^i states whether or not each camera is detecting the robot:

$$C_t^i = \begin{cases} 1, & \text{if camera } i \text{ detects the robot at time } t \\ 0, & \text{if camera } i \text{ does not detect the robot at time } t \end{cases} \tag{5.50}$$

With this information, we can reconstruct the FOV coverage of each camera over the map. To this extent, we construct a training set for each camera $D_i^C = \{\,(x_t, y_t)\,;\,C_t^i\}$. Each sample of the training set consists on a sample output C_t^i (whether or not the camera i detected the robot at time t), associated with a sample input (x_t, y_t) (position at time t).

With this training set, we compute the FOV map of each camera using the GP regression technique again. This gives us the functions $\mu_i^C(x,y)$ and $\sigma_i^C(x,y)$ for each camera i: average and typical deviation of the likelihood that the camera detects the robot when the robot is at each position. Therefore, the areas with the highest $\mu_i^C(x,y)$ values actually correspond to the FOV of camera i (Fig. 5.21)

Taking this into account, we propose the following observation model for the cameras:

$$p(z_t^C|s) = \prod_{j \in \{i | C_t^i = 1\}} \mu_j^C(x,y) \tag{5.51}$$

This model will be used whenever the robot receives a detection message from a camera (only taking into account the cameras that detect the robot at that instant, $C_t^i = 1$). Therefore,

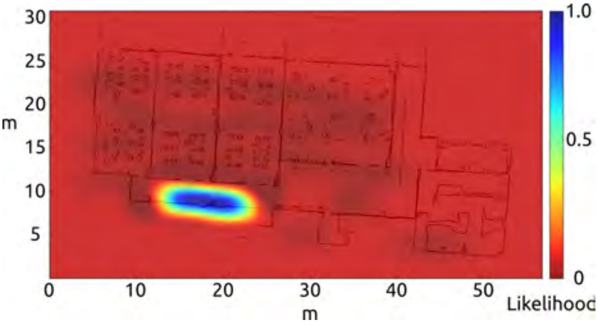


Figure 5.21: Likelihood of robot detection from a sample camera: $\mu^{C}(x,y)$. The highest probability corresponds to the FOV of the camera (in blue).

the most likely robot positions will be those within the FOV of the cameras that detected the robot. Note that in this case we are not using the typical deviation $\mu_{j}^{C}(x,y)$, because we are only interested in determining the FOV of each camera and in weighting each particle depending on whether it falls within the FOV of the cameras that are detecting the robot.

We have discarded the use of a model able to give a more precise estimation of the real robot position (e.g. taking into account the robot position in the image). This is because we have observed that these models require more calibration points than our simple model in order to give accurate estimates (therefore they are less recommendable in the context of fast deployment with non expert users, as we will see in next section). Moreover, this simple model has proven to be reliable enough for our purposes.

In this thesis, we are interested on studying whether the fusion of different sensors improves the performance of a positioning algorithm (even if their sensor models are simple and naive). In this regard, despite the simplicity of the suggested model, the cameras contribute notably to the correct performance of the system (see experiments in Sec. 5.7). When fused with other sensors, even this simple model can provide sufficiently accurate and robust pose estimates. Obviously, camera models able to provide more accurate pose estimates [90, 91, 52, 92, 121] should improve these results even more.

5.5.5 Scene recognition from robot camera - Observation model

One of the limitations of today's robots is related to scene recognition. Robots are still unable to understand their environments, and therefore they are not aware if they are moving in a room that is similar to another one where they have been moving previously. The automatic detection of representative situations -a room without people that can be tidied up, people sitting in a sofa or children playing, people that have just entered home, amongst others- would represent an important qualitative leap forward, as robots would stop from being passive and transform into robots with "initiative". On the other hand, scene recognition might be useful to help the robot in estimating its pose: identifying the scene where the robot is could help to determine the robot position faster and in a more reliable way.

In this thesis, we have collaborated with David Santos-Saavedra in order to develop and integrate an scene recognition system as a sensor for our localisation algorithm. The scene recognition system assumes that the robot will take one observation from a camera (Microsoft Kinect) at some location and this image will be classified to identify the environment/scene. The different classes represent general categories of places and not particular instances. For instance, in a house the classifier may assign different labels to a kitchen and a corridor, but the same label to two corridors.

We have carried an exhaustive analysis of the performance of different SVM classifiers which use different image descriptors to solve the task of scene recognition [122]. As a result of this analysis we found out that working with a representation that combines global and local features is the best option [122]. The global descriptor we have chosen is the *Local Difference Binary Pattern (LDBP)*. This descriptor is composed by the *Local Difference Sign Binary Pattern* (LSBP) and the *Local Difference Magnitude Binary Pattern* (LMBP).

The LSBP, also referred to as the *Census Transform* (CT) [123], is a non-parametric local transform based on the comparison amongst the intensity value of each pixel of the image with its eight neighbouring pixels, as illustrated in Fig. 5.22. As we can see in this Fig., if the centre pixel is bigger than (or equal to) one of its neighbours, a bit 1 is set in the corresponding location. Otherwise a bit 0 is set. The eight bits generated after all the comparisons have to be put together following always the same order, and then they are converted to a base-10 number in the interval $[0,255]$. This process maps a 3×3 image patch to one of 256 cases, each corresponding to a special type of local structure, and it is repeated for every pixel of the original image.

Regarding the LMBP, its computation is very similar to the LSB, but in this case the intensity (magnitude value) of every pixel is compared with its eight neighbouring pixels. If the difference in intensity amongst the centre pixel and one of its neighbours is higher than a

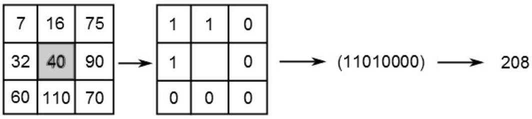


Figure 5.22: Illustration of the LSBP Process on a 3×3 image patch.

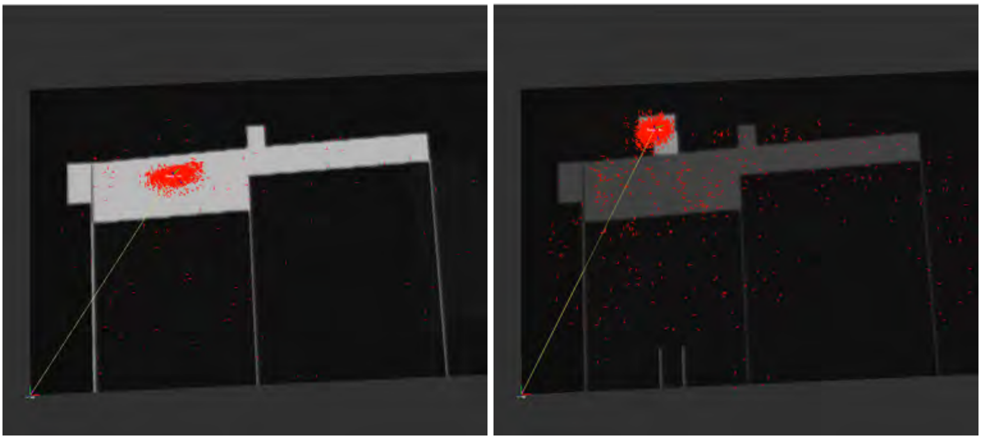
threshold, a bit 1 is set, otherwise a bit 0 is set. Like in the case of the LSBP, after conversion, we will obtain a base-10 number as result.

Thus, for every image we can compute the holistic representation given by the combination of the *LMBP* and the *LSBP*. Both the LSBP and the LMBP histograms 256-dimensional, therefore their combination is 512-dimensional (the bins of the histograms are each one of the values that the LMBP and LSBP can take). It is a common practice to suppress the first and the last bins of these histograms, due to noise cancellation and the removal of not significant information, obtaining a final size of 508. We have also used *spatial pyramids* to get an holistic representation at different abstraction levels, and principal component analysis (PCA) to reduce the dimension of the final descriptor.

Regarding the use of local descriptors (i.e. the discovering of salient points in the image), we have used a bag-of-visual-words model [124]. This model works on three stages: 1) extraction of the local descriptors; 2) quantization of the descriptors into a codebook, 3) description of the image as a collection of words. We have used SURF [125] to detect the interest points and their description, and *K-means* to cluster them into a codebook. Finally, we represent images using a histogram of the visual words.

For each image, the concatenation of the descriptors (histograms) previously described (LDBP and bag-of-visual-words) will be used as the input of our classifier, a Support Vector Machine (SVM). This classifier will assign each image a label from the set $L = \{L_1, \dots, L_{n^L}\}$ (n^L is the total number of classes, i. e. different scenes recognised in the environment). Taking into account that each robot pose s can be assigned univocally with a label L_s (the label of the true scene corresponding to that pose), the observation model of the scene classifier is the probability of receiving a certain observation $z_t^S = L_t$ when the true label is L_s:

$$p(z_t^S|s) = p(z_t^S = L_t|L_s) \tag{5.52}$$



(a)

(b)

Figure 5.23: Likelihood of two different scenes as provided by the scene recognition observation model (white represents a higher likelihood).

This probability is given by the confusion matrix obtained in the training process of the classifier. In Fig. 5.23 we show the likelihood provided by the scene recognition observation model with two different scenes.

5.6 Fast deployment and scalability

In this section, we describe how our system can be deployed quickly in any environment. Then, we discuss the scalability of our system with respect to the number of robots and to the number of sensors.

5.6.1 Fast deployment

The deployment of our system and the construction of the maps is fast, easy and can be done even by non specialists. Following the method suggested in Sec. 5.5:

1. The user deploys the cameras in the environment.

2. The user moves the robot within the environment while the robot captures information from every sensor (*data collection* in Fig. 5.15). The user may move the robot either:

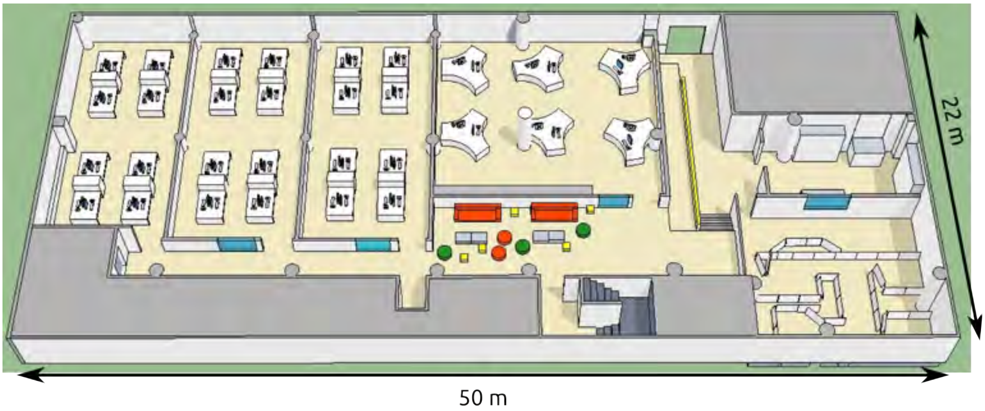


Figure 5.24: Representation of the basement floor of the CITIUS research centre.

a) with a joystick or, 2) with our person-following behaviour with gesture based interaction and voice feedback (Sec. 7). We have seen that non-expert users are able to perform this step successfully using any of both methods.

3. The robot builds the occupancy map, the path followed during calibration, the training sets and calibrates the observation models (*off-line data processing* in Fig. 5.15).

We have deployed our system in different environments. In this thesis, we will focus on a deployment carried out in the basement floor of the CITIUS research centre (Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela, Spain). Figure 5.24 shows an sketch of the experimental area. A typical experimental setup is described in Sec. 5.7.2.

A single user can deploy and calibrate our system in less than one hour in a 1100 m^2 building like this one. Since we want the deployment to be fast, the number of robot runs in the environment will be limited. Therefore, we will only be able to capture a limited number of observations from each sensor (e.g. 20 robot detections per camera on average). Nevertheless, even if the training sets obtained are very small, we have found that our system works properly.

5.6.2 Scalability with the number of robots

Theoretically, our system could be used by more than one robot. As a simple and naive approach, each robot can construct its own map and sensor models, but this is not very practical. Alternatively, one robot can construct all the maps and models, and transfer them to the rest.

Even more, the map can be constructed collaboratively using multi-robot SLAM techniques [126], and the models can be trained by joining the calibration data of the robots. In these last two cases, the camera network model will be valid for any robot out of the box. The same applies to the compass model, because although there might be differences among their compasses, they will not be large in any case.

Similarly, the laser model of one robot can be used in others provided that lasers are mounted at a similar height in all of them. On the contrary, the transference of WiFi localisation models is a bit trickier, because there will exist reception differences across different WiFi cards [127]. However, the impact of these errors is not larger than others (e.g. people walking around), and they can be compensated with other sensors. Furthermore, there exist methods to correct these differences [127].

5.6.3 Scalability with the number of sensors

We have already shown that our fusion methodology is scalable to the number of sensors (Eq. 5.9, Sec. 5.4.1), at least theoretically. In practice, expanding the number of sensors might increase: a) time required for deployment, data collection and model training, b) the power consumption of the robot, c) the computational cost of the algorithm.

Deployment, data collection and model training time

Information sources that require the deployment of elements (e.g. wireless beacons, cameras, landmarks, etc.) increase the deployment time. The deployment and configuration of each information source is different, therefore each one affects the deployment effort differently.

Similarly, each information source requires a training set. All the training sets can be captured at once as described in Sec. 5.6, and this is true for other sensors such as 3D lasers, sonar range-finders, on-board cameras, etc. Therefore, the collection of training data does not limit the number of sensors we can use.

Finally, each sensor model must be trained individually. This increases the global training time, but this increment depends on the nature of each sensor. For instance, the creation of the occupancy map and laser ranges map of the environment represented in Fig. 5.24 took 10 minutes each (approximately). Then, the calibration of the WiFi, compass and camera-network model took less than 1 minute each. Other sensor models, such as 3D lasers or cameras on the robot might need even more time.

As a conclusion, in practice we should use our best judgement and not deploy any number of sensors, but just those that contribute to increase the precision, robustness and redundancy of the system.

Power consumption

The sensors of the robot usually require a fraction of the energy consumed by motors and processing units. Nevertheless, batteries have a limited capacity, therefore in practice an arbitrarily large number of sensors may affect the robot's autonomy.

Computational cost

Robots typically operate on a periodic control loop (e.g. period of $250\,ms$). At each iteration, the robot receives the sensor information, processes it, decides the best action to take and executes this action. The sum of time required by localisation (T_{loc}) and non localisation processings (T_{noloc}) must not exceed the period of the loop (T_{loop}):

$$T_{loc} + T_{noloc} \leqslant T_{loop} \tag{5.53}$$

T_{loc} is approximately the sum of the processing time of each particle filter step: prediction, update, resampling and pose estimation.

$$T_{loc} \approx T_{prediction} + T_{update} + T_{resampling} + T_{pose-estimation} \tag{5.54}$$

Only T_{update} depends on the number of sensors. More concretely, during the update step we compute $p(z^k|s^i)$ for each sensor measurement z^k provided the pose s^i of each particle. The time $T_{sensor}(z^k, s^i)$ needed for this is different for each sensor: negligible for some (e.g. compass or WiFi localisation), and very expensive for others (e.g. 3D laser). We know that:

$$T_{update} = \sum_{i=1}^{n^p} \sum_{k=1}^{n^z} T_{sensor}(z^k, s^i) \tag{5.55}$$

This shows that for fixed number of particles n^p, the T_{update} grows linearly with the number of sensors. Moreover, the computation of this step is highly paralelisable. Therefore, the most important constraint is not the number of sensors, but the inclussion of sensors whose observation models are very expensive to compute.

5.7 Experimental results

We have performed several localisation experiments to test our system with the following objectives: 1) evaluate the performance of the fusion algorithm with different sensor combinations, 2) evaluate each observation model, 3) see whether the fusion of complementary sensors tends to improve the performance of the system, and 4) evaluate whether our proposal works in the context of fast deployment.

All the experiments followed a similar methodology. First of all, we deployed our system in the experimental environment. Then, we moved the robot around the environment to build a map and collect all the information needed to calibrate the sensor models. After that, we moved the robot along a different trajectories, while it captured information from all the sources. With this information, we run our algorithm with different sensor combinations. To out carry an statistical analysis, we run our algorithm several times for each sensor combination, using the same data and sensor observation models, but different random initialisation conditions. In each execution, the robot starts with no knowledge about its pose (global localisation), until the algorithm converges to a pose estimate (tracking). Finally, we compared the trajectory estimated by the algorithm with the real trajectory (ground-truth), and we evaluated a number of the following statistics:

1. *Error in position and orientation* (e_{xy} and e_{θ}): difference between the pose estimated by the algorithm and the ground-truth (the lower the better).

2. *Convergence time* (t_{conv}): time that passes until the algorithm provides the first estimation of the pose (the lower the better). We consider that the algorithm has converged when the clustering step discovers a sufficiently important cluster of particles (Sec. 5.4.2).

3. *Convergence percentage* ($\%t_{loc}$): percentage of time that the algorithm provides an estimation of the pose (the higher the better).

4. *Stability* ($\%t_{loc-conv}$): percentage of time that the algorithm provides an estimation of the pose *after the first convergence* (the higher the better). This is similar to the previous

metric, but *stability* gives direct information about the probability that the algorithm loses track of the robot pose.

In order to collect the ground truth of each trajectory, we have followed a procedure inspired in other works [33]. First of all, an expert initialises the localisation algorithm with the initial true pose of the robot during the trajectory, which is known. Then, the algorithm processes the laser and odometry information, and generates the trajectory followed by the robot. Finally, each pose of the trajectory is either accepted or rejected by the expert, who compares the real laser signature with the signature expected from that pose (visual inspection).

We have evaluated the accuracy of the ground-truth obtained by this procedure. We have moved the robot around the environment, forcing its trajectory to pass over 8 checkpoints. Then, we measured the real robot pose at each checkpoint. Finally, we have compared each real pose with the corresponding pose estimated by the ground-truth construction procedure. We have obtained a median difference of $0.254\,m$ in position and $0.033\,rad$ in orientation. We are aware that the use of a high precision localisation system would have been a better option to build the ground truth. However, we did not have access to a system able to work with such mechanism on such big areas. Moreover, we believe that the results obtained are adequate for the purposes of this thesis.

From now on, we will represent each sensor by a letter (L=laser, W=WiFi, C=cameras, c=compass, S=scene classifier) and the combination of sensors as the sum of letters (e.g. L+W is the combination of laser and WiFi). In the following subsections we analyse each experiment in detail.

In the following sections, we will describe three different sets of experiments that we have performed. First of all, in Sec. 5.7.1 we evaluate the performance of our algorithm using a 2D laser, an ε-SVR WiFi positioning system and a compass with spatial stationary distortion. Then, in Sec. 5.7.2 we evaluate the performance of our algorithm using a 2D laser, a GP regression WiFi positioning system, a compass with non-stationary distortion and a network of cameras. Finally, in Sec. 5.7.3 we have tested the performance of our algorithm when using an scene recognition algorithm.

5.7.1 Laser, WiFi ε-SVR and compass with spatial stationary distortion

The objective of the first set of experiments is to validate our sensor fusion algorithm experimentally. Moreover, we will evaluating the performance of the following observation models:

laser, WiFi ε-SVR and compass with spatial stationary distortion. We have performed the experiment at the CITIUS research centre (Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela, Spain), in an area of $1100\,m^2$ approximately. Figure 5.25 shows an sketch of the experimental area, and Fig. 5.26 the corresponding occupancy map. We have used a Pioneer P3DX robot equipped with a SICK-LMS100 laser, a Devantech CMPS10 magnetic compass, a Cisco WUSB54GCv3 Wifi USB card, and 15 WiFi APs (only those that belonged to the building).

Model calibration

During the "deployment" phase, we have moved the robot around the environment to build the occupancy map and gather experimental data. This data was used to calibration the observation models of the different sensors. From the data, we determined (heuristically), the parameters of the laser and compass sensor models: $max_{LD} = 2$ and $\sigma^c = 1$. Then, we trained the WiFi ε-SVR model with the following sets: training set of 1891 patterns, validation set of 722 patterns, and test set of 644 patterns.

The parameter ε was set to 0.001, and the remaining parameters (C and γ) were computed by grid search with the validation set [115]. The best combination was achieved with the values $C = 8$ and $\gamma = 2^{-16}$). Using these parameters, we have achieved an average error of 4.87 *meters* in the test set, which is similar to the Ekahau solution, but without the drawbacks that we have mentioned earlier (Sec. 5.3). Finally, we have estimated the noise parameters of the *measurement model* through a *5-fold cross validation* with the training and validation sets [115], which yielded the values of $\sigma^w(x) = 2.59$ and $\sigma^w(y) = 2.42$. In the experiments we will assume higher values to accommodate errors that might not be present in the training data.

Localisation results

We have recorded two different robot trajectories in the experimental environment. The first one is a 100 meters trajectory (performed in 161 seconds), that starts in the centre of region A (Fig. 5.25), and goes trough the main corridor and down the ramp to finish in the lowest level (region E, Fig. 5.25). The second is a trajectory where the robot was just spinning for 51 seconds inside region A (Fig. 5.25).

We have evaluated the localisation algorithm with different sensor combinations: 1) laser only (L), 2) laser and compass ($L+c$), 3) Wi-Fi only (W), 4) Wi-Fi and compass ($W+c$), 5)

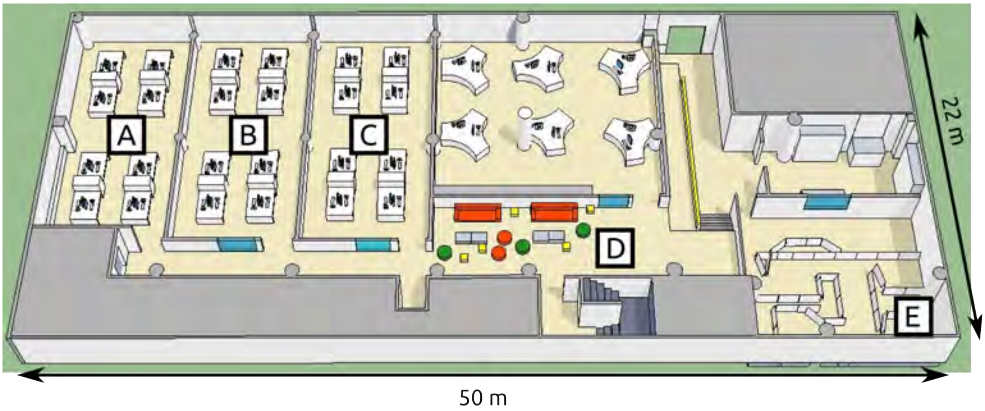


Figure 5.25: Representation of the experimental area in the first set of localisation experiments (basement floor of the CITIUS research centre).

Wi-Fi and laser ($W+L$), and 6) Wi-Fi, laser and compass ($W+L+C$). We ran our algorithm 10 times with each combination, without providing any information about the initial robot pose within the map. Each localisation step always took less than $100\,ms$ in the robot's processing unit (Intel Core i7-3610QM). Taking into account that the period of the robot control cycle is $333\,ms$, we conclude that our algorithm can work in real time. In all the experiments, we used the following parameters: $n^p = 1000$ particles, $\alpha^l = 0.1$, $\alpha^s = 0.3$, $dist_{th-xy} = 5m$, $dist_{th-\theta} = \pi/2$, $\Omega_{th} = 3/4$, $max_{LD} = 2m$, $\sigma_x^w = \sigma_y^w = 5m$, $\sigma^c = 1\,rad$.

The results that we have obtained from these experiments can be seen in tables 5.3 and 5.4, which allow us to extract the following conclusions:

1. *Laser-only (L).* The low $\%t_{loc}$ indicates that the location algorithm needs excessive time to converge to a position (as it is illustrated in Fig. 5.26(a)). The low percentages of $e_{xy} < 5m$ and $e_\theta < \pi/4$ also demonstrate that there is a big probability of incorrect convergence. This happens because the laser alone can not distinguish amongst similar rooms. This causes the algorithm to either not to select an hypotheses, or to select one randomly.

2. *Laser and compass (L+c).* The compass deteriorates the performance of the laser: it tends to reduce the convergence time, at the cost of increasing e_{xy}. Again, none of the sensors can discriminate amongst similar rooms, but the compass information tends to

Traj.1	L		$L+c$		W		$W+c$		$W+L$		$W+L+c$	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
e_{xy} (m)	1.81	2.25	2.67	2.88	2.72	0.30	2.54	0.36	0.77	0.38	**0.52**	0.15
e_θ (rad)	0.14	0.18	0.07	0.02	0.25	0.05	0.12	0.03	0.15	0.12	**0.08**	0.02
$e_{xy} < 5m$ (%)	87.5	18.2	76.5	23.0	93.4	2.37	98.6	0.07	97.6	3.45	**99.3**	0.00
$e_\theta < \pi/4$ (%)	97.8	5.80	100	0.00	95.0	3.50	98.3	0.67	97.2	0.45	**99.8**	0.06
t_{loc} (%)	74.4	15.6	79.9	7.7	76.6	6.4	87.0	3.68	84.2	2.63	**93.3**	2.18

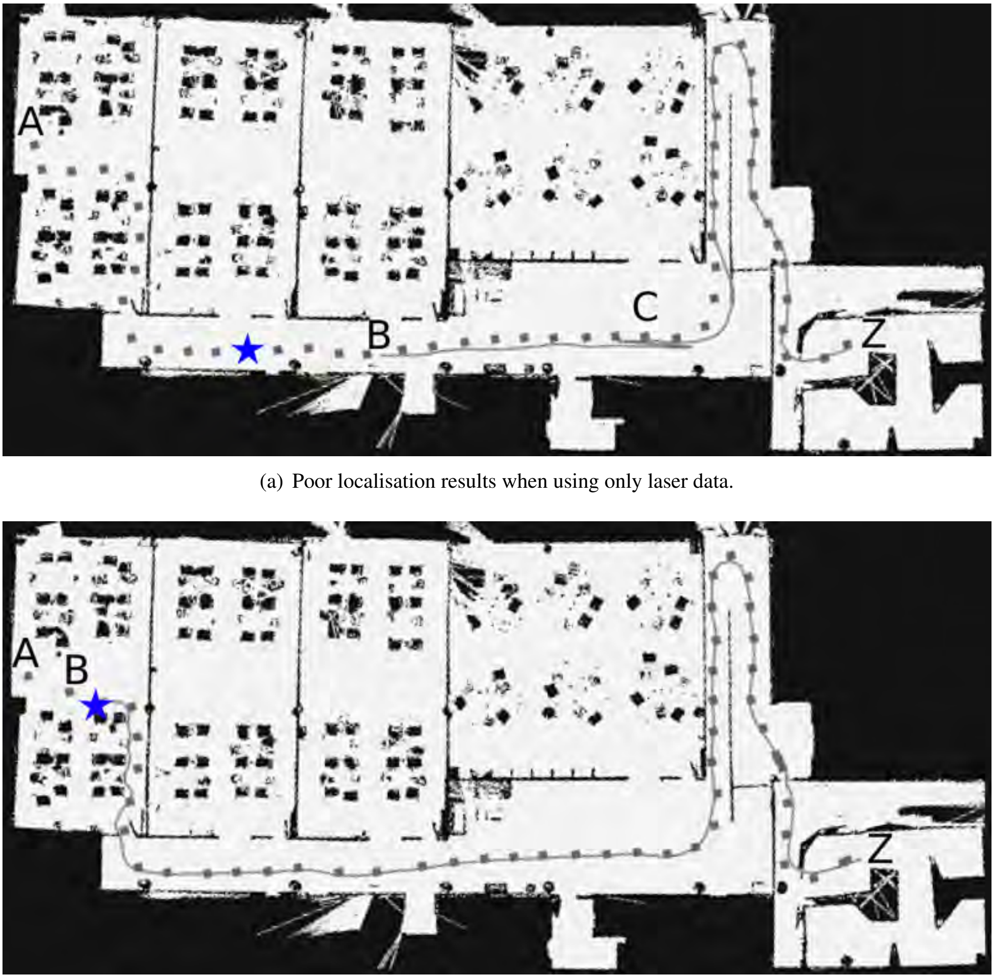
Table 5.3: Results for trajectory 1 on the first set of experiments. We provide averages (μ) and standard deviations (σ) of: error in position (e_{xy}) and orientation (e_θ), percentage of time that the error in position is below 5 meters, percentage of time that the error in orientation is below $\pi/4$ *rad*, and percentage of time that the algorithm is able to provide a localisation estimation (t_{loc}). Bolded numbers highlight the best results.

Traj.2	L		$L+c$		W		$W+c$		$W+L$		$W+L+c$	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
e_{xy} (m)	**0.42**	0.44	5.65	5.77	1.22	0.42	1.83	0.71	0.92	0.61	0.59	0.30
e_θ (rad)	0.14	1.68	0.07	0.05	1.80	0.57	0.17	0.06	1.62	1.37	**0.18**	0.11
$e_{xy} < 5m$ (%)	100	0.00	44.4	52.7	100	0.00	99.0	2.23	100	0	**100**	0.00
$e_\theta < \pi/4$ (%)	66.7	57.7	100	0.00	13.4	21.5	100	0.00	48.9	48.4	**100**	0
t_{loc} (%)	13.9	25.2	41.0	25.3	45.1	15.9	70.2	18.6	60.4	13.6	**80.6**	4.59

Table 5.4: Results for trajectory 2 on the first set of experiments. We provide averages (μ) and standard deviations (σ) of: error in position (e_{xy}) and orientation (e_θ), percentage of time that the error in position is below 5 meters, percentage of time that the error in orientation is below $\pi/4$ *rad*, and percentage of time that the algorithm is able to provide a localisation estimation (t_{loc}). Bolded numbers highlight the best results.

reduce the diversity of particles. This increases the probability of convergence towards a random hypothesis.

3. *Wi-Fi only (W)*. It has high e_{xy} values, but the percentage of $e_{xy} < 5m$ is also quite high. This means that even though Wi-Fi might not have a high accuracy in short distances it is very adequate to discriminate amongst spatially distant hypotheses. In the example, amongst the laboratories (A, B, C in Fig. 5.25) and the large corridor in our building.



(a) Poor localisation results when using only laser data.

(b) High localisation performance example when using all the sensors in our robot (laser, wifi and compass).

Figure 5.26: Two trajectory samples (grey line) generated with our localisation algorithm on the first set of experiments. The ground truth trajectory is represented with grey squares. Letters A and Z represent the start and end points, respectively, of the trajectory. Letter B represents the point where our algorithm starts to output a pose estimate, and the grey star represents the actual position of the robot at that moment. Letter C indicates a zone in Fig. 5.26(a) where our algorithm had to correct its pose estimate.

4. *Trajectory 1 vs 2.* If we compare the results obtained for the two trajectories, we can see that the main difference is that $\%t_{loc}$ is much higher in the first one. This happens because in the second trajectory the robot is just spinning, continuously obtaining the same (or similar) sensor data, therefore the data is not discriminative enough to obtain a fast first estimate of the robot's pose.

5. *Wi-Fi and compass (W+c).* The compass improves slightly the performance of the Wi-Fi. Noticeably, it reduces the percentage of $e_\theta > \pi/4$ to a 0%.

6. *Wi-Fi and laser (W+L).* The Wi-Fi helps the laser to discard spatially distant hypotheses. This improves $\%t_{loc}$, and increases the percentage of $e_{xy} < 5$. However, the Wi-Fi adds noise, which tends to increase the values of e_{xy} and e_θ.

7. *Wi-Fi, laser, and compass (W+L+c).* This is the best combination, as we illustrate in Fig. 5.26(b), which overcomes the drawbacks shown in Fig. 5.26(a). It shows a percentage of $e_{xy} < 5$ above the 99%, and a percentage of $e_\theta < \pi/4$ of 100%. Moreover, the high value of t_{loc} (80.59% in trajectory 2 and 93.3% in trajectory 1), and the low σ values indicate a fast and stable convergence (convergence time below 11 seconds).

Discussion of the first set of experiments

We have proved that the combination of information sources with non-correlated errors tends to perform better than solutions based on single sensors alone [128]. It avoids the confusion between similar spaces, and it provides accurate localisation estimates. We have also tested some of our observation models, and validated our multisensor fusion algorithm for robot localisation.

Although not reflected in the experimental data, we have observed that the WiFi observation model can be improved. We have observed that sometimes the robot receives similar WiFi measurements in different areas of the environment. This causes the ε-SVR to predict that the robot is in any of those areas almost randomly, which causes the robot estimate to "jump" continuously. This problem is not unique to ε-SVR, but to any regression method that aims to predict a single intermediate position. Instead, a method such as GP regression (Sec. 5.5.2), that estimates the potentially multi-modal probability distribution of receiving a certain measurement z_t^w across the whole environment might be more adequate for WiFi localisation.

On the other hand, we have noted that the compass model works well in general, but there are certain areas where the magnetic interferences are too strong and distort the measurements completely. In this case, orientation values should be discarded, but the stationary model does not allow that. Therefore, we believe that a non-stationary model could improve the results in these situations.

5.7.2 Laser, WiFi GP Regression, multi-camera network and compass with non-stationary distortion

Again, we have deployed our system in the basement floor of the CITIUS research centre. Figure 5.27 shows an sketch of the experimental area. We have used a Pioneer P3DX robot equipped with a SICK-LMS100 laser, a Devantech CMPS10 magnetic compass, a Cisco WUSB54GCv3 Wifi USB card, 9 WiFi APs (not belonging to the building), and a network of 5 Microsoft HD-5000 usb cameras (C1 to C5 in Fig. 5.27), each one connected to a laptop. Table 5.5 shows a summary of a calibration trajectory in this environment, which took less than 20 minutes. Figs. 5.28-(a) and 5.28-(b) show two training sets gathered to calibrate the WiFi and the cameras following our methodology (respectively). Figure 5.29 shows the result of the WiFi calibration, and Fig. 5.30 the result of the camera calibration.

This time, we have performed a total of four localisation experiments, each reflecting different common situations of robot operation in environments where people works and lives. In experiment 1 (trajectory 1, Table 5.5) the robot moves within an area that is very similar to

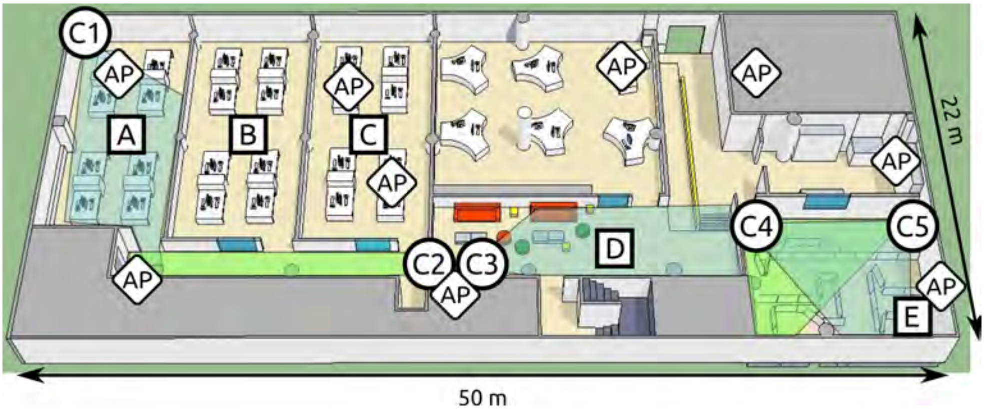
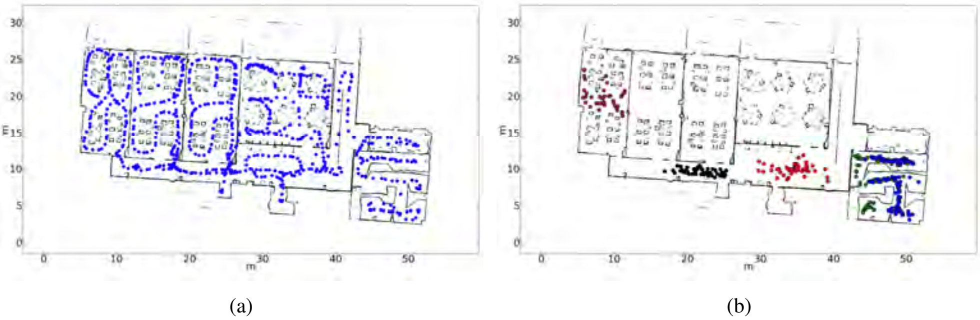


Figure 5.27: Representation of the experimental area on the second set of experiments. Tags A, B, C, D and E correspond to relevant zones (see Sec. 5.7) Tags C1, C2, C3, C4, and C5 represent the locations of the cameras. We also show the APs position and the cameras FOV.

Trajectory	Length	Odometry	Laser	WiFi	Cameras	Compass
Calibration	1148s/646m	11462	56334	841	109	11483
1	198s/76m	1972	9830	137	29	1979
2	185s/65m	1857	8271	128	27	1858
3	110s/40m	1100	5476	76	38	1101
4	87s/27m	703	3492	60	20	872
5	133s/50m	1331	6644	92	9	1331
6	90s/25m	908	4513	62	16	907

Table 5.5: Summary of each trajectory in the second set of experiments: length (in seconds and meters), and number of readings of odometry, laser, WiFi, cameras (cams), and compass.



(a)

(b)

Figure 5.28: Calibration points recorded on the setup of the second set of experiments: a) WiFi, b) network of cameras (each camera is shown with a different colour).

others (environment symmetries). In experiment 2 (trajectory 2, Table 5.5), we have analysed the impact on our algorithm of several furniture rearrangements (environment changes). In experiment 3 (trajectories 3 and 4, Table 5.5) the robot was following a group of persons (group following). Finally, experiment 4 (trajectories 5 and 6, Table 5.5) was performed during a real robotics demonstration in front of visitors (crowded environment).

In all the localisation experiments, we have used the following parameters: $n^p = 4000$ particles, $\alpha^l = 0.1$, $\alpha^s = 0.3$, $dist_{th-xy} = 5m$, $dist_{th-\theta} = \pi/2$, $\Omega_{th} = 3/4$, $max_{LD} = 2m$, $\lambda^c = 1$, $\lambda^w = 4$. We have executed our algorithm with a period of 333 ms (control cycle), enough to ensure correct performance of the tasks carried out by our robot (e.g. planning and navigation). The execution of the localisation algorithm takes approximately 30 ms of the control

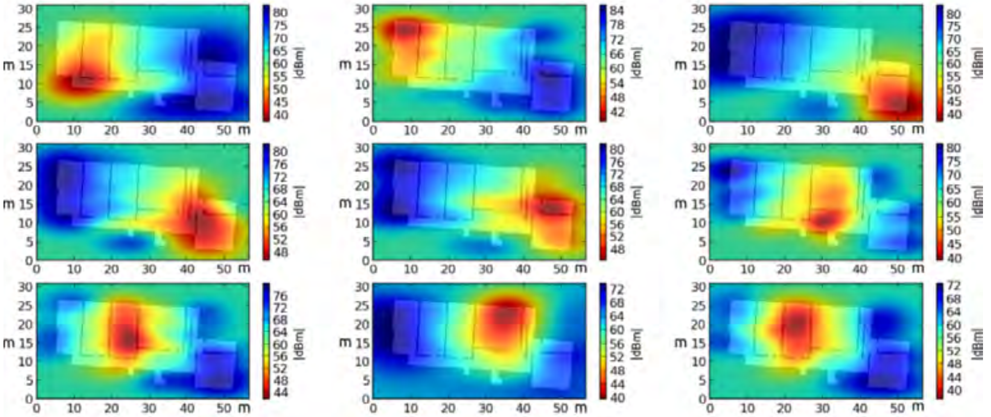


Figure 5.29: Average power (in absolute dBms) of each WiFi AP computed from the data captured during the setup of the second set of experiments. The lower (red), the more powerful. The higher (blue), the less powerful.

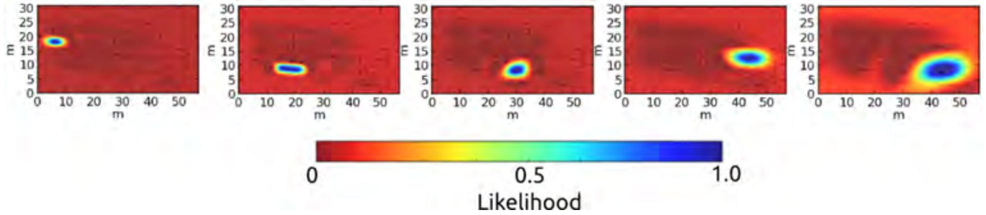


Figure 5.30: Camera maps M_C computed from the calibration data captured during the second set of experiments. A high likelihood (blue) indicates camera coverage. From left to right, we show the maps of cameras C1, C2, C3, C4, and C5 (see Fig. 5.27).

cycle (less than in the previous experiment, due to code optimisation). This indicates that our algorithm can work in real time.

Experiment 1. Environment similarities.

The first experiment corresponds with the trajectory shown in Fig. 5.31. Figs. 5.31-(a) and 5.31-(b) show the ground truth and represent the error at each position when using only laser or the combination of all the sensors (respectively). In addition, Figs. 5.31-(c), 5.31-(d), and 5.31-(e) show the performance boxplots for the 3 sensor combinations mentioned before.

We observe that the laser was useless during this experiment: it is not able to converge most of the time, and when it does the error is extremely high. This happens because the trajectory is confined only to room A, which is very similar to rooms B and C (see Fig. 5.27). Indeed, the algorithm chooses randomly among these rooms when it uses only the the laser information.

On the contrary, we observe that using all the sensors improves dramatically the results. This happens because the WiFi and the cameras break the influence of the similarity among the rooms.

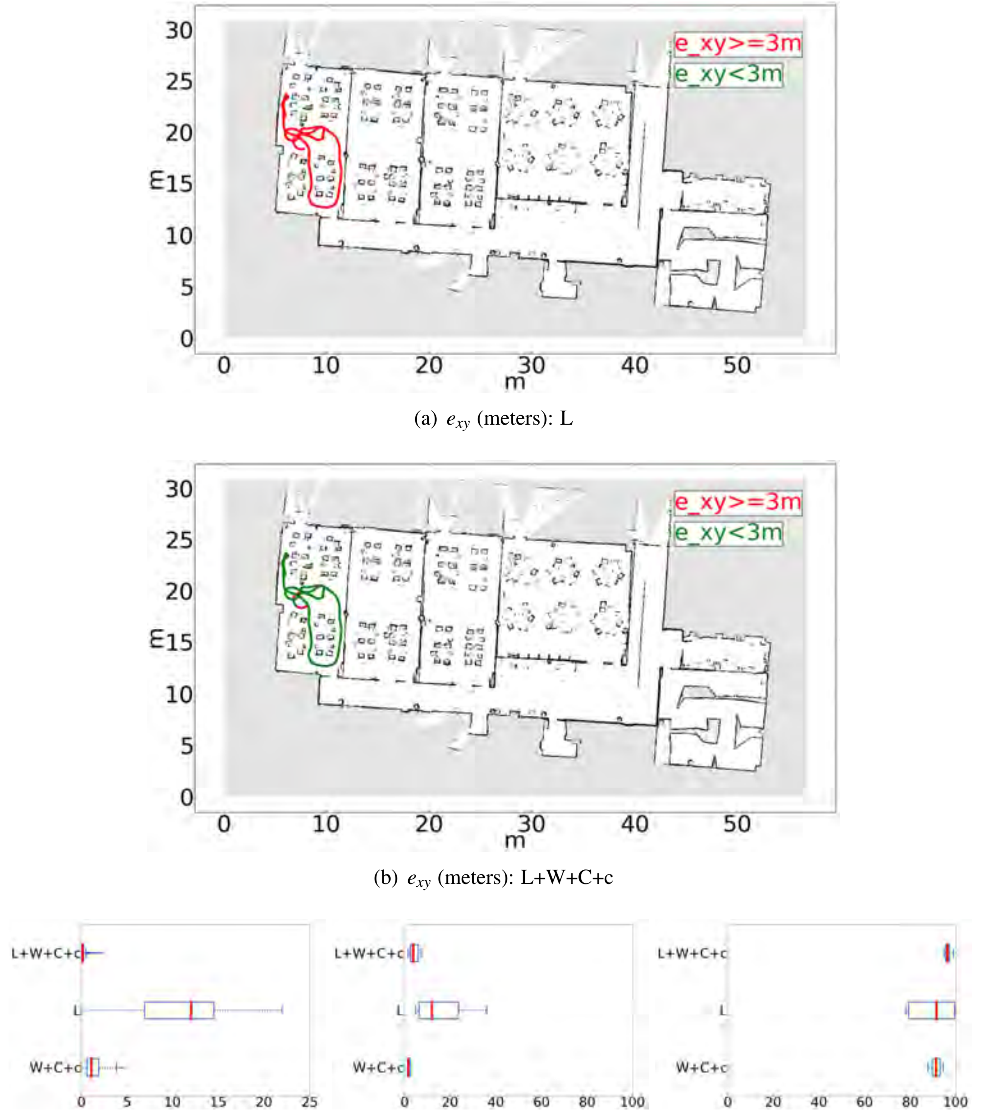
Experiment 2. Changes in the environment.

The second experiment corresponds with the trajectory in Fig. 5.32. This trajectory is even more challenging, because we have rearranged the layout of the sofas and tables of area D (Fig. 5.27). Again, the experimental results show that the laser is useless on its own: since we have changed the environment, the signature of the laser can not match to the signature expected from the map. The fusion of all the sensors improves the results. Note that in this case, the combination W+C+c outperforms the rest. This indicates that the laser is introducing severe errors. However, the fusion of all the sensors is able to mitigate them.

Experiment 3. Group following.

The third experiment corresponds to two trajectories, one of which is shown in Fig. 5.33 (the other is similar, but in an opposite direction). These trajectories are ideal for the laser, because the robot traverses very different areas, which mitigates the errors derived from similarities in the environment. During the trajectories the robot was following a person, while other four were walking by the robot (see Fig. 5.34-(a)). This is a typical situation for our guide-tour robot [129]. We expected that the legs of the people around the robot would distort significantly the laser signatures.

Figure 5.33 shows the experimental results when the robot follows these trajectories. The accuracy of the laser is very high, therefore we conclude that the presence of small groups of people do not lead to failure when using the laser alone. On the one hand, this indicates that a substantial part of the laser signature still contains valid information, despite of the occlusion. On the other, this demonstrates the robustness of the laser observation model that we have suggested. Given the good performance of the laser alone, the main contribution of the sensor fusion is the reduction of the convergence time (Fig. 5.33-(c)). This is because the laser alone



(a) e_{xy} (meters): L

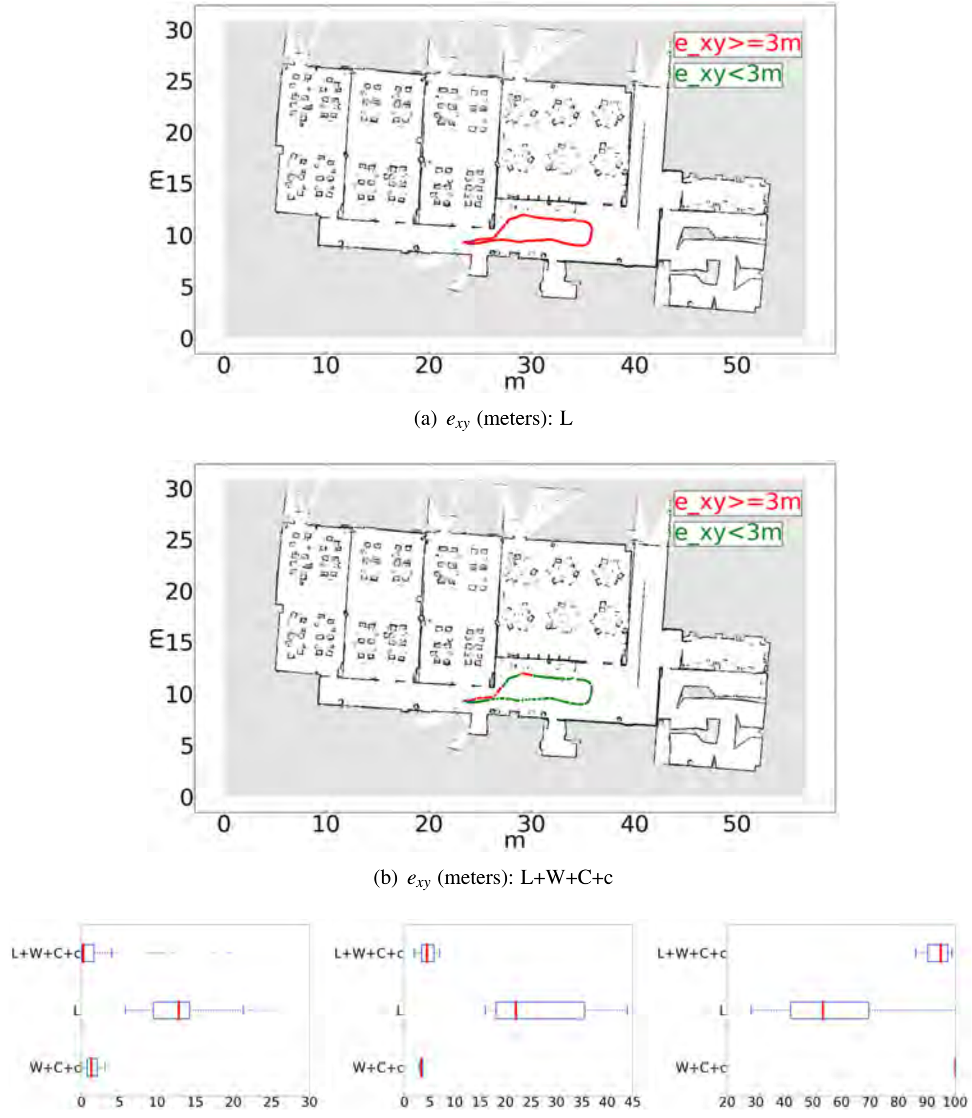
(b) e_{xy} (meters): L+W+C+c

(c) e_{xy} (meters)

(d) t_{conv} (seconds)

(e) $\%t_{loc-conv}$ (percentage)

Figure 5.31: Results of the trajectory of experiment 1 from the second set of experiments (Sec. 5.7.2). a-b) Colour red represents $e_{xy} \geq 3m$, and colour green represents $e_{xy} < 3m$. c-d-e) e_{xy}, t_{conv} and $\%tloc$ boxplots for the sensor combinations ($L+W+C+c$, L and $W+C+c$).



(a) e_{xy} (meters): L

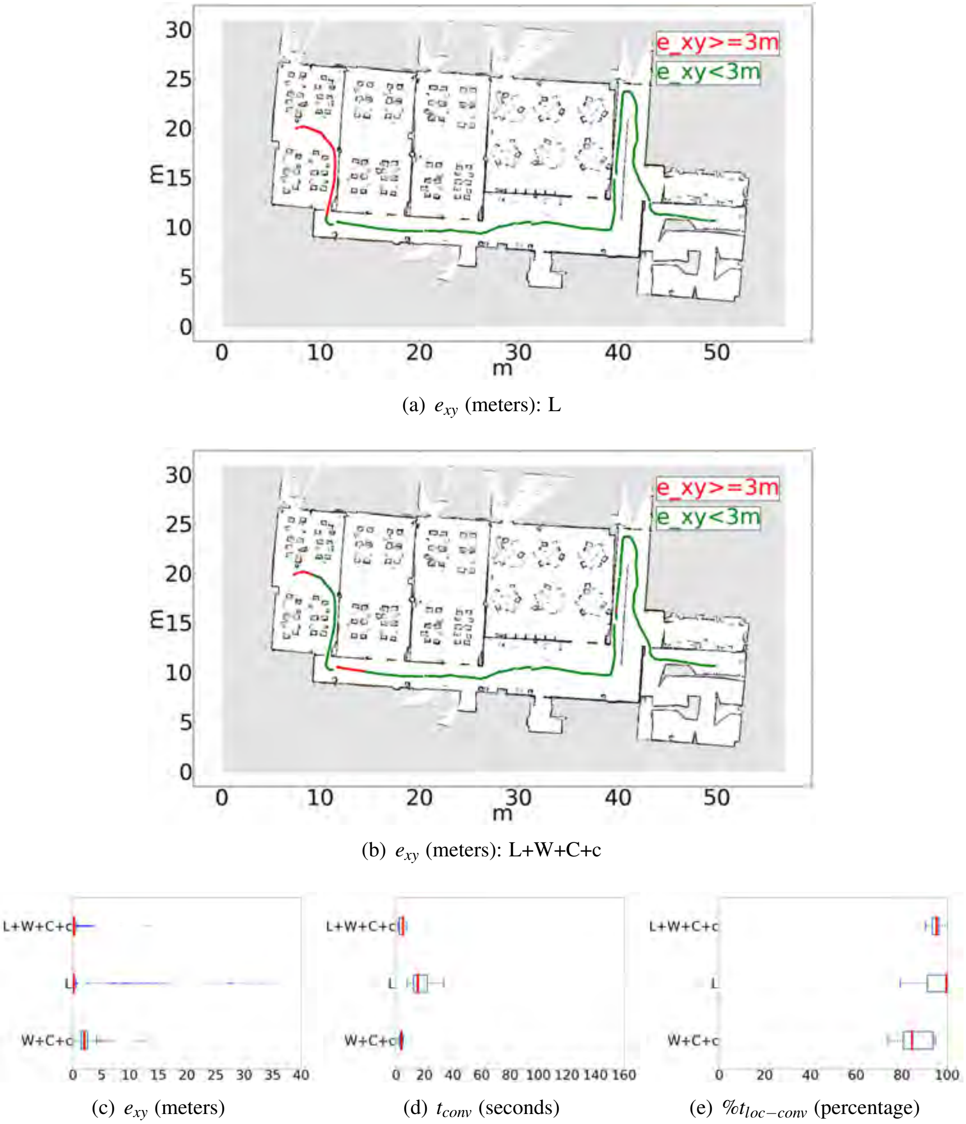
(b) e_{xy} (meters): L+W+C+c

(c) e_{xy} (meters)

(d) t_{conv} (seconds)

(e) $\%t_{loc-conv}$ (percentage)

Figure 5.32: Results of the trajectory of experiment 2 from the second set of experiments (Sec. 5.7.2). a-b) Colour red represents $e_{xy} \geq 3m$, and colour green represents $e_{xy} < 3m$. c-d-e) e_{xy}, t_{conv} and $\%tloc$ boxplots for the sensor combinations ($L+W+C+c$, L and $W+C+c$).



(a) e_{xy} (meters): L

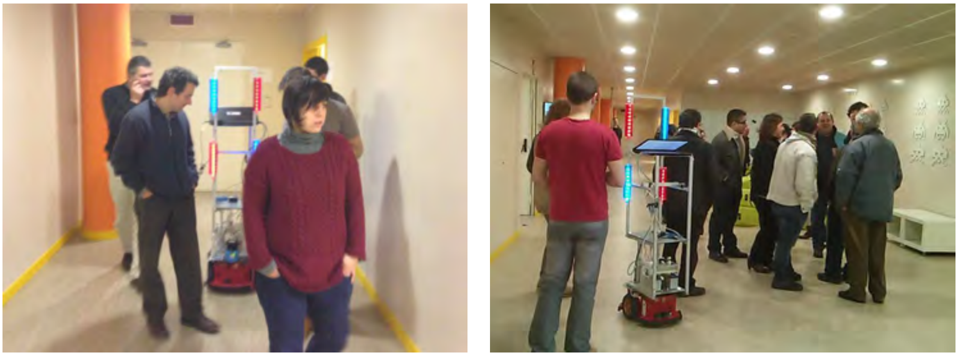
(b) e_{xy} (meters): L+W+C+c

(c) e_{xy} (meters)

(d) t_{conv} (seconds)

(e) $\%t_{loc-conv}$ (percentage)

Figure 5.33: Results of the trajectory of experiment 3 from the second set of experiments (Sec. 5.7.2). a-b) Colour red represents $e_{xy} \geq 3m$, and colour green represents $e_{xy} < 3m$. c-d-e) e_{xy}, t_{conv} and $\%tloc$ boxplots for the sensor combinations ($L+W+C+c$, L and $W+C+c$).



(a)

(b)

Figure 5.34: a) Picture taken while the robot is following a person and other four people walk nearby (experiment 3 of the second set of experiments , Sec. 5.7.2). b) Picture taken while the robot is moving in a crowded area during a robotics demonstration (experiment 4 of the second set of experiments , Sec. 5.7.2)

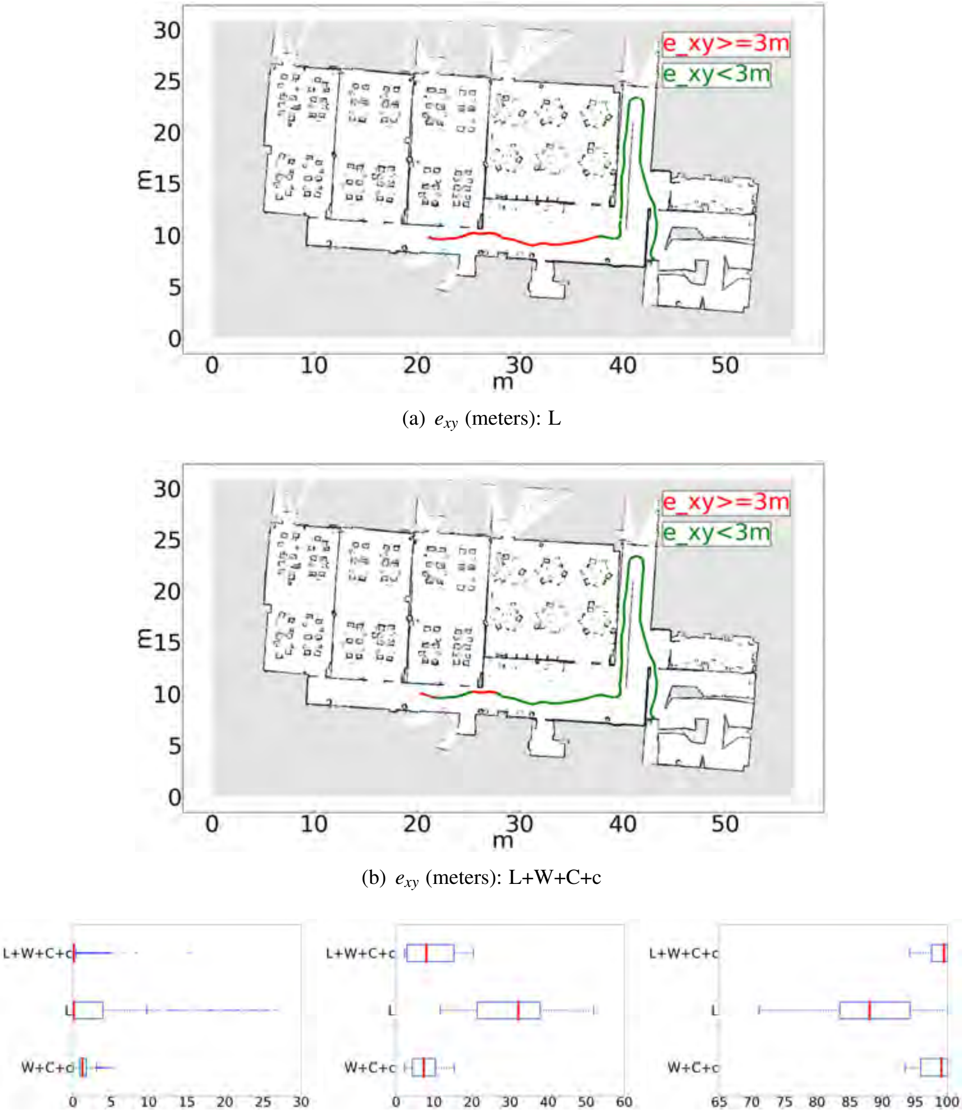
can not converge while moving inside room A due to its similarities with rooms B and C, as we have seen in Sec 5.7.2.

As a curiosity, in Fig. 5.33-(b) we see that the error of sensor fusion grows after the convergence always in the same region. We have observed that this is due to errors introduced by the compass, due to very strong magnetic interferences in that area (see Fig. 5.20).

Experiment 4. Crowded environment.

This experiment corresponds with two trajectories, one of which is shown in Fig. 5.35 (the other one is similar). We have carried out this experiment while the robot was operating in a real social event (Fig. 5.34-(b)). We have recorded the trajectories during a robotics demonstration in front of 20 external visitors approximately. Some visitors were moving around the robot in different directions, others were standing still alone or in groups, etc. Therefore, we would expect that they would occlude the laser with their legs significantly.

The results of this experiment are shown in Fig. 5.35. First of all, we observe a clear difference between the performance of the laser in this experiment and the previous one (Sec. 5.7.2). In the previous experiment, the moderate presence of people did not affect the laser significantly. However, the current experiment shows that the density of people that we might find in real situations does affect the laser performance in terms of accuracy, convergence



(a) e_{xy} (meters): L

(b) e_{xy} (meters): L+W+C+c

(c) e_{xy} (meters)

(d) t_{conv} (seconds)

(e) $\%t_{loc-conv}$ (percentage)

Figure 5.35: Results of the trajectory of experiment 4 from the second set of experiments (Sec. 5.7.2). a-b) Colour red represents $e_{xy} \geq 3m$, and colour green represents $e_{xy} < 3m$. c-d-e) e_{xy}, t_{conv} and $\%tloc$ boxplots for the sensor combinations ($L+W+C+c$, L and $W+C+c$).

time and stability. In fact, the part of the trajectory in Fig. 5.35-(a) where the error is higher corresponds to the area shown in Fig. 5.34-(b), where there are many people around the robot. On the other hand, during the rest of the trajectory there were no people around the robot. Finally, we observe that the fusion of all the sensors works very robustly even in these challenging situations. The algorithm converges very fast, it is very stable and the errors are very low.

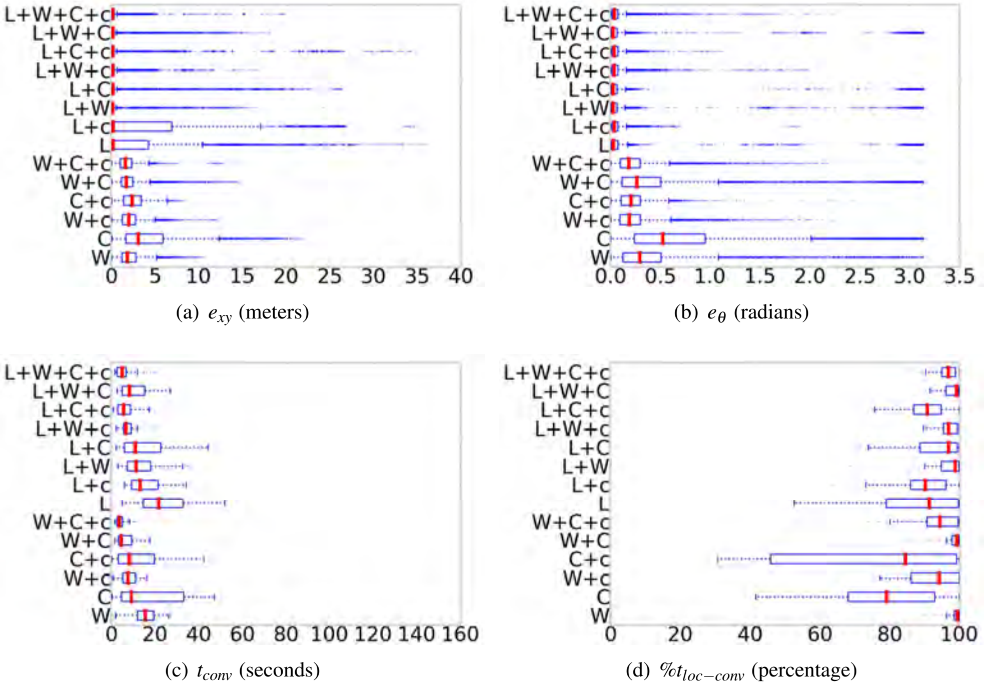
Discussion of the second set of experiments

In Fig. 5.36 and Table 5.6 we show the average and standard deviation of the performance of the system for every metric (e_{xy}, e_θ, t_{conv}, $\%tloc$). We have considered each sensor combination and we have used the data from all the experiments. On the other hand, in Table 5.7 we rank the performance (with statistical significance) of each sensor combination. Considering these results, we can conclude that the best combination is laser, WiFi, cameras, and compass (L+W+C+c), since it is the only one that is among the bests in all the categories. Moreover, we can draw the following conclusions:

- The WiFi tends to produce reasonably good results in terms of position accuracy, convergence time and stability. For instance, it is always present in the combinations that maximise $\%tloc$ (Table 5.7). Since the WiFi does not provide orientation information, the orientation accuracy is poorer (although acceptable in most cases). Overall, it has proven to be a very robust and stable information source.
- The cameras (C) are always present in the combinations that minimise the t_{conv} (Table 5.7). On the other hand, they seem to perform worse than the WiFi in position and orientation error and stability. However, experiments have shown that only the stability differences are significant. Nevertheless, we believe that a wider experimental analysis would have pointed out accuracy differences among them. This is because, on the one hand, the cameras are the sensors that provide less information. This happens because they provide information at a lower rate, the presence of people occludes the robot, and the range of detection is limited to 15 meters in practice. On the other hand, the WiFi APs have bigger ranges which overlap significantly, providing redundancy. Moreover, the presence of people affects the WiFi less than the cameras.
- The laser alone performed poorly because of its inability to deal with symmetries, changes in the environment, and crowded spaces. On the other hand, we have found

out that the laser works well when: a) the trajectory is sufficiently large and diverse, b) there is few or no people around the robot.

- The compass improves the orientation accuracy. The robotics community tends to assume that a compass can not be used for robot localisation, because of the magnetic interferences of the environment [64, 31]. However, we have demonstrated that even a low cost compass can be used if we model adequately the magnetic interferences of the environment (Sec. 5.5.3). On the other hand, the compass may cause errors in areas where the magnetic interference is very strong (see Fig. 5.20). Nevertheless, this should be solved with a finer adjustment of the compass model.



(a) e_{xy} (meters)

(b) e_θ (radians)

(c) t_{conv} (seconds)

(d) $\%t_{loc-conv}$ (percentage)

Figure 5.36: Boxplots of the performance of the localisation in all the experiments of the second set with different sensor combinations.

	$\mu_{e_{xy}}$	$\sigma_{e_{xy}}$	μ_{e_θ}	σ_{e_θ}	$\mu_{t_{conv}}$	$\sigma_{t_{conv}}$	$\mu_{\%tloc}$	$\sigma_{\%tloc}$
L+W+C+c	**0.66**	1.66	**0.05**	0.06	**9.51**	6.71	**98.71**	1.54
L+W+C	**0.88**	2.46	0.25	0.77	16.35	8.41	**98.07**	2.44
L+C+c	2.66	6.33	**0.05**	0.07	**13.60**	11.53	90.52	11.19
L+W+c	**0.83**	1.71	**0.05**	0.06	13.19	14.32	**99.26**	0.89
L+C	2.85	5.96	0.37	0.95	28.67	12.90	92.34	7.60
L+W	**0.94**	2.52	0.16	0.56	18.65	8.89	**97.44**	3.23
L+c	5.44	9.05	**0.07**	0.08	19.06	11.21	86.25	17.87
L	3.28	6.57	0.25	0.75	31.57	11.84	87.61	9.42
W+C+c	1.48	0.88	0.23	0.19	**7.46**	3.84	97.82	2.20
W+C	1.65	1.07	0.43	0.56	**9.46**	4.59	**99.26**	0.76
C+c	1.98	1.04	0.17	0.10	28.80	13.71	72.98	27.40
W+c	1.70	0.90	0.24	0.19	15.52	24.95	93.23	21.49
C	3.26	2.50	0.49	0.37	38.79	6.13	87.52	16.78
W	1.76	1.12	0.39	0.44	20.83	24.96	**94.30**	21.65

Table 5.6: Performance of the localisation in all the experiments of the second set with different sensor combinations. μ represents the average value and σ the standard deviation. We represent in boldface the best combinations, according to the ranking of Table 5.7.

– The combination of WiFi and compass (W+c) leads to very satisfactory results. We believe that this combination could be used in low-cost robots that do not carry a 2D laser range finder.

– The fusion of WiFi and cameras (W+C) improves their individual results, but just slightly. This is because they provide similar information, therefore they do not complement each other significantly. However, they can be used to increase the redundancy of the system.

– There is no perfect sensor. In fact, Table 5.7 shows that there is not a significant difference in performance among C, W, and L alone. On the contrary, L+W is always present in the combinations that minimise e_{xy}, and L+c in those that minimise e_θ. We have observed that the cameras and the WiFi solve the global localisation problem efficiently providing coarse position estimations. On the other hand, the laser is able to solve the local localisation problem by refining these estimations. Finally, the compass helps on refining the orientation. This leads to the best combination, L+W+C+c.

- The algorithm performs correctly with any subset of sensors available, even if they work at different rates or if they stop working: if a sensor is not available, the performance just degrades gracefully.

	e_{xy}	e_{θ}	t_{conv}	$\%t_{loc-conv}$
Ranking	**L+W+C+c**	**L+W+c**	**L+W+C+c**	**W+C**
	L+W+c	**L+C+c**	**W+C**	**W**
	L+W+C	**L+W+C+c**	**W+C+c**	**L+W+c**
	L+W	**L+c**	**L+C+c**	**L+W+C**
	W+C+c	L+W+C	L+W+c	**L+W**
	L+C+c	C+c	W+c	**L+W+C+c**
	W	L+W	L+W+C	L+C
	W+C	W+C+c	C+c	W+C+c
	W+c	W+c	L+W	W+c
	L+C	L+C	C	L+c
	L+c	C	W	L+C+c
	L	W+C	L+C	C+c
	C	W	L+c	L
	C+c	L	L	C
Friedman Aligned Ranks p-value	0.00017	0.0000017	0.00020	0.0072
Holm p-value	0.00000020	0.000000019	0.00039	0.000034

Table 5.7: Ranking of each sensor combination according to different statistical measurements using the Friedman Aligned Ranks method [130]. We highlight in boldface the best combinations. We separate combinations with horizontal rows when there is a significant different among them (post-hoc Holm test [130] with $\alpha = 0.05$). These statistical tests were performed using the STAC web tool [131]. The Holm p-value refers to the adjusted p-value of the first combination of the ranking that performs significantly worse than one of the best group.

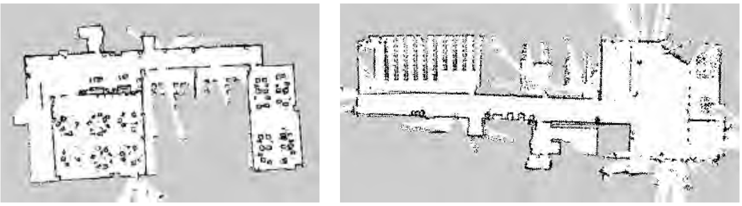
5.7.3 Scene recognition

In this experiment, we have analysed the performance of scene recognition in mobile robot positioning. To have a baseline for comparison, we have also analysed the performance of other sensors in the experiment: laser, WiFi with GP regression, compass with non-stationary distortion, and a network of cameras. We have performed 2 experiments at the CITIUS research building, one in the basement floor and one in the ground floor. We have used a Pioneer P3DX robot equipped with a SICK-LMS100 laser, a Devantech CMPS10 magnetic compass, a Kinect camera, a laptop with Wi-Fi connectivity (Intel Core i7-3610QM @ 2.3GHz, 6GB RAM), 5 WiFi APs, and a network of 5 Microsoft HD-5000 USB cameras (each one connected to a laptop). We have established a period of 1000ms (control cycle) to execute our algorithm.

In this experiment, we have followed a methodology similar to other experiments. First, we have moved the robot along the environment following different trajectories and collecting all sensors information (training data sets). Then, we have run the localisation algorithm over the trajectories with different the sensor combinations (10 times with each combination). We have considered the error in position e_{xy}, the convergence time t_{conv}, and the stability $t_{loc-conv}$ of the algorithm. To explain the experiments, we will use L to refer to the laser, W to the WiFi positioning, C to the cameras, c to the compass and K to the Kinect with scene recognition.

Both experiments took place at the basement and ground floor of the CITIUS building (Fig. 5.37), respectively. Figure 5.38 shows the performances achieved with all the sensor combinations in the first experiment, and Fig. 5.39 the performances achieved in the second experiment. We are mainly interested in combinations that include the Kinect sensor but we also show the performance of the localisation system achieved with other sensors.

It is interesting to study how Kinect sensor works compared to Wi-Fi or the network of cameras, since all of them provide a probability grid for the localisation system. As we can see in Fig. 5.38-(a), Kinect has a slightly worse performance to sensors such as WiFi positioning and the network of cameras when looking at e_{xy} and $t_{loc-conv}$. However, the convergence times t_{conv} of Kinect are significantly worse than those of WiFi and cameras. This result is directly



(a) (b)

Figure 5.37: Occupancy maps created using GMapping SLAM algorithm for the third set of experiments: a) basement floor map, b) ground floor map.

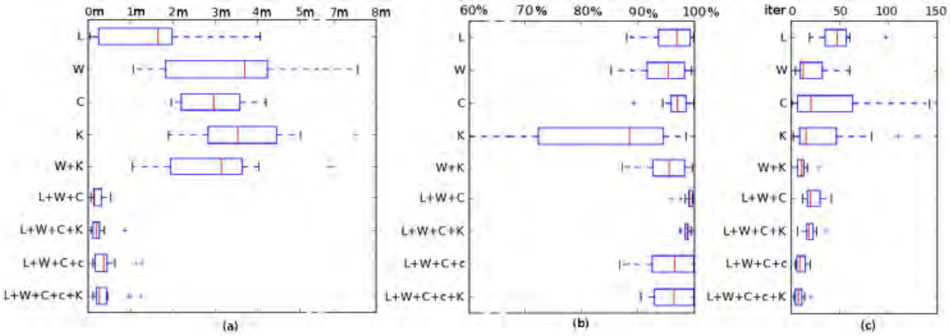


Figure 5.38: Results of the basement floor experiment from the third set of experiments: a) e_{xy}, b) t_{conv}, c) $t_{loc-conv}$.

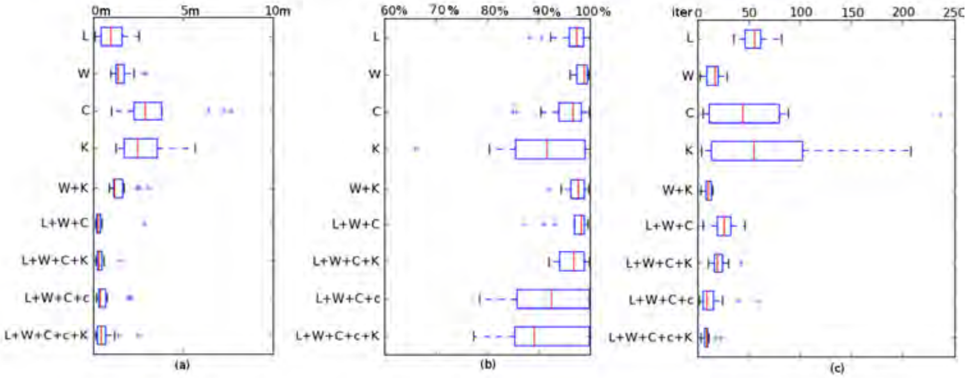


Figure 5.39: Results of the ground floor experiment from the third set of experiments: a) e_{xy}, b) t_{conv}, c) $t_{loc-conv}$.

related to the accuracy of the image classifier (88% success rate in these experiments [122]). Even more, the convergence time of Kinect in the basement floor (Fig. 5.38) is significantly worse than in the ground floor, because this last environment is much more diverse and there exist less symmetries, therefore the classifier performs more robustly and accurately.

The combination W+K shows how WiFi and Kinect work together. The result reveals that they provide a similar information and, therefore, their combination does not achieve better accuracy. However, the combination always improves the stability. Finally, the performance when grouping many sensors together (e.g. like L+W+C+c+K or L+W+C+K), including Kinect as part of the system, improves slightly the accuracy in pose estimation and time required to get the first robot pose. Similarly, the stability is always improved when combining sensors.

These experimental results show that scene recognition can be used for mobile robot localisation. Moreover, once more the experiments show how adding information sources can improve the robustness of the positioning system.

5.8 Discussion

In this chapter, we have presented a localisation algorithm based on particle filters that fuses the information of several sensors of different natures. Our algorithm is able to fuse both on-board and external sensors, even if they are not synchronised or work at different data rates. Our algorithm is scalable (able to handle a variable number of sensors) and highly parallelisable. We have demonstrated the use of this algorithm with multiple sensors, and we have suggested one or more probabilistic observation models for each of them:

- 2D laser range-finder: we have suggested a model that takes into account the differences between the expected and received laser signature, both regarding their shape and each of their concrete values.
- Magnetic compass: we have suggested a model that takes into account the magnetic distortions present in the environment. This allows us to minimise their influence in the orientation readings.
- WiFi receiver card: we have suggested three different techniques that allow us to pinpoint the position of the robot using the signal power received from the WiFi APs in the environment.
- Camera network: we have designed a model that is able to estimate the position of the robot based on the list of cameras that detect it at each instant.
- Camera on-board of the robot: we have developed a model based on scene recognition techniques, that allows us to estimate the most probable robot position from an image taken by the camera.

We have also demonstrated how to calibrate each sensor model with few data without overfitting (context of fast deployment). The algorithm, the methodology, the mapping strategy and the probabilistic modelling can be extended to other sensors as well.

The suggested methodology was designed taking into account the casuistry of service robots (particularly, tour-guide robots). Our robots learn informative routes by following an instructor around. During this process, the robot maps the whole working environment, so the mapping does not add much overhead to the deployment. Moreover, if we wanted to deploy not one but many similar robots, all of them could use same calibration models (at least theoretically), therefore the calibration effort would not increase. Nevertheless, we believe that automatic exploration would be a nice feature to have. The robot could explore the environment on its own or supported by the cameras [132, 133] prior to the route learning process. Even more, we could explore hybrid alternatives, where the robot would be able to complete the mapping directed by the instructor with autonomous exploration.

We have performed experiments in both controlled and real operation conditions. We have demonstrated that our system can be deployed in a fast and easy manner even by non expert users. Moreover, experiments have shown that our system is able to cope with different challenges typical of social environments: a) people walking by the robot, b) crowded environments, c) changes in the environment, and d) similarities among different areas. We have seen that there are two main kinds of sensors: a) those that provide rough global estimates of the robot position, and b) those that provide precise local estimates. With sensors of the first group, the robot will receive an estimation of its position almost guaranteed, although it might not have the highest precision. On the contrary, with sensors of the second group, the robot might be subject to temporary loses, although the precision of the estimation is usually higher than with the first group. With statistical significance, we can affirm that the robustness of the system increases as more sensors are fused (specifically when fusing sensors from both groups). In fact, the fusion of all the sensors is always among the top performers for all the parameters analysed (with statistical significance). Finally, we have demonstrated that our algorithm performs correctly with any subset of sensors available, even if they work at different rates or if they stop working: if a sensor is not available, the performance just degrades gracefully.

With this in mind, it is clear that our system no longer depends exclusively on the information of the multi-camera network. Now, the robot can localise itself and navigate without their help, that is, it can be a “smart” robot. However, this approach is not perfect either, because the system depends completely on the occupancy map that we generate at deployment time. There will be situations where we might not be able to build such a map accurately, at least in certain areas (e.g. in environments with slippery floors where the robot wheels might slide). In this case, the external cameras can still be used to support the robot navigation, even if the localisation algorithm can not work momentarily or during the whole operation. Moreover, the cameras will still be used to detect situations of interest to which the robot should attend.

CHAPTER 6

WIRELESS LOCALISATION WITH VARYING TRANSMISSION POWER

In this thesis, one of our objectives is to be able to deploy robots in different environments, in a fast and easy manner. We have seen that having a robust localisation system is important when we want our robots to perform their tasks correctly. In the previous chapter, we have demonstrated the potential of WiFi positioning, specifically when combined with other sensors, such as 2D laser range-finders. In this chapter, we will explore a more flexible wireless positioning alternative.

WiFi transmitters used in WiFi positioning systems are mostly commercial. Therefore, they use similar transmission powers that can not be modified. We consider that the use of transmitters that can vary their transmission power could lead to a number of benefits. On the one hand, each environment is different, and using a default transmission power may not be the most optimum solution. On the other, different parts of the environment or different situations might require different levels of accuracy and robustness, and therefore different transmission powers. For instance, the signal attenuation of a high power transmitter might not be noticeable in a small room, so it might not be possible to distinguish where the robot is within the room. On the contrary, a low power transmitter might not be received out of a region close to it, so we could know whether the robot is within this region. Even more, different transmission powers lead to different signal propagations, which provide different information. Therefore, the use of transmitters with varying transmission power will increase the discrimination ability of the localisation system. In the extreme, these transmission powers could be changed by the robot itself. This would allow the robot to discard localisation hypotheses actively depending on different criteria: the quality of the current localisation estimate, the task at hand, the part of the environment where the robot is, etc. In fact, the ability of the robot to change the transmission power of the motes would be more profound than that. This ability would take us back to the collective intelligence paradigm: now, not only the intelligent environment would be able to modify the robot behaviour, but the robot would be able to change the way the environment works as well. Taking this into account, in this thesis we will explore the use of wireless transmitters that are able to vary their transmission power.

In this chapter, we describe the wireless motes that we have designed in the context of this thesis, and how we have integrated them into our localisation algorithm (Sec. 6.1). After that, in Sec. 6.2 we describe the experiments that we have carried out in order to test whether or not varying the transmission power increases the quality of wireless positioning systems.

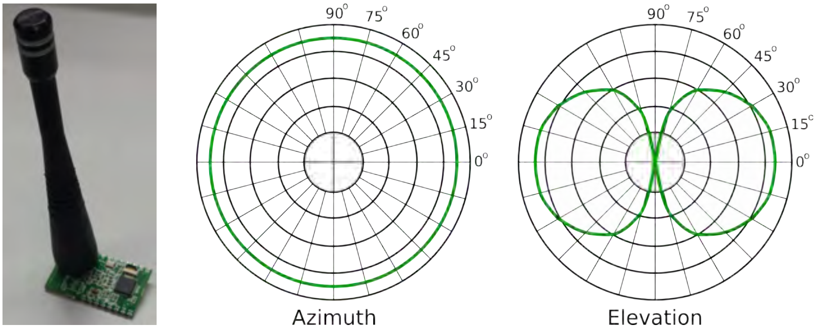
6.1 Wireless sensor nodes (motes) for mobile robot localisation

In this thesis, we will use specifically designed wireless sensor nodes [134, 135], also known as motes. We could have used other alternatives, such as commercial WiFi APs or Bluetooth beacons, but motes are a flexible and convenient solution for our purposes. Moreover, motes have several characteristics that make them very appealing for mobile robot applications:

- Control over the software and hardware of the mote. This enables the adjustment of the transmission power, among other properties.
- Communication capabilities. Motes can be used for communication among robots and environment elements.
- Low power consumption. Motes usually consume much less power that their WiFi APs. What is more important, designers have full control over it.
- Homogeneous hardware. It is known that different WiFi receptors have different reception properties, which constitute a challenge for wireless localisation algorithms [136]. Having homogeneous hardware limits this phenomenon, and simplifies the working conditions of the robots, which enhances the robustness and reduces the cost of design and deployment.

– Several frequency bands. This might be interesting if we want to avoid interferences with existing systems (such as WiFi networks), or to comply with radiation policies of the application environment (e.g. in hospitals or industrial plants).

We have designed and developed motes with the help of Dr. Jose Manuel Rodríguez Ascariz, from the University of Alcalá. Our motes use a CC1110 SoC (Fig. 6.1-(a)), that combines an industry-standard enhanced 8051 MCU and an excellent performance RF transceiver CC1101 that operates in the 433 MHz band. These motes can be produced for less than 15 € each, allow the modification of several transmission parameters, and have a very small power consumption. More concretely, our motes operate at 3V and consume less than 10 uA in sleep mode, 34 mA when transmitting at 10 dBm (maximum power), and 0.1 mA when configured for our robot localisation purposes (assuming a 2 ms data burst every 1 s at maximum transmission power, 10 dBm). Therefore, they consume approximately $0.1W$ in the worst case, and 0.1 mW in a typical setting. Just to give a comparison, we have measured that the power consumption of a commercial router (Linksys WAG200G) is 4.5W even when it is not transmitting information. Every mote is powered by two AAA batteries with an estimated battery life of several months in a typical setting. We have programmed the motes to operate at a 10 kbps data rate, using GFSK modulation in the 433 MHz ISM band with 19 kHz of deviation and 100 kHz of RX bandwidth filters.



(a) (b)

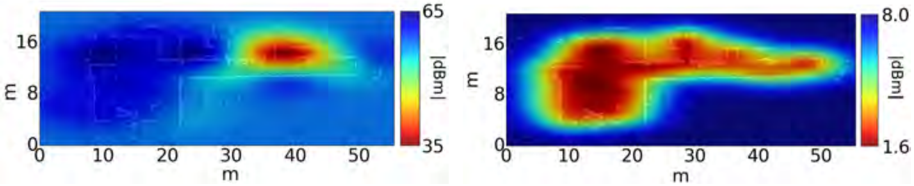
Figure 6.1: a) Prototype of a mote using a CC1110 sub-1GHz SoC with a monopole antenna. b) Radiation pattern of the monopole antenna for an arbitrary transmission power.

We have equipped each mote with a 1.8 dBi monopole antenna from MaxStream (Fig. 6.1-(a)). The radiation pattern of this antenna is illustrated in Fig. 6.1-(b). In the azimuth radiation graph, we observe that the antenna radiates equally in all the directions of the plane parallel to the ground. We will estimate the position of the robot in this plane (2D robot localisation). In the elevation radiation graph, we observe that the antenna also radiates in directions that are not parallel to the ground. Therefore, in principle the motes could be used to estimate the attitude of a receiver. For instance, they could distinguish among different floors, or the attitude of a drone, although this goes beyond the scope of this thesis. Moreover, in this last case other sensors (e.g. sonar sensors) could provide higher precision, because indoor ceilings are rarely high and the attenuation from the ground to the ceiling would not be perceivable.

Our motes can be used both for communications and localisation purposes, but in this thesis we will focus in the latter. In this regard, we have programmed some of the motes to send data packages periodically (transmitter motes). The data sent in the packages indicates the output power at which the packages are transmitted, and the ID of the transmitting mote. On the other hand, the receiver motes were programmed to receive these packages and measure the power of the signal received. These motes are placed on board of the robot.

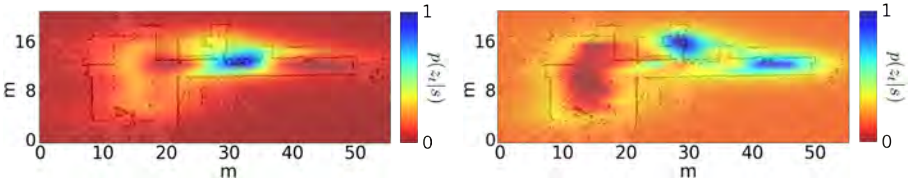
Most of the positioning works with motes have been devoted to the localisation of the static motes that form the sensor network [137, 138]. There have been a few attempts to use motes in person [139] and robot positioning applications [140, 31], but examples are scarce. Therefore, we will propose our own localisation system based on motes. More concretely, the observation model of these motes will be the same used with the WiFi technology described in Sec. 5.5.2 based on the GP Regression technique. Therefore, we do not need to make any changes in the algorithm nor in the deployment methodology.

Figure 6.2 shows a representation of the functions $\mu^k(x,y)$ and $\sigma^k(x,y)$ for a sample mote. Figure 6.3 shows two examples of this likelihood distribution when the robot is at two different positions. Note that in both cases the distribution is multi-modal, but there is usually an area that concentrates most of the likelihood.



(a) $\mu_i^w(x,y)$ (b) $\sigma_i^w(x,y)$

Figure 6.2: Output of the GP regression for a sample mote. a) The lowest values (red) correspond to the highest power. It is clear that the mote was located near the position (40m,16m). b) The lowest values (red) correspond to the lowest typical deviation power.



(a) (b)

Figure 6.3: Likelihood distribution provided by the mote observation model at two different positions (highest values in blue, lowest values in red).

6.2 Experimental results

The objective of these experiments is to test whether the use of varying transmission powers increases the quality of a wireless robot positioning system. To these extent, we will perform two different experiments:

1. Analysis of the performance of localisation with motes and WiFi localisation using a fixed transmission power in both cases. We want to compare their performance to see whether localisation with motes could be used in the context of mobile robot localisation. It is known that the performance of WiFi localisation depends on the quality of the WiFi card attached to the robot [136]. Therefore, to obtain a more reliable result, we will perform this experiment by using 3 different WiFi cards.

2. Analysis of the performance of the localisation with motes when using different transmission powers. We will also explore if using motes that change their transmission

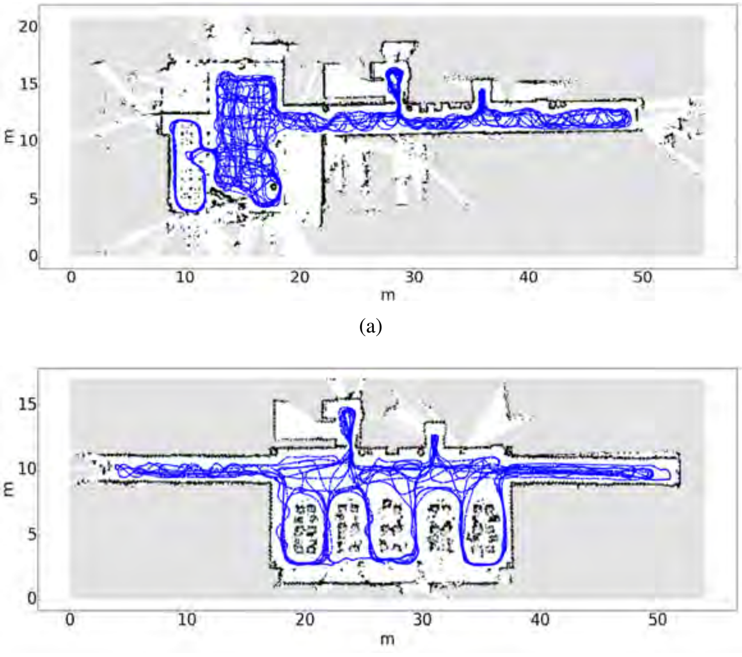
power dynamically increases the performance of the localisation algorithm. This is important, because if this is the case, robots could change the transmission power of motes to achieve a better performance (e.g. active localisation, where the robot may ask the motes to modify the transmission power in order to validate or reject localisation hypotheses).

We have performed experiments in two different floors of the CITIUS research centre (Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela, Spain). We have deployed 6 WiFi Access Points and 6 motes at each floor (each mote was placed near to an AP, to achieve a fair comparison). The WiFi APs are TP-Link TD-W8968 with EIRP<20dBm (Equivalent Isotropically Radiated Power). These APs have 2 dipole antennas of 5dBi each, while each mote has a 1.8dBi monopole antenna (Sec. 6.1). The radiation pattern of each WiFi antenna in the azimuth plane (the relevant in our system) is similar to the pattern of each mote antenna (Fig. 6.1). On the other hand, we have used a Pioneer P3DX robot equipped with a SICK-LMS100 laser, 1 mote (receiver), and 3 different commercial WiFi receiver cards.

To collect the experimental data, we have moved the robot around the environment with a joystick. During this process, the robot recorded the information received from the odometry encoders, the laser, and the signal power received by the mote and the WiFi cards. Table 6.1 shows a summary of the data collected, and Fig. 6.4 a representation of the trajectory followed by the robot. Note that one of our experiments involves changing the power transmission of the motes. In principle, we would have to repeat the data capture stage as many times as the number of transmission powers that we would like to explore (re-configuring the motes every time). Instead, we have programmed the transmission motes to send a new package of data each 250*ms*. Each package is transmitted with a different power: -40 dBm, -30 dBm, -20 dBm, -10 dBm, 0 dBm, 5 dBm, 7 dBm, and 10 dBm. Therefore, the first package is transmitted at -40dBm, after 250*ms* a new package is transmitted at -30dBm, and so on. The

Experiment	Length	Odometry	Laser	Motes	WiFi0	WiFi1	WiFi2
1	2330s/878m	23245	115846	1150	1673	1691	1155
2	2894s/1245m	28898	144136	2756	2093	2108	1416

Table 6.1: Summary of each experimental trajectory: length (in seconds and meters), and number of readings of odometry, laser, and WiFi.



(a)

(b)

Figure 6.4: Trajectory followed by the robot during the capture of the experimental data for localisation with varying transmits power in the CITIUS building. a) Ground floor. b) First floor.

cycle is repeated after 2 seconds, after the package at 10 dBm is transmitted. On the other hand, the robot's mote receives this packages, and every 2 seconds it sends to the robot a vector containing the signal power of the last packages received. This way, to analyse the performance under a certain transmission power, we just have to discard all packages except those transmitted at the power that we want to analyse. Moreover, this same data can be used to analyse the performance when the transmission power changes periodically.

After the experimental data was captured, we divided it in two subsets:

- Training set. Represents approximately the first 33% of the data set. We used this set to construct the map (with the laser and odometry information) and to calibrate the observation models of the mote and the 3 WiFi cards (with the signal power received by each of them).

– Testing sets. The remainder of the data set was divided randomly in 20 parts of 120 seconds each. Each part was used to test the performance of our localisation algorithm.

In each experiment, we execute the algorithm using each of all the test sets (20 times each, to carry out statistical analyses). In each execution, the robot starts with no knowledge about its pose (global localisation), until the algorithm converges to a pose estimate (tracking). The localisation algorithm estimates the trajectory followed by the robot using the robot odometry and the signal power received from the WiFi APs or motes (depending on the case). Finally, we compare this trajectory with the real trajectory (ground-truth), in order to evaluate:

1. *Error in position and orientation* (e_{xy} and e_{θ}): difference between the pose estimated by the algorithm and the ground-truth (the lower the better).

2. *Convergence ratio* ($\% t_{loc}$): percentage of time that the algorithm provides an estimation of the pose (the higher the better). We consider that the algorithm has converged when the clustering step discovers a sufficiently important cluster of particles (Sec. 5.4.2).

We have always used the following parameters: $n^p = 2000$ particles, $\alpha^s = 0.01$, $\alpha^f = 0.1$, $dist_{th-xy} = 5m$, $dist_{th-\theta} = \pi/2$, $\Omega_{th} = 3/4$. We have executed our algorithm with a period of $333\,ms$ (control cycle), enough to ensure the correct performance of the tasks carried out by our robot (e.g. planning and navigation). The execution of the localisation algorithm takes approximately $15\,ms$ of the control cycle. This indicates that our algorithm can work in real time.

6.2.1 Results - Performance of motes vs. WiFi at a fixed transmission power

In this experiment, we have analysed the performance of the localisation algorithm using motes and WiFi APs at a fixed transmission power (we have used the same power for motes and WiFi APs). As we have explained, the performance of WiFi localisation depends on the quality of the WiFi card attached to the robot, so we will use 3 different WiFi cards in this experiment. Figure. 6.5 shows the radio maps of one floor, for each receiver. We can see that the radio maps depend greatly on the hardware of the receiver [136]: each radio map is different from the rest, not only between motes and WiFi, but among the 3 WiFi receivers as well.

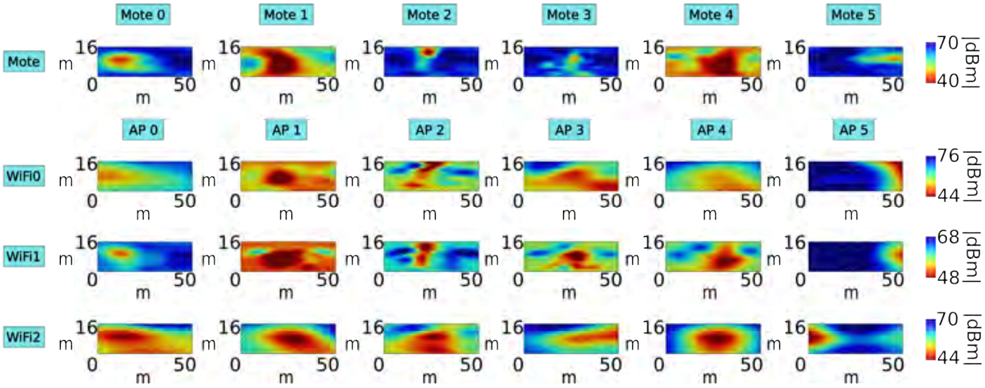


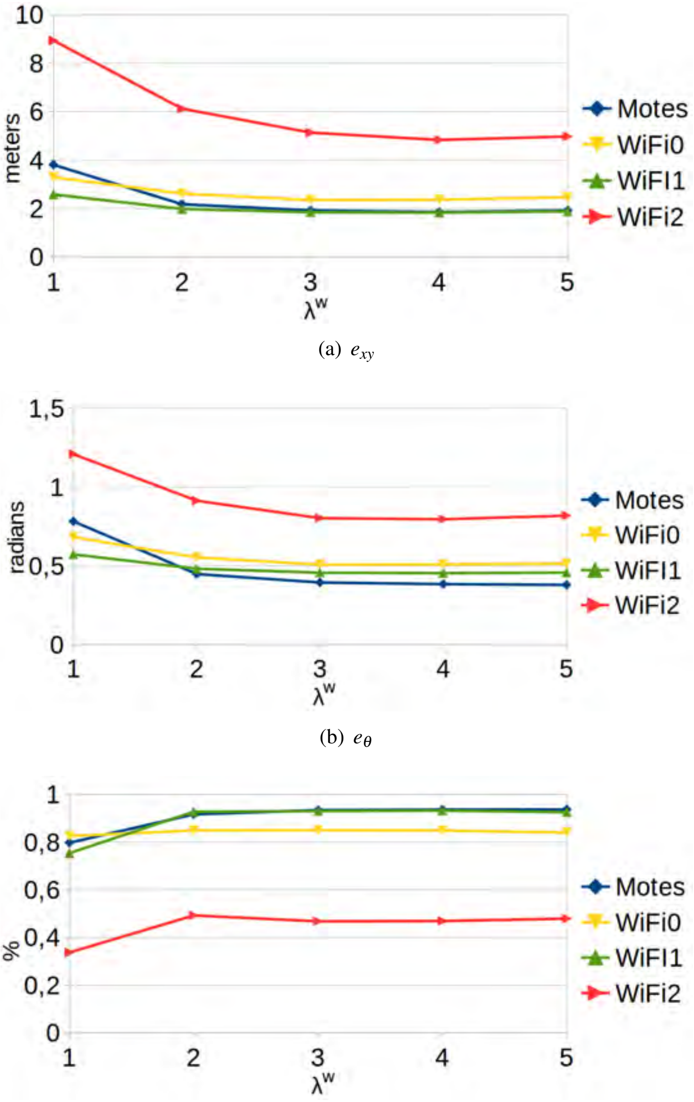
Figure 6.5: Radio maps generated with one of the data sets.

Figure 6.6 shows the performance results of the localisation with motes and WiFi. The experiments were performed using different values of the parameter λ^w. This parameter scales the typical deviation estimated by the GP regression (Eq. 5.35). Therefore, it modifies the confidence that we have in the sensor model (the greater the λ^w, the lower the confidence and the higher the tolerance towards noise). We can draw the following conclusions:

1. Higher values of λ^w tend to give better results. On the one hand, this happens because particle filters work better when they are conservative about the confidence put in observation models [63]. On the other, the training data was captured during a limited amount of time under very stable circumstances. Therefore, any regression algorithm will tend to over-fit, but the λ^w parameter helps to mitigate this problem. However, the value of the parameter can not be arbitrarily large: the larger λ^w, the less information the observation model provides. Results show that the best trade-off is a value of λ^w between 3 and 4.

2. WiFi localisation results depend greatly on the hardware that we use: we have obtained very different results with each WiFi receiver.

3. Performance results of localisation with motes are at least as good as the best results of WiFi localisation. Therefore, our proposal is just as valid as WiFi localisation to be used for wireless robot localisation.



(a) e_{xy}

(b) e_{θ}

(c) $\%t_{loc}$

Figure 6.6: Comparisson of the performance of the algorithm using the motes (blue) and each WiFi card (yellow, green, red) with fixed transmission power.

6.2.2 Results - Varying the transmission power

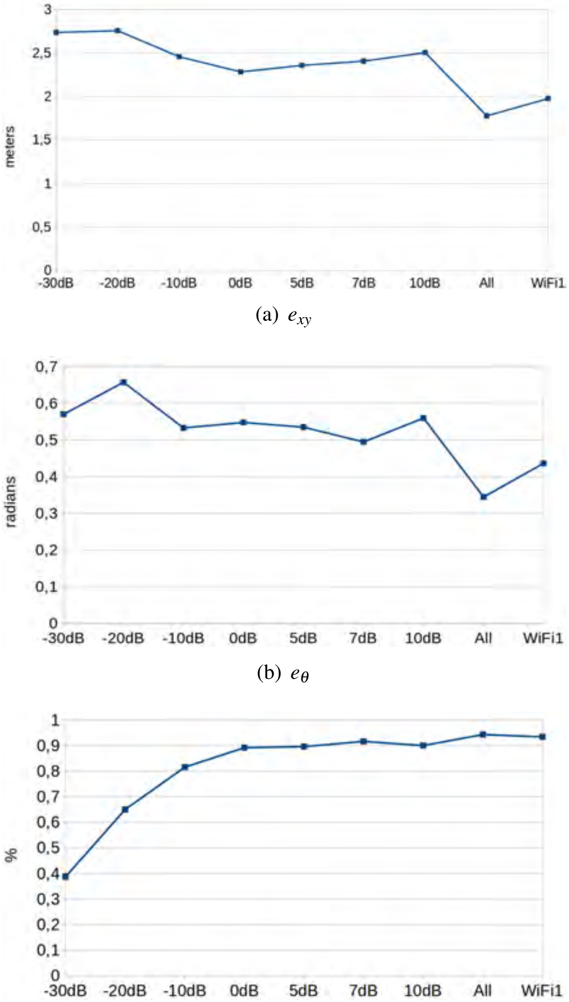
In this experiment, we have compared the performance of the localisation algorithm using: 1) motes with different transmission powers, 2) motes that change the transmission power periodically, and 3) the best WiFi receptor of the previous experiments, whose transmission power can not be modified. In this case, we could only use 4 out of the 6 total transmitter motes, because the other 2 did not allow us to change the transmission power due to firmware restrictions. Obviously, we used only the 4 WiFi APs that were near the selected motes, to ensure a fair comparison.

Note that we did not have to repeat the data capture stage, because the motes were already sending a new package of data each 250*ms*, each package with a different power (-40 dBm, -30 dBm, -20 dBm, -10 dBm, 0 dBm, 5 dBm, 7 dBm, and 10 dBm). Therefore, to analyse the results with a certain fixed transmission power, we just have to discard all the packages, except those transmitted with that power. Similarly, to analyse the results with a varying transmission power, we can consider that we have 8 "virtual" motes for every real mote (because each real mote transmits at 8 different powers per cycle). Therefore, to analyse the performance when varying the transmission power, we have trained a total of 32 "virtual" observation models, and we have just integrated each mote package using the observation model of its corresponding "virtual" mote.

Figure 6.7 shows the performance results in this experiment, and Table 6.2 ranks the performance of each configuration. The experiments were performed using a values of λ^w. We can draw the following conclusions:

1. The best configuration is always the one that uses varying transmission power, and the difference is significant when considering e_{xy} and e_θ. This suggests that the use of varying transmission powers provides useful information that improves the localisation results. This opens the door for future improvements in the line of active localisation. Under this paradigm, the robot would be able to modify the transmission power of the motes in order to discard localisation hypotheses proactively.

2. The second best configuration is, in this case, the one obtained when using the WiFi card. This contradicts partially the results obtained in the previous section, where the motes performed just as good as the best WiFi card. Nevertheless, just as in the previous case, the results may vary depending on a number of factors: number of APs, distribution in the environment, environmental conditions, etc.



(a) e_{xy}

(b) e_{θ}

(c) $\%t_{loc}$

Figure 6.7: Comparison of the performance of the algorithm using: a) motes with different fixed transmission powers (tagged as *-30 dBm*, *-20 dBm*, *-10 dBm*, *0 dBm*, *5 dBm*, *7 dBm*, or *10 dBm*), b) motes that change their transmission power periodically (tagged as *All*), and c) the WiFi card that performed better in the previous experiment (tagged as *WiFi1*). We have discarded the results obtained when using a transmission power of -40 dBm, because the algorithm did not converge most of the times.

Ranking	e_{xy}	e_θ	$\%tloc$
1 (best)	**All**	**All**	**All**
2	WiFi1	WiFi1	**WiFi1**
3	0dB	7dB	**7dB**
4	5dB	-10dB	**10dB**
5	7dB	10dB	**0dB**
6	-10dB	5dB	**5dB**
7	10dB	0dB	-10dB
8 (worst)	-20dB	-20dB	-20dB

Table 6.2: Ranking of the second experiment according to different statistical measurements using the Friedman Aligned Ranks method [130] ($p-value = 0.05$). We highlight in boldface the winning configuration. We separate configurations with horizontal rows when there is a significant different among them (post-hoc Holm test [130] with $\alpha = 0.05$). These statistical tests were performed using the STAC web tool [131]. We have analysed performance of the algorithm using the motes with different transmission powers (tagged as *-20 dBm*, *-10 dBm*, *0 dBm*, *5 dBm*, *7 dBm*, and *10 dBm*), with varying transmission power (tagged as *All*), and with the best WiFi card of the previous experiment (tagged as *WiFi1*). We have discarded the results obtained when using a transmission power of -40 dBm (because the algorithm did not converge most of the times) and -30 dBm (because the algorithm did not converge sometimes).

3. There is no significant difference among the different transmission powers analysed, considering e_{xy} and e_θ. However, powers above 0 dBm perform better on the convergence rate results ($\%t_{loc}$). This makes sense, because motes with low transmission power can only be received in the surroundings of the mote. Therefore, the algorithm will only converge when passing near one of them. We can extract two recommendations from these results. First of all, motes should use a transmission power of 0 dBm, which is the lowest one with the best results. Second, lower transmission powers might be used as well (e.g. proximity based localisation), but a greater number of motes should be used.

6.3 Discussion

We have demonstrated that the use of transmitters with varying transmission powers can improve the performance of wireless positioning systems. We have proven this result with a wireless localisation system for mobile robots. In particular, we have designed a system based on motes that operate in the 433 MHz band. The motes allow to modify several transmission parameters, and have a very small power consumption. We have presented the hardware and software design of the motes, and we have described how to integrate them with a localisation algorithm based on particle filters.

The experimental results described in this thesis show that the localisation with motes is at least as good as WiFi localisation. This shows that mote localisation is a viable alternative to WiFi localisation. Nevertheless, working with motes provides some interesting characteristics that make them very appealing: a) control over the software and hardware of the mote, b) homogeneous hardware, c) full control of the communication capabilities, d) low power consumption, etc. However, our objective is not to substitute WiFi localisation systems, therefore we believe that both technologies could even co-exist in a robotics application, in order to increase the redundancy of the system.

In addition, experimental results have shown that different transmission powers result in different performances. Particularly, the performance was optimum when using transmission powers above 0 dBm. Most importantly, we have analysed the performance of the algorithm when using motes that vary their transmission power periodically. The results show that this configuration improves the results significantly with respect to any other configuration. This suggests that the use of varying transmission powers provides useful information that improves the localisation results. This result is as valid for our motes as it is for other wireless technologies, provided that they allow the modification of the transmission power. This opens a door to explore new research lines such as active localisation, where the robot can modify proactively the power of the transmitters, in order to discard localisation hypothesis actively.

In the future, we will use the mote system to provide communications among our robots and other intelligent elements, such as a network of intelligent cameras that we have developed [141, 120]. In addition, we will further explore the use of patterns of varying transmission powers to improve the robot localisation. Finally, we will explore the use of the motes in active localisation strategies, where the robot will ask the motes to vary their power to improve the localisation results.

Chapter 7

Integration with a tour-guide robot

In Chapter 1, we have indicated that our proposal is a generic solution that can be applied to many different service robot applications. In this thesis, we have integrated our intelligent space with a general purpose guide robot that we have developed in the past [35], as an specific example of application. In this chapter, we will present such tour-guide robot, and how it is connected to the work that we have presented so far. The development of this tour-guide robot was carried out in collaboration with Dr. Victor Alvarez-Santos.

Our tour-guide robot has four main abilities:

– Person following (Sec. 7.1). Our robot is able to follow a human around the environment. Therefore, the robot must be able to detect and track this human, distinguish him from others, and follow him. To this extent, it uses the information provided by a 2D laser range-finder and a RGB-D camera (e.g. Microsoft Kinect).

– Interact with humans (Sec. 7.2). The robot recognises human's commands based on gestures, and it executes different behaviours according to the command recognised. This interaction reduces the learning curve needed to operate with the robot.

– Route recording (Sec. 7.3.1). Using the person following ability, an instructor can teach the robot different routes of interest in the event where the robot operates. Moreover, the robot will be able to record voice messages at points of interest within the route.

– Route reproduction (Sec. 7.3.2). Our robot can reproduce the routes and voice messages under command from a visitor of the event. To this extent, our robot must be able to

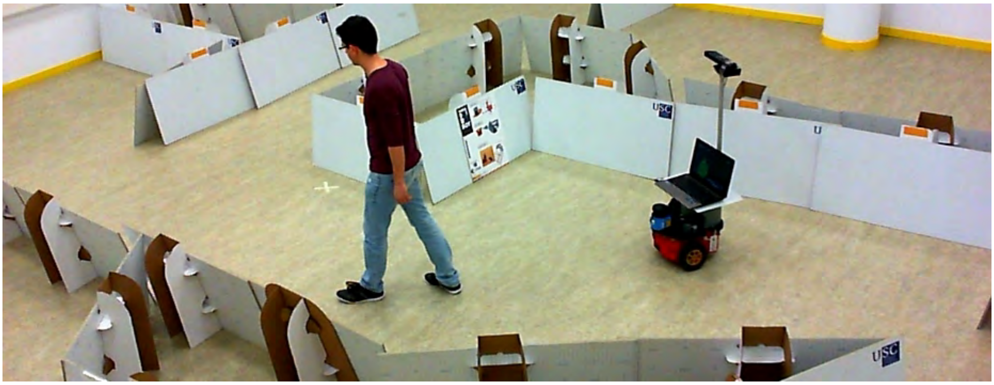


Figure 7.1: The tour-guide robot following an instructor in a narrow test-environment.

compute paths dynamically, and navigate in an environment that might be crowded or might present variations on the furniture.

Some of the aforementioned behaviours use the results described in the previous chapters. For instance, both the route recording and the route reproduction behaviours require the robot to be able to localise itself. This ability is provided by the multi-sensor localisation algorithm presented in Chapter 5. Moreover, our cameras will help the robot on the detection of the events that require their presence, such as a person that is waving at them. This person might be an instructor that wants to call the robot to show it a route, or a visitor that wants information about the event.

7.1 Person following

Person-following ability is a key element for most service robots that interact with humans. For instance, service robots will be usually required by humans for help in a different location than where the robot is. In this case, the most natural way for the human to get robot help would be to ask the robot to follow him to that new location (Fig. 7.1).

Figure 7.2 shows an overview of the software architecture of our person following behaviour. We can see that the person following behaviour (shaded in grey) consists of three main processes:

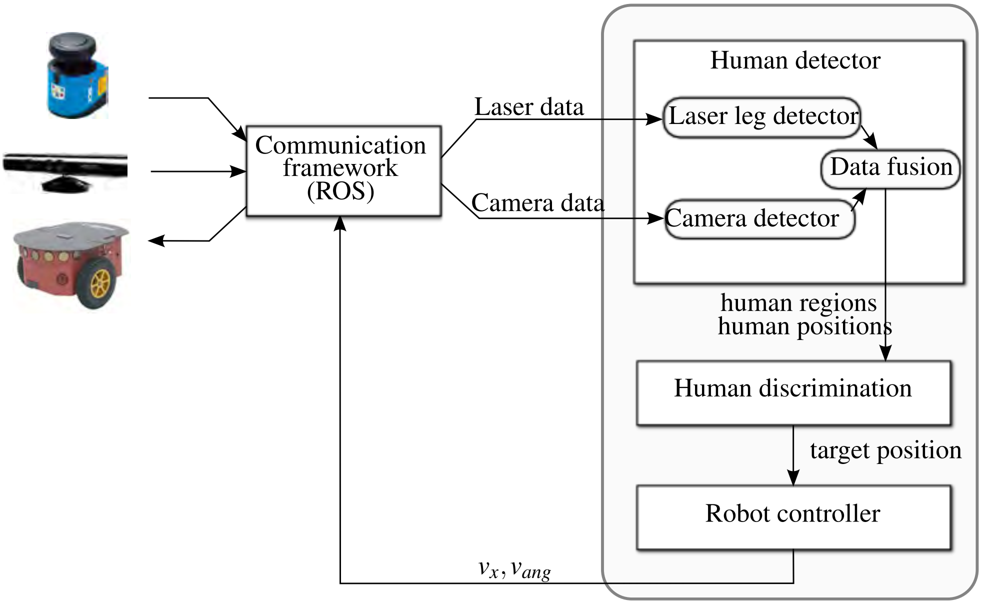


Figure 7.2: Schematic representation of the person following behaviour (shaded in grey) and its interaction with the communication framework (ROS [142]) and the robot sensors. The system uses information from two sensors: a RGB-D camera and a laser scanner. Then, it processes this information to obtain the position of the target, and send to the robot the desired angular and linear velocities.

1. Human detector (Sec. 7.1.1). First of all, our robot must be able to detect humans in order to follow them. Contrary to most related works, we will not base human detection on the detection of their faces. Instead, human's bodies will be detected [143, 144] from the data provided by a RGB-D camera (human detection) and a 2D laser rangefinder (leg detection). This removes a limitation from other works: the humans who are being followed do not have to face the robot anymore.

2. Human discrimination or tracking (Sec. 7.1.2). Our robot must be able to track an specific human (target) in order to follow him, while avoiding any confusion of this human with others around him (distractors). We will introduce an adaptive strategy that is able to enhance the discrimination between the target and the distractors. This strategy consists on the dynamic weighting of a set of selected visual features according to their discrimination ability at each instant.

3. Person following controller (Sec 7.1.3). Finally, our robot must be able to follow the target safely. The person following controller avoids obstacles and maintains the target at a safe distance and centred in the robot's field of view. Our controller is very simple, based on the Potential Fields controller [145].

7.1.1 Human detector

Our human detector, represented in Fig. 7.3, combines the results of 2 modules: 1) a human detector that uses depth images from a RGB-D camera [144], and 2) a laser leg detector.

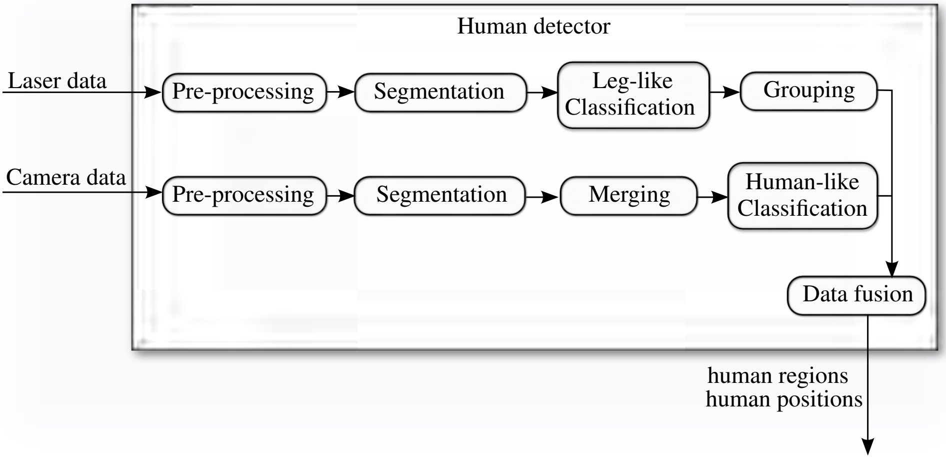


Figure 7.3: Flowchart of the full human detection process that takes place in our tour-guide robot.

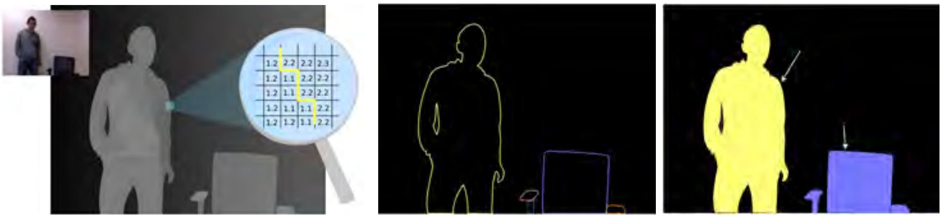
Human detector from depth images (RGB-D camera)

The main goal of our human detection algorithm is to detect standing humans from a mobile platform, regardless of their orientation. Since the robot is moving continuously, we can not use background subtraction or common motion detection techniques. Strategies that look for certain parts of the human body have been discarded as well: a face detector or a head-shoulders detector would not work when those parts are occluded due to the orientation of the human. In this context, we have decided to design an algorithm able to detect humans on depth images (provided by a RGB-D sensor such as Microsoft Kinect). Assuming that humans are not touching each other, the method proceeds as follows [144] :

1. Pre-processing of the depth image. First of all, we remove the floor from the depth image. This is done by erasing the pixels below a certain height from the ground (e.g. 10 cm). This height can be computing taking into account the height at which we have placed the RGB-D camera. Then, we remove low precision data (any pixel farther away than 4 meters from the camera).

2. Depth difference segmentation. In Fig. 7.4-(a) we can see an illustration of a depth image and its corresponding RGB image. The depth image is coloured using a grey scale, where black represents a far distance and white a close one. We compare each pixel value with the values of its adjacent pixels. If we find a depth difference greater than a threshold, we mark that pixel as being part of the contour of an object. We have illustrated this step with the magnifying glass of Fig. 7.4-(a). The result of this process are the contours of multiple objects in the image as we illustrate in Fig. 7.4-(b).

3. Merging of broken objects. There are some objects that due to its pose or orientation might be segmented in several smaller objects. An example of this segmentation can be seen in Fig. 7.4-(b) with the chair, where the armrests were identified as individual objects. To correct this, we compute the positions of every small object identified in the previous step, and if they are close enough to a big object (e.g. less than 0.3 m) we



1.2	2.2	2.2	2.3
1.2	1.1	2.2	2.2
1.2	1.1	2.2	2.2
1.2	1.1	1.1	2.2
1.2	1.2	1.1	2.2

(a) (b) (c)

Figure 7.4: Different stages in our human detector algorithm for depth images. (a) The algorithm starts searching for big depth different amongst adjacent pixels. (b) Then, it connects those regions defining the contours of multiple objects. (c) Finally, the algorithm merges the contours which might be part of another to create big objects, and selects only those which meet the properties of a human (size, shape, etc).

join them together. Figure 7.4-(c) illustrates this process with an example, where the different parts of the chair were put together.

4. Classification of objects. In this final stage, we compute several properties of each object such as height, average width and area. Using these properties we classify the objects into human or non-human. In the example of Fig. 7.4-(c), we filter out the chair for being too wide for such a short height.

It is obvious that an object with a volume and shape of a human can be misclassified. Fortunately, we have found very few objects in real world experiments with these properties, and even when one is misclassified it did not represent a real problem for the tracking of the actual target of our robot.

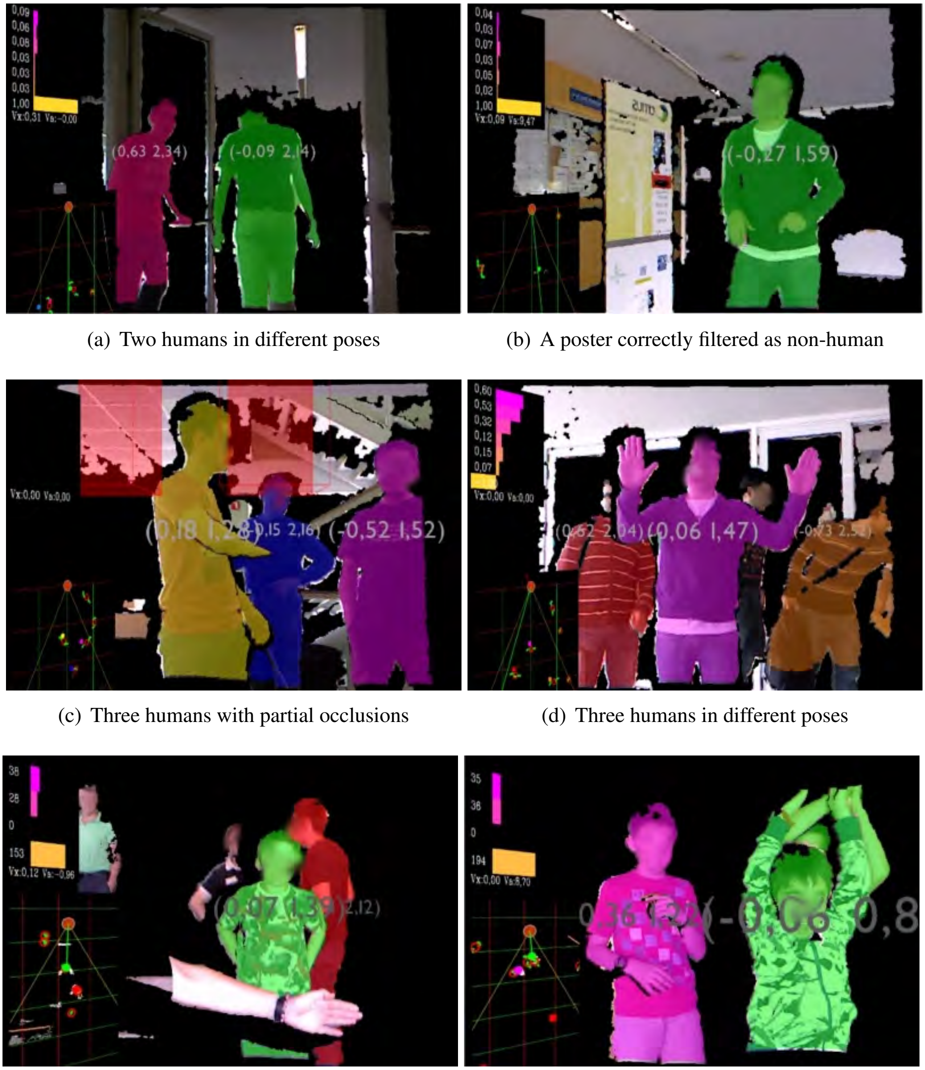
The biggest advantages of this method are: a) it requires small computational power (uses less than 10 ms on a 2012 Intel®Core i7 laptop), b) it is able to detect humans regardless of their orientation (Fig. 7.5), c) it is able to detect humans partially occluded by other humans (Fig. 7.5-(c-e)), or when they are close to the robot and only a part of their body is visible, and d) it is able to reach pixel level segmentation of the human body, leaving most of the background outside the detected region. On the other hand, the major drawback of this algorithm is that if two humans are touching each other it will either: a) not detect them as humans, or b) detect them as a single human (Fig. 7.5-(f)).

The laser leg detector

The role of this second detector is to improve the robustness of the human detection algorithm presented above. Our laser leg detector is based on the work by Gockley et al. [146]. Essentially, it works in a similar way than our human detector for depth images:

1. First, we pre-process the laser data to remove noisy data. It is quite frequent that a part of laser beam detects the edge of an object, while the other part detects the background. When this happens, the distance value that we obtain is the average of both points, which introduces noise in our algorithm.

2. Then, we break up the laser data into segments. The start and end points of a segment are defined by the points where there is a depth difference higher than a threshold (we set it to 0.1 m).



(a) Two humans in different poses

(b) A poster correctly filtered as non-human

(c) Three humans with partial occlusions

(d) Three humans in different poses

(e) People is detected up to a level of occlusion

(f) Two humans are detected as one (green)

Figure 7.5: Different examples of our human detector for depth images. The samples are taken from different scenes recorded in several real demonstrations where we have showcased our tour-guide robot. A human is detected only if his body is shaded with some colour, and his position drawn in front.

3. Next, we extract legs segments and group them by pairs. A leg is any segment with a length between 0.05 m and 0.45 m (no separation between legs in this last case).

The results of this algorithm are the positions of the detected pairs of legs.

Increasing the field of view of the camera

The field of view of the camera is narrower than that of the laser. This might be a problem when the robot is following a human who moves faster than the robot. For this reason, we have decided to include an small actuator to rotate the robot's camera horizontally towards the position of the human target. We have attached a Dynamixel AX-12A motor on top of the mast that holds the Kinect camera in our robot (Fig. 7.6). With this motor, we can rotate horizontally the camera between -70 and +70 degrees.

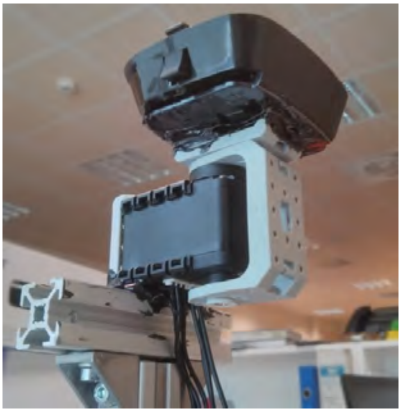


Figure 7.6: The Dynamixel AX-12A installed on top of the mast that holds the camera. This configuration allow us to pan the robot's camera.

Sensor fusion in the human detector

Finally, we fuse the information from both detectors to remove false positives. Figure 7.7 illustrates the sensor fusion process with one example. In the left side of the image, we can see how the laser leg detector has found three pairs of legs located at the positions L_1, L_2 and L_3. On the right side of the image, our human detector for depth images detects three humans located at the positions H_1, H_2 and H_3. A simple matching of both sets of positions

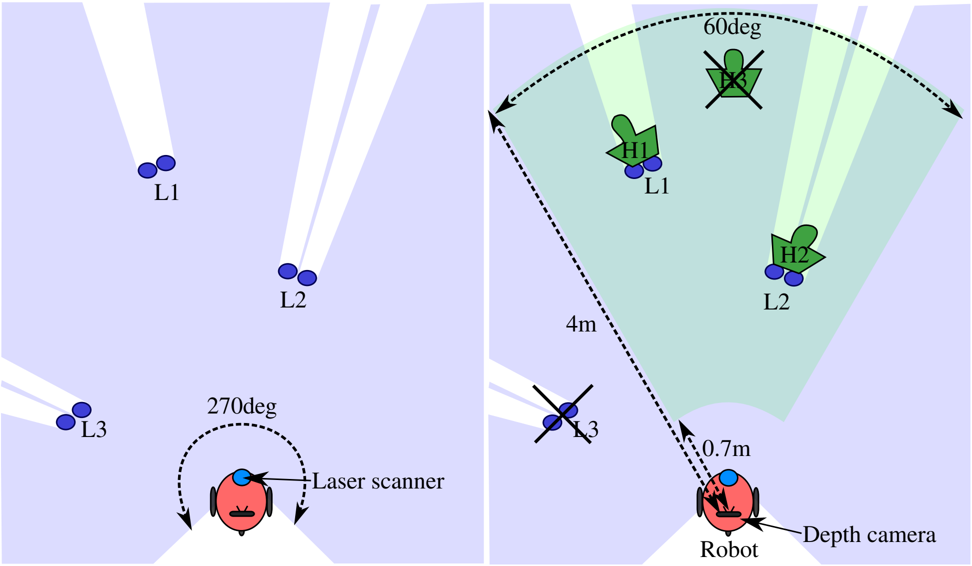


Figure 7.7: Illustration of the fusion of the results from the laser leg detection algorithm and the depth human detector in our tour-guide robot.

is able to discard humans detected by only one detector (e.g. H_3 and L_3). In this matching we use a threshold of 0.5 m. In the end, this process leaves the tour-guide robot with two robust detections (H_1, H_2). Note that, since the camera might be rotated, we need to compute the transform between the frame of references of camera and the laser at each iteration.

Our human detector has proved to be very effective in the tests that we have carried out with our tour-guide robot, as we have shown with the samples of Fig. 7.5. Moreover, the whole human detection task takes no more than 15ms in a 2012 Intel®Core i7 laptop, which leaves enough processing power for other tasks. This is very important in any service robot, since they usually have to deal with several problems at the same time.

7.1.2 Human discrimination

In service robots, human discrimination is often called human identification, human recognition or human tracking. All these designations refer to the problem of identifying the humans in the robot's surroundings, i.e. which one should a robot follow, avoiding mistaking him with other users. This discrimination is a critical task in our robot and in service robots in general.

Therefore, we have made a considerable effort to obtain a robust human discrimination solution, that is able to cope with changes in illumination and temporary occlusions of the target. Our solution takes into account the previous positions of target and distractors, as well as a history of their visual features.

The discrimination of the target from the distractors can be seen as a classification problem, where $R = \{r_1, r_2, ..., r_{n_r}\}$ is the set of detected human regions at the current instant (as given by the human detector), $H = \{h_1, h_2, ..., h_{n_h}\}$ is the set of recognised humans that our robot has recently seen, and h_1 is the robot's target. Our task is to classify each of the human regions r_i into its corresponding human h_j. The probability of a human region r_i of being an instance of a human h_j will be:

$$P(h_j|r_i) \propto P_{pos}(h_j|r_i) + P_{ap}(h_j|r_i) \tag{7.1}$$

where $P_{pos}(h_j|r_i)$ is the probability according to $d_{pos}(h_j, r_i)$ (distance in meters between r_i and h_j):

$$P_{pos}(h_j|r_i) = \begin{cases} 1 & \text{if } d_{pos}(h_j, r_i) \leq 0.3 \\ \frac{1-d_{pos}(h_j,r_i)}{0.7} & \text{if } d_{pos}(h_j, r_i) > 0.3 \;\&\; d_{pos}(h_j, r_i) < 1 \\ 0 & \text{if } d_{pos}(h_j, r_i) \geq 1 \end{cases} \tag{7.2}$$

On the other hand, $P_{ap}(h_j|r_i)$ is the probability according to the appearance, or similarity between r_i and h_j:

$$P_{ap}(h_j|r_i) = similarity(h_j, r_i) = \frac{1}{n_f} \sum_{f=1}^{n_f} d_f(h_j, r_i) \quad \in [0,1] \tag{7.3}$$

where n_f is the number of visual features that are compared, and $d_f(h_j, r_i)$ is the Bhattacharyya distance between the histogram of feature f in the human region ($hist_f(r_i)$), and the histogram of feature f in the human model ($hist_f(h_j)$):

$$d_f(h_j, r_i) = \sqrt{\sum_{k=0}^{b-1} \sqrt{hist_f^k(h_j) hist_f^k(r_i)}} \tag{7.4}$$

where b is the number of bins in those histograms. The comparison of visual features using histograms has the advantage of being invariant to the size of the human region, and to the pose of the human [61, 62].

Finally, a region $r_i \in R$ will be classified as belonging to the class $h_j \in H$ when it maximises $P(h_j|r_i)$, that is:

$$h_j^* = \arg\max_h P(h|r_i) \tag{7.5}$$

with the restriction that we will only assign a human region to the target class when the probability is higher than 0.8, and 0.5 for any other class distinct from the target. Finally, once we have classified the human regions we will update each human with the properties of its corresponding region.

Analysis of visual features for human discrimination

It is obvious that the visual features play an important role on a correct discrimination between the robot's target and the distractors. The visual properties of an object can be classified into colour and texture, and each one can be described with several features. However, many of these features are redundant (i.e. colour spaces), and computing all of them is not a feasible option for our tour-guide robot. Therefore, we have carried out an experimental study to determine which features should be included in our visual representation of the human (visual model). In this section, we will provide a brief insight of the study and its results (for more details, please refer to [147]).

The features that we have considered have been presented in other works and have shown good performance in object recognition or pattern classification problems. We had to discard some popular features due to the real-time computational requirements of our robot (e.g. Fourier, Gabor or Radon [148] transforms, and SIFT/SURF features [149, 150]). We have analysed the performance of a subset of 27 features:

- Texture features: a) Local Binary Patterns (LBP) [151], b) Census-LBP (Census) [152], c) Centre-Symmetric-LBP (csLBP) [59], d) Semantic-Local Binary Patterns (SLBP) [153], e) Edge density (ED) [154], f) Gradient orientation (HOG) [36], g) Edge descriptor (EHD, inspired on the MPEG-7 Edge Histogram Descriptor [155]), h) first and second grey level derivatives($\partial x, \partial y, \partial^2 x, \partial^2 y$) along the horizontal and vertical axes of a grey scale image [60].

- Colour features: *HLS, LAB, YCbCr, HSV, and Grey*. These are all common representations of colour (colour spaces).

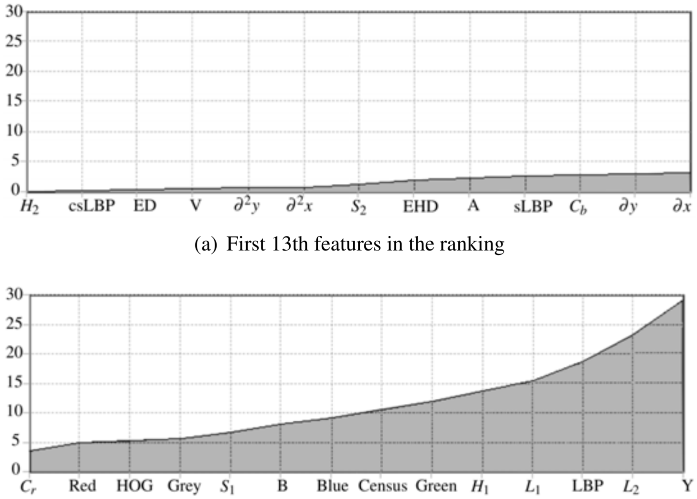
Computing all these features is not a feasible option for a mobile robot with limited computational capabilities. Therefore, we have selected the most appropriate subset of features to accomplish our human discrimination task. The selection was based on four criteria:

1. Redundancy: features must provide different information than those that we have already selected.

2. Relevancy: features must contain useful information for the task.

3. Performance: features must perform adequately in the problem.

4. Computation time: in case that there are two models of similar performance, it will prevail the one with the lowest number of features.



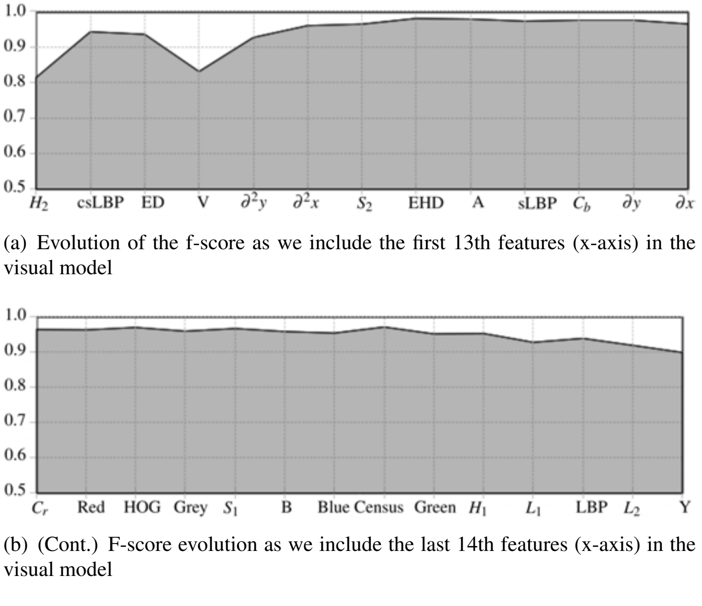
(a) First 13th features in the ranking

(b) (Cont.) Next 14th features in our ranking

Figure 7.8: Increase of mutual information within the features in the visual model, every time that a new feature is included. For space reasons the top graph continues in the bottom one. The x-axis represents the ordered list of features, i.e. H2 from the top table is the first feature in our ranking, while Y is the last one. The y axis shows the amount of mutual information between each feature on the x-axis, and the rest of features that have already been selected (features on the left).The mutual information increases each time that a feature is selected because there is always at least an small redundancy with the previous ones.

More concretely, the selection was carried out as follows. First of all, we ranked our features according to the Minimum Redundancy-Maximum Relevance algorithm (mRMR [156]). The Minimum Redundancy-Maximum Relevance criterion uses Mutual Information [157] to measure both redundancy between the features and relevance to the task. Figure 7.8 presents the increase of mutual information within the features in the visual model, every time that a new feature is included in the model. Then, we selected the most appropriate number of features according to their performance in the task and the computation time that they required. The performance was measured based on the F-score [158]. In Fig. 7.9 we show the evolution of the f-score as we increase the number of features in the visual model of the human.

This selection process resulted on the following subset of eight features (in this order): the three components of the HSV colour space, the second derivatives in both image axis ($\partial^2 x$,$\partial^2 y$), the edge density image based on Canny (ED), the centre-symmetric LBP (csLBP)



(a) Evolution of the f-score as we include the first 13th features (x-axis) in the visual model

(b) (Cont.) F-score evolution as we include the last 14th features (x-axis) in the visual model

Figure 7.9: Evolution of the f-score as we increase the number of features in the visual model of the human. The y-axis shows, for each point in the graph, the f-score obtained when the model is built using the set of all features in the x-axis on the left of the point which is being analysed (i.e. the f-score shown for the V-feature corresponds to the f-score when a set of four features is used: H2, CSLBP, Canny and V). The best performance is obtained when the first 8th features are used. For space reasons, the top graph continues in the bottom one.

and the edge histogram based on the MPEG-7 edge histogram (EHD). This subset of visual features will be used from now on to identify and distinguish the robot's target from the distractors.

Online feature weighting

We have just described how we have selected a suitable set of visual features to distinguish the robot's target from the distractors. All of these features describe different properties of an image, and most of the time all of them are useful to discriminate between humans. However, there might be some complex situations in which some features are more discriminant than others, or even situations where some features will not allow that discrimination. Since the number of possible situations that a robot might encounter when working with humans is large, we believe that the robot itself must be able to adapt its visual models for each situation. The algorithm that we describe in this section provides our tour-guide robot with the necessary adaptability to these real world situations.

Starting from the procedure described in Sec. 7.1.2, we replace Eq. 7.3 with a new measurement that considers the weights w_f of the different features:

$$P_{ap}(h_j|r_i) = similarity(h_j, r_i) = \sum_{f=1}^{n_f} w_f d_f(h_j, r_i) \quad \in [0,1] \tag{7.6}$$

These weights will be updated dynamically according to:

$$w_f \propto w_f + score_f \tag{7.7}$$

where $score_f$ measures the ability of each feature to distinguish target from distractors. We must bear in mind that these weights must be normalised. After testing several score functions [147], we have decided to use a score function based on the Bhattacharyya distance among histograms (defined in Eq. 7.4). Basically, the best feature should be the one that: a) minimises the Bhattacharyya distance between the feature histogram of the region classified as target and the target model ($d_{f,1}$, in Eq. 7.8), and b) maximises the average distance between the histograms of the distractors and that of the current target model ($\bar{d}_{f,2}$ in Eq. 7.8):

$$score_f = \bar{d}_{f,2} - d_{f,1} \tag{7.8}$$

This way, the online feature weighting process adapts dynamically the weight of the visual features that compose the visual model to enhance the discrimination between the target and the distractors. This is very useful when target and distractors share a similar distribution on some features but differ on the others (e.g. clothes of the same colour but with different patterns). For instance, Fig. 7.10 shows the evolution of the weights of three features during a person following test. Observe how H, and V (colour features) are the most important most of the time, but there are certain situations where the algorithm needs to rely on EHD features (texture) to keep track of the target.

We have conducted a study [147] to test several other classical scoring functions [159], as well as to compare our solution with the case when the features are not weighted. Experimental results have shown that: a) the scoring function that we have designed provides one of the best results, b) the online feature weighting improves the results with respect to the case where features are not weighted.

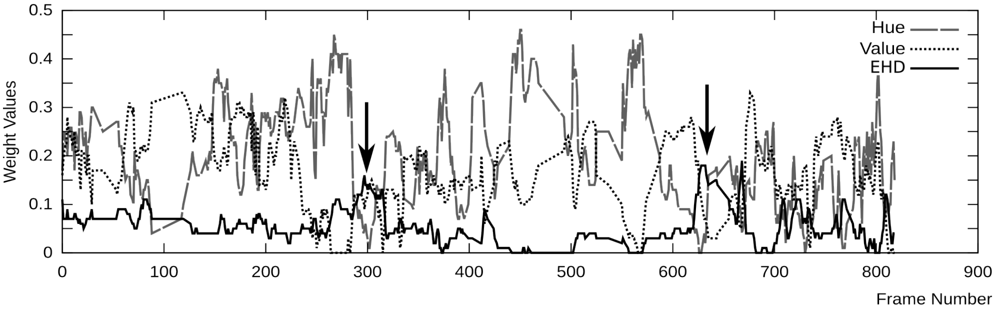
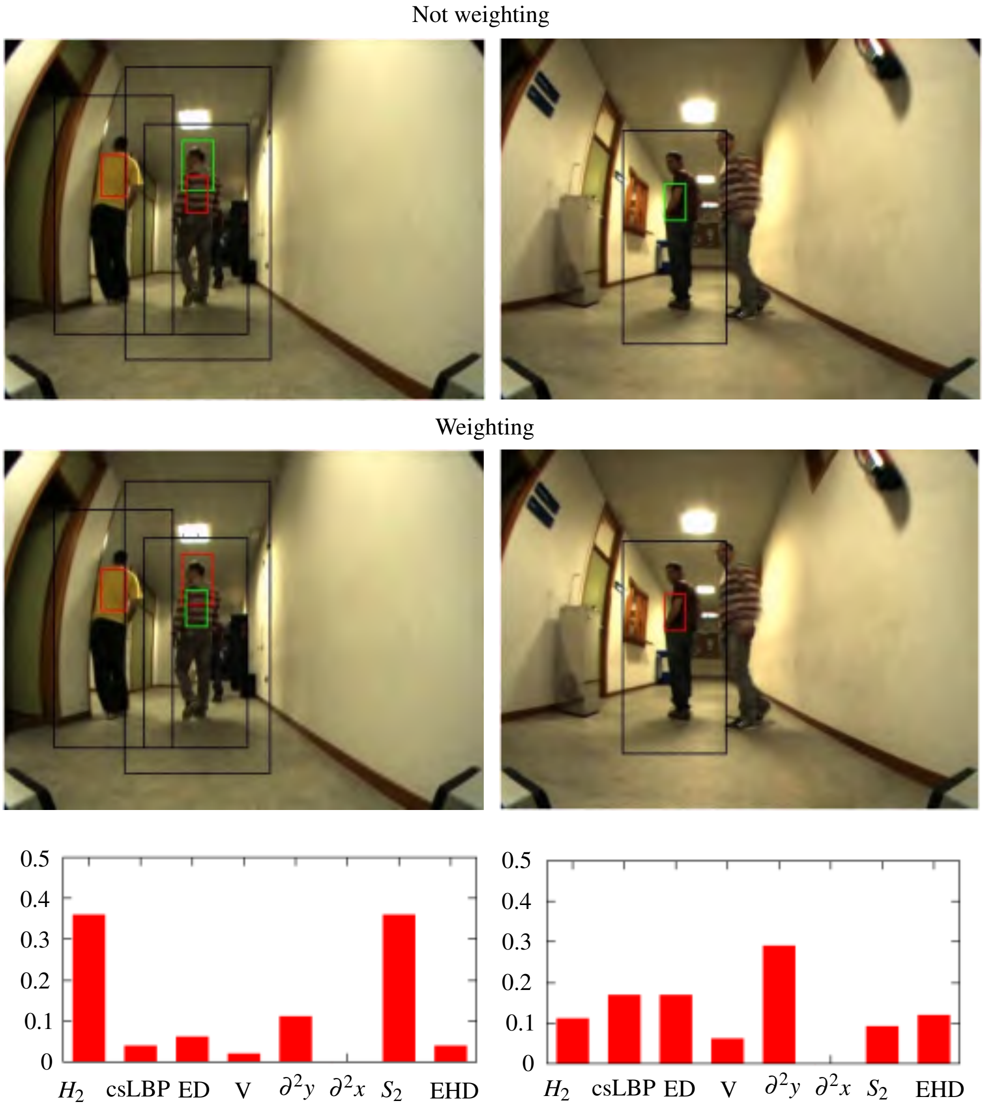


Figure 7.10: Evolution of the Hue, Value and EHD features weights during the first sequence. The arrows in the figure highlight those frames in which the EHD feature gets a higher weight than hue and and value features.

Figure 7.11 shows a graphical example of the results of this algorithm and the importance of weighting the features. We can see how the relevance of the features depends on which target is being followed and which distractors are present at each moment. In the left column of this figure we can see how the relevance of the colour features (hue-saturation) is the highest when it comes to discriminate between a yellow and a brownish torso. Nevertheless, some seconds later, in the right column, we can see how the most relevant feature is the vertical derivative ($\partial^2 y$), which is used to discriminate between a stripped torso and a flat torso with similar colour (both are brownish).



Not weighting

Weighting

Figure 7.11: Two situations where the online feature weighting selects the most appropriate features to discriminate the target (brownish with horizontal stripes) from the distractor present at each moment. Left column: the distractor's torso is of a different colour, therefore H_2 and S_2 are selected. Right column: the distractor's torso colour (when mixed with the background) is similar to the target colour, however the horizontal stripes are a good choice to discriminate amongst both torsos, therefore the weight of $\partial^2 y$ is increased.

7.1.3 Person following controller

Once the robot is able to detect and track the target, it must follow him around the environment. In this case, the robot's controller must: a) avoid obstacles and b) maintain the human at a safe distance (e.g. 1 meter), centred in the robot's field of view. We have designed an ad-hoc controller based on the potential fields technique [145]: the target position exerts an attractive force, and the obstacles around the robot determine the repulsive forces. The target position is provided by the human tracking module (Sec. 7.1.2), and the repulsive vector by the laser signature. From the sum of both attractive and repulsive forces we obtain the resultant force, that can be translated into a linear and angular velocity in order to follow the human target. On the other hand, we also want to maintain a safe distance with the human. Therefore, we will not send linear velocities other than zero to the robot when the distance with the target has been met.

The main advantage of this method is that it is very simple to implement and fast to compute, while it gives good results in most situations. However, we have observed some drawbacks, such as when many people is surrounding the robot. In this case, the repulsive forces are strong and might modify the robot's trajectory in a way that makes the robot lose its target. A similar problem arises when the robot is working in narrow spaces or when it tries to cross some doors: either it moves really slow or it is unable to follow the human.

7.2 Human robot-interaction

In the previous section we have described an essential part of the human-robot interaction: how to detect, identify, discriminate and track humans from our robot. In this section, we will describe how the robot communicates with the users. We will consider two kinds of users: the route instructor, and the visitor of an event where the robot operates. The robot's instructor will be able to: a) teach new routes to our robot, and b) record voice messages that describe points of interest within the route. On the other hand, the visitor will have the opportunity to select a route from those that were taught by the instructor. Then, our robot will follow the route path, stopping at each point of interest and reproducing the corresponding message.

We can classify the human-robot interaction of our tour-guide robot depending on the direction of the communication:

– From human to robot (gesture interaction). The instructor will be able to: a) tell the robot that he will be teaching a route, b) tell the robot when to start and stop recording a voice message within the route, and c) tell the robot that the route teaching process has finished. On the other hand, the visitors will be able to: a) ask the robot about the routes available, and b) ask the robot to show them the desired route.

– From robot to human. Our robot provides information of two different kinds: a) graphical user interface with augmented reality and voice feedback (to ease the interaction with the users), and b) voice messages and sounds while reproducing a route.

The interaction scheme that we have designed [160] for our tour-guide robot is intended to let everyone use our robot with an very short training period. Our solution consists on: a) the hand interaction (gestures), and b) an augmented reality graphical user interface (AR-GUI) with voice feedback.

7.2.1 Hand gesture recognition

The users of our tour-guide robot will be able to send commands to it using hand movements. To this extent, our robot needs to localise and track the hands and recognise the gestures. We currently recognize the following gestures: *wave, push, join hands, and the swipe-left/right* gestures. Whenever the robot recognizes one of these gestures, it carries out the actions associated to it.

First of all, when a new user is in front of the robot, the robot will ask him to hold his hands up towards the robot as illustrated in Fig. 7.12. Then, the robot analyses the range images provided by the RGB-D camera in order to look for two hand-sized objects located in front of the user at a certain height. This initialisation was in general accepted by the users who tested the system, who did not consider it an awkward requirement.

Then, the robot tracks the hands and tries to recognise gestures from the history of positions. To this extent, we model each gesture with a Finite State Machine (FSM) [161]. This reduces complexity and overhead, and has proved to perform reasonably well. The inputs of the FSM are series of points that describe lines in the 3D space filtered with the least-squares method. A gesture usually comprises between 10 and 30 consecutive points (this margin introduces speed tolerance and flexibility in the gesture trajectory).

Figure 7.13 shows four trajectory samples and the outcome of each one when it matches a swipe-left gesture. Examples A and B are accepted as swipe-left gestures showing certain gesture size and speed tolerance. Example C is rejected because although the hand trajectory

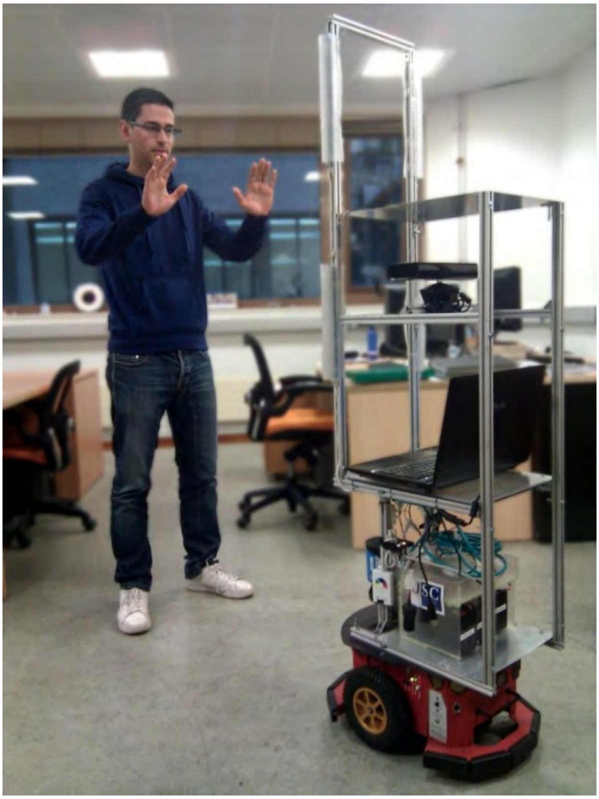


Figure 7.12: The robot is detecting a human who stopped in front of it. The human raises his hands up towards the robot. This allows the robot to detect the hands and start tracking them, in order to recognise commands from the human.

could fit the pattern, it is performed very slowly. Example D is rejected because the hand trajectory does not fit the line.

The tracking and recognition processes run in less than 30ms. Therefore, it is highly unlikely that we will lose the hands due to rapid hand movements. On the other hand, these processes employ only range data, so we will be able to detect the hands in any environment, regardless of the background colour (there is no interference between it and the hands).

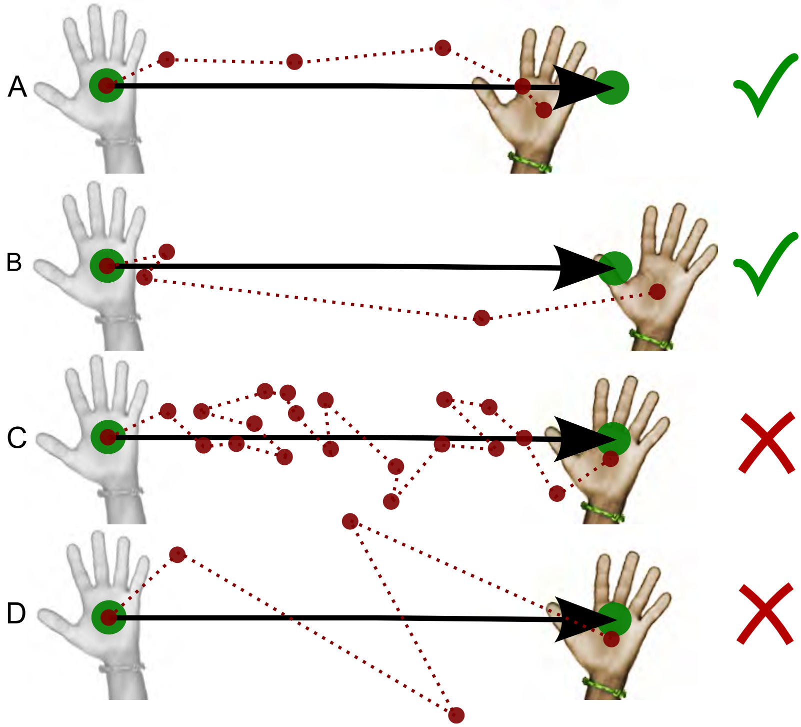
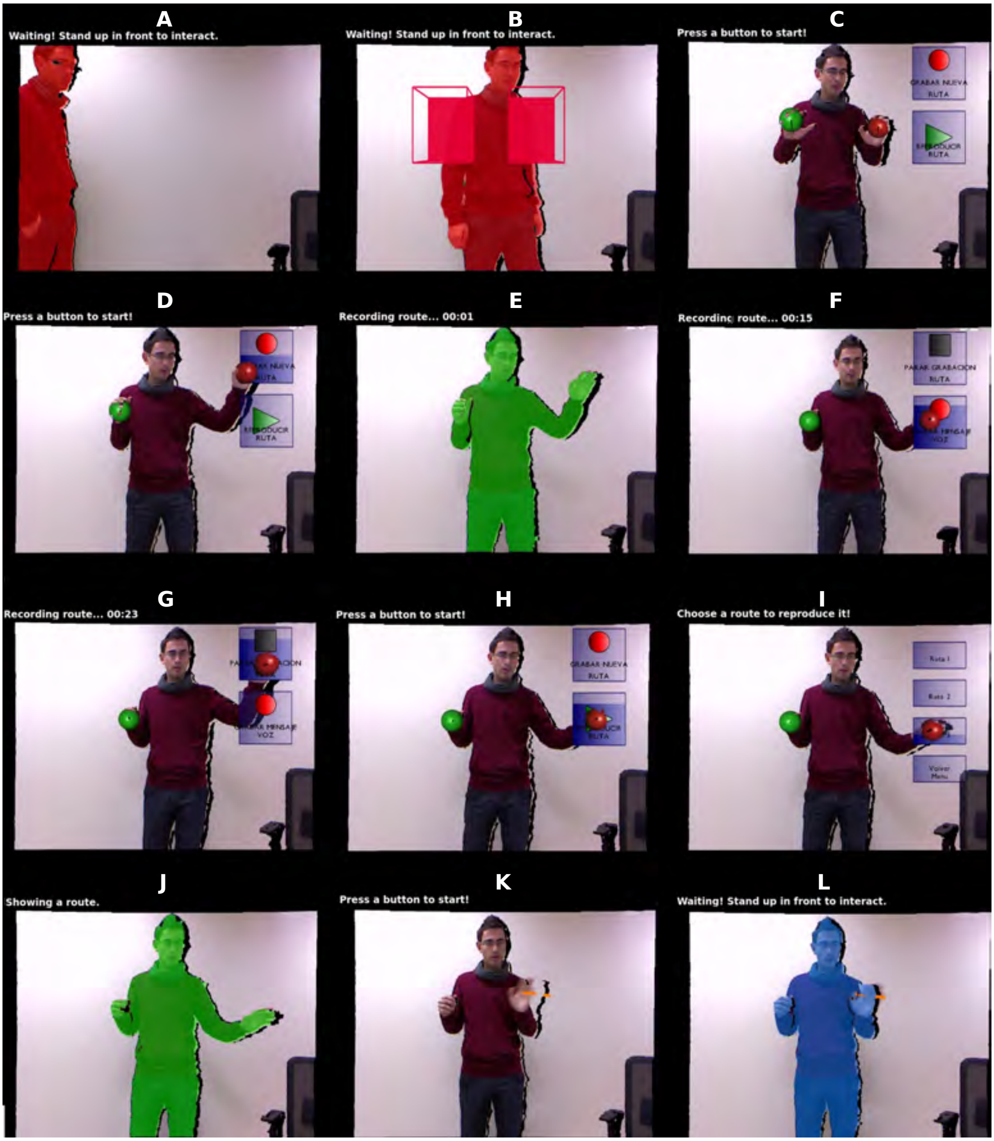


Figure 7.13: Different examples of gesture recognition and rejection for the swipe-left pattern (simplified in the 2D space). The gesture start and end points are drawn with green points. The gesture path is represented with the dotted line and the red points. The black arrow simulates the straight line that fits the points.

7.2.2 Augmented reality graphical user interface with voice feedback

In order to help the user and guide his communication with the robot, we have designed an augmented reality graphical user interface (AR-GUI) with virtual buttons which can be activated using hand movements. The AR-GUI provides on-screen information and voice feedback to keep the user informed about what the robot is sensing and what he expects from the user at every moment. The AR-GUI is the users' window to what the robot sees, thinks, and expects from him.

The AR-GUI is the central element of our robot's interaction capabilities. It is displayed in the robot's screen, and its main characteristic is that the user can see himself as in a mirror, along with several virtual elements which ease the interaction. Currently, the virtual elements that we use are (see Fig. 7.14):



Waiting! Stand up in front to interact.

Waiting! Stand up in front to interact.

Press a button to start!

Press a button to start!

Recording route... 00:01

Recording route... 00:15

Recording route... 00:23

Press a button to start!

Choose a route to reproduce it!

Showing a route.

Press a button to start!

Waiting! Stand up in front to interact.

Figure 7.14: Augmented reality graphical user interface (AR-GUI) displayed on the robot's screen. We can see how a user enters the robot's field of view, initialises his hand positions and starts giving commands to the robot.

– The human body is shaded with a random colour when he is being detected by the robot. Moreover, in case that a human is the robot's target (i.e. when following him), his body is shaded in green.

– When a human is performing the initialisation pose to let the robot detect his hands (Fig. 7.12), two red boxes will pop up, to indicate where the user should put his hands.

– Blackout of the areas not visible for the robot: i.e. if a human is outside the working range of the robot's human detector, he will not appear in the screen.

– Two virtual balls, drawn over each hand while they are being detected and tracked.

– Virtual buttons to command actions to the robot: i.e. the instructor of our robot will be able to start recording a new route by holding his hand over a virtual button for a couple of seconds.

In Fig. 7.14 we can see a sequence of screenshots from the AR-GUI taken when performing a typical interaction with our guide robot. First, when a human walks into the robot's field of view (Fig. 7.14.A), his body is coloured in a random colour distinct from green. Then, when that user is standing in front of the robot a voice message asks him to *please, put your hands up towards the robot*, and immediately two red boxes are drawn at the positions where the user should put his hands (Fig. 7.14.B). This step helps a robot's user to correctly perform the initialisation pose.

When the robot has detected the user's hands, the hand positions are marked with two virtual balls. At the same time, two buttons (one for recording and another for reproducing routes), appear on the screen (Fig. 7.14.C), and a voice message requests the user to select an action. Then, the user of our example can activate the *record a new route* button by holding his hand over it for a couple of seconds, as shown in Fig. 7.14.D. At that point, the AR-GUI shades his body in green (Fig. 7.14.E) and the robot starts following the user around the environment. Again, the user receives information via voice about the action: the robot is now *recording a route*. At any point, the user may record a voice message by activating the button to record a voice message (Fig. 7.14.F), speaking loudly to the robot, and then stopping the recording by pressing a button again. The message will be reproduced immediately afterwards to offer the user a chance to record it again. The voice message will be stored with the route information, and can thus be reproduced at the same point of the route where it was recorded. Once the user has taught the new route to the robot, he activates the *stop the route recording* button (Fig. 7.14.G), and the information on the route is stored in the robot, so that the robot can reproduce it afterwards.

Once the robot has learnt several routes, it can offer its guidance services to anyone who stops in front of it, and asks for a route to be reproduced by activating the button as shown in Fig. 7.14.H. This button leads to a sub-menu with as many buttons as routes are available (Fig. 7.14.I), while the robot reproduces a voice message asking to *choose one route*. When the user choses a route, the robot finishes the hand interaction (Fig. 7.14.J) and starts reproducing this route. Finally, when a user decides that he no longer wishes to interact with the robot, he can wave off. A voice message saying *goodbye* is reproduced and his body will appear shaded in a colour other than green. This process is shown in Fig. 7.14.K and Fig. 7.14.L.

It is important to note that our guide robot allows humans to walk freely. In the case that the user is walking too fast for the robot, he is informed by a voice message to *please, walk slower*. Also, if the user walks outside the working range, he is informed with a voice message saying, *I lost you*, and then the user is responsible for coming back to be recognised by the system.

7.3 The route recording and reproduction architecture

In this section, we will describe the architecture that allows our robot to learn routes from an instructor and to reproduce them in front of users.

7.3.1 Route recording architecture

Figure 7.15 illustrates the route recording process, as well as the rest of the elements involved in the process (the localisation algorithm, the AR-GUI, and the voice recording process), and the communication between all of them. We have integrated our system using ROS (Robot Operating System). The localisation algorithm is an asynchronously, always-working, independent node. That is, it is continuously estimating the robot's pose with every new sensor information available. During the entire recording, the pose estimation should be correct and precise (errors below 1 meter). This is one of the reasons why we have made an special effort into obtaining a robust localization solution. On the other hand, the route recording process may be launched by the instructor at any time. Then, it remains in the background, waking up in two cases:

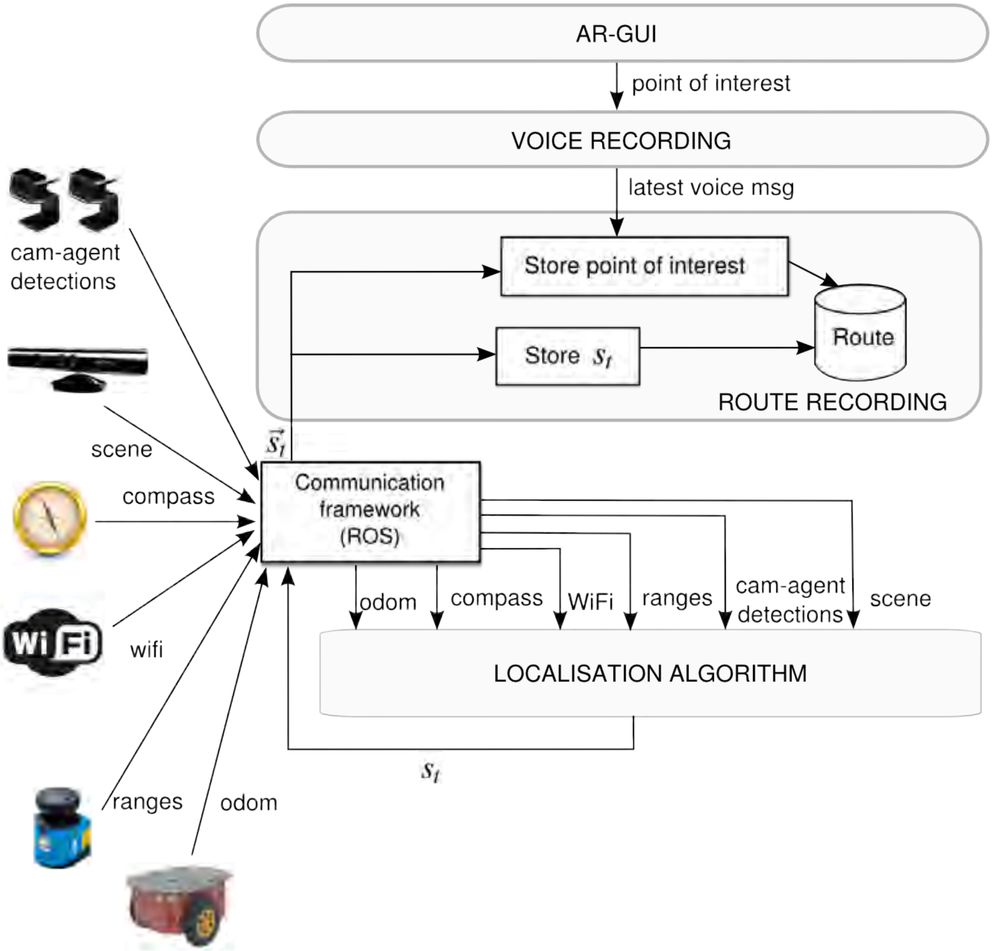


Figure 7.15: Schematic representation of the route recording process of our tour-guide robot, as well as the rest of the elements that take part in this process.

– When the localisation algorithm estimates a new pose ($\vec{s_t}$). In this case, it will decide whether to store $\vec{s_t}$ or not based on how far $\vec{s_t}$ is from the latest pose stored in the route.

– When the AR-GUI informs about a *point of interest*. In this case, it will store the voice message recorded by the instructor, alongside the latest estimated robot pose ($\vec{s_t}$).

Therefore, the routes of our robot are composed of a sequence of robot poses (which we call *way-points*), intercalated with *point of interest* (which consist of a robot's pose and a voice message). This data will be used by the route reproduction process to mimic the path of the robot and to reproduce the voice messages at the same places where they were originally recorded.

7.3.2 Route reproduction architecture

A visitor of an event can demand any route from our tour-guide robot, regardless of the place where the robot is. This means that the first task that our robot has to accomplish is to travel to the first point of the route from its current position. Then, it will need to reproduce the route in the most accurately, dealing with the dynamic nature of the environment.

In order to describe this process, we can define a route r as a list of robot's poses $\{\vec{s_1^r}$, ..., $\vec{s_{n_r}^r}\}$ where n_r is the number of stored poses (or *way points*). Some of these *way points* can actually be *points of interest*, if they are associated with voice messages. The goal of the route reproduction behaviour of our robot is to visit each of those points in the specified order, and to reproduce the associated voice message if there is one.

Global planner: navfn

In Fig. 7.16 we can see that the route reproduction process starts when the AR-GUI demands a route (*start route*). At this moment, the behaviour picks a *way point* ($\vec{s_1^r}$) and plans a valid path from the robot's pose ($\vec{s_t}$) to that point. These points can be separated several dozen of meters, and planning a valid path might not be a trivial problem. In our case, we have chosen a classical solution, Dijkstra's algorithm [162]. It is a good solution in terms of efficiency and simplicity, and it was able to provide good results in all cases that we have tested. More specifically, we have used an improved version which is available in the ROS global navigation planner *navfn* [1].

This algorithm requires us to build a cost-map of the environment where the robot operates (*global costmap* in Fig. 7.16). This costmap will be used as the graph in which Dijkstra's algorithm will search for a minimum cost plan between the two points. In order to build the costmap of the environment we need: a) an occupancy map of the environment (which is created during the deployment stage of the robot, as described in Sec. 5.5), b) the robot's

[1] `http://wiki.ros.org/navfn`

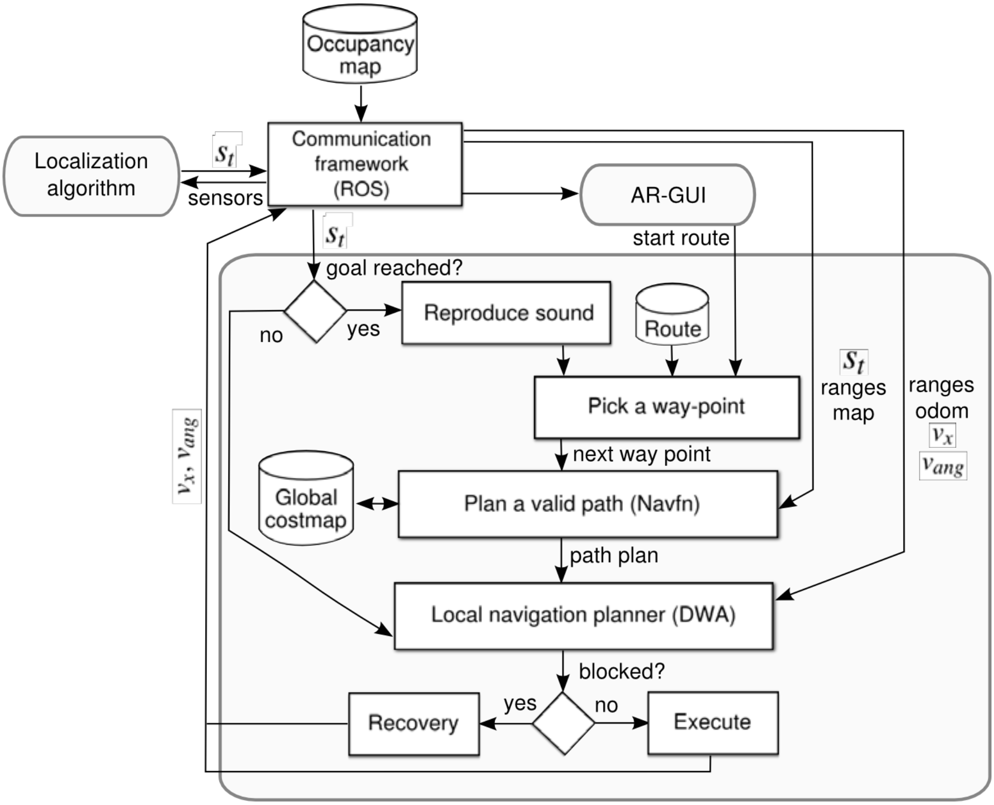


Figure 7.16: Schematic representation of the navigation node used for route reproduction in our tour-guide robot. It is a ROS based node with a global planner for path planning (navfn) and a local planner based on the Dynamic Window Approach (DWA) for navigation.

estimated pose $\vec{s}_t$, c) the current data from the laser scanner, and d) the radius of the robot min_r.

The costmap is initialised using the occupancy map of the environment: we assign the highest value in the costmap to those cells that are marked as occupied in the occupancy map, and a low value to those cells that are marked as free space. Then, using the robot's estimated pose and the laser signature, we create new occupied cells in the costmap, and in the same way, we also release cells that were previously marked as occupied. We need to do this because there might be obstacles in the environment that are not in the map, such as humans, but there might also be furniture which has been moved, releasing previously occupied space. The last stage to build the costmap consists in marking as occupied those free cells that are closer than min_r to an occupied cell, as the robot will not be able to visit those cells without colliding. This process is known as *inflating the obstacles* of the map.

Local planner: Dynamic Window Approach

At this point, our algorithm will use the costmap to plan a valid path from the robot's position towards the start point of the route. This plan will be handed out to a second element: the local navigation planner (Fig. 7.16). The local planner is the element responsible for moving safely our robot in the environment, while it follows the path plan. In order to face this problem, we use a popular algorithm that is able to solve the task with an small computational overhead: the Dynamic Window Approach (DWA) [2] [163]. Basically, this algorithm simulates various trajectories which are generated by sampling the linear and angular velocities of the robot. Then, each simulated trajectory is scored according to different criteria. In the end, the trajectory with the highest score is chosen, and the velocities which generated that trajectory are executed in the robot's base. This algorithm allows us to continuously re-evaluate the robot's trajectory, and thus avoid dynamic obstacles such as persons walking in the robot's path.

An important element of DWA is how we score the simulated trajectories. In each cycle, we need to create a second costmap, which we call the local costmap. In order to do this, we use the laser signature and the computed path plan. First, obstacles are inflated in the same way than in the other costmap, and then we assign different weights to the free space cells, according to how far they are from the planned path. Therefore, using this local costmap, trajectories can be scored taking into account how close they are to the desired path and on different parameters like proximity to obstacles or achieved speed.

Route execution

When our tour-guide robot has reached the start point of the route, it will make a short beep (*Reproduce sound* in Fig. 7.16), pick the next *way point* and start a similar process to navigate towards it. That is, we will compute a new path plan using Dijkstra's algorithm and safely

[2] `http://wiki.ros.org/dwa_local_planner`

execute it using the Dynamic Window Approach. We repeat this process until the last point of the route is reached or the visitor stops the robot.

Additionally, our route might contain *points of interest*, which are treated similarly to a normal way-point. The main difference is that the robot will stop in that point for a while, until the associated voice message is fully played back (*Reproduce sound* in Fig. 7.16).

Finally, we have also implemented several rules to increase the flexibility of the robot, and to deal with unreachable route points. First, we consider a point as visited when the robot moves at a close distance (0.5m from the route point). This prevents our robot from spending many seconds moving slowly to reach a goal pose with a high precision, and thus increases the overall robot's speed. Second, when our robot's detect that a point within the route is unreachable, i.e. a human is blocking the path, we proceed in two different ways. In the first one, the robot will clear his local costmap from the nearest obstacles and perform an in-place rotation to check for alternative paths. If there is still no valid path, the robot will perform a more aggressive reset of the costmap, completely clearing it and performing another in-place rotation to rebuild it. If this also fails, we will consider the second alternative: the robot is completely blocked, and therefore we will allow it to skip that *way-point* and move to the next one. In case that the robot skips a point, it will make a recognisable error sound to let the user's know that there was a problem.

7.4 Tests in the robotics laboratory

In this section, we want to illustrate the performance of the route reproduction module that we have designed for our tour-guide robot. For this reason, we will describe two routes that we have recorded in our robotics laboratory (Fig. 7.17). In this laboratory we have settled an indoor environment with five rooms. We have chosen this laboratory as the first test-bed for the route-management in our robot because it is similar in size ($100m^2$) and space arrangement (five rooms) with most homes or offices nowadays.

The routes that we have recorded are illustrated in Fig. 7.18 with a dotted line (robot's trajectory during recording) and small dark grey circles (recorded way-points). The result obtained when reproducing the route is illustrated in Fig. 7.18 with a light grey line. In these examples we can see how the robot has to previously find a valid path from its initial pose (C in Fig. 7.18) to the initial point of the route (A in Fig. 7.18). Once the robot has reached the first point of the route, it moves from one *way-point* to the next one, avoiding obstacles in the environment (two humans walking randomly) without separating from the path more than a few dozens of centimetres. Moreover, when the robot reached a point of interest, it played the voice message recorded by the instructor. Finally, when the robot finished the reproduction of the route, it played a bell sound and waited for new commands from any other human.

The first route (Fig. 7.18, left) was recorded in 56 seconds, and it is 15.3 m long. In this route no points of interest were recorded by the instructor, but it is interesting in order to notice how fast can anyone teach a route path to the robot in a home-like environment with narrow corridors. In this experiment, the robot had to travel 29.2 m in order to find the first point of the route and mimic the complete route path.

The second route (Fig. 7.18, right) was recorded in 155 seconds, and it is 40.8 m long. When reproducing this route the robot travelled 52.8 m. In this route, the instructor recorded three voice messages at the points of interest. These messages were very short: around 5 seconds each one. Our tour-guide robot allows the record and reproduction of a complex route in few time by non experts.

Moreover, we have showcased our tour-guide robot during many demonstrations in our research centre (CITIUS). These demonstrations are part of a program for visitors, in which we show them the results of our research activity. In this regard, Fig. 7.19 shows a picture of a visitor teaching our robot a route inside our laboratory (Fig. 7.17). Then, the rest of the

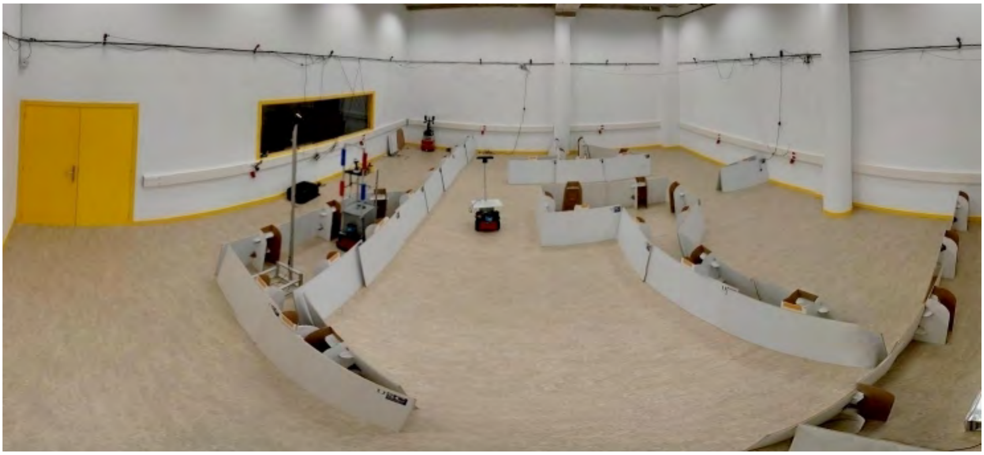


Figure 7.17: CITIUS robotics laboratory ($100m^2$) where we have conducted several route reproduction tests. We set up an environment with 5 rooms, and recorded and reproduced several routes while two humans were walking around.

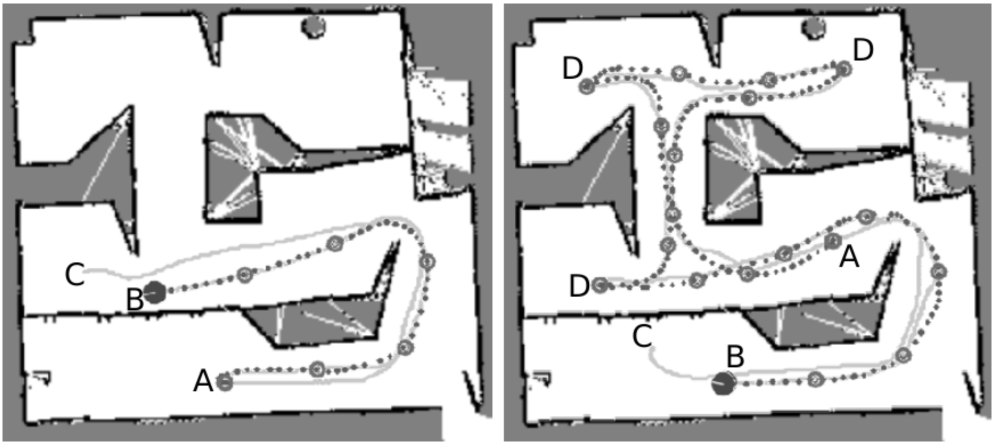


Figure 7.18: Robot trajectories while recording (dotted) and reproducing (light grey) two routes. The small circles represent the stored way-points. The points marked with the letters *A* and *B* represent the first and last points of each route, respectively. The points marked with letter *C* are the location of the robot when it was asked to reproduce the route, and the points marked with the letter *D* in the second route are the locations of *points of interest* as specified by the route teacher.

pictures show how our robot reproduces a route that it has just learnt. In many demonstrations our tour-guide robot did not have problems to reproduce the routes, however in some of them it had to perform in-place rotations to clear out blocked space. This situation happens when the environment differs significantly from the one in the occupancy map of the robot, which might be a common problem when the visitors are mainly kids who play around with furniture in our laboratory.

7.5 Discussion

In this thesis, we have integrated our intelligent space with a general purpose guide robot that we have developed in the past, as an specific example of application. This robot is aimed to operate in different environments, such as museums, conferences, or robotics demonstrations in research centres. Our robot is able to detect and track people around him, follow an instructor around the environment, learn routes of interest from the instructor, and reproduce them for the visitors of the event. Moreover, the robot is able to interact with humans using gesture recognition techniques and an augmented reality interface.

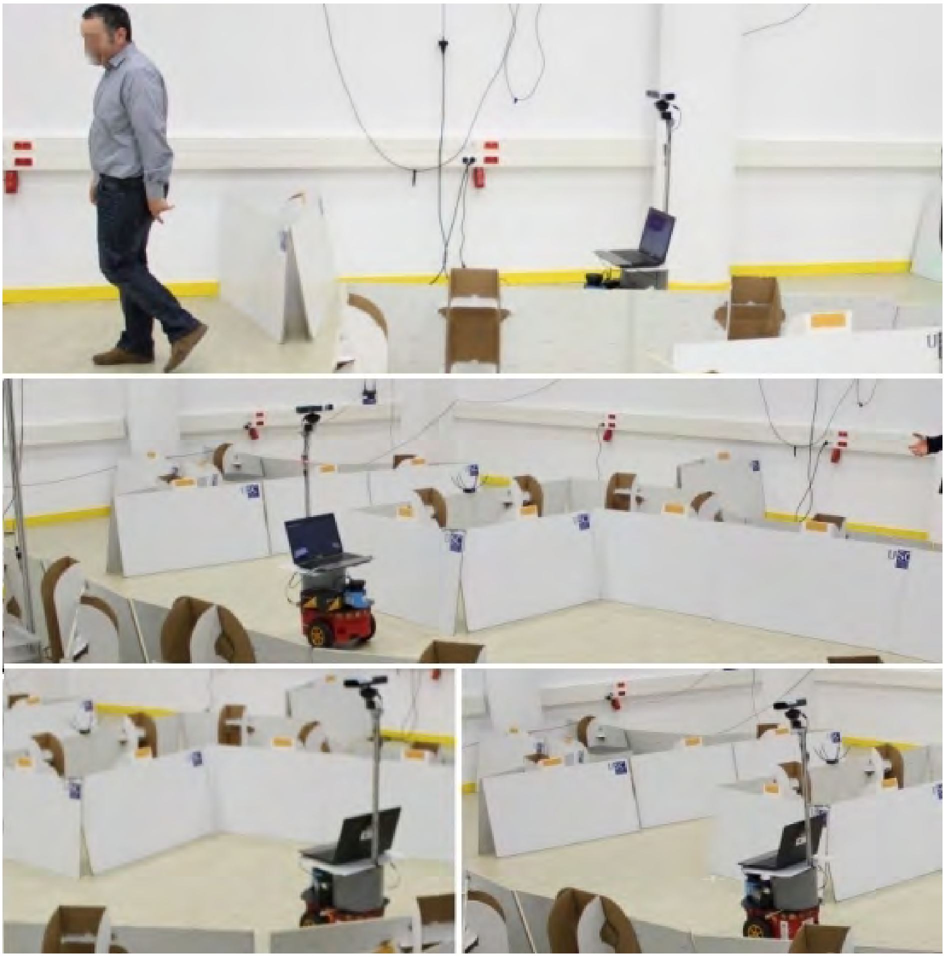


Figure 7.19: Pictures of several demonstrations in which we have tested the route recording and reproduction behaviours of our tour-guide robot. In the top picture the human is teaching a route to the robot, in the rest of the pictures the robot is reproducing it.

This robot has been shown and tested multiple times in research centres, as well as in other environments such as museums, primary schools, high-schools and universities. During these demonstrations, our robot has been used successfully by people of different ages and backgrounds. Moreover, we have tested the usability of our robot based on the quality standard ISO/IEC-9126-4, by monitoring the users while they interacted with our robot and by gathering their opinions through polls.

CHAPTER 8

SITUM: INDOOR POSITIONING FOR SMARTPHONES

We believe that we, as researchers, have the responsibility to transfer the knowledge that we create to society. This means, of course, to publish articles, to educate students, to fill patents and to participate in divulgative activities. In addition, we believe that one of our main responsibilities is to actually apply the technologies we create to solve the problems of society, to create commercial value.

For this reason, in 2013 we started studying the possibility of creating an indoor positioning technology for smartphones. Nowadays, this germinal idea has given birth to a company called Situm Technologies, a spin-off of the University of Santiago de Compostela. Situm has grown up to a team of 11 members and is already selling its products and generating revenue.

In this chapter, we will provide a brief description of Situm: its initial idea, its technology and products, its business model, its team, its achievements and its future.

8.1 The problem, the idea, and the first contact with the market

Every product must solve a real problem or need for real people. This is a basic and fundamental law when the goal is to launch a product into the market. No matter how unique, technologically advanced, elegant, disruptive and well designed the product is, if it does not solve a real problem for which customers are willing to pay, it is worthless.

In our case, the problem was that the GPS does not work indoors. We knew that the GPS had enabled the proliferation of Location Based Services (LBS), such as guidance for drivers and pedestrians, but these services could only be used outdoors. If we had a reliable indoor location technology (a GPS for indoor spaces), a new world of services would be possible. The market growth estimations agreed with this perspective. In fact, the global indoor location market was estimated to be $935.05 million in 2014 and expected to grow to $4,424.1 million by 2019. This represents an estimated Compound Annual Growth Rate (CAGR) of 36.5% from 2014 to 2019 (source: MarketsAndMarkets[1]). This told us two things: 1) that the indoor positioning problem was relevant for a big number of potential customers, and 2) that customers were willing to pay for solutions related to this problem.

Under this perspective, we identified a number of sectors where indoor positioning technologies could be applied: shopping malls and commercial areas, supermarkets, museums, hospitals, airports, etc. We envisioned a number of use cases for each sector. With an adequate solution, passengers on airports could get step-by-step guidance to their boarding gates, hospitals and factories could know the position of their employees to respond faster on emergency situations, commercial centres and supermarkets could know the position and trajectory of each costumer and obtain relevant information for marketing studies, etc. Then, we identified local people involved in each sector, and met with some of them to present them our ideas. Our objective was to gather information about the different applications of an indoor positioning technology in each sector, and to propose the development of pilot projects to test the use cases. The feedback from the first meetings was very positive, so we decided to build a prototype of an indoor positioning technology.

8.2 The technology

In 2013, indoor positioning companies (e.g. Ekahau) were offering technologies that required the installation of expensive hardware and/or could only track the position of specific hardware devices (e.g. hardware tags). These technologies could only be applied to domains where it was be possible to afford the installation costs and where the users would agree on carrying the tracking devices (e.g. hospitals or industrial plants). Moreover, from our experience with Ekahau, an state of the art system, we knew that these technologies could not offer enough precision and robustness for demanding applications, such as turn-by-turn guidance.

[1] http://www.marketsandmarkets.com/PressReleases/indoor-location.asp

We believed that an indoor positioning technology that aspired to reach the mass market should be compatible with modern smartphones, because their adoption was on track to become universal. Therefore, we developed our own indoor positioning technology for these devices. This technology is based on:

1. Intelligent sensor fusion. Our technology is able to fuse the information received from the different sensors of the smartphone: WiFi, Bluetooth Low Energy (BLE) and magnetometer. This makes our technology robust and versatile.

2. Inertial navigation. To achieve the highest precision, our technology estimates the displacement of the user from the readings of the accelerometer and gyroscope of the smartphone.

3. Cloud computation. A server receives the sensor information from every smartphone, estimates its position and sends it back (Fig. 8.1).

The results were very positive. First of all, our technology could achieve high precision (up to 1 meter) and robustness. Second, it could be configured very quickly in any building, mainly because it could work with existing WiFi networks without installing additional hardware. Finally, unlike most competitors, our technology could work in multi-level buildings.

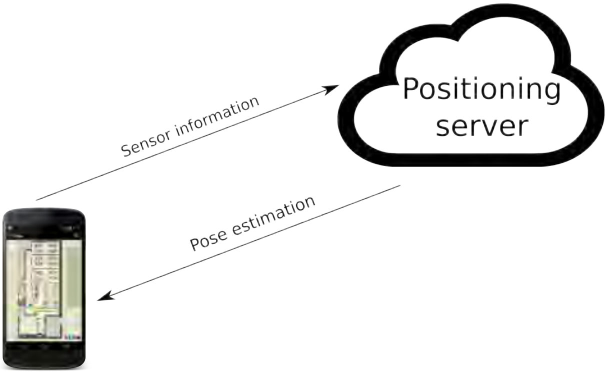


Figure 8.1: Cloud-based indoor positioning technology.

This technology, by itself, did not solve any problem for which a customer was willing to pay. Therefore, we created a product that used this technology, and that could provide value to potential customers.

8.3 The product

On top of our indoor positioning technology, we built a Software Development Kit (SDK), illustrated in Fig. 8.2. The idea was to enable any developer to build apps and location based services based on our technology. Other than the cloud-based positioning server, the SDK consists on two components:

1. Situm Dashboard. Web panel that allows to configure our indoor positioning system: create a new building and upload its map, create points of interest to where users can be guided (e.g. emergency exits), create alerts that users will receive when passing through a certain area (e.g. shopping promotions), etc. In addition, this panel shows positioning statistics: number of visitors within a period, popularity of each area of the building, etc.

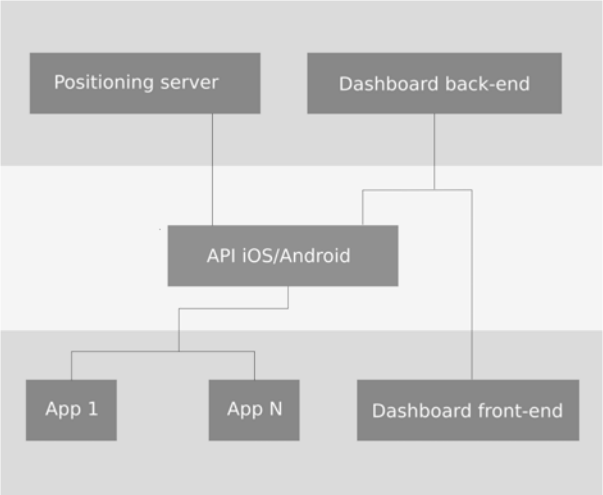


Figure 8.2: Situm Software Development Kit.

2. Situm API. Set of tools oriented to the development of apps for Android/iOS. These tools allow any developer to integrate our technology into any smartphone app, in a fast and easy manner.

In paralell, we worked on the definition of a business model that would allow us to sell this product to our customers.

8.4 The business model

Currently, we offer our products to software development companies that want to create location based apps for their clients. The revenue model is based on license fees and royalties on the final product revenues. In addition, in some sectors we are exploring the possibility of developing final products ourselves.

8.5 The team

Situm has 11 team members at the moment. All the founders are or have been researchers of the CITIUS: Dr. Victor Alvarez-Santos (Chief Executive Officer), Adrian Canedo (Chief Technology Officer) and Dr. Cristina Gamallo (R&D manager). Each of them has worked for several years in the topics of indoor positioning and robotics. They have more than 25 scientific publications, a wide experience in software development, and business training. The development team is completed by 6 developers with a wide experience on their respective areas: Felix Queiruga (web back-end), Maria Reino (web front-end), Pablo Vazquez (web front-end), Abraham Barros (mobile), Javier Casanova (mobile), and Dario Villar (C++). Finally, Miguel Perez is in charge of corporate communication, and Daniel Sanmartin of finance.

The company also counts with the invaluable support of the other three founders, that play the role of scientific advisors on the topics of indoor positioning and machine learning: Dr. Senén Barro and Dr. Roberto Iglesias (University of Santiago de Compostela, Spain) and Dr. Carlos V. Regueiro (University of A Coruña, Spain).

8.6 Achievements

Situm has already developed the core of the indoor positioning technology and product. It has installed the technology in 9 buildings: 4 in Santiago de Compostela, 4 in Madrid and 1 in Barcelona. It has also several projects going on with clients of different sectors, the most important as a technology provider of Telefónica. In this project, it will install the indoor positioning system in 22 Galician hospitals and health centers.

In addition, Situm has received several recognitions:

1. Winners of the Smart Ideas contest 2014 (Smart City program, Council of A Coruña).
2. First national price in Yuzz 2014 (Banco Santander, CISE).
3. First price in Galicia Open Future 2015 (Xunta de Galicia, Telefónica).
4. Best elevator pitch at the Singularity University events (supported by entities such as NASA, Google and the IMPACT accelerator).

Finally, Situm has been backed financially by S. C. R. Unirisco (venture capital) and ENISA.

8.7 Future

Situm started 2 years ago as a compromise between 3 PhD students and 3 senior researchers, that wanted to apply their know-how to build a product that could enter the market. Nowadays, Situm has a team of 11 members that has developed a tangible product that is already generating revenue (more than 100.000 € so far). In the future, Situm aspires to maintain a technologically advanced product able to evolve according to the requirements yet to come, and to provide its technology in buildings all around the world.

Chapter 9

Conclusions

The goal of this thesis was to develop robotic systems that are able to operate robustly in dynamic and non structured environments. In addition, we wanted to be able to deploy these systems in different environments in a fast and easy manner, and to provide robots with initiative to respond proactively to the needs of the users. To this extent, we have combined technologies from ubiquitous computing, ambient intelligence, and robotics. More specifically, we have built an intelligent space that consists of a distributed network of intelligent cameras and autonomous robots. The cameras are spread out on the environment, detecting situations that might require the presence of the robots (call events), informing them about these situations, and also supporting their movement in the environment. The robots, on the other hand, must navigate safely within this space towards the areas where these situations happen. With this proposal, our robots are not only able to react to events that occur in their surroundings, but to events that occur anywhere. As a consequence, the robots can react to the needs of the users regardless of where the users are. This will look as if our robots are more intelligent, useful, and have more initiative. In addition, the network of cameras will support the robots on their tasks, and enrich their environment models. This will result on a faster, easier and more robust robot deployment and operation. The intelligence needed to achieve a robust performance of the system can be distributed amongst all the agents, cameras and robots in our case. In this thesis, we have explored two alternatives: collective intelligence and centralised intelligence.

Under the collective intelligence paradigm, intelligence is fairly distributed among agents. Global intelligence arises from individual interactions without the intervention of a central agent, similarly to self-organization processes that occur in nature. We have assumed that robots can operate in a priori unknown environments when their behaviour emerges from the interaction amongst an ensemble of independent agents (cameras), that any user can place in different locations of the environment. These agents, initially identical, observe human and robot behaviour, learn in parallel, adapt and specialise in the control of the robots. To this extent, our cameras were able to detect and track robots and humans robustly, under challenging conditions: changing illumination, crowded environments, etc. Moreover, they could discover their camera neighbours, and guide the robot navigation through routes of these cameras. Meanwhile, the robots only had to follow the instructions of the cameras and negotiate obstacles in order to avoid collisions. In order to accomplish this, we have achieved the following milestones:

1. We have designed and implemented a network of cameras that can be deployed in a fast and easy manner in different environments. Our cameras have high computation capabilities and can work connected to a wall plug or using their own batteries with an autonomy of up to 4 hours. Moreover, they can communicate wirelessly among them and with the robots.

2. We have developed a software architecture that controls the interaction amongst all the agents of the system. This architecture considers the existence of two different agents: robot-agents and camera-agents. The architecture is fully distributed and very scalable. This allows us to remove or introduce agents in the system with minimum reconfiguration requirements.

3. We have studied the possibility of detecting and tracking the robot from the cameras using object recognition techniques and techniques based on the detection of passive markers mounted on the robot (markers that do not emit light). However, none of these techniques has given satisfactory results in the context of application of this thesis. Therefore, we have designed and developed an algorithm for robot detection and tracking based on active markers. Our algorithm has shown to be very robust in real experiments, and can work in real-time. Moreover, we have shown that we can detect and track multiple robots, and even identify each of them, while the computational requirements scale linearly with the number of robots.

4. We have developed a system to detect situations that may require the presence of the robots from the cameras. Specifically, we have developed algorithms for the detection

of: 1) people waving at the cameras, 2) people standing in certain areas. Both algorithms have been proposed as specific examples of call events that might require the presence of our robots. We would like to point out that the concept of call event can be accommodated to a wide range of robotics applications. For instance, we could detect water spills to aid a cleaning robot, potential threats to aid a security robot, etc. Moreover, call events could be triggered by other kind of devices: for instance, the user could use a "tag" or an smartphone to call the robot.

5. We have developed a set of algorithms that allows the system to work under the collective intelligence paradigm. With these algorithms, the cameras where able to: 1) detect their camera neighbours, 2) construct routes of cameras through which the robot can navigate, 3) assign a call event to an available robot, 4) support the robot navigation towards this call event. We have shown that our cameras can establish neighbourhood links when their Fields of View (FOVs) overlap, attending at simultaneous detections of the robot. In other case: 1) the robot can construct maps for navigation between neighbour cameras (e.g. occupancy maps from a 2D laser scanner), 2) the cameras can detect their neighbourhood relationships by re-identifying people walking around. Real world experiments have shown the feasibility of our proposal to work with "naive" robots.

On the other hand, under the centralised intelligence paradigm, one type of agent is assigned much more intelligence than the rest. In this case, the network of cameras will be still responsible of detecting the call events, and they will support the movement of the robots in the environment. In particular, they will inform the robot when they see it. Nevertheless, the robot will be the agent that makes most decision making, and its performance will have the highest importance. This option implies that the robot is able to estimate its position accurately and robustly, therefore most of the intelligence was devoted to the task of self-localisation and navigation. Our robots implemented multi-sensor localisation strategies that fuse the information of complementary sources, in order to achieve a robust robot behaviour in dynamic and unstructured environments. Under this paradigm, we have achieved the following milestones:

1. We have performed an experimental study about the strengths and weaknesses of different information sources to be used for the task of robot localisation: a 2D laser range finder, a magnetic compass, a commercial WiFi localisation system, and the network of external cameras. The study shown that no source performs well in every situation,

but the combination of complementary sensors may lead to more robust localisation algorithms.

2. We have developed a robot localisation algorithm that combines the information from multiple sensors. This algorithm is able to provide robust and precise localisation estimates even in situations where single-sensor localization techniques usually fail. It can fuse the information of an arbitrary number of sensors, even if they are not synchronised, work at different data rates, or if some of them stop working. We have tried this algorithm with the following sensors: a 2D laser range finder, a magnetic compass, a WiFi reception card, a radio reception card (433 MHz band), the network of external cameras, and a camera mounted in the robot.

3. We have tested our positioning algorithm with multiple sensors, and we have proposed observation models for each of them:

   a) 2D laser scanner: we have proposed a model that takes into account the differences among the expected and received laser signature, both regarding their shape as well as their concrete values.

   b) Compass: we have proposed a model that takes into account the magnetic distortions of the environment, in order to minimise their influence in orientation readings.

   c) WiFi reception card and radio reception card (433MHz band): we have proposed and analysed three different techniques that allowed us to perform indoor positioning based on the power of the wireless signals received.

   d) Camera network: we have proposed a model able to estimate the position of the robot based on the list of cameras that detect it at every instant.

   e) Camera mounted in the robot: we have developed a model based on scene recognition that allows us to estimate the most probable robot position based on an image captured by the camera mounted on the robot.

4. We have performed a complete experimental study of the localisation algorithm, in both controlled and real conditions during robotics demonstrations in front of users. We have studied the behaviour of the algorithm when using each single sensor and most of their combinations. Our main conclusion was that, with statistical significance, the fusion of

complementary sensors tends to increase the precision and robustness of the localisation estimates.

5. We have designed wireless transmitters (motes) and we have integrated them into our indoor positioning algorithm. Our motes use an inexpensive, low-power sub-1-GHz system-on-chip (CC1110) working in the 433-MHz ISM band.

6. We have studied the performance of our positioning algorithm when the wireless transmitters in the environment are able to vary their transmission power. Through an experimental study, we have demonstrated that this ability tends to improve the performance of a wireless positioning system. This opens the door for future improvements in the line of active localisation. Under this paradigm, the robot would be able to modify the transmission power of the transmitters in order to discard localisation hypotheses proactively.

7. We have integrated our intelligent space with a general purpose guide robot, as an specific example of application. Our tour guide robot is aimed to operate in different social environments, such as museums, conferences, or robotics demonstrations in research centres. Our robot is able to detect and track people around him, follow an instructor around the environment, learn routes of interest from the instructor, and reproduce them for the visitors of the event. Moreover, the robot is able to interact with humans using gesture recognition techniques and an augmented reality interface.

8. Finally, we have designed a methodology that allows any user to deploy and calibrate this system in a short period of time in different environments.

The architectures, behaviours and algorithms that have been proposed within this thesis can go beyond the scope of robotics. In the coming years, robots are called to populate new environments like hospitals, museums, or offices. These new generations of robots will need to be able to navigate autonomously and perform robustly under difficult conditions: challenging illumination, people walking around the robot, etc. Moreover, robots will need to be aware of humans regardless of where they are, interact with them, follow them, and cooperate with them. Even more, there is a current trend (“Internet of Things”) to populate our environments with networked devices with processing and sensing capabilities, such as cameras, Bluetooth beacons, and all sorts of sensors. The robotics community should not miss the opportunity of enriching and improve the robots’ behaviours and services with the possibility that this new trend offers. For all these reasons, we believe that what has been proposed within this thesis (in part or as a whole) can be the basis for future robot architectures and behaviours that might even reach the market in the coming years.

Finally, we would like to highlight that the indoor positioning knowledge derived from this thesis is the base of Situm Technologies, a company that provides indoor positioning technologies for smartphones. This company has received several recognitions, financial backing from a venture capital fund, and has already generated more than 100.000 € in revenue.

9.1 Future work

A thesis is not a closed work, but it should open lines of work that may lead to interesting results in the future. In this regard, we believe that this thesis leaves room for future improvement and for new interesting research lines. Just to mention but a few:

1. We have explored two different solutions: collective intelligence and centralised intelligence. We demonstrated that it was possible to obtain robust navigation of robots under the collective intelligence paradigm, by supporting them with a network of intelligent cameras. We have seen as well that this could lead to poor performance if the cameras failed. As an alternative, we moved most of the intelligence to the robot, leading to a robot that was able to localise themselves and navigate autonomously. This approach had a problem too: the whole system depended on the construction of a previous map of the environment, so if the map could not be built for any reason, the robot would fail. We believe that it would be worth to explore an intermediate solution. The cameras could help the robot during the deployment stage, in order to explore the environment autonomously and calibrate all the systems. Then, the robot could rely on its localisation estimates, unless the localisation algorithm failed, in which case the cameras would still be able to support its navigation.

2. We would like to continue exploring the detection of call events. The Computer Vision community has developed dozens of algorithms aimed at the characterisation of people behaviour. This could be used as a way to infer whether or not the assistance of the robot is needed.

3. We believe that the introduction of smartphones in this system would be very interesting. Smartphones have perception and processing capabilities, and they can interact

with the user. Moreover, as we have demonstrated with our experience at Situm Technologies, they can be localised within the environment with sufficient precision. Therefore, users could use them to call a robot, and the system could track them to infer the behaviour of the users and trigger call events.

4. We would like to improve the observation models of the external cameras and the camera mounted on top of the robot. Both showed good results, but we think that the information that they provide has much more potential. For instance, in our current proposal, when a camera sees the robot, the robot only knows that it is within the FOV of the camera. However, cameras could be used to give a more precise estimation of the position of the robot in the map.

5. We would like to explore the use of other sensors in our localisation system. Particularly, we would like to include an omni-directional camera pointing to the ceiling. Gamallo et al. [68] demonstrated that this sensor could be used to obtain very robust global and local localisation estimates.

6. We have seen that 3D perception could help our robot in order to avoid dangerous situations (e.g. falling down stairs) and to achieve a more robust and safe navigation. In addition, it would also be interesting to integrate 3D perception in the positioning process.

7. We have seen that the use of varying transmission powers improves significantly the performance of wireless positioning systems. We will continue on exploring the use of different power transmission patterns. In addition, we will explore active localisation strategies, where the robot can modify the power of the transmitters proactively, in order to discard localisation hypotheses. This opens the possibility of robot-agents influencing the behaviour of other agents of the intelligent environment in order to improve its own behaviour and being more effective.

8. We would like to explore the use of radio technologies in the 433 MHz band for communications amongst the elements of our system.

9. Finally, we will continue on improving our tour-guide robot. Particularly, we would like to: 1) improve the human detector so that it is able to segment humans that are touching each other, 2) provide a more robust and natural hand-tracking (now it requires

initialisation), 3) enable the robot to learn gestures and associate them with actions, 4) introduce speech recognition techniques.

CHAPTER 10

DERIVED WORKS

This thesis has generated several results of great relevancy. First of all, the contents of this thesis have been reported in 7 publications in journals indexed in JCR, in 1 publication in a journal with other quality index, in 6 publications in international congresses and in 4 publications in national congresses. We have also registered our multi-sensor indoor positioning software at the "Oficina Española de Patentes y Marcas" (Spanish Patent and Trademark Office) and we have created the company Situm Technologies, an spin-off of the University of Santiago de Compostela.

10.1 Publications in journals indexed in JCR

1. **A. Canedo-Rodriguez**, J. M. Rodriguez, V. Alvarez-Santos, R. Iglesias and C. V. Regueiro. "Mobile Robot Positioning with 433-MHz Wireless Motes with Varying Transmission Powers and a Particle Filter". *Sensors*, vol. 15, pp. 10194–10220, 2015. Impact factor: 2.048 (10 of 57 - Q1 in category: Instruments & instrumentation).

2. **A. Canedo-Rodriguez**, V. Alvarez-Santos, C.V. Regueiro, R. Iglesias, S. Barro and J. Presedo. "Particle filter robot localisation through robust fusion of laser, WiFi, compass, and a network of external cameras". *Information Fusion*, available online. Impact factor: 3.472 (12 of 121 - Q1 in category: Computer Science, Artificial Intelligence).

3. V. Alvarez-Santos, **A. Canedo-Rodriguez**, R. Iglesias, X.M. Pardo, C.V. Regueiro and M. Fernandez-Delgado. "Route learning and reproduction in a tour-guide robot".

Robotics and Autonomous Systems, vol. 63, p. 2, pp. 206-213, 2015. Impact factor: 1.105 (11 of 21 - Q3 in category: Robotics).

4. V. Alvarez-Santos, R. Iglesias, X.M. Pardo, C.V. Regueiro and **A. Canedo-Rodriguez**. "Gesture-based interaction with voice feedback for a tour-guide robot". *Journal of Visual Communication and Image Representation*, no. 2, vol 13, pp. 499–509, 2014. Impact factor: 1.361 (30 of 105 - Q2 in category: Computer Science, Software Engineering)

5. V. Alvarez-Santos, X.M. Pardo, R. Iglesias, **A. Canedo-Rodriguez** and C.V Regueiro. "Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot". *Robotics and Autonomous Systems*, no. 8, vol. 60, pp. 1021–1036, 2012. Impact factor: 1.156 (9 of 21 - Q2 in category: Robotics).

6. **A. Canedo-Rodriguez**, R. Iglesias, C.V. Regueiro, V. Alvarez-Santos and X.M. Pardo. "Self-organized multi-camera network for a fast and easy deployment of ubiquitous robots in unknown environments". *Sensors*, no. 1, vol 13, p. 426–454, 2013. Impact factor: 2.048 (10 of 57 - Q1 in category: Instruments & instrumentation).

7. **A. Canedo-Rodriguez**, C.V. Regueiro, R. Iglesias, V. Alvarez-Santos and X.M. Pardo. "Self-organized multi-camera network for ubiquituous robot deployment in unknown environments". *Robotics and Autonomous Systems*, no. 7, vol. 61, pp. 667–675, 2012. Impact factor: 1.105 (11 de 21 - Q3 in category: Robotics).

10.2 Publications in journals with other quality indexes

1. **A. Canedo-Rodriguez**, V. Alvarez-Santos, C.V. Regueiro, X.M. Pardo and R. Iglesias. "Multi-agent system for fast deployment of a guide robot in unknown environments". *Special Issue on Advances on Physical Agents. In Journal of Physical Agents*, vol. 6 no. 1, pp. 31-41, 2012.

10.3 Publications in international conferences

1. D. Santos-Saavedra, **A. Canedo-Rodriguez**, X. M. Pardo, R. Iglesias, C. V. Regueiro. "Scene recognition for robot localization in difficult environments". *Foundations on*

Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2015, under review.

2. D. Santos-Saavedra, X. M. Pardo, R. Iglesias, **A. Canedo-Rodriguez** and V. Alvarez-Santos. "Scene recognition invariant to symmetrical reflections and illumination conditions in robotics". *Proceedings of the 7th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2015)*, accepted.

3. V. Alvarez-Santos, **A. Canedo-Rodriguez**, R. Iglesias, X.M. Pardo and C.V. Regueiro. "Route learning and reproduction in a tour-guide robot". *Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2013*, vol. 6686 pp. 112-121, 2013.

4. **A. Canedo-Rodriguez**, V. Alvarez-Santos, D. Santos-Saavedra, C. Gamallo, M. Fernandez-Delgado, R. Iglesias and C.V. Regueiro. "Robust multi-sensor system for mobile robot localization". *Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2013*, vol. 6686 pp. pp. 112-121, 2013.

5. V. Alvarez-Santos, X. M. Pardo, R. Iglesias, **A. Canedo-Rodriguez** and C. V. Regueiro. "Online Feature Weighting for Human Discrimination in a Person Following Robot". *Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2011*, vol. 6686, pp. 222-232, 2011.

6. **A. Canedo-Rodriguez**, R. Iglesias, C. V. Regueiro, V. Alvarez-Santos, X. M. Pardo. "Self-organized Multi-agent System for Robot Deployment in Unknown Environments". *Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2011*, vol. 6686, pp. 165-174, 2011.

10.4 Publications in spanish conferences

1. D. Santos-Saavedra, X. M. Pardo, R. Iglesias, V. Alvarez-Santos, **A. Canedo-Rodriguez** and C. V. Regueiro. "Global image features for scene recognition invariant to symmetrical reflections in robotics". *Proceedings of the XV Workshop of Physical Agents (WAF2014)*, pp. 29-37, 2014.

2. V. Álvarez-Santos, R. Iglesias, X. M. Pardo, C. V. Regueiro and **A. Canedo-Rodriguez**. "Gesture based interface with voice feedback for a guide robot". *Proceedings of the XIII Workshop of Physical Agents (WAF2012)*, pp. 135-144, 2012.

3. **A. Canedo-Rodriguez**, D. Santos-Saavedra, V. Alvarez-Santos, C. V. Regueiro, R. Iglesias and X.M. Pardo. "Analysis of different localization systems suitable for a fast and easy deployment of robots in diverse environments". *Proceedings of the XIII Workshop of Physical Agents (WAF2012)*, pp. 39-46, 2012.

4. **A. Canedo-Rodriguez**, V. Alvarez-Santos, C.V. Regueiro, X.M. Pardo and R. Iglesias. "Multi-agent system for fast deployment of a guide robot in unknown environments". *Proceedings of the XII Workshop of Physical Agents (WAF2011)*, pages 23-30, 2011.

10.5 Computer software

1. Software "Motor de localización en interiores", registered at the Oficina Española de Patentes y Marcas (Spanish Patent and Trademark Office) with number SC-089-15.

10.6 Entrepreneurship

1. Situm Technologies S. L., a company that provides indoor positioning technologies for smartphones.

Bibliography

[1] I. F. of Robotics, “World robotics 2014 executive summary,” tech. rep., International Federation of Robotics, 2014.

[2] C. Balaguer, A. Barrientos, P. Sanz, R. Sanz, and E. Zalama, “Libro blanco de la robotica,” *De la investigación al desarrollo y futuras aplicaciones. CEA-GTROB subencionado por el MEC. Espana*, 2007.

[3] M. Weiser, “Some computer science issues in ubiquitous computing,” *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, 1993.

[4] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser, “The parctab ubiquitous computing experiment,” in *Mobile Computing*, pp. 45–101, Springer, 1996.

[5] K. Ara, N. Kanehira, D. O. Olguin, B. N. Waber, T. Kim, A. Mohan, P. Gloor, R. Laubacher, D. Oster, A. Pentland, *et al.*, “Sensible organizations: Changing our businesses and work styles through sensor data,” *Information and Media Technologies*, vol. 3, no. 3, pp. 604–615, 2008.

[6] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, “Easyliving: Technologies for intelligent environments,” in *Handheld and ubiquitous computing*, pp. 12–29, Springer, 2000.

[7] M. Mozer, *Lessons from an adaptive house*. PhD thesis, University of Colorado, 2004.

[8] K. Irie, K. Umeda, and M. Wada, *Construction of an Intelligent Room using Distributed Camera System*. INTECH Open Access Publisher, 2008.

[9] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun, “The mobile robot rhino,” *AI Magazine*, vol. 16, no. 2, p. 31, 1995.

[10] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, *et al.*, “Minerva: A second-generation museum tour-guide robot,” in *IEEE international conference on Robotics and automation*, vol. 3, IEEE, 1999.

[11] R. Siegwart, K. O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, *et al.*, “Robox at expo. 02: A large-scale installation of personal robots,” *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 203–222, 2003.

[12] P. Trahanias, W. Burgard, A. Argyros, D. Hahnel, H. Baltzakis, P. Pfaff, and C. Stachniss, “Tourbot and webfair: Web-operated mobile robots for tele-presence in populated exhibitions,” *IEEE Robotics & Automation Magazine*, vol. 12, no. 2, pp. 77–89, 2005.

[13] D. Rodriguez-Losada, F. Matia, R. Galan, M. Hernando, J. M. Montero, and J. M. Lucas, “Urbano, an interactive mobile tour-guide robot,” *Advances in Service Robotics, Ed. H. Seok. In-Teh*, pp. 229–252, 2008.

[14] J.-H. Kim, K.-H. Lee, Y.-D. Kim, N. S. Kuppuswamy, and J. Jo, “Ubiquitous Robot: A New Paradigm for Integrated Services,” in *IEEE International Conference on Robotics and Automation*, pp. 2853–2868, 2007.

[15] J.-H. Lee and H. Hashimoto, “Intelligent spaceâconcept and contents,” *Advanced Robotics*, vol. 16, no. 3, pp. 265–280, 2002.

[16] J. H. Lee, K. Morioka, N. Ando, and H. Hashimoto, “Cooperation of distributed intelligent sensors in intelligent environment,” *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 3, pp. 535–543, 2004.

[17] J. H. Lee and H. Hashimoto, “Controlling mobile robots in distributed intelligent sensor network,” *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 890–902, 2003.

[18] P. Steinhaus, M. Strand, and R. Dillmann, "Autonomous robot navigation in human-centered environments based on 3D data fusion," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 224–235, 2007.

[19] P. Steinhaus, M. Walther, B. Giesler, and R. Dillmann, "3d global and mobile sensor data fusion for mobile platform navigation," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, pp. 3325–3330, IEEE, 2004.

[20] D. Pizarro, M. Mazo, E. Santiso, M. Marron, D. Jimenez, S. Cobreces, and C. Losada, "Localization of mobile robots using odometry and an external vision sensor," *Sensors*, vol. 10, no. 4, pp. 3655–3680, 2010.

[21] C. Losada, M. Mazo, S. Palazuelos, D. Pizarro, and M. Marrón, "Multi-camera sensor system for 3d segmentation and localization of multiple mobile robots," *Sensors*, vol. 10, no. 4, pp. 3261–3279, 2010.

[22] C. Losada, M. Mazo, S. E. Palazuelos, D. Pizarro, M. Marron, and J. F. Velasco, "Identification and tracking of robots in an intelligent space using static cameras and an XPFCP," *Robotics and Autonomous Systems*, vol. 61, pp. 75–85, Dec. 2012.

[23] I. Fernandez, M. Mazo, J. L. Lazaro, D. Pizarro, E. Santiso, P. Martin, and C. Losada, "Guidance of a mobile robot using an array of static cameras located in the environment," *Autonomous Robots*, vol. 23, no. 4, pp. 305–324, 2007.

[24] A. Saffiotti and M. Broxvall, "Peis ecologies: Ambient intelligence meets autonomous robotics," in *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pp. 277–281, ACM, 2005.

[25] A. Saffiotti, M. Broxvall, M. Gritti, K. LeBlanc, R. Lundh, J. Rashid, B. Seo, and Y.-J. Cho, "The peis-ecology project: vision and results," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2329–2335, IEEE, 2008.

[26] R. Lundh, L. Karlsson, and A. Saffiotti, "Autonomous functional configuration of a network robot system," *Robotics and Autonomous Systems*, vol. 56, no. 10, pp. 819–830, 2008.

[27] M. Shiomi, T. Kanda, H. Ishiguro, and N. Hagita, “Interactive Humanoid Robots for a Science Museum,” *IEEE Intelligent Systems*, vol. 22, pp. 25–32, Mar. 2007.

[28] M. Shiomi, T. Kanda, D. F. Glas, S. Satake, H. Ishiguro, and N. Hagita, “Field trial of networked social robots in a shopping mall,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2846–2853, IEEE, 2009.

[29] M. Shiomi, D. Sakamoto, T. Kanda, C. T. Ishi, H. Ishiguro, and N. Hagita, “Field trial of a networked robot at a train station,” *International Journal of Social Robotics*, vol. 3, no. 1, pp. 27–40, 2011.

[30] A. Sanfeliu, J. Andrade-Cetto, *et al.*, “Ubiquitous networking robotics in urban settings,” in *Proceedings of the IEEE/RSJ IROS Workshop on Network Robot Systems*, pp. 9–15, 2006.

[31] A. Sanfeliu, J. Andrade-Cetto, M. Barbosa, R. Bowden, J. Capitan, A. Corominas, A. Gilbert, J. Illingworth, L. Merino, J. M. Mirats, P. Moreno, A. Ollero, J. Sequeira, and M. T. J. Spaan, “Decentralized sensor fusion for Ubiquitous Networking Robotics in Urban Areas,” *Sensors*, vol. 10, pp. 2274–2314, Jan. 2010.

[32] M. Barbosa, A. Bernardino, D. Figueira, J. Gaspar, N. Gonçalves, P. U. Lima, P. Moreno, A. Pahliani, J. Santos-Victor, M. T. Spaan, *et al.*, “Isrobotnet: A testbed for sensor and robot network systems,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2827–2833, IEEE, 2009.

[33] A. Corominas Murtra, *Map-based localization for urban service mobile robotics*. PhD thesis, Universitat Politècnica de Catalunya, 2011.

[34] A. Sanfeliu, N. Hagita, and A. Saffiotti, “Network robot systems,” *Robotics and Autonomous Systems*, vol. 56, no. 10, pp. 793–797, 2008.

[35] V. Alvarez-Santos, *Development of a general purpose tour-guide robot able to learn routes from people and to adapt and move in unstructured and crowded environments*. 2014.

[36] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, IEEE, 2005.

[37] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–511, IEEE, 2001.

[38] R. Lienhart and J. Maydt, “An extended set of haar-like features for rapid object detection,” in *International Conference on Image Processing*, vol. 1, pp. I–900, IEEE, 2002.

[39] P. KaewTraKulPong and R. Bowden, “An improved adaptive background mixture model for real-time tracking with shadow detection,” in *Video-based surveillance systems*, pp. 135–144, Springer, 2002.

[40] V. Lepetit and P. Fua, “Monocular model-based 3d tracking of rigid objects: A survey,” *Foundations and trends in computer graphics and vision*, vol. 1, no. CVLAB-ARTICLE-2005-002, pp. 1–89, 2005.

[41] R. Cassinis and F. Tampalini, “Amirolos an active marker internet-based robot localization system,” *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 306–315, 2007.

[42] G. Mayer, H. Utz, and G. K. Kraetzschmar, “Playing robot soccer under natural light: A case study,” in *RoboCup 2003: Robot Soccer World Cup VII*, pp. 238–249, Springer, 2004.

[43] D. Kim, J. Choi, and M. Park, “Detection of multi-active markers and pose for formation control,” in *International Conference on Control Automation and Systems*, pp. 943–946, IEEE, 2010.

[44] H. Freeman, “On the encoding of arbitrary geometric configurations,” *IRE Transactions on Electronic Computers*, no. 2, pp. 260–268, 1961.

[45] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O’reilly, 2008.

[46] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, “Assessing the accuracy of prediction algorithms for classification: an overview,” *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[47] P. Quintia, *Robots able to learn and adapt to the environment based on their own experiences*. PhD thesis, Universidade de Santiago de Compostela, 2013.

[48] I. Cabado-Lago, *Reconocimiento de acciones mediante un sistema de vision por ordenador*. 2012.

[49] J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, IEEE, 1994.

[50] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, pp. 1–10, 2001.

[51] S. Thrun, W. Burgard, D. Fox, *et al.*, *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2005.

[52] I. Rekleitis, D. Meger, and G. Dudek, "Simultaneous planning, localization, and mapping in a camera sensor network," *Robotics and Autonomous Systems*, vol. 54, no. 11, pp. 921–932, 2006.

[53] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, pp. 317–323, 2003.

[54] J. M. Abuin-Mosquera, *Reconocimiento de personas desde diferentes puntos de vista en un entorno con mÃºltiples cÃ¡maras*. 2012.

[55] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 205–210, IEEE, 2004.

[56] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Tracking multiple people under global appearance constraints," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 137–144, IEEE, 2011.

[57] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1 – 27:27, 2011.

[58] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[59] M. Heikkilä, M. Pietikäinen, and C. Schmid, "Description of interest regions with center-symmetric local binary patterns," in *Computer Vision, Graphics and Image Processing*, pp. 58–69, Springer, 2006.

[60] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 685–692, IEEE, 2010.

[61] G. Cielniak, T. Duckett, and A. J. Lilienthal, "Data association and occlusion handling for vision-based people tracking by mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 435–443, 2010.

[62] N. Bellotto and H. Hu, "Multimodal people tracking and identification for service robots," *International Journal of Information Acquisition*, vol. 5, no. 03, pp. 209–221, 2008.

[63] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, MA: MIT Press, 2005.

[64] U. Nehmzow, *Mobile robotics*. Springer, 2003.

[65] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[66] H. Morioka, S. Yi, and O. Hasegawa, "Vision-based mobile robot's slam and navigation in crowded environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3998–4005, Sept 2011.

[67] J. D. Tardos, J. Neira, P. M. Newman, and J. J. Leonard, "Robust Mapping and Localization in Indoor Environments Using Sonar Data," *The International Journal of Robotics Research*, vol. 21, pp. 311–330, Apr. 2002.

[68] C. Gamallo, M. Mucientes, and C. V. Regueiro, "Omnidirectional visual slam under severe occlusions," *Robotics and Autonomous Systems*, vol. 65, pp. 76–87, 2015.

[69] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, “Multisensor data fusion: A review of the state-of-the-art,” *Information Fusion*, vol. 14, pp. 28–44, Jan. 2013.

[70] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: Real-time single camera slam,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[71] M. Labbe and F. Michaud, “Online global loop closure detection for large-scale multi-session graph-based slam,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2661–2666, Sept 2014.

[72] J. Wolf, W. Burgard, and H. Burkhardt, “Robust vision-based localization by combining an image-retrieval system with monte carlo localization,” *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 208–216, 2005.

[73] A. Canedo-Rodriguez, V. Alvarez-Santos, D. Santos-Saavedra, C. Gamallo, M. Fernandez-Delgado, R. Iglesias, and C. Regueiro, “Robust multi-sensor system for mobile robot localization,” in *Natural and Artificial Computation in Engineering and Medical Applications*, vol. 7931 of *Lecture Notes in Computer Science*, pp. 92–101, 2013.

[74] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[75] G. Deak, K. Curran, and J. Condell, “A survey of active and passive indoor localisation systems,” *Computer Communications*, vol. 35, no. 16, pp. 1939–1954, 2012.

[76] Y. Gu, A. Lo, and I. Niemegeers, “A survey of indoor positioning systems for wireless personal networks,” *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 13–32, 2009.

[77] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 6, pp. 1067–1080, 2007.

[78] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach, “Robotics-based location sensing using wireless ethernet,” *Wireless Networks*, vol. 11, no. 1-2, pp. 189–204, 2005.

[79] J. Fink and V. Kumar, “Online methods for radio signal mapping with mobile robots,” in *IEEE International Conference on Robotics and Automation*, pp. 1940–1945, May 2010.

[80] F. Duvallet and A. D. Tews, “WiFi position estimation in industrial environments using Gaussian processes,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2216–2221, Sept. 2008.

[81] F. Della Rosa, M. Pelosi, and J. Nurmi, “Human-induced effects on rss ranging measurements for cooperative positioning,” *International Journal of Navigation and Observation*, vol. 2012, pp. 1–13, 2012.

[82] Y. Chapre, P. Mohapatra, S. Jha, and A. Seneviratne, “Received signal strength indicator and its analysis in a typical wlan system (short paper),” in *IEEE 38th Conference on Local Computer Networks*, pp. 304–307, Oct 2013.

[83] K. Kaemarungsi and P. Krishnamurthy, “Properties of indoor received signal strength for wlan location fingerprinting,” in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 14–23, Aug 2004.

[84] Y.-C. Chen, J.-R. Chiang, H.-h. Chu, P. Huang, and A. W. Tsui, “Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics,” in *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pp. 118–125, 2005.

[85] G. Deak, K. Curran, J. Condell, and U. Londonderry, “Device-free passive localization using rssi-based wireless network nodes,” in *PGNeT 2010-The Eleventh Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, pp. 241–246, 2010.

[86] J. Ryckaert, P. De Doncker, R. Meys, A. de Le Hoye, and S. Donnay, “Channel model for wireless communication around human body,” *Electronics letters*, vol. 40, no. 9, pp. 543–544, 2004.

[87] P. Corke, R. Peterson, and D. Rus, “Localization and navigation assisted by networked cooperating sensors and robots,” *The International Journal of Robotics Research*, vol. 24, no. 9, pp. 771–786, 2005.

[88] D. Brscic, T. Kanda, T. Ikeda, and T. Miyashita, “Person tracking in large public spaces using 3-d range sensors,” *IEEE Transactions on Human-Machine Systems*, vol. 43, pp. 522–534, Nov 2013.

[89] D. Devarajan, Z. Cheng, and R. Radke, “Calibrating distributed camera networks,” *Proceedings of the IEEE*, vol. 96, pp. 1625–1639, Oct 2008.

[90] Y. He, X. Shen, Y. Liu, L. Mo, and G. Dai, “Listen: Non-interactive localization in wireless camera sensor networks,” in *IEEE 31st Real-Time Systems Symposium*, pp. 205–214, Nov 2010.

[91] I. Rekleitis and G. Dudek, “Automated calibration of a camera sensor network,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3384–3389, Aug 2005.

[92] D. Meger, D. Marinakis, I. Rekleitis, and G. Dudek, “Inferring a probability distribution function for the pose of a sensor network using a mobile robot,” in *IEEE International Conference on Robotics and Automation*, pp. 756–762, 2009.

[93] X. Zhang, A. B. Rad, and Y.-K. Wong, “Sensor fusion of monocular cameras and laser rangefinders for line-based Simultaneous Localization and Mapping (SLAM) tasks in autonomous mobile robots,” *Sensors*, vol. 12, pp. 429–452, Jan. 2012.

[94] J. Biswas and M. Veloso, “Multi-sensor mobile robot localization for diverse environments,” in *RoboCup 2013: Robot Soccer World Cup XVII*, pp. 468–479, 2013.

[95] C. Gamallo, P. Quintia, R. Iglesias, J. V. Lorenzo, and C. V. Regueiro, “Combination of a low cost GPS with visual localization based on a previous map for outdoor navigation,” in *11th International Conference on Intelligent Systems Design and Applications*, pp. 1146–1151, 2011.

[96] M. Quigley, D. Stavens, A. Coates, and S. Thrun, “Sub-meter indoor localization in unmodified environments with inexpensive sensors,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2039–2046, 2010.

[97] A. Sanfeliu, N. Hagita, and A. Saffiotti, “Network robot systems,” *Robotics and Autonomous Systems*, vol. 56, pp. 793–797, Oct. 2008.

[98] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, “Robust monte carlo localization for mobile robots,” *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[99] D. Fox, “Adapting the Sample Size in Particle Filters Through KLD-Sampling,” *The International Journal of Robotics Research*, vol. 22, pp. 985–1003, Dec. 2003.

[100] M. Klaas, N. D. Freitas, and A. Doucet, “Toward practical N2 Monte Carlo: The marginal particle filter,” pp. 308–315, 2005.

[101] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[102] F. Gustafsson, “Particle filter theory and practice with positioning applications,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, pp. 53–82, July 2010.

[103] R. Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE transactions on neural networks*, vol. 16, pp. 645–678, May 2005.

[104] D. Pollard, *A user’s guide to measure theoretic probability*. Cambridge University Press, 2002.

[105] M. Kjaergaard, “A taxonomy for radio location fingerprinting,” in *Location-and Context-Awareness*, vol. 4718 of *Lecture Notes in Computer Science*, pp. 139–156, 2007.

[106] V. Honkavirta, T. Perala, S. Ali-Loytty, and R. Piché, “A comparative survey of wlan location fingerprinting methods,” in *6th Workshop on Positioning, Navigation and Communication*, pp. 243–251, 2009.

[107] J. Yim, S. Jeong, K. Gwon, and J. Joo, “Improvement of kalman filters for wlan based indoor tracking,” *Expert Systems with Applications*, vol. 37, no. 1, pp. 426–433, 2010.

[108] M. Ocana, L. Bergasa, M. Sotelo, J. Nuevo, and R. Flores, “Indoor robot localization system using wifi signal measure and minimizing calibration effort,” in *Proceedings of the IEEE International Symposium on Industrial Electroncs*, vol. 4, pp. 1545–1550, 2005.

[109] R. Battiti, A. Villani, and T. Le Nhat, “Neural network models for intelligent networks: deriving the location from signal patterns,” *Proceedings of the First Annual Symposium on Autonomous Intelligent Networks and Systems*, 2002.

[110] K. Derr and M. Manic, “Wireless based object tracking based on neural networks,” in *3rd IEEE Conference on Industrial Electronics and Applications*, pp. 308–313, IEEE, 2008.

[111] M. Brunato and R. Battiti, “Statistical learning theory for location fingerprinting in wireless lans,” *Computer Networks*, vol. 47, no. 6, pp. 825–845, 2005.

[112] B. Li, J. Salter, A. G. Dempster, and C. Rizos, “Indoor positioning techniques based on wireless lan,” in *First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, pp. 13–16, 2006.

[113] J. Krumm and E. Horvitz, “Locadio: inferring motion and location from wi-fi signal strengths,” in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 4–13, Aug 2004.

[114] J. Biswas and M. Veloso, “Wifi localization and navigation for autonomous indoor mobile robots,” in *IEEE International Conference on Robotics and Automation*, pp. 4379–4384, IEEE, 2010.

[115] C.-C. Chang and C.-J. Lin, “Libsvm: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[116] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[117] B. Ferris, D. Haehnel, and D. Fox, “Gaussian processes for signal strength-based location estimation,” in *Proceedings of Robotics Science and Systems*, Citeseer, 2006.

[118] T. B. Welch, R. L. Musselman, B. A. Emessiene, P. D. Gift, D. K. Choudhury, D. N. Cassadine, and S. M. Yano, “The effects of the human body on uwb signal propagation in an indoor environment,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 9, pp. 1778–1782, 2002.

[119] C. M. Bishop *et al.*, *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

[120] A. Canedo-Rodriguez, R. Iglesias, C. V. Regueiro, V. Alvarez-Santos, and X. M. Pardo, "Self-organized multi-camera network for a fast and easy deployment of ubiquitous robots in unknown environments.," *Sensors*, vol. 13, pp. 426–454, Jan. 2013.

[121] M. Rampinelli, V. B. Covre, F. M. de Queiroz, R. F. Vassallo, T. F. Bastos-Filho, and M. Mazo, "An intelligent space for mobile robot localization using a multi-camera system," *Sensors*, vol. 14, no. 8, pp. 15039–15064, 2014.

[122] D. Santos-Saavedra, X. M. Pardo, R. Iglesias, A. Canedo-Rodríguez, and V. Álvarez-Santos, "Scene recognition invariant to symmetrical reflections and illumination conditions in robotics," in *Accepted in IbPRIA 2015: 7th Iberian Conference on Pattern Recognition and Image Analysis*, 2015.

[123] J. Wu and J. M. Rehg, "Centrist: A visual descriptor for scene categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1489–1501, 2011.

[124] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pp. 197–206, ACM, 2007.

[125] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[126] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[127] A. W. Tsui, Y.-H. Chuang, and H.-H. Chu, "Unsupervised learning for solving rss hardware variance problem in wifi localization," *Mobile Networks and Applications*, vol. 14, no. 5, pp. 677–691, 2009.

[128] A. Canedo-Rodriguez, D. Santos-Saavedra, V. Alvarez-Santos, C. V. Regueiro, R. Iglesias, and X. M. Pardo, "Analysis of different localization systems suitable for a fast and easy deployment of robots in diverse environments," in *Workshop of Physical Agents*, pp. 39–46, 2012.

[129] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro, “Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot,” *Robotics and Autonomous Systems*, vol. 60, pp. 1021–1036, Aug. 2012.

[130] J. Derrac, S. Garcia, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, pp. 3–18, Mar. 2011.

[131] I. Rodríguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarín, “STAC: a web platform for the comparison of algorithms using statistical tests,” 2015.

[132] D. Meger, I. Rekleitis, and G. Dudek, “Heuristic search planning to reduce exploration uncertainty,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3392–3399, 2008.

[133] I. Rekleitis, “Simultaneous localization and uncertainty reduction on maps (slurm): Ear based exploration,” in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 501–507, 2012.

[134] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[135] J. Yick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[136] G. Lui, T. Gallagher, B. Li, A. Dempster, and C. Rizos, “Differences in rssi readings made by different wi-fi chipsets: A limitation of wlan localization,” in *International Conference on Localization and GNSS*, pp. 53–57, June 2011.

[137] M. Allen, E. Gaura, R. Newman, and S. Mount, “Experimental localization with mica2 motes,” in *Technical Proceedings of the 2006 NSTI Nanotechnology Conference and Trade Show*, vol. 3, pp. 435–440, 2006.

[138] L. Wang and Q. Xu, “Gps-free localization algorithm for wireless sensor networks,” *Sensors*, vol. 10, no. 6, pp. 5899–5926, 2010.

[139] L. Klingbeil and T. Wark, “A wireless sensor network for real-time indoor localisation and motion monitoring,” in *International Conference on Information Processing in Sensor Networks*, pp. 39–50, 2008.

[140] G. Fu, J. Zhang, W. Chen, F. Peng, P. Yang, and C. Chen, “Precise localization of mobile robots via odometry and wireless sensor network,” *International Journal of Advanced Robotic Systems*, pp. 1–13, 2013.

[141] A. Canedo-Rodriguez, C. V. Regueiro, R. Iglesias, V. Alvarez-Santos, and X. M. Pardo, “Self-organized multi-camera network for ubiquitous robot deployment in unknown environments,” *Robotics and Autonomous Systems*, vol. 61, pp. 667–675, July 2013.

[142] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, 2009.

[143] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro, “Online feature weighting for human discrimination in a person following robot,” in *Foundations on Natural and Artificial Computation*, pp. 222–232, Springer Berlin Heidelberg, 2011.

[144] V. Alvarez-Santos, R. Iglesias, X. Pardo, C. Regueiro, and A. Canedo-Rodriguez, “Gesture-based interaction with voice feedback for a tour-guide robot,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 499–509, 2014.

[145] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[146] R. Gockley, J. Forlizzi, and R. Simmons, “Natural person-following behavior for social robots,” in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 17–24, ACM, 2007.

[147] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro, “Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot,” *Robotics and Autonomous Systems*, vol. 60, no. 8, pp. 1021–1036, 2012.

[148] H. Zhou, P. Miller, and J. Zhang, “Age classification using radon transform and entropy based scaling svm.,” in *BMVC*, pp. 1–12, 2011.

[149] H. Zhou, Y. Yuan, and C. Shi, “Object tracking using sift features and mean shift,” *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[150] D. Gossow, P. Decker, and D. Paulus, “An evaluation of open source surf implementations,” in *RoboCup 2010: Robot Soccer World Cup XIV*, pp. 169–179, Springer, 2011.

[151] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[152] R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondence,” in *Computer VisionâECCV’94*, pp. 151–158, Springer, 1994.

[153] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou, “Discriminative local binary patterns for human detection in personal album,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[154] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[155] C. S. Won, D. K. Park, and S.-J. Park, “Efficient use of mpeg-7 edge histogram descriptor,” *Etri Journal*, vol. 24, no. 1, pp. 23–30, 2002.

[156] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[157] C. E. Shannon, “A mathematical theory of communication,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[158] C. van Rijsbergen, *Information Retrieval. 1979.* Butterworth, 1979.

[159] V. S. N. Prasad, A. Faheema, and S. Rakshit, “Feature selection in example-based image retrieval systems.,” in *ICVGIP*, Citeseer, 2002.

[160] V. Alvarez-Santos, R. Iglesias, X. M. Pardo, C. V. Regueiro, and A. Canedo-Rodriguez, "Gesture-based interaction with voice feedback for a tour-guide robot," *Journal of Visual Communication and Image Representation*, vol. 25, pp. 499–509, Feb. 2014.

[161] M. Roccetti, G. Marfia, and A. Semeraro, "Playing into the wild: A gesture-based interface for gaming in public spaces," *Journal of Visual Communication and Image Representation*, vol. 23, no. 3, pp. 426–440, 2012.

[162] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[163] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

List of Figures

List of Tables