Universidad de Granada

Escuela Técnica Superior de Ingeniería Informática

Departamento de Ciencias de la Computación

e Inteligencia Artificial

Soft-Computing and Computer Vision Techniques Applied to Autonomous Robot Navigation and Human-Robot Interaction

MEMORIA DE TESIS PRESENTADA POR

Rafael Muñoz Salinas

COMO REQUISITO PARA

OPTAR AL GRADO DE DOCTOR

EN INFORMÁTICA

Editor: Editorial de la Universidad de Granada
Autor: Rafael Muñoz Salinas
D.L.: Gr. 964 - 2006
ISBN: 84-338-3888-1

Universidad de Granada

Escuela Técnica Superior de Ingeniería Informática

Departamento de Ciencias de la Computación

e Inteligencia Artificial

Soft-Computing and Computer Vision Techniques Applied to Autonomous Robot Navigation and Human-Robot Interaction

MEMORIA DE TESIS PRESENTADA POR

Rafael Muñoz Salinas

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

DIRECTORES:

Eugenio Aguirre Molina

Miguel García Silvente

Fdo:Rafael Muñoz Salinas Fdo:Eugenio Aguirre Molina Fdo:Miguel García Silvente

Granada, Abril de 2006

Dedicado a mi madre y a la memoria de mi abuela,
quienes más han hecho para que yo llegara aquí.

Agradecimientos

Me gustaría transmitir mi más sincero agradecimiento a mis directores Eugenio y Miguel, por el apoyo prestado no solo en el plano científico sino también en el personal. En estos tres años, además de adquirir la formación necesaria para poder realizar este trabajo, he tenido la oportunidad de haber compartido mi tiempo con dos buenos y pacientes amigos.

Quiero agradecer a Antonio González el haberme dado la oportunidad de conseguir la financiación necesaria para el desarrollo de este trabajo y a Oscar Cordón por la colaboración que hemos mantenido. También quiero agradecer a Enrique Herrera y al Grupo de Robótica de la DeMontfort University de Leicester por haberme permitido realizar parte del trabajo con ellos.

Finalmente quisiera agradecer a todos mi familiares y amigos el apoyo y la atención prestada. Y en especial, quisiera agradecer a Carmen haber estado conmigo siempre que la he necesitado.

Contents

Resumen

El trabajo de esta memoria se engloba dentro del área de la robótica móvil autónoma. El propósito de esta disciplina consiste en desarrollar robots capaces de moverse de forma autónoma para realizar las tareas que les sean encomendadas. En este sentido, el trabajo de investigación realizado y que se expone en esta memoria tiene un doble objetivo. Por un lado, explorar mecanismos que permitan aumentar el grado de autonomía de robots móviles. Y por otro lado, el desarrollo un conjunto base de habilidades percepto-motoras que doten a los robots con capacidades primitivas para interactuar con humanos de forma natural.

En relación al primer objetivo, un aspecto fundamental para dotar a los robots de autonomía para moverse de forma libre y segura es conseguir mecanismos robustos de navegación. Un robot móvil ha de ser capaz de determinar el conjunto de acciones necesarias para ir desde su posición a otro lugar sin comprometer su seguridad ni la de los objetos o personas a su alrededor. Con este propósito, los robots móviles hacen uso de sus sensores para construir un modelo del entorno que les permita decidir la mejor secuencia de movimientos posible. Sin embargo, los sensores utilizados están sujetos a errores que sesgan el éxito del sistema de navegación. Los ultrasonidos son el sensor empleado por excelencia para la navegación debido su bajo precio y coste computacional. Pese a ello, sus fuente de errores hacen que los sistemas de navegación basados únicamente en ellos sufran de problemas de precisión serios. En esta memoria, exploramos el uso combinado de ultrasonidos y visión con la idea que las flaquezas de un subsistema sensorial puedan ser suplidas con las fortalezas del otro. Para ello, desarrollamos un sistema capaz de integrar información visual y de rango para desarrollar la tarea de navegación en entornos de interior. El sistema tiene un diseño muy modular basado en el concepto de *sistemas multiagente* que permite crear

sistemas altamente flexibles con la posibilidad de ser físicamente distribuidos en diferentes máquinas. El sistema de navegación crea un plan de movimiento, utilizando para ello un mapa topológico, basado en la secuencia de puertas y pasillos que se han de recorrer para llevar al robot desde su posición inicial a una habitación destino. Para ello, el sistema visual es capaz de detectar puertas del entorno que son identificadas mediante la colocación de marcas artificiales junto a ellas.

Las marcas artificiales son una herramienta muy utilizada para realizar las tareas de navegación y localización debido a su facilidad para distinguirlas en el entorno. Sin embargo, no son una solución adecuada cuando no es posible modificar el entorno (algo que ocurre con frecuencia). Por ello, otro de los trabajos que se desarrollan en esta memoria es la creación de un sistema visual capaz de detectar las puertas por sí mismas, sin la necesidad de utilizar marcas artificiales. El sistema opera extrayendo los segmentos más relevantes de la imágenes capturadas para analizarlos utilizando un clasificador jerárquico basado en lógica difusa. El éxito del detector de puertas depende de dos parámetros: la fracción de verdaderos positivos (el éxito de encontrar puertas en imágenes con puertas) y la fracción de verdaderos negativos (el éxito en detectar que no hay puertas en imágenes sin puertas). El detector diseñado obtiene buenos resultados de clasificación para las puertas de nuestro entorno. Sin embargo, su rendimiento en otros entornos (con puertas de diferentes tamaños) o utilizado en un sistema hardware distinto (un robot de altura distinta o una cámara con diferente distancia focal), puede verse afectado. El rendimiento del detector de puertas en un entorno determinado, utilizando un robot y cámaras particulares, puede maximizarse mediante un procedimiento de ajuste que lo adapta a las condiciones de operación.

El ajuste de sistemas difusos proporciona un mecanismo automático para incrementar su rendimiento a partir de un conjunto de datos validados del problema. El ajuste automático de sistemas difusos es un problema que ha sido tratado en la literatura relacionada utilizando varios enfoques. Cuando el rendimiento del sistema difuso a ajustar es evaluado mediante una función multiobjetivo (como es nuestro caso), el uso de enfoques evolutivos parece ser la herramienta más apropiada. Parte del trabajo desarrollado en esta memoria está dedicado al desarrollo y comparación de diversas técnicas evolutivas multiobjetivo y su uso para el

ajuste de nuestro sistema de detección de puertas.

Como segundo objetivo de esta memoria, nos interesa el desarrollo de un conjunto base de habilidades percepto-motoras para dotar a los robots con capacidades primitivas de interacción. Nuestro propósito es que estas habilidades constituyan una base útil para futuras aplicaciones robóticas que requieran interactuar con usuarios humanos de forma natural. Si deseamos que los humanos y los robots trabajen de forma conjunta en la sociedad del futuro, es necesario que los primeros puedan dirigir a los segundos utilizando medios de comunicación flexibles. El uso de medios de interacción clásicos como ratón, teclado o pantallas táctiles puede llegar a ser una barrera para la adaptación real de los robots a las tareas de nuestra vida diaria. Por tanto, es preferible la utilización de mecanismos de comunicación más humanos, es decir, comunicación verbal, gestos faciales o posturales. En este sentido, la visión juega un papel muy importante ya que permite a los humanos detectar la presencia de otros humanos así como la detección de multitud de pistas visuales que se dan durante una comunicación. Por ello, dotar a los robots con la capacidad de detectar y seguir a personas es un paso inicial requerido para permitir a los robots interactuar de forma natural con las personas. La detección y seguimiento visual de personas son problemas ya abordados en la literatura. Y en particular, el uso de visión estéreo para este propósito es un tema muy interesante que concentra una gran atención hoy en día. La visión estéreo es un sensor capaz de solucionar muchos de los problemas que aparecen al usar información monocular. En primer lugar, la información proporcionada por un sistema estéreo es más invariante a la iluminación, por lo que un sistema basado en esta tecnología será más robusto en aplicaciones reales. En segundo lugar, la información de profundidad que proporcionan puede ser de gran utilidad para el desarrollo de sistemas de seguimiento.

En esta memoria desarrollamos un sistema de detección y seguimiento de personas utilizando información estéreo. El sistema es capaz de detectar y seguir visualmente a múltiples personas que se mueven libremente en el entorno, combinando información de profundidad con información acerca del color de sus trajes. El sistema de detección y seguimiento es posteriormente integrado dentro de una arquitectura multiagente mayor. La arquitectura multiagente desarrollada implementa una serie de habilidades de interacción básicas, pero de gran

reusabilidad en un amplio conjunto de aplicaciones robóticas que requieran de la interacción con humanos. Las habilidades que diseñadas combinan información estéreo con información de ultrasonido para permitir: (i) la detección mediante visión estéreo de un usuario principal que desea interaccionar con el robot; (ii) el seguimiento visual de este usuario sin confundirlo con otras personas en el entorno; y (iii) la habilidad de seguir al usuario por un entorno de interiores evitando colisionar con los obstáculos presentes. Estas tres habilidades constituyen un conjunto inicial básico para dotar a robots con capacidades de atención e interacción, que puede ser completado con funcionalidad adicional dependiendo de la aplicación concreta que se requiera implementar. En este sentido, la filosofía multiagente empleada supone una ventaja para la rápida y fácil adaptación de la arquitectura a distintas plataformas y aplicaciones.

Podemos concluir afirmando que los trabajos realizados y que se presentan en esta memoria nos han permitido cumplir los objetivos inicialmente fijados. Además, el trabajo realizado apunta hacia futuras vías de investigación principalmente dirigidas hacia la creación de sistemas robóticos capaces de interaccionar de forma natural con las personas.

Motivación y Estructura de la Memoria

"El objetivo la robótica móvil autónoma es la construcción de sistemas físicos capaces de moverse con un propósito y sin intervención humana en entornos no modificados".

Alessandro Saffiotti [186].

Motivación

La robótica móvil constituye un campo ideal para el desarrollo de técnicas de Inteligencia Artificial (IA) y los investigadores pueden ven en los robots móviles un doble desafío. Por un lado, la oportunidad de proveer a una máquina con la capacidad de emular el comportamiento, percepción y razonamiento humanos. Por otro lado, la oportunidad de dotar a un sistema robótico con la habilidad de moverse de forma autónoma en un entorno real. Podría decirse que la robótica es uno de los lugares donde la IA toma contacto con el mundo real para validar sus propuestas.

Un robot móvil, analizado desde una perspectiva muy simple, es una máquina que debe percibir su entorno y decidir autónomamente las acciones a realizar en base a los objetivos encomendados y a las percepciones que toma del mundo real. En este sentido, hay dos aspectos fundamentales que deben tenerse en cuenta: (i) conseguir que los robots utilicen de forma eficiente sus sensores, atendiendo a la incertidumbre en las medidas que estos llevan asociados; y (ii) decidir el conjunto de acciones a realizar entre un conjunto ilimitado de ellas.

Desde los primeros trabajos desarrollados con Skakey [168] en el Instituto de Investigación de Standford a medidados de los años 60, se ha realizado un considerable esfuerzo de

investigación con el propósito de aumentar el grado de autonomía de los robot móviles. Los avances conseguidos en este ámbito han contribuido a los primeros intentos para utilizar los robot móviles en una gran variedad de aplicaciones de la vida real como robots personales [20, 68, 130], mascotas robóticas [77], robots guias en museos [37, 194], y asistentes robóticos personales para personas con discapacidades o ancianas [101, 178], entre otras. Poco a poco, los robots van ocupando un lugar no solo en el mercado sino también en nuestros hogares y aulas de enseñanza. Muestra de ello es el creciente número de robots disponibles en la actualidad para su uso recreativo, de investigación y educativo (ver Figura 1).

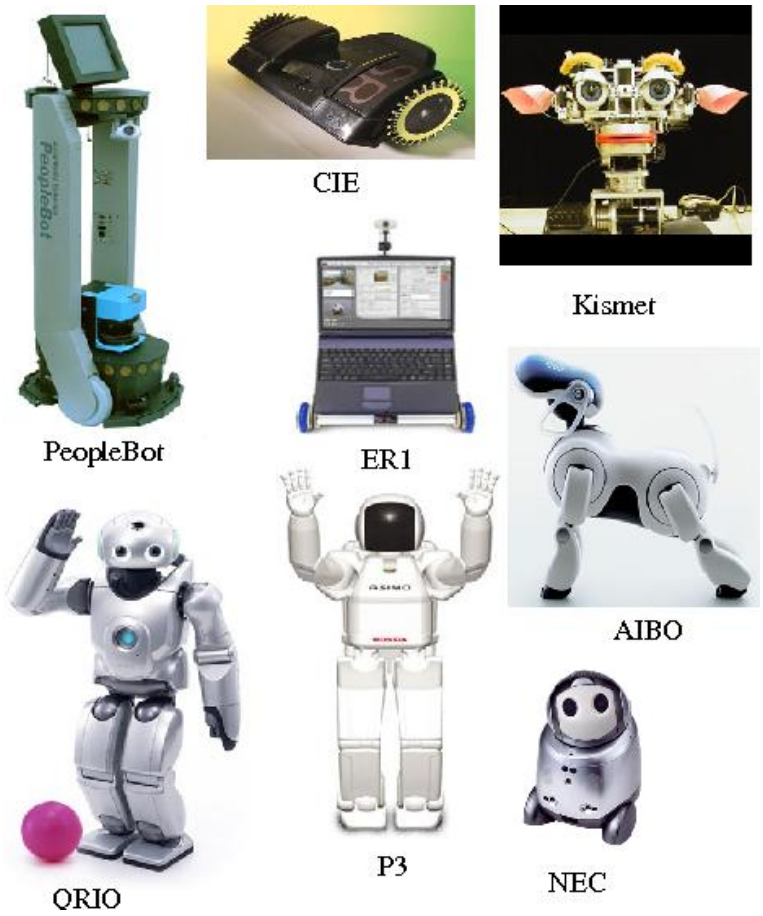


Figure 1: Ejemplos de robots recreativos, de investigación y educativos.

Sin embargo, a pesar de los grandes avances conseguidos en los últimos cincuenta años, aún queda un importante trabajo por realizar para conseguir que los robots estén presentes en nuestra vida diaria. El trabajo que ha sido desarrollado y que se expone en esta memo-

ria es nuestra contribución con el objetivo de hacer que los robots sean herramientas más accesibles en nuestros hogares y lugares de trabajo. En realidad, el objetivo de nuestro trabajo es doble. En primer lugar, el desarrollo de técnicas que ayuden a incrementar el nivel de autonomía de los robot móviles. Los robots móviles deben ser capaces de trabajar de forma autónoma, es decir, sin intervención humana. Esta memoria explora el uso combinado de técnicas de visión y de computación flexible aplicadas a los problemas de navegación y percepción con el objetivo de dotar a los robots con capacidades de movimiento autónomo más robustas. El segundo objetivo que perseguimos está relacionado con el desarrollo de habilidades percepto-motoras que doten a los robots móviles con habilidades básicas de interacción natural. Aunque los robots deban ser capaces de operar de forma autónoma, su propósito es el de servir a los humanos. Por tanto, los robots deben ser capaces de detectar a los humanos e interactuar con ellos, en lugar de considerarlos meros obstáculos en el camino. Además, esa comunicación debe ser natural. El uso de dispositivos como pantallas, teclados o ratones para realizar la comunicación puede ser una limitación a la hora de introducir los robots en nuestras vida diaria. Es preferible el uso de mecanismos de comunicación más similares a los que usamos los humanos: comunicación visual, voz, gestos, etc. En esta memoria se muestra el diseño de una serie de habilidades percepto-motoras que constituyen un conjunto básico de utilidad para un amplio de rango de aplicaciones robóticas que requieran de la interacción natural con usuarios humanos.

Incrementar el nivel de automía de los robots (primer objetivo de nuestro trabajo) es una tarea difícil que implica el trabajo en una gran variedad de áreas de investigación. Para hacer que los robots realicen tareas cotidianas de forma autónoma los diseñadores deben desarrollar sistemas complejos capaces de manejar de forma continua sensores y actuadores. Los diferentes subsistemas que componen un robot (sistemas de visión artificial, planificadores de trayectorias, controladores de hardware, etc) deben compartir eficientemente la información utilizando para ello mecanismos apropiados de comunicación, estar adecuadamente acoplados y coordinados, y trabajar en tiempo real. Al mismo tiempo, la arquitectura del sistema empleada debe ser lo suficientemente flexible para adaptarse a cambios de diseño y nuevas funcionalidades que pudiera ser necesario añadir en fases posteriores. Por

tanto, el diseño e implementación de los sistemas de control de robots es un trabajo que se hace más difícil conforme la complejidad de las tareas a realizar aumenta. Las arquitecturas clásicamente empleadas para el control de robots móviles pueden dividirse en: deliberativas [25, 168], reactivas [14, 34, 146] y arquitecturas híbridas [12, 84]. A pesar de que estos enfoques han mostrado la habilidad de solventar la mayoría de los requisitos anteriormente mencionados, siguen teniendo debilidades que inducen a continuar la investigación en este área. La flexibilidad y la escalabilidad son dos requisitos muy importantes para las arquitecturas de control. La flexibilidad hace referencia a la capacidad de una arquitectura de control para adaptarse a nuevos requisitos y funcionalidades que puede ser demandados tras el diseño inicial de la misma. La escalabilidad hace referencia a la habilidad de una arquitectura de aumentar su funcionalidad y seguir trabajando adecuadamente, respondiendo en los tiempos necesarios. Parte del trabajo desarrollado en esta memoria está dedicado al diseño de arquitecturas de control flexible capaces de tratar con estos problemas. En ese sentido, hacemos uso del concepto de *agente* como el componente básico para implementar nuestra arquitectura de control. Un agente puede entenderse como un programa independiente que trabaja para alcanzar un objetivo implícito en su diseño, y capaz de comunicarse con otros agentes. El concepto de agente es una extensión natural de la programación orientada a objetos para sistemas distribuidos. En particular, nosotros utilizamos arquitecturas multiagente, es decir, arquitecturas compuestas de diferentes agentes que se reparten la funcionalidad del sistema completo. Tres son las principales ventajas de las arquitecturas multiagente. En primer lugar, la habilidad poder distribuir físicamente los agentes en distintas máquinas si se necesita. En segundo lugar, estas arquitecturas proporcionan la flexibilidad suficiente para incrementar y reducir de forma sencilla la funcionalidad de un sistema (agregado o eliminando agentes), permitiendo así una gran adaptabilidad a los cambios de diseño. Finalmente, este tipo de arquitecturas son potencialmente muy robustas dado que ante el eventual fallo de un agente, el resto de los agentes pueden hacerse cargo de la misión y preservar la integridad del robot así como la de los que le rodean.

Otro aspecto clave en robótica, que aún sigue siendo objeto de investigación, es el de la navegación autónoma. Un robot debe ser capaz de percibir el entorno para poder decidir el

conjunto de acciones apropiadas que lo conduzcan hasta una posición deseada. Sin embargo, los sensores con que cuentan los robots están normalmente sujetos a incertidumbre y errores que hace de la tarea de navegación un problema difícil ya que los robots no pueden confiar en sus percepciones de forma absoluta. El caso más representativo es el de los sensores de ultrasonidos. Los ultrasonidos han sido ampliamente empleados como el principal sensor para la navegación debido a que proporcionan una información muy útil a un muy bajo coste económico y computacional. Sin embargo, el principal inconveniente de este tipo de sensores es que la información que proporcionan esta sujeta a un alto grado de incertidumbre. Por ello, son muchos los trabajos de investigación que se centran en el modelado y manejo de la incertidumbre e imprecisión con el objetivo de desarrollar mecanismos robustos de navegación. En este sentido, los comportamientos difusos han sido una buena solución a este problema [186]. Sin embargo, hay una limitación en el rendimiento de un sistema de navegación basado únicamente en ultrasonidos que viene dada por las propias limitaciones de este sistema sensorial. Es por ello, que el uso de distintas fuentes de información parece ser la mejor opción para manejar este problema. Es decir, el uso combinado de diferentes subsistemas sensoriales puede permitir compensar las deficiencias de un sensor con las fortalezas de otros. En este sentido, la visión es un sensor barato y disponible en los robots actuales que puede proporcional una gran cantidad de información útil para la navegación autónoma en combinación con los sensores de ultrasonidos. Es por esto, que parte del trabajo presentado en esta memoria está enfocado hacia el desarrollo de técnicas de navegación autónoma capaces de integrar la información proporcionada por un sensor visual y de ultrasonidos. En particular utilizaremos un enfoque de navegación basado en el *seguimiento visual de marcas* (*visual-landmark-tracking* [58]).

La navegación basada en el seguimiento de marcas es una técnica que consiste en la detección y seguimiento visual de determinados elementos del entorno que ayudan al proceso de localización de lugares relevantes para la tarea de navegación. Por *marca* se entiende un elemento representativo del entorno y suelen clasificarse en *marcas artificiales* y *marcas naturales*. Las primeras hacen referencia a elementos fabricados por el hombre que son colocados en lugares estratégicos del entorno con el objetivo de facilitar el proceso de navegación.

Las segundas hacen referencia a lugares representativos que hay en el propio entorno y que pueden ser utilizados para la navegación (como paredes, puertas o cuadros), pero que no están ahí por esa razón. En esta memoria presentamos un sistema multiagente capaz controlar la tarea de navegación de un robot móvil en entornos de oficina. El plan de navegación, que se crea haciendo uso de un mapa topológico del entorno, consiste en la secuencia de puertas a atravesar y pasillos que recorrer hasta llegar a la habitación deseada. Para manejar la incertidumbre e imprecisión de los sensores de ultrasonido empleados y poder así hacer que el robot navegue de forma segura, hemos hecho uso de comportamiento difusos. Además, se hace uso de comportamientos visuales capaces de localizar la puerta que se debe cruzar y realizar un seguimiento visual de ella para así indicar la dirección apropiada de navegación. En este trabajo, para facilitar la detección de las puertas se han colocado marcas artificiales junto a ellas. Las marcas artificiales han sido aprendidas utilizando técnicas de computación flexible y nuestro sistema de visión es capaz de detectarlas y realizar un seguimiento visual en tiempo real.

Aunque el uso de marcas artificiales es una solución apropiada cuando es posible modificar el entorno, esta opción no es siempre asumible bien por cuestiones practicas o estéticas. En este caso, la habilidad de un robot para detectar puertas sigue siendo una ventaja para la tarea de navegación e incluso para la creación autónoma de mapas. Es por ello, que también hemos diseñado un detector de puertas capaz de encontrarlas sin necesidad de utilizar marcas artificiales. Las puertas son encontradas en imágenes de nivel de gris mediante la detección de los bordes de sus marcos. El detector que proponemos se basa en las extracción de los principales segmentos de la imagen y su posterior análisis mediante un clasificador jerárquico basado en lógica difusa. El sistema visual difuso creado permite que la arquitectura de navegación diseñada pueda ser empleada incluso en entornos sin marcas artificiales. El éxito del detector de puertas depende de dos parámetros: la fracción de verdaderos positivos (el éxito de encontrar puertas en imágenes con puertas) y la fracción de verdaderos negativos (el éxito en detectar que no hay puertas en imágenes sin puertas).

El detector diseñado obtiene buenos resultados de clasificación para las puertas de nuestro entorno. Sin embargo, su uso en otros entornos (con puertas de diferentes tamaños) o

utilizado en un sistema hardware distinto (un robot de altura distinta o una cámara con diferente distancia focal), puede requerir de una fase previa de entrenamiento con el objeto de adaptarlo a las condiciones particulares de operación. El ajuste manual de sistemas difusos es una tarea tediosa, así que se han propuesto en la literatura distintos enfoques. El detector de puertas que ha sido diseñado es multiobjetivo debido a que su rendimiento es evaluado mediante dos criterios distintos y en conflicto. Para solucionar este tipo de problemas, el uso de enfoques evolutivos ha demostrado ser una herramienta muy ventajosa. Parte del trabajo de esta memoria está dedicado al desarrollo y comparación de diversas técnicas evolutivas de optimización multiobjetivo aplicadas a sistemas difusos, y en particular para poder adaptar nuestro detector a una amplia variedad de entornos de interior. La integración del detector de puertas como parte de la arquitectura de navegación diseñada permite a nuestro robot navegar de forma autónoma en entornos sin modificar en absoluto, puesto que se prescinde del uso de marcas artificiales.

Hasta este momento, hemos explicado el trabajo desarrollado en esta memoria que hace referencia al primer objetivo. Pero como ya indicamos, esta memoria persigue un doble objetivo. El segundo objetivo es el desarrollo de un conjunto de habilidades percepto-motoras elementales que doten a los robots con mecanismos de interacción naturales. Si deseamos que los robots se encuentren presentes en nuestra vida diaria es necesario poder interactuar con ellos de forma natural e intuitiva [31, 32, 33, 38, 73, 137, 192]. El área de investigación denominado interacción humano-robot (IHR) (*human-robot interaction HRI*) tiene por objeto hacer que tanto los robots como los humanos sean capaces de comunicarse sin que el humano requiera un proceso previo de aprendizaje. El uso de medios de interacción clásicos como ratón, teclado o pantallas táctiles puede llegar a ser insuficiente para una natural adaptación de los humanos a los robots. Es por tanto muy deseable, que los robots puedan comunicarse con los humanos tal y como los últimos lo hacen entre sí, es decir, mediante un dialogo multimodal y natural (voz, gestos, expresiones faciales, etc). Visto desde la perspectiva del robot, un aspecto crucial para conseguir una adecuada IHR es el poder detectar la presencia humana. Los robots deben ser capaz de detectar y seguir visualmente a los humanos a su alrededor en lugar de considerarlos como meros obstáculos. Parte de nue-

stro trabajo se dedica al desarrollo de habilidades percepto-motoras básicas que doten a los robots con primitivas de interacción válidas para un amplio rango de aplicaciones que requieran de una comunicación natural con los humanos. Las habilidades que se han diseñado en este trabajo combinan información estereoscópica con información de ultrasonidos para permitir a los robots : (i) detectar usuarios; (ii) poder realizar un seguimiento de los usuarios sin confundirlos entre ellos, y (iii) ser capaz de moverse junto con los usuarios para realizar cualquier tipo de tarea. Esas habilidades constituyen un conjunto mínimo cuya utilidad es común a un amplio conjunto de aplicaciones. Sin embargo, estas deberán ser utilizadas junto con habilidades adicionales, especificas de la aplicación a desarrollar.

A pesar de que la detección y seguimiento visual de personas es un tema ya tratado en la literatura empleando visión monocular, cuando la cámara se coloca sobre un sistema móvil el problema se hace más complejo. Para ese problema particular, la visión estereoscópica puede ser un sensor muy interesante. Además, el decremento en el precio de estos sistemas los convierten en un sensor muy atrayente para crear nuevos sistemas visuales capaces de ofrecer una serie de ventajas sobre el uso de visión monocular. Por un lado, la información de disparidad que un sistema estéreo proporciona es más invariable a cambios de iluminación, de forma que se pueden crear sistemas más robustos en situaciones reales. Por otro lado, es posible concocer la posición tridimensional del usuario y de esta manera conseguir un mejor seguimiento incluso cuando se producen oclusiones parciales o totales.

En esta memoria presentamos un sistema para la detección y seguimiento de múltiples personas combinando información estereoscópica con información de color de la ropa de las personas que se sigue. El sistema ha sido especialmente diseñado para detección y seguimiento cuando el sistema estéreo es colocado en posiciones bajas (por debajo de la altura de las personas), lo cual es un requisito para los robots que han de interactuar con los humanos [73]. Este mismo sistema de detección y seguimiento ha sido implantado en varios robots como parte de una arquitectura multiagente que implementa las tres habilidades percepto-motoras anteriormente indicadas. Aunque esas habilidades son útiles para un amplio conjunto de aplicaciones robóticas que desean interactuar con humanos, será necesaria la adición de otras habilidades para desarrollar la aplicación deseada. En este sentido, la

filosofía multiagente empleada constituye una ventaja para una fácil y rápida adaptación de la arquitectura a otras aplicaciones.

Podemos concluir afirmando que los trabajos realizados y que se presentan en esta memoria nos han permitido cumplir los objetivos inicialmente fijados. Además, el trabajo realizado apunta hacia futuras vías de investigación principalmente dirigidas hacia la creación de sistemas robóticos capaces de interaccionar de forma natural con las personas utilizando las técnicas desarrolladas en esta memoria.

Estructura de la Memoria

A continuación explicamos la estructura de esta memoria mediante un breve resumen de los contenidos de cada capítulo:

- El Capítulo 2 proporciona una visión general del estado del arte en las principales áreas relacionadas con la presente memoria.

- El Capítulo 3 expone la primera contribución de esta memoria al incremento de la autonomía los robots móviles. En este capítulo se presenta un sistema multiagente, basado en comportamiento visuales y difusos, para controlar la tarea de navegación de un robot móvil en entornos de oficina. El conjunto de agentes diseñado ha sido estructurado en una arquitectura híbrida de tres niveles. El nivel de mayor grado de abstracción, nivel *Deliberativo*, es encargado de crear un plan de movimiento utilizando un mapa topológico del entorno. El mapa está compuesto por el conjunto de habitaciones y pasillos a recorrer, y puertas a cruzar para llegar a una determinada habitación del entorno. El nivel de *Ejecución y Monitorización* (nivel intermedio) transforma el plan en un conjunto secuenciado de habilidades a activar para alcanzar el objetivo del plan y mantiene una monitorización de la ejecución del mismo. En el nivel más bajo de la arquitectura, nivel de *Control*, se localiza un conjunto de agentes que se ejecutan de forma concurrente para mover al robot. Para conseguir una navegación segura, se han utilizado comportamientos difusos capaces de manejar la incertidumbre y la

imprecisión de los ultrasonidos. Además, comportamientos visuales se encargan de localizar la marca artificial que indica la posición de la puerta a cruzar y realizar un seguimiento visual de ésta, indicando al robot la dirección de avance apropiada. El sistema ha sido validado en un numerosos experimentos en entornos de oficina reales.

- El sistema de navegación desarrollado en el capítulo anterior adolece de un problema: no puede ser empleado en entornos no modificables, es decir, se han de colocar las marcas artificiales. El Capítulo 4 muestra nuestra propuesta para la detección de puertas en entornos de interior utilizando visión y lógica difusa, evitando así la necesidad de marcas artificiales. El capítulo presenta un sistema visual difuso basado en una estructura jerárquica de varios clasificadores que permite al robot detectar la presencia de puertas en las imágenes capturadas sin necesidad de utilizar marcas artificiales. Las puertas son encontradas en imágenes de nivel de gris mediante la detección de los bordes de sus marcos. La extracción de bordes se realiza utilizando una variación de la transformada de Hough después de aplicar un detector de fronteras. El sistema propuesto analiza características de los segmentos tales como su longitud, dirección o distancia para descubrir si la relación entre ellos indica la existencia de una puerta. El sistema ha sido diseñado, utilizando conocimiento experto, para detectar puertas rectangulares típicas de muchos entornos de interior. Para probar el rendimiento del sistema se ha creado una base de datos que contiene imágenes de puertas de nuestro entorno de trabajo vistas desde distintas posiciones, ángulos y distancias. Las pruebas realizadas al sistema demuestran su capacidad para detectar puertas incluso en presencia de fuertes deformaciones debidas a la perspectiva. Además, el proceso de análisis es lo suficientemente rápido como para ser empleado en tiempo real en aplicaciones para robots móviles.

- En el Capítulo 5, tratamos el problema del ajuste de las etiquetas lingüísticas del sistema difuso presentado en el anterior capítulo. Si bien el conocimiento experto expresado en la base de reglas difusas da buenos resultados en la tarea de detección de puertas, el ajuste del sistema a las condiciones especificas del entorno donde el robot

trabaja (altura y tamaño de las puertas, distancia focal de la cámara empleada) puede incrementar significativamente el éxito del mismo. Sin embargo, el ajuste del sistema difuso que se presenta en el capítulo anterior se ve complicado por el hecho de que su éxito depende de dos objetivo (la fracción de verdaderos positivos y la fracción de falsos negativos). El ajuste es por tanto un problema de optimización multiobjetivo. En este capítulo se presenta una metodología basada en algoritmos evolutivos para el ajuste de sistemas difusos y se realiza la comparación de diversos algoritmos de optimización monoobjetivo y multiobjetivo para este propósito. El mecanismo de ajuste evolutivo desarrollado permite que el detector de puertas pueda ser adaptado para su uso en un amplio rango de entornos de interior.

- El Capítulo 6 está dedicado al segundo objetivo de nuestro trabajo: el desarrollo de un conjunto de habilidades percepto-motoras que constituyan una base para robots móviles que deseen establecer una natural comunicación con los humanos. En este capítulo se muestra un sistema capaz de detectar y seguir múltiples personas mediante un sistema visión estéreo colocado en posiciones por debajo de la altura de una persona. Esta posición de la cámara es especialmente apropiada para aplicaciones de interacción humano-máquina en que se requiera la interacción con personas ya que facilita la visión de sus gestos y cara. En una fase inicial, el entorno es modelado utilizando un mapa de altura que despues será empleado para extraer fácilmente los objetos que no son del entorno y buscar entre ellos a personas utilizando un detector de caras. Cuando una nueva persona es localizada, el sistema es capaz de seguirla mientras se continúa buscando a nuevas personas en el entorno y siguiendo a las ya encontradas. El seguimiento se basa en el filtro de Kalman para estimar la posición futura de las personas en seguimiento. Sin embargo, cuando dos o más personas se aproximan entre sí, la información acerca del color de sus trajes es también empleada para evitar posibles confusiones (siempre que las personas vistan ropas de distinto color). El sistema desarrollado ha sido probado en numerosos experimentos y su baja latencia lo hace apropiado para su aplicación en tiempo real.

- El Capítulo 7 presenta un sistema multiagente reconfigurable que implementa las tres habilidades percepto-motoras previamente indicadas: (i) detectar a un usuario interesado en interactuar con el robot; (ii) realizar un seguimiento visual de dicho usuario mientras éste se mueve por el entorno; y (iii) moverse siguiendo al usuario por el entorno evitando los obstáculos presentes en el camino. La arquitectura multiagente que se presenta en este capítulo está estructurada en tres niveles. Los agentes del nivel inferior constituyen una capa de abstracción del hardware de forma tal que el sistema puede ser fácilmente portado a otras plataformas robóticas. El nivel intermedio cuenta con un conjunto de comportamientos que les permite controlar el movimiento del robot y realizar un seguimiento visual de los usuarios humanos que interactúan con el robot (utilizando las técnicas desarrolladas en el capítulo anterior). Finalmente, la capa superior combina de forma secuencial o concurrente la activación de los comportamientos de la capa intermedia para desarrollar comportamientos complejos también llamados *habilidades*. Las tres habilidades diseñadas constituyen un conjunto mínimo de herramientas que son de utilidad en diversas aplicaciones robóticas que requieran de la interacción con humanos. Además, la arquitectura multiagente propuesta puede ser fácilmente modificada para agregar a posteriori nueva funcionalidad. El sistema propuesto ha sido probado en diferentes experimentos reales con varios usuarios obteniendo muy buenos resultados y un rendimiento en tiempo real.

- Finalmente, el Capítulo 8 concluye esta memoria exponiendo las principales contribuciones de nuestro trabajo, las publicaciones derivadas de él, y posibles vías de trabajo futuro.

Trabajos Relacionados y Estado del Arte

Este capítulo está dedicado al estudio del estado del arte en las áreas relacionadas con el trabajo desarrollado en esta memoria. En primer lugar, realizamos una breve explicación de los sensores y actuadores más frecuentemente utilizados. En segundo lugar, realizamos una revisión de las principales aproximaciones empleadas para la representación del entorno así como para las tareas de localización. En tercer lugar, realizamos una revisión de las arquitecturas de control. En cuarto lugar, se da una visión de los principales usos de la lógica difusa en tareas de robótica, haciendo especial hincapié en trabajos que usan algoritmos evolutivos para ajuste de sistemas difusos. En quinto lugar, proporcionamos una visión general del uso de algoritmos evolutivos en aplicaciones de robótica. Finalmente, realizamos una revisión de los principales trabajos en visión para robots móviles, así como una breve explicación del uso de visión estéreo para aplicaciones de detección y seguimiento de personas.

Percepción del entorno

Los robots cuentan con un sistema sensorial que les permite obtener información acerca del entorno con el objetivo de extraer información de utilidad. Existe una gran variedad de sensores disponibles, cada uno de los cuales con sus particulares fuentes de error, y capaces de medir distintas propiedades físicas. Los sensores pueden ser clasificados en dos grandes ejes funcionales como *propioceptivos/extereoceptivos* y *pasivos/activos* [195].

Los sensores *proprioceptivos* miden valores internos del robot; por ejemplo: nivel de carga de las baterías, temperatura, carga de las ruedas. Por otro lado, los sensores *extereoceptivos* recogen información sobre el entorno del robot; por ejemplo, medidas de distancia,

intensidad lumínica, amplitud de una onda de sonido emitida. Por tanto, estos sensores son utilizados por el robot con el objeto de obtener información de utilidad sobre el entorno.

Los sensores *pasivos* miden propiedades del entorno midiendo señales externas; por ejemplo, micrófonos, cámaras CCD o CMOS. Por el contrario, los sensores *activos* emiten energía y luego esperan a la reacción que esta produce en el entorno; ejemplos de estos sensores son el sonar, láser o infrarrojos.

A continuación se expone una lista de los sensores más comúnmente utilizados en robots móviles, así como una indicación de sus deficiencias y limitaciones:

- Los sensores de *ultrasonidos* permiten detectar el eco de una señal ultrasónica. Este es uno de los sensores más frecuentemente empleados para la medición de distancias. Poseen una buena precisión en condiciones de operación óptimas y permiten un rango de operación que varía desde los 20 cm a los 10 m. Es un sensor muy barato, pero que tiene diversas deficiencias. En primer lugar, es difícil estimar de forma precisa la posición de los objetos que producen el eco de la señal, dado que normalmente la señal emitida lo hace en un cono que varía entre los 15^{o} y 30^{o}. En segundo lugar, la señal no ha de rebotar necesariamente de forma perpendicular a la superficie. Por tanto, el eco puede ser perturbado, perderse o ser absorbido. En tercer lugar, superficies colocadas muy cerca del emisor de sonido pueden provocar problemas en la recepción de la señal.

- Los sensores de *infrarrojos* emiten una señal de luz en lugar de una señal de sonido. De esta manera consiguen una mejor precisión angular y la habilidad de detectar objetos cercanos. Sin embargo, requieren de una precisa calibración y se ven afectados por la fuentes externas de luz (entre ellas la luz natural).

- Los sensores de *láser* trabajan de forma similar a los anteriores pero emitiendo una señal de láser. El rayo rebotado es medido para calcular la distancia a los objetos de alrededor. La principal ventaja de este sensor es su gran precisión angular (approx. 1^{o}) y su bajo error a largas distancias.

- La *información visual* es probablemente uno de los sensores más potentes que puede emplear un robot móvil. La visión es el sensor principal del ser humano y es fun-

damental para una adecuada interacción en nuestra vida diaria. Por ello, la visión ha sido ampliamente explorada para proporcionar a los robot capacidades visuales. Mas adelante se da una introducción más detallada del uso de visión por computador en los robots.

- Los *giroscopos* son otro sensor muy utilizado para medir o mantener la orientación, basándose en el principio de conservación del momento angular. Hoy en día, los giroscopos electrónicos son aparatos pequeños y baratos que pueden ser fácilmente colocados en plataformas robóticas.

- Los *acelerometros* son aparatos capaces de medir la aceleración. Suelen utilizarse junto con giroscopios en sistema de guiado inercial. Sin embargo, los acelerómetros son aparatos poco precisos y muy sensibles a la inclinación del suelo así como a la aceleración de la gravedad.

- Las *brújulas* son un instrumento para determinar direcciones en la tierra. Este aparato consiste en un puntero magnetizado y libre para alinearse en función del campo magnético terrestre. Su principal desventaja es que este aparato puede verse afectado por la presencia de campos magnéticos y por tanto su uso suele descartase en entornos de interior.

El lector interesado en más detalles acerca de las características de cada sensor es referido a [112, 195] para una lectura más profunda.

Actuadores

Los actuadores son mecanismos mediante los cuales los robots reaccionan ante los eventos de su entorno. Un robot puede contener un amplio conjunto de distintos actuadores, cada uno de ellos con una tarea específica. Los actuadores son normalmente utilizados para tareas de locomoción o de manipulación. A pesar de la gran variedad de posibles actuadores, la mayoría consiste en motores eléctricos. Los motores eléctricos son actuadores que producen movimiento a partir de la electricidad utilizando el efecto electromagnético. Los robots

pueden ser clasificados en función de su habilidad para moverse en: (i) robots con ruedas, (ii) robots con piernas, y (iii) robots deslizantes.

Representación del entorno

El diseño de un adecuado modelo de representación del entorno es necesario para el desarrollo de tareas de navegación o manipulación. El uso de un mapa del entorno sirve a los robots para conocer su propia posición, así como para crear un adecuado plan para la navegación. Tres son los enfoques que pueden verse en la literatura para solucionar el problema.

- Los enfoques geométricos [64] consisten en el uso de complejos modelos 3D o mapas de celdas para la representación del entorno. En particular, estos últimos son los más populares. Estos representan el entorno como una matriz de celdas, cada una de ellas representando un región plana del espacio. Cada celda suele llevar asociado un valor de ocupación que indica si en dicho lugar del espacio hay obstáculos. Esta información es empleada por el robot para poder realizar planificación de trayectorias. La principal ventaja del uso de enfoques geométricos, es que pueden utilizarse para generar trayectorias muy eficientes. Sin embargo, requieren del uso de mecanismos precisos de localización.

- Los enfoques topológicos [120] consisten en detectar diversos elementos del entorno e identificar la relación entre ellos. En estos casos, los mapas normalmente representan el entorno una mayor nivel de abstracción que los enfoques geométricos. Es muy común la utilización de grafos para la representación del entorno en este tipo de enfoques. Thrun, en [205], crea un mapa topológico a partir de una división en regiones de un mapa de celdas. Sin embargo, el significado asociado a los nodos y los arcos varía de unos autores a otros. Kuipers y Byun, en [126], utilizan nodos para representar lugares destacados en el entorno desde el punto de vista del sistema sensorial, y los arcos representan caminos entre esos lugares. En algunos casos, la representación puede incluir información acerca de la propia forma de navegar entre los nodos. En [145],

Mataric utiliza nodos para representar lugares destacables y son asociados a comportamientos. Aguirre y González, en [3], utilizan arcos para indicar los comportamientos necesarios para navegar entre nodos.

- Finalmente, otros autores [3] han propuesto el uso de aproximaciones híbridas para combinar las ventajas de ambos enfoques.

Localización

Una de las principales problemáticas que se presentan en la construcción de robots móviles es el conocimiento de la posición del vehículo. Un robot móvil ha de ser capaz de conocer su posición para el desarrollo de diversas tareas, y en especial para la de navegación.

La precisión requerida en la localización depende la aplicación y en el tipo de arquitectura de control empleado. Mientras que ciertas aplicaciones requieren del uso de mecanismos precisos de localización, otros solo hacen uso de posiciones aproximadas. Por ejemplo, los enfoques geométricos de representación del entorno suelen requerir el uso de mecanismos precisos de localización. Sin embargo, los enfoques topológicos usualmente requieren conocer la posición del robot en el grafo, lo cual suele requerir menos precisión que los enfoques geométricos.

La localización es un problema bien estudiado en la literatura y diversos autores han propuesto diferentes técnicas para solucionar el problema. A continuación, damos una breve visión de las técnicas más comunes para localización. Para una revisión más completa, el lector interesado es referido a [216].

- *Odometría*: La odometría es el sistema de posicionamiento de robots más usado, por ser más económico y por su bajo coste de computación. El problema principal que presenta este método es la dificultad de saber el error que se comete, lo que provoca que a medida que el robot se mueve se van acumulando los errores, provocando grandes errores de posicionamiento. A pesar de estas limitaciones, se piensa en este método como una parte importante de los sistemas de navegación y, así mismo, las tareas de

navegación se simplificarían mucho si la precisión de este método mejorara.

La odometría está basada en ecuaciones simples consistentes en hacer lineal el giro que da la rueda del robot, midiendo así la distancia lineal recorrida por la rueda sobre el suelo. El principal problema es el derrape de las ruedas, ya que el desplazamiento de éstas no puede ser linealmente aplicado al movimiento del robot sobre el suelo.

Los errores que se producen en este tipo de modelos pueden ser categorizados en dos tipos: errores sistemáticos y errores no sistemáticos. Los *errores sistemáticos* son aquellos que surgen como consecuencia de las imperfecciones del robot como, por ejemplo, un desigual diámetro de las ruedas. Los *errores no sistemáticos* son aquellos que resultan de la interacción de las ruedas sobre el suelo como, por ejemplo, el deslizamiento de la rueda o un bache en el suelo. La principal dificultad que presenta este método es medir los errores que se cometen. Sin embargo, existen diferentes propuestas [26, 27] para medir ambos tipos de error.

- *Navegación inercial*: La navegación inercial se vale de giroscopios y acelerómetros para medir las rotaciones y las aceleraciones del robot en su movimiento.

  La ventaja que ofrecen estos sistemas es que no necesitan del uso de referencias externas. Los acelerómetros para la navegación de robots móviles generalmente han sido pobres. Los acelerómetros sufren de grandes desviaciones y son muy sensibles a la inclinación del suelo ya que ésta hace que los sensores detecten parte de la aceleración de la gravedad provocando que se cometan errores. Los giroscopios son de particular importancia en el posicionamiento de robots móviles porque pueden compensar los errores de la odometría. En los métodos basados en la odometría, una pequeña desviación momentánea provoca que los errores crezcan de forma constante durante el movimiento del robot. Por esta razón, es muy beneficioso detectar los errores de navegación y corregirlos inmediatamente. Hasta hace muy poco los giroscopios de alta precisión tenían un coste muy elevado, siendo imposible su utilización en robots. Pero, recientemente, los giroscopios de fibra óptica (que tienen una gran precisión) han hecho caer los precios de forma drástica.

- *Brújulas magnéticas*: Este método consiste en dar la orientación en base al campo magnético terrestre que se detecte en el lugar.

  El principal problema que tiene este tipo de posicionamiento es que el campo magnético de la Tierra es distorsionado por las líneas eléctricas. Esto hace que el uso de las brújulas magnéticas en recintos cerrados sea difícil. Existen diferentes métodos para este tipo de medición basándose en los diferentes efectos físicos que se dan en relación al magnetismo de la Tierra. El método más utilizado es el de la brújula de inestabilidad (*fluxgate compass*) que mide la componente horizontal del campo magnético de la Tierra. Éste método tiene una gran cantidad de ventajas: bajo consumo de potencia, no contiene partes móviles, no se ve afectado por vibraciones y un relativo bajo coste.

- *Balizas activas*: La navegación usando balizas ha sido la forma de navegación más usada desde el principio de los tiempos en el campo de la aeronáutica. Las balizas pueden ser detectadas con fiabilidad y proporcionar una gran precisión en la posición, con un bajo coste de procesamiento. El principal problema que presenta esta técnica es el alto coste de instalación y de mantenimiento del material. La precisión en la instalación de los balizas es primordial para después obtener buenos resultados. Son dos los métodos que se pueden distinguir, la trilateración y la triangulación.

  La trilateración es la determinación de la posición del vehículo basándose en la medición de la distancia a las balizas colocadas en posiciones conocidas. En la navegación usando trilateración se suelen usar tres o más transmisores que se colocan en posiciones conocidas y un receptor en el robot. Se hace conveniente la colocación de un emisor en el robot y un receptor colocado en las paredes. Un ejemplo de este método de posicionamiento es el Sistema de Posicionamiento Global que será explicado más adelante.

  En la triangulación hay tres o más balizas de transmisión, y un sensor en constante rotación colocado en el robot. Este sensor rotatorio registra los tres ángulos relativos a los ejes longitudinales del robot. A partir de esas mediciones se puede calcular las

coordenadas del robot.

- *Sistemas de posicionamiento global*: Este sistema es una técnica revolucionaria para la navegación en espacios abiertos. Está constituido por un conjunto de satélites que transmiten señales codificadas, usando unos avanzados métodos de trilateración. Unos sensores colocados en la Tierra pueden medir su posición midiendo el tiempo que tarda la señal en llegar. Sabiendo la distancia del sensor a los tres satélites se puede saber la latitud, longitud y altitud en que se encuentra el sensor. El gobierno de los EE.UU., responsable de su desarrollo, puede aplicar pequeños errores para evitar que posibles países hostiles utilicen esta tecnología como base para armas de gran precisión.

- *Posicionamiento usando marcas*: Las marcas han sido muy utilizadas para tareas de navegación y localización de robots. Es posible clasificarlas en *artificiales* y *naturales*.

  Las *marcas artificiales* se han utilizado con éxito en la literatura como un medio para proporcionar información adicional a un robot y facilitar los procesos de localización así como en el desarrollo de otro tipo de tareas [58, 132]. Distintos diseños de marcas se han propuesto en función del propósito perseguido [117, 191] e incluso se ha estudiado la mejor disposición espacial de ellas para minimizar el error de posicionamiento [203].

  Las *marcas naturales* son aquellos elementos del entorno del robot que, de alguna manera, pueden ser utilizados como referencias. Estamos hablando de paredes, puertas o pasillos en entornos de interior, o de calles o montañas en entornos de exterior.

- *Posicionamiento basado en mapas*: El posicionamiento basado en mapas, también conocido como *map matching*, es una técnica en la que el robot usa sus sensores para crear un mapa *local* del entorno que se compara con uno *global* previamente almacenado para determinar su posición. El mapa *global* puede ser un modelo *CAD* (Computer Asisted Design) del entorno o bien puede haber sido construido con los mismos sensores del robot. La principal ventaja de esta técnica es que el robot usa para posicionarse información sobre el propio entorno en el que se mueve sin necesidad de

adaptarlo. Además, se pueden implementar algoritmos de reconocimiento del entorno de tal forma que el robot aprenda a mejorar su precisión mediante inspecciones del terreno. Las desventajas que tiene esta técnica son los requerimientos de precisión de los sensores y llevar a cabo el proceso de "emparejamiento" con el mapa global.

Arquitecturas del Control

En la medida en que las tareas encomendadas a un robot autónomo son cada vez más complejas, los algoritmos que se ejecutan para llevar a cabo dichas tareas se complican en la misma proporción, llegando a convertirse en sistemas de difícil construcción y mantenimiento.

Una forma común de reducir la complejidad de un gran sistema software es descomponiéndolo en un número determinado de módulos menores que realizan, cada uno de ellos, una tarea específica. El método de construcción de este gran sistema global, a partir de dichos módulos, es lo que a menudo se conoce con el nombre de *Arquitectura de Control* de un robot autónomo [61].

En los últimos años han surgido diferentes arquitecturas de robots autónomos. Cada una de ellas ofrece ventajas específicas para un conjunto de aplicaciones. La adecuación de una arquitectura varía dependiendo de la aplicación que el robot va a desarrollar y del entorno en que ésta se lleve a cabo.

Se pueden agrupar las arquitecturas existentes en la actualidad en tres categorías desde el punto de vista funcional: deliberativas, reactivas e híbridas. Otra clasificación, aunque similar, es la propuesta en otros trabajos como en [14] o [129], donde se habla de tres arquitecturas: funcionales (o deliberativas), basadas en comportamientos (o reactivas) y las arquitecturas híbridas.

Desde el punto de vista topológico se pueden clasificar en arquitecturas verticales y horizontales. Las horizontales tienen la característica de que todos los niveles tienen acceso a las percepciones, y determinan de esta forma la actuación. En las verticales, existe un nivel de percepción y otro de acción. Las percepciones adquiridas se procesan en los diferentes niveles y se determina la acción a realizar.

A continuación se describen en mayor profundidad la clasificación desde el punto de vista funcional, y se realiza una introducción a las arquitecturas de agentes.

Arquitecturas Deliberativas

Las arquitecturas deliberativas, también conocidas como arquitecturas jerárquicas, se basan la hipótesis del sistema de símbolos [166]. La esencia de esa hipótesis puede ser resumida de la siguiente forma: un sistema de simbolos físicos tiene los medios necesarios y suficientes para la acción inteligente.

Los sistemas de control basados en las arquitecturas deliberativas tienden a estar muy estructurados y siguen el enfoque tradicional de descomposición jerárquica de arriba a abajo. Suelen tener planificadores y módulos de resolución de problemas más sofisticados que mantienen una representación compleja del mundo real que les permite, junto con la información sensorial, generar un conjunto de acciones apropiadas. Entre otros trabajos que se basan en este modelo se encuentra el trabajo de *J.S. Albus* en el desarrollo de la arquitectura jerárquica RCS, [7, 8, 9].

Una importante ventaja de estas arquitecturas es la mayor capacidad que poseen de representar el conocimiento necesario para la realización de una tarea haciendo uso de estructuras más complejas. Además, su naturaleza más estructurada permite un mejor entendimiento del componente de control existente en el sistema [129].

Estas arquitecturas dominaron los primeros sistemas de control en los comienzos de los robots autónomos. El primer robot *inteligente*, Shakey [168], fue desarrollado en el Stanford Research Institute en los años 60. Su arquitectura de control estaba basada el enfoque SPA (Sense-Plan-Act) [25] (ver Fig. 2). Los sensores eran empleados para percibir el mundo externo con el propósito de crear un modelo completo y preciso del mundo. Despues, se calculaba el conjunto de acciones a realizar para alcanzar su objetivo. Cada acción era pasada de forma directa al controlador que la ejecutaba. Por tanto, las acciones que el sistema era capaz de manipular eran acciones de bajo nivel relativas al hardware.

Los sistemas de control basados en las arquitecturas deliberativas son muy eficientes en situaciones en las que el entorno es totalmente conocido y no encontramos incertidumbre.

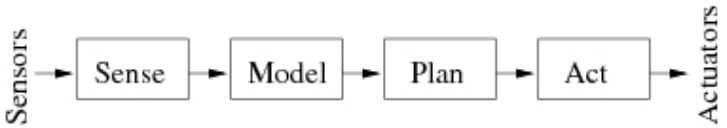


Figure 2: Modelo Sense-plan-act.

Pero cuando la tarea a realizar no está bien definida en tiempo de diseño o existen indeterminaciones en tiempo de ejecución, estos sistemas no pueden manejar las contingencias adecuadamente.

Arquitecturas Reactivas

En fuerte constaste con las arquitecturas deliberativas encontramos las arquitecturas reactivas, en que la percepción está fuertemente ligada a la acción. Estas arquitecturas también se denominan arquitecturas basadas en comportamientos [14, 34, 146]. *Brooks* puede considerarse el investigador que más ha influido en esta corriente de diseño basándose en las consideraciones de que: (i) el comportamiento inteligente puede ser generado sin una representación explicita del tipo que propone la IA simbólica; (ii) el comportamiento inteligente puede ser generado sin la necesidad de un mecanismo abstracto de razonamiento; y (iii) la inteligencia es una propiedad emergente de ciertos tipos de sistemas emergentes. *Brooks* identifica dos ideas clave que guían su investigación:

- Situabilidad y Corporeidad: la inteligencia "real" esté situada en el mundo, no en sistemas confinados a la memoria de un ordenador como demostradores de teoremas y sistemas expertos.

- Inteligencia y Emergencia: un comportamiento "inteligente" aparece como resultado de la interacción de los agentes con su entorno. Además, la inteligencia está en el ojo del que mira y no es una propiedad aislada e innata.

Para ilustrar sus ideas, crea la arquitectura de subsunción [34] (ver Figura 3). Esta arquitectura es una jerarquía de comportamientos orientados a tareas. Cada comportamiento, es

un sistema que reacciona a los estímulos externos y que compite con los demás para tomar el control del robot. Dentro de la propia arquitectura localiza agentes de más bajo nivel (como evitación de obstáculos) y otros de más alto nivel. El sistema resultante es extremadamente rápido en su funcionamiento y realiza tareas que no podrían ser realizadas por la IA simbólica. Otros ejemplos destacables de este tipo de arquitectura los encontramos en las arquitecturas de *selección de acción* de Maes' [139, 140] y los *esquemas motores* de Arkin [13].

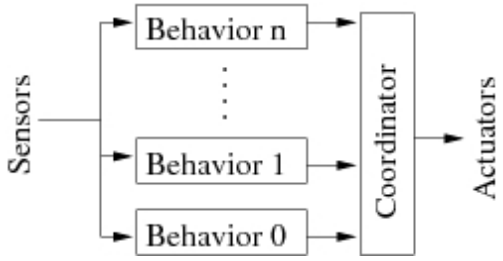


Figure 3: Arquitectura basada en comportamientos de Brooks.

Arquitecturas híbridas

Las arquitecturas híbridas se encuentran entre los dos extremos. Mientras que los sistemas de control basados en ellas mantienen las características de reacción rápida ante cambios del entorno, puesto que comparten elementos de las arquitecturas reactivas, su mayor capacidad computacional y poder de representación permite la realización de tareas más complejas, acercándose en este aspecto a los sistemas basados en arquitecturas deliberativas. Así, y a diferencia de las arquitecturas reactivas, estos sistemas no están limitados en su capacidad de representación.

La idea fundamental es descomponer el comportamiento inteligente global buscado para un sistema en una colección de comportamientos más simples. Cada uno de estos comportamientos más primitivos se construirán computacionalmente y su unión dará lugar a un comportamiento cada vez más complejo y cercano al deseado para todo el sistema.

Usualmente, estas arquitecturas tienen un estructura de tres niveles (ver Figura 4). La capa inferior suele estar compuesta por un conjunto de comportamientos reactivos que es-

tablecen una relación directa entre la acción y la percepción. La capa superior está al cargo de crear un plan de alto nivel de acuerdo con los objetivos y creencias del robot. Esta capa es capaz de manejar símbolos y de razonar acerca del estado del robot. Finalmente, la capa intermedia suele servir de puente entre la capa superior y la inferior. La capa intermedia puede recibir una plan creado a un alto nivel de abstracción que es descompuesto en subobjetivos. Después, en función del estado de ejecución del plan, esta capa puede coordinar la activación o desactivación de los comportamientos en la capa inferior.

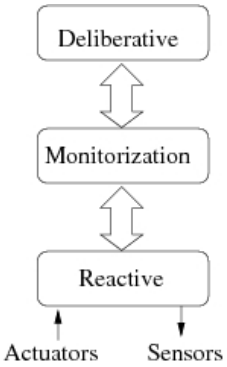


Figure 4: Arquitecturas híbridas.

Ejemplos de este tipo de arquitectura son Aura [12] de Arkin o el sistema Gat's Atlantis [84] empleado el vehículo de exploración marciana JPL's rovers.

Arquitecturas basadas en agentes

La teoría de agentes se ha popularizado en los últimos veinte años. A pesar de que el concepto de agente ha estado siempre muy relacionado con la Inteligencia Artificial, su estudio no ha atraído la atención de los investigadores hasta los años 80. Definir el término *agente* es algo complejo dado que ha sido comúnmente empleado en áreas muy diversas y con diferentes propósitos. Para nuestros propósitos, un agente puede ser entendido como un proceso software destinado a conseguir o mantener un objetivo que es implícito a su propio diseño. Puede decirse, que un agente debe cumplir las siguientes características [217]:

- *Autonomía*: los agentes deben tener la capacidad de operar independientemente de ser humano y tener control sobre sus acciones.

- *Habilidades sociales*: los agentes debe ser capaces de comunicarse con otros agentes o con seres humanos utilizando algún *lenguaje de comunicación de agentes*.

- *Reactividad*: un agente percibe su entorno (sea este físico o no) y reacciona ante los eventos que suceden.

- *Pro-actividad*: las acciones de los agentes están dirigidas por sus objetivos.

Desde un punto de vista práctico, los agentes pueden verse como la extensión natural del paradigma orientado a objetos. Los agentes encapsulan funcionalidad, objetivos e intenciones y se comunican con otros agentes para establecer una cooperación. Algunas de las ventajas de emplear agentes para el diseño de sistemas complejos son: (i) la posibilidad de crear sistemas distribuidos como un conjunto heterogéneos de agentes; y (ii) la facilidad para agregar, eliminar o alterar la funcionalidad de un sistema de forma incremental.

Las arquitecturas basadas en agentes han sido comúnmente empleadas para coordinar y controlar conjuntos de robots móviles encargados de realizar una tarea de forma coordinada. Colecciones de dos o más robots móviles que trabajan de forma conjunta es lo que en la literatura suele denominarse como equipos, sociedades o sistemas multiagente. Los equipos robóticos tienen usos potenciales en tareas como la extinción de incendios o misiones de rescate o exploración, entre otras. Cada robot individual puede ser un aparato de reducido coste cuyo fallo puede ser fácilmente superado mediante el trabajo del resto del grupo. Por tanto, los equipos de robots son sistemas muy robustos ante fallos. Este campo de investigación emergente se engloba dentro de lo que se denomina Inteligencia Artificial Distribuida. Un famoso ejemplo del esfuerzo de investigación en este sentido es la competición RoboCup, donde equipos formados por varios perros robots compiten en una liga de fútbol.

Otro uso posible de un conjunto de agentes es para modularizar el diseño de arquitecturas de control de robots móviles. En esta memoria, nosotros hacemos uso de agentes para diseñar nuestras arquitecturas de control. En este sentido, nosotros utilizamos el término *arquitectura multiagente* para referirnos a sistemas cuya funcionalidad es distribuida entre un conjunto de agentes heterogéneos capaces de cooperar para alcanzar un objetivo.

Lógica Difusa y Robótica

La lógica difusa se ha convertido en una potente herramienta en diversas áreas de la robótica. Las razones principales de la generalización de su uso son que la lógica difusa permite: (i) manejar y propagar la incertidumbre y vaguedad asociada a las medidas dadas por los sensores de forma natural; y (ii) definir conocimiento experto utilizando un conjunto de reglas del tipo *si-entonces*. En esta sección, proporcionamos una breve introducción a los usos de la lógica difusa en robótica. El lector interesado es referido a [186] para mayor información sobre este tópico.

- Diseño y coordinación de comportamientos: debido a que la lógica difusa es capaz de manejar la incertidumbre y la vaguedad de los sistemas sensoriales, ha sido ampliamente empleada para el diseño de comportamientos. Un ejemplo representativo de esto es la arquitectura Shaphira [119]. Shaphira es un arquitectura consistente en un conjunto de comportamientos difusos (codificados como reglas Mandani) que son capaces de desarrollar tareas complejas. En esa arquitectura, el grado de creencia del antecedente de la regla es empleado para calcular su nivel de activación. Los autores Aguirre et. al. emplean en [1] un enfoque muy similar mediante la definición de varios comportamientos básicos: (*seguir-pared*, *navegar-en-pasillo*, *evitar-obstáculo*, etc.) expresados como conjuntos difusos. Los comportamientos son adecuadamente coordinados en función del contexto permitiendo una suave transición entre ellos. En el trabajo de Bonarini y Basso [23], se realiza el control del robot mediante un conjunto de comportamientos básicos agrupados en dos clases en función del contexto de aplicación. Él distingue entre comportamientos *globales* y comportamientos de *contexto*. Los primeros son empleados para alcanzar objetivos del robot, como *seguir-pared*, mientras que los segundos son empleados para solucionar situaciones inesperadas, como *evitar-obstáculo*.

- Construcción de mapas: la lógica difusa ha sido empleada de forma satisfactoria para la construcción de mapas del entorno utilizando tanto enfoques geométricos como topológicos. Gasós y Saffiotti, en [83], proponen un técnica para la construcción

de mapas geométricos utilizando sonar. En una fase inicial, el robot crea un mapa global del entorno que posteriormente es actualizado utilizando percepciones locales. El mapa está basado en la utilización de bordes difusos proyectados en un mapa de celdas. De forma parecida, Aguirre et. al. presenta en [2] una aproximación híbrida (geométrica y topológica) para la construcción de mapas utilizando segmentos difusos construidos utilizando sensores de ultrasonidos. Otros trabajos emplean lógica difusa para crear mapas son [81, 172].

- Localización: la lógica difusa ha sido empleada para calcular la posición de un robot como una medida difusa. En lugar de representar la posición utilizando medidas probabilísticas, algunos autores han empleado una región de posibilidad para indicar la posición del robot. Este enfoque permite una fácil integración con los mapas difusos previamente explicados. Ejemplos de este enfoque son [83, 188]

- Percepción: La lógica difusa también ha sido utilizada para diseñar sistemas perceptuales. Howard et al. proponen en [103] un método basado en lógica difusa para analizar las características del terreno por el que navega un coche autónomo. Ellos utilizaron un sistema visual para calcular características como *transversabilidad y discontinuidad* como medidas difusas. Esas variables eran posteriormente empleadas por el controlador de un robot para determinar la mejor estrategia de movimiento. Le et. al. muestran en [133] un sistema visual difuso para detectar los bordes de una carretera utilizando imágenes de su robot THMR-III.

Ajuste Evolutivo de Sistemas Difusos

Tal y como se ha comentado anteriormente, los sistemas difusos han sido ampliamente empleados para multitud de aplicaciones en robótica. Probablemente, una de las razones que los hacen tan populares es que permiten crear sistemas que incluyen el conocimiento de un experto como un conjunto de reglas. Por consiguiente, los sistemas difusos pueden ser fácilmente creados e interpretados por expertos. Sin embargo, la selección del conjunto apropiado de valores para las etiquetas difusas es un trabajo difícil y tedioso. La configu-

ración óptima de un sistema difuso para un contexto particular de aplicación puede diferir de la configuración óptima para otro contexto. Por ejemplo, imaginemos un controlador difuso que implementa el comportamiento *seguir-pared*. Aunque las reglas y las variables del comportamiento pueden ser apropiadas para una amplia variedad de entornos, el ajuste de las etiquetas puede mejorar las condiciones de operación para cada entorno particular.

El ajuste de sistemas difusos proporciona un mecanismo automático para incrementar su rendimiento a partir de una conjunto de datos validado que describe el problema. Tres son los principales enfoques de ajuste que pueden encontrarse en la literatura [52]: ajuste de las funciones de escalado que mapean las entradas y salidas de las variables al rango en que las variables difusas están definidas [55, 179, 224]; ajuste de las funciones de pertenencia mediante su movimiento, estrechamiento o alargamiento [87, 170, 213]; y el ajuste de reglas difusas modificando las etiquetas difusas en la parte *entonces* de las reglas [24]. A pesar de que el ajuste de sistemas difusos ha sido abordado con técnicas clásicas de optimización [155], el uso de algoritmos genéticos (AGs) se ha hecho muy popular en este campo [39, 50, 94, 99, 141]. El término *Sistema Difuso Genético* (SDG) ha sido empleado en la literatura para referirse a aquellos sistemas difusos que de alguna manera se apoyan en el uso de AG, bien para definir su estructura, o para adaptar algunos de sus parámetros [49, 52, 177].

A pesar del gran esfuerzo realizado para la optimización de sistemas difusos utilizando aproximaciones monoobjetivo, hay la necesidad en aplicaciones reales de cumplir con varios objetivos al mismo tiempo. A ese tipo de problemas se les denomina *multiobjetivo* (en [43] puede encontrarse una interesante revisión de este tipo de trabajos). El concepto de *Pareto-óptimo*, formulado por Vilfredo Pareto [176], constituye el origen de la investigación en la optimización multiobjetivo basada en Paretos. Una solución se dice que es Pareto-optimal (o *no dominada*) si ninguna otra solución encontrada es mejor que ella en resolver al menos uno de los objetivos del problema. El término OEM (Optimización Evolutiva Multiobjetivo) hace referencia a un conjunto de técnicas evolutivas destinadas a resolver este tipo de problemas utilizando enfoques inspirados en la evolución natural. La principal ventaja de los algoritmos OEM (AsOEM), es la habilidad de buscar simultáneamente varias soluciones no dominadas. Los AsOEM son capaces de encontrar varias soluciones no dominadas de

un problema en una única ejecución del algoritmo, mientras que varias ejecuciones de un algoritmo monoobjectivo sería necesarias para obtener resultados similares.

Los AsOEM también han sido empleados en la optimización de SDG. La gran parte del trabajo en este sentido ha estado enfocado hacia el aprendizaje de la base de reglas del sistema difuso [51, 80, 108, 110]. Sin embargo, también hay enfoques para el ajuste de las funciones de pertenencia sistemas difusos. En [22], un sistema difuso que controla un misil es ajustado utilizando un AsOEM. Con ello se consiguió optimizar de forma simultánea el tiempo de ascensión, el tiempo de estabilización y el error estacionario del controlador. En [42], se ajustó un sistema difuso para controlar un tren urbano autónomo. El sistema estaba destinado a controlar la operación de un tren destinado al transporte de personas entre estaciones. Estaba compuesto por un total de 16 parámetros y los resultados demostraron que el método de optimización empleado permitía optimizar el sistema de acuerdo con los objetivos de: puntualidad, ahorro energético y confort de los pasajeros.

Algoritmos evolutivos aplicados a la robótica

El término robótica evolutiva (RE) se aplica a la utilización de algoritmos evolutivos (AE) con el propósito de crear estructuras de control, conocimiento o incluso cuerpos de robots. Existen multitud de razones que inspiran esta tendencia. Por un lado, los AE permiten una optimización tanto estructural como paramétrica de un sistema. En segundo lugar, la evolución (tanto natural como artificial), tiene la habilidad de evitar el estancamiento en mínimos locales del espacio de búsqueda. De esta manera, con el tiempo suficiente, un AE suele encontrar una solución cercana al óptimo global. Debido a su naturaleza estocástica, la evolución es capaz de encontrar un conjunto de soluciones distintas pero válidas a un mismo problema. Todo esto junto al hecho de que el diseño manual de sistemas inteligentes, incluso en los casos más simples, es una tarea muy difícil, ha provocado que este campo concentre una parte importante de la investigación en el área de la robótica. La RE se ha empleado en diferentes ámbitos dentro del extenso campo de la robótica:

- En [70], Floreno y Mondada evolucionan comportamientos para la navegación en un

robot Khepera. Los experimentos tenían como objetivo el desarrollo de comportamientos para la navegación evitando obstáculos y para ello realizaron el proceso evolutivo en un entorno real con obstáculos estáticos. Los autores definen una función de evaluación que premia (i) alta velocidad de movimiento, (ii) el movimiento rectilíneo y (iii) la evitación de obstáculos. El *cerebro* del comportamiento consistía en una red neuronal con algunas conexiones recurrentes en la capa de salida. El proceso de evolución consistía en la determinación del número de neuronas, así como del peso de sus conexiones. En el transcurso de la evolución, se dieron cuenta de que los mejores individuos exhibían un comportamiento inteligente, más allá del inicialmente previsto al diseñar la función de evaluación. Moderaban la velocidad al encontrarse en las proximidades de obstáculos.

- Nelson et al. [165] realiza una experimentación similar pero en simulación e incorporando mecanismos más sofisticados para el modelado temporal de la información. El objetivo era la creación de comportamientos para la navegación en un laberinto y evitando obstáculos. Para ello, utilizaron redes neuronales con varias capas, además de conexiones recurrentes que procesaban las salidas del comportamiento utilizando una memorización de estados anteriores. En estos experimentos la función de evaluación premiaba la distancia recorrida y penalizaba aquellas situaciones en las que el robot quedaba bloqueado en el laberinto. Las mejores redes contenían entre 1 y 3 capas ocultas con 5-10 neuronas con capa y fueron finalmente implantadas en un robot real equipado con cinco sensores táctiles.

- Miglino et al. en [149] evolucionó en simulación una simple red neuronal para implementar un comportamiento de exploración. La red estaba formada por un par de neuronas en la capa de entrada (una para cada sensor del robot), dos unidades ocultas y dos unidades de salida (una para cada motor). Adicionalmente, se agregó una neurona de memoria mediante un conexión recurrente en la capa oculta. El entorno de simulación consistía en un mundo de cuadrículas que tenía que ser explorado por el robot. La función de evaluación premiaba el número de cuadrículas visitadas y penal-

izaba el tiempo empleado. En el proceso de los experimentos se dieron cuenta que si colocaban al robot siempre en la misma posición de partida, las redes mejores tendía a aprender buenos recorridos. Para evitar esto, cada individuo era evolucionado para distintas posiciones aleatorias de partida. Tras el proceso evolutivo las redes fueron implantadas en robots Lego y probadas en un entorno real. Comprobaron que el comportamiento que exhibían las redes en simulación difería del observado en la realidad y que la razón era la falta de ruido a la hora de realizar la evolución. Al evolucionar las redes utilizando ruido, el sistema implantado en el robot exhibía un comportamiento más similar al observado en la realidad.

- Nolfi y Marocco en [169], crean una red retro-alimentada para el control de un Kepera guiado por sus sensores de sonar y visión. El entorno de trabajo era un cuadrilátero blanco en el que se habían dibujado dos bandas de color negro de distinto tamaño en lados opuestos del mismo. El objetivo del robot era acercarse a la banda mayor, para lo cual tenía que ser capaz de distinguir, de alguna forma, su tamaño, lo cual es una tarea en principio compleja al utilizar únicamente una cámara. Los resultados que se obtuvieron, fueron que los mejores controladores eran capaces de realizar la tarea explotando de manera muy inteligente dos características: (i) el hecho de que el cambio en el tamaño angular de un objeto, al aproximarse hacia él, varía con la distancia al objeto y (ii) que la duración de un barrido visual a lo largo de un objeto es proporcional a su tamaño. De esta manera, el comportamiento que se observó en los mejores individuos, consistían en girar hasta tener contacto visual con alguna de las bandas, entonces se desplazaban hacia ella girando ligeramente. Al perder el contacto visual, giraban sin desplazarse hasta encontrar la otra en el extremo opuesto y volvía a realizar la misma operación de giro con desplazamiento. De esta manera el controlador era capaz de aproximarse hacia la banda más grande.

- Reynolds en [183] evoluciona el comportamiento *navegar-en-pasillo* utilizando *Programación Genética*. En su propuesta añade un adecuado modelo de ruido en los sensores así como en los actuadores para conseguir un comportamiento realista en su

traspaso al mundo real. Como entradas del sistema se tenían los valores proporcionados por los sensores de sonar y como salida valores limitados para el ángulo de giro de las ruedas del robot. El objetivo del sistema es proporcionar al robot un comportamiento de moverse en un pasillo sin chocar con las paredes del mismo. De esta forma, durante el proceso evolutivo, los individuos eran automáticamente descartados en cuanto exhibían un comportamientos que colisionaba. Los resultados obtenidos corroboraron que los sensores más importantes para esa tarea eran los frontales en la dirección de avance del pasillo y no tanto los laterales ni traseros.

- Mucientes y Casillas [157] presentan un método para la evolución de comportamientos representado mediante reglas difusas utilizando colonias de hormigas. En este trabajo, evolucionan el comportamiento *seguir-pared* usando una metodología propia (COR) que es capaz de construir la base de reglas del controlador difuso con alta interpretabilidad. Una ventaja que aporta este método es la posibilidad de analizar el comportamiento evolucionado (analizando las reglas generadas) en lugar de tener que deducirlo a partir del funcionamiento del mismo (como es el caso cuando se usan redes neuronales). El comportamiento creado se componía de 77 reglas y sólo fue probado en simulación, pero permitía hacer navegar de forma segura al robot siguiendo paredes con esquinas tanto cóncavas como convexas.

Además de estos trabajos, los AE también han sido satisfactoriamente utilizados para robots futbolistas en la Robocup [131, 215], brazos robóticos [156], coordinación de comportamientos [70, 89, 214] e incluso en robots con patas [79, 181].

Visión por Ordenador

La visión por ordenador es el estudio y el uso de los métodos que permiten a las computadoras *entender* el contenido de los datos multidimensionales de una imagen. El término *entender* hace referencia a que la información específica se está extrayendo de los datos de la imagen para un propósito específico: para presentarla a un operador humano (e.g., si se

han detectado células cancerosas en una imagen de microscopía), o para controlar un cierto proceso (e.g., un robot industrial o un vehículo autónomo). Los datos de la imagen que alimentan a sistema visión por ordenador son a menudo imágenes a color o en niveles de gris, pero pueden también estar en la forma de dos o más imágenes (e.g., un par estéreo), una secuencia vídeo, o un volumen 3D (e.g., de un dispositivo del tomografía). En la mayoría de los usos prácticos de la visión de ordenador, las computadoras se programan para solucionar una tarea particular, pero los métodos basados en aprendizaje están llegando a ser cada vez más comunes.

La visión por ordenador se puede también describir como el complemento (pero no necesariamente el opuesto) de la visión biológica. En la visión biológica y la percepción visual, los sistemas visuales de seres humanos y de varios animales son estudiados, dando como resultado modelos de cómo estos sistemas son implementados en términos de procesamiento neuronal a varios niveles. La visión por ordenador, por otro lado, estudia y describe sistemas técnicos que son implementados en hardware o software, tanto en computadoras como el procesadores de señal.

A pesar de que existen trabajos tempranos, no es hasta finales de los años 70 que comienza un estudio más profundo de cómo los computadores pueden ser empleados para manejar grandes conjuntos de datos como imágenes. Sin embargo, esos estudios fueron originados por el avance en otros campos, por consiguiente no hay una formulación estándar de qué es el problema de la visión por ordenador ni de cómo se deben solucionar sus problemas. En lugar de ello, existe una abundancia de métodos para resolver varias tareas bien definidas, donde los métodos son normalmente poco generalizables. Muchos de esos métodos y aplicaciones continúan en estado de investigación básica, pero muchos están comenzando a aparecer como productos comerciales.

La visión por ordenador es normalmente estudiada como un campo dentro de la IA, donde la imagen es normalmente la entrada a un sistema en lugar del uso de entrada de texto, con el propósito de entender o controlar el comportamiento de un sistema. Algunos de los métodos de aprendizaje que se usan en visión por ordenador están basados en técnicas de aprendizaje desarrolladas dentro de la inteligencia artificial.

El análisis y procesado de imágenes son los principales campos de la visión por ordenador. La distinción entre ellos no es muy clara, es decir, el análisis de imágenes utiliza muchos de los métodos que tradicionalmente pertenecen al procesado de imágenes. Una distinción formal podría ser decir que el procesamiento de imágenes trata de transformar imágenes, produciendo una imagen a partir de otra, o proporcionar información de bajo nivel sobre una imagen (bordes, lineas, etc). Ninguna de las tareas comentadas requieren ni proporcionan una interpretación acerca del contenido de la imagen en términos de objetos o eventos. El análisis de imágenes, por otra parte, puede ser utilizado para controlar la acción sobre objetos en escenas. Y en sistemas más avanzados, esos modelos pueden ser aprendidos en lugar de programados.

Elementos de un Sistema de Visión por Ordenador

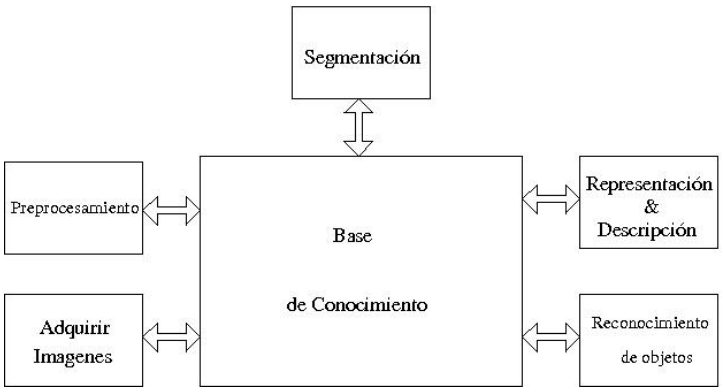


Figure 5: Elementos típicos de un sistema de visión por ordenador.

Un sistema de visión por ordenados típico puede ser dividido en los subsistemas mostrados en la Figura. 5. El elemento central, denominado *Base de Concimiento*, representa el conocimiento acerca del problema a resolver mediante el uso de visión. Este conocimiento puede ser tan simple como detallar las regiones de interés de la imagen, limitando así la zona donde realizar la búsqueda. Pero sin embargo, la base de conocimiento puede también ser algo complejo como una lista interrelacionada de los posibles defectos que se producen en un problema de inspección industrial de componentes, o una base de datos conteniendo imágenes de alta resolución por satélite de una región para detectar cultivos.

El módulo de *adquisición de imágenes* está encargado de capturar las imágenes que sirven de entrada al sistema. Existen multitud de dispositivos que pueden ser empleados como entrada al sistema de visión: cámaras, radares, sistemas de tomografía, etc. Sin embargo, las cámaras son el sensor empleado por excelencia para tareas de visión.

En la etapa de preprocesamiento, la imagen es tratada utilizando operaciones de bajo nivel. Este es un proceso cuya entrada y salida son imágenes. El objetivo de esta etapa es doble. En primer lugar, trata de realizar un *realce* de la imagen y una *restauración* si fuese necesario. La idea tras el realce es la de destacar detalles oscurecidos, o simplemente resaltar ciertas características de interés en una imagen. Un ejemplo típico de realce es el ajuste de contraste. La tarea de restauración también trata de mejorar la apariencia de una imagen. Sin embargo, el objetivo es recuperar cierta información o la eliminación de ruido. Operaciones típicas incluidas en esta etapa son: resampleo de la imagen, filtrado digital (convolución y correlación), descomposición en frecuencias, detección de bordes o estimación de la orientación local, estimar la disparidad en imágenes estéreo o análisis multiresolución.

La etapa de *segmentación* está presente en la mayoría de los sistemas de visión por ordenador. Consiste en particionar la imagen en sus partes u objetos constituyentes. Por ejemplo, en la tarea de inspección automática de elementos electrónicos, el interés recae en el análisis de imágenes de productos con el objetivo de determinar la presencia ausencia de anomalías especificas, como componentes perdidos o conexiones rotas. Los algoritmos de segmentación son normalmente muy generales y basados en una o dos propiedades básicas de valores de intensidad: discontinuidad y similaridad. En la primera categoría, el enfoque consiste en particionar la imagen basándose en cambios de intensidad, como los bordes de una imagen. En el segundo enfoque, las técnicas se basan en particionar la imagen en regiones similares de acuerdo con un conjunto de criterios predefinidos. La segmentación, crecimiento de regiones, partición y agrupación son ejemplos de métodos en esta categoría. En general, la segmentación autónoma es una de las tareas más difíciles en visión por ordenador. Una buena segmentación hará que la tarea a realizar se vea encaminada hacia buenas soluciones, mientras que en caso de una mala segmentación, el proceso abocará probable-

mente en fracaso.

La fase de *representación y descripción* es un paso que normalmente se produce tras la segmentación. La entrada es normalmente un conjunto de pixels en crudo, representando bien el contorno de una región o todos los puntos de la región en sí. Hay básicamente dos enfoques en esta fase: (i) representar una región en base a sus características externas (su contorno), o (ii) representarla en función de sus características internas (su interior). La siguiente tarea consiste en describir la región basándose en el medio de representación seleccionado. Una representación de las características externas suele ser tomada cuando el interés princial está en conocer la forma. Cuando lo que se desea es conocer propiedades de la región tales como el color o la textura, se eligen representaciónes internas. Por supuesto, en algunas aplicaciones es presiso el uso de ambas. En cualquier caso, el descriptor empleado debe ser tan invariante como fuera posible a variaciones en tamaño, traslación y rotación. Ejemplos de descriptores externos son los códigos de cadenas, aproximaciones poligonales de formas, segmentos del contorno, esqueletos, etc. Ejemplos de descriptores internos incluyen descriptores topológicos, medidas estadísticas, momentos y componentes principales.

El *reconocimiento* puede verse, en un sentido amplio, como el proceso de asignar una etiqueta (e.g. vehiculo) a un objeto (tambíen llamado patrón) basandose en sus descriptores. Existen multitud de técnicas que pueden ser empleados para la clasificación de patrones de varias clases, todos ellos cubiertos bajo el término *aprendizaje por ordenador* [150]. Ejemplos de técnicas de aprendizaje por ordenador son: redes neuronales, máquina de vectores soporte, algoritmos evolutivos, métodos estadísticas, métodos de agrupamiento, arboles de decisión, etc.

Para dotar al lector de una visión más completa acerca de los elementos típicamente involucrados en los sistemas de visión por ordenador, se recomienda la lectura de [90].

Visión Estéreo

La *stereopsis* es el proceso de percepción visual que lleva a la detección de la profundidad o distancia de los objetos. La palabra viene de la raíz latina, *stereo* que significa solidez, y *opsis* significando visión. Esto significa que se refiere a cualquier tipo de percepción visual

de la profundidad, pero no es hasta los años 60 que el término a quedado restringuido a la visión binocular. Antes, ese término esra denominado "stereopsis binocular".

La profundidad a partir de la stereopsis surge de las ligeramente distintas posiciones que cada ojo ocupa en la cabeza. La stereopsis fue descubierta por Charles Wheatstone en 1833. El reconoció que dado que cada ojo observa las escenas del mundo desde lugares ligeramente diferentes, las imágenes capturadas por cada ojo son ligeramente diferentes. Por tanto, objetos colocados a distintas distancias de los ojos se proyectan en lugares que difieren en el eje horizontal, dando una pista acerca la profundidad del objeto. Wheatstone provó que esa distancia horizontal entre objetos, también llamada *disparidad*, provocaba la percepción de profundidad utilizando de pinturas planas mostrando objetos ligeramente desplazados. Para que estos dibujos fueran mostrados de forma independiente en los dos ojos, Wheatstone inventó el estereoscopio.

Base de la computación estereoscópica

En esta sección, explicamos brevemente las bases de la computación estéreo. Para conocer mejor el estado del arte en esta materia, el lector interesado es referido a [36].

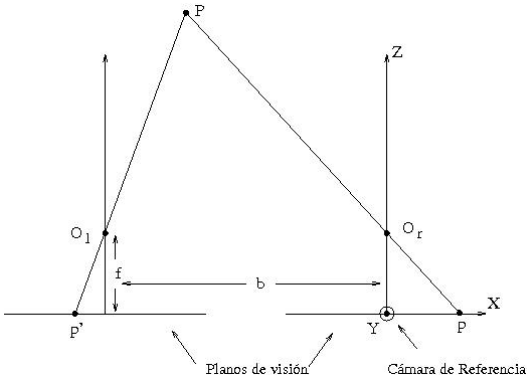


Figure 6: Sistema estéreo mínimo compuesto por dos cámaras

El sistema estéreo mínimo está compuesto por un par de cámaras (llamemoslas derecha e izquierda) cuyos centros ópticos (O_l y O_r) están separados por una distancia b. Asumamos, para facilitar la explicación, que ambas cámaras tienen propiedades ópticas idénticas y sus planos de visión son coplanares y colineales (ver Figura 6). Asumiremos que el centro del

sistema de referencia empleado está en la cámara derecha (cámara que denominaremos de *referencia*).

Un punto $P = (X, Y, Z)$ en el espacio se proyecta en dos localizaciones ($p = (x, y)$ y $p' = (x', y')$) en la misma linea horizontal en ambas cámaras. El desplazamiento en la proyección de una imagen respecto de la otra es lo que se denomina disparidad y el conjunto de todas las disparidades entre dos imágenes es lo que se conviene en llamar *mapa de disparidad*. Las disparidades solo pueden ser calculadas en aquellos puntos que son vistos en ambas imágenes y aun así es difícil de calcular en regiones que no tienen suficiente textura. Aquellos puntos cuya disparidad no puede ser calculada se denominan *sin correspondencia* y al problema de calcular el mapa de disparidad se le denomina *el problema de la correspondencia*.

Conocidos los parámetros intrínsecos de un sistema estéreo es posible realizar una reconstrucción tridimensional de la escena capturada a partir del mapa de disparidad (*problema de reconstrucción*). La posición de un punto puede ser calculada mediante triangulación como:

$$Z = \frac{fb}{d}\ ;\ X = \frac{xZ}{f};\ Y = -\frac{yZ}{f}. \tag{1}$$

donde f representa la distancia focal y d la disparidad calculada como $d = p - p'$.

La Figura 7 muestra una escena capturada con un sistema estereo formado por dos cámaras separadas unos $b = 12$ cm. Las Figuras 7(a) y 7(b) muestra las imágenes capturadas por ambas cámaras, mientras que la Figura 7(c) muestra el mapa de disparidad calculado. En la Figura 7(c), los pixels más claros muestran valores de alta disparidad mientras que los oscuros representan valores de baja disparidad. Aquellos puntos sin correspondencia son mostrados en negro. Finalmente, la Figura 7(d) muestra la reconstrucción tridimensional de la escena.

Visión y Robótica

La visión es probablemente el sensor más importante tanto para los humanos como para los robots. La visión es capaz de proporcionar información muy útil que puede ser de gran

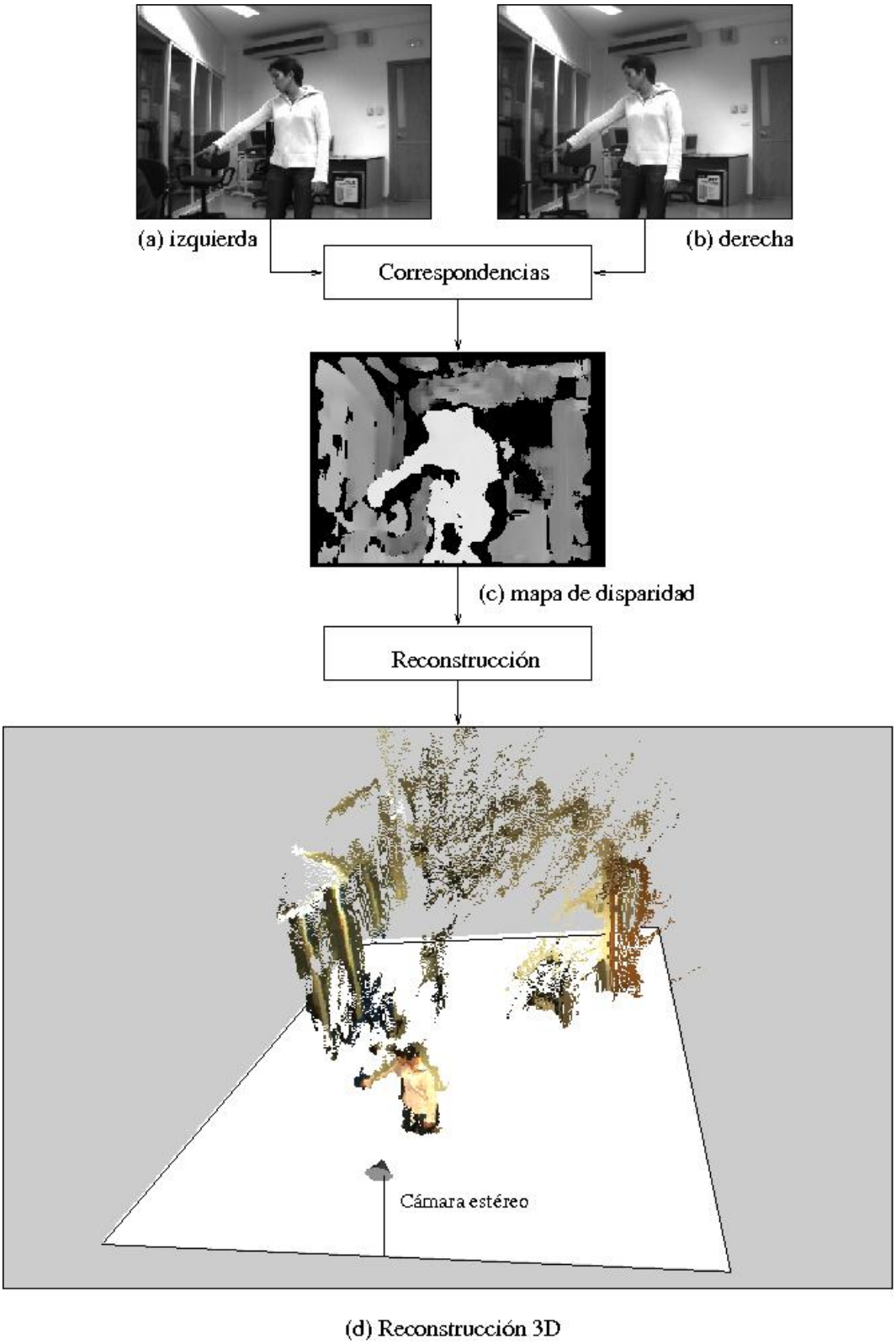


Figure 7: (a) Imagen de la cámara izquierda (b) Imagen de la cámara derecha (c) Mapa de disparidad (d) Reconstrucción tridimensional de la escena.

ayuda para los robots móviles y por ellos ha sido investigado su uso para tareas de navegación. Las aplicaciones de navegación que emplean visión pueden ser clasificadas en dos categorías: navegación guiada por visión en exterior o interior. Aunque esta memoria está más relacionada con el segundo tópico, el lector interesado puede obtener un interesante estudio del estado del arte en [58]. A continuación se proporciona una pequeña introducción a las principales contribuciones de la navegación guiada por visión en entornos de interior.

La navegación guiada por visión hace referencia al uso de visión con el propósito de ayudar en la tarea de navegación aun robot móvil en entornos de interior. En un sentido amplio, la navegación en interior basada en visión puede dividirse en las tres categorías explicadas a continuación:

- La *vision aplicada a la navegación basa en mapas* consiste en proporcionar al robot con un mapa del entorno. El mapa puede estar definido con diversos grados de detalle (modelos CAD, mapas de ocupación, mapas topológicos). La idea central de la navegación basada en mapas en proporcionar al robot con un conjunto de marcas que puede esperar encontrar durante la navegación. La tarea pues del sistema de visión consiste en buscar e identificar las diferentes marcas encontradas en las imágenes capturadas. Una vez que son identificadas, el robot puede emplear su mapa para determinar su posición. Tres grandes ramas pueden distinguirse entonces en este tipo de navegación utilizando mapas: *localización global*, *localización incremental* y localización derivada del *seguimiento visual de marcas*.

  - La *localización global* consiste en determinar la posición absoluta del robot en el entorno. Sugihara [201] realizó uno de los estudios pioneros en la autolocalización utilizando una cámara. Más tarde, Krotkov [122] extendió su trabajo proponiendo un modelo capaz de integrar la incertidumbre en la observación.
  - La *localización incremental* asume que se conoce de forma aproximada la posición del robot al inicio de la sesión, y que el objetivo del sistema de visión consiste en refinar las coordenadas. Ejemplos de este tipo de sistemas pueden verse en el trabajo de Matthies y Shafer [148], donde un sistema estéreo de visión es

empleado para la reducción del error; el sistema FUZZY-NAV de Pan et al. [174] extiende su trabajo previo NEURO-NAV mediante la incorporación de un controlados de alto nivel basado en reglas para el control de los comportamientos de un robot; el sistema de Horn y Schmidt [102] describe un sistema de localización para el robot móvil MACROBE–Mobile and Autonomous Computer-Controlled Robot Experiment–utilizando una cámara de láser 3D.

- La localización derivada del *seguimiento visual de marcas* puede ser realizada para aproximar la localización del robot si la identidad de las marcas vistas con la cámara es conocida y se puede hacer un seguimiento de ellas. Las marcas a emplear puede ser tanto artificiales como naturales. En el trabajo de Kabuka y Arenas [113] las marcas empleadas son círculos con un único código de barras, mientras que en el trabajo de Tsumura [210] se emplearon bandas reflectantes colocadas a lo largo de la ruta del robot.

- *Visión aplicada a la creación de mapas.* Los mapas del entorno no siempre son fáciles de generar, especialmente si es necesario incluirles información métrica. Por tanto, muchos investigadores han propuesto robot autónomos o semiautónomos capaces de explorar su entorno para construir una representación interna de él. El primer intento de construcción autónoma de mapas fue realizado el Stanford Cart de Moravec [153]. El Stanford Cart utilizaba una única cámara para tomar imágenes espaciadas 50 cm. Después, un operador era empleado para extraer características distinguidas de las imágenes. Después, esas características eran correlacionadas para generar sus coordenadas $3D$. El *mundo* era representado mediante las coordenadas $3D$ de las características que eran marcadas en un mapa de celdas de (de dos metros cuadrados cada una). Aunque el grid representaba de forma indirecta la posición del los obstáculos y era útil para la planificación de trayectorias, no proporcionaba un modelo detallado y por tanto demasiado útil del entorno. Por esta razón, Moravec y Elfes presentaron más tarde una nueva estructura de datos que llamaron *mapas de ocupación* [154] y emplearon sonar para adquirir información sobre la estructura del entorno. En los robots

modernos, los mapas de ocupación permiten que se incluyan medidas provenientes de diversos sensores (incluyendo estéreo) y así manejar la incertidumbre [158, 185].

La tarea de creación de mapas de ocupación puede ser contrastada con trabajos donde robots móviles generan modelos para representar el entorno como mapas topológicos [44, 164]. Este tipo de representación tienen normalmente información métrica incluida para el reconocimiento de los nodos y facilitar así la tarea de navegación. Uno de los principales problemas de los enfoques basados en mapas topológicos es reconocer el nodo donde se encuentra el robot. En [205], Thurn propone la integración de mapas de ocupación y mapas topológicos para aunar las ventajas de ambos enfoques. Otra contribución notable relacionada con la creación de mapas es la de Ayacha y Faugeras [15] que emplean un sistema trinocular de visión y el filtro de Kalman, y el trabajo de Zhang and Faugeras [223] que realiza una reconstrucción $3D$ a partir de la secuencia de imágenes tomadas por la cámara del robot.

- La *visión aplicada a la navegación sin mapa* incluye a todos aquellos sistemas en que la navegación se realiza sin una descripción previa del entorno. Nos referimos a robots que nunca usan mapas. En su lugar, los movimientos del robot son determinados mediante la observación y extracción de información relevante acerca de los elementos del entorno. Esos elementos pueden ser paredes, objetos como sillas, puertas o escritorios, etc. Entre los trabajos destacables que ha echo uso de este enfoque, los más destacables son los basados en flujo óptico [190] y las técnicas basadas en la apariencia [85].

Detección y seguimiento visual de personas utilizando visión estéreo

El tópico interacción humano-robot (IHR) ha atraído un gran atención en la última década. El objetivo es poder crear interfaces más inteligentes capaces de extraer información sobre el contexto o las acciones a realizar mediante una interacción natural con el usuario, por ejemplo por medio del uso de gestos o la voz.

Un aspecto fundamental en este sentido es la detección y seguimiento de personas, ex-

istiendo una gran cantidad de literatura al respecto [86, 134, 197, 218]. Las técnicas para realizar la localización suelen basarse en la integración de múltiples fuentes de información: color de la piel, detectores de caras, análisis del movimiento, etc.

Si bien la detección y seguimiento de personas con una única cámara es un tema ampliamente tratado, el uso de la tecnología estéreo para este propósito concentra ahora un importante interés. La aparición de hardware comercial para el cálculo estereoscópico, así como el abaratamiento de este tipo de sistemas, los convierte en una atractivo sensor con el que desarrollar sistemas inteligentes. El uso de visión estéreo dota de un grado más de información que es de gran utilidad en la tarea de detección e interacción hombre-máquina. Por un lado, la información de disparidad es más invariable a cambios de iluminación que la de una sola cámara, siendo un elemento muy ventajoso para la estimación del entorno (*background*) [53, 67, 97]. Además, la posibilidad de conocer la distancia al sujeto puede ser de gran ayuda para su seguimiento así como para un mejor análisis de sus gestos.

Entre los trabajos más destacados sobre detección y seguimiento de personas usando visión estéreo encontramos el trabajo de Darrel et al [54]. Este trabajo presenta un sistema de display interactivo capaz de detectar y seguir múltiples personas. La detección de las personas se basa en integrar la información de un detector de piel, un detector de caras y el mapa de disparidad de la escena. Por un lado se detectan objetos (*blobs*) independientes en la imagen de disparidad que serán candidatos a personas. Por otro lado, se analiza el color de la imagen para detectar aquellas zonas que pudieran pertenecer a piel y se aplica un detector de caras. Estas tres informaciones son fusionadas para detectar a las personas que se encuentran visibles. Para realizar el seguimiento se utiliza información sobre el pelo, la ropa y la historia pasada de las personas localizadas. De esta manera los sujetos pueden ser reconocidos aunque desaparezcan de la imagen por un tiempo. Sin embargo, dos aspectos pueden ser criticados de su trabajo. En primer lugar, el sistema requiere que las caras de los usuarios sean visibles para poder hacerles un seguimiento apropiado. En segundo lugar, el sistema confía en el uso de un modelo predefinido del color de piel, y por tanto, se puede esperar una degradación del algoritmo de seguimiento conforme las condiciones de iluminación difieran de las de entrenamiento [144]. El uso de modelos dinámicos de color podría

haber solucionado el problema [196].

Grest y Koch [91] muestran un sistema para la detección y seguimiento de personas para la interacción en entornos virtuales. El sistema permite a un usuario navegar por un entorno virtual simplemente andando por una habitación colocándose unas gafas de realidad virtual. En este trabajo la detección de la cara es un punto crucial que se resuelve utilizando el detector de caras propuesto por Viola y Jones [212]. Una vez localizado el sujeto se emplea un histograma del color de la cara y del pecho del sujeto utilizando un filtro de partículas para estimar la posición de la cabeza. La información estéreo es utilizada para conocer la posición del sujeto en la habitación y poder así determinar su posición en un entorno virtual. En este trabajo, el procesamiento estéreo se realiza utilizando la información de varias cámaras situadas en distintos puntos del entorno. Como en el trabajo de Darrel et. al, la principal limitación del sistema es que requiere que la cara de la persona a seguir sea visible para poder realizar el seguimiento.

En [96] se presenta un método para la localización y seguimiento de personas en imágenes estéreo utilizando mapas de ocupación. Antes de realizar la detección de personas se crea una imagen del entorno mediante un sofisticado método de análisis de imágenes. Una vez que se tiene la imagen del entorno se separan fácilmente los objetos que no pertenecen a él y se crea tanto un mapa de ocupación como un mapa de altura de la zona visible. La información de ambos mapas es fusionada para detectar personas utilizando heurísticas simples. El seguimiento de las mismas se realiza utilizando el filtro de Kalman combinado con plantillas deformables. En este trabajo se utiliza sistema estereoscópico que es situado a 3 metros del suelo y en una posición fija. La principal crítica a este trabajo está en el uso de simples heurísticas para determinar si una objeto es persona que puede llevar a clasificar incorrectamente a objetos de tamaño similar a seres humanos como una caja grande o un abrigo colgado en un perchero (tal y como indica el autor).

Hayashi et al [98] presentan un sistema de detección y seguimiento de personas especialmente indicado en tareas de vídeo vigilancia. En su trabajo, el entorno es modelado utilizado un mapa de ocupación donde los puntos capturados por el sistema estéreo son proyectados como *voxels variables* para tratar con los errores del estéreo. La detección de las personas

se realiza sobre el mapa de ocupación utilizando heurísticas simples como asumir que una persona se corresponde con un máximo de ocupación en el mapa y que tiene una altura determinada. Como en el caso anterior es de esperar falsos positivos debido a la simplicidad del método de detección empleado.

Tanawongsuwan presenta en [202] los pasos iniciales para diseñar un agente robótico capaz de seguir a una persona y reconocer sus gestos. El sistema emplea técnicas básicas para encontrar los brazos y cabeza de la personas combinando información de un filtro de color de piel, un detector de movimiento y profundidad. Una vez la persona es localizada, utilizan Modelos Ocultos de Markov para reconocer los gestos realizados por el usuario de entre un conjunto de gestos aprendidos. A pesar de todo, el trabajo no trata en profundidad el problema de la detección de personas y asume que hay una persona al detectar tres regiones de color piel (correspondientes a cabeza y manos).

Abstract

The work developed in this thesis is framed into the Autonomous Robots field. The aim of this field consists in the development of robots able to move purposefully and without human intervention in unmodified environments. In that sense, this thesis pursues a double objective. Firstly, exploring the use of new mechanisms able to increase the autonomy level of mobile robots. Secondly, developing of a basis set of primitive perceptual-motor skills that provide robots with basic capabilities for interacting with human users naturally.

Regarding the first objective, a key aspect in order to provide robots with the ability to move freely and safely, is the development of robust navigation mechanisms. Mobile robots must be able to determine the set of appropriate actions to go from their current location to a desired one, without putting at risk their integrity nor the integrity of the people in their surroundings. Therefore, mobile robots use their sensors in order to build an environment model that help them to decide the best action to take. Ultrasound sensors are the devices most frequently used for navigation purposes because of their low price and computational cost. Nevertheless, ultrasound sensors are strongly affected by several sources of errors. Thus, navigation systems that rely on them uniquely suffer from their weakness. Part of this thesis explores the combined use of ultrasound sensors and computer vision in order to solve the weakness of one sensorial subsystem with the strengths of the other. A navigation system able to fuse both visual and range information in order to do the navigation task in indoor environments is developed in this thesis. The uncertainty and vagueness present in the sensorial systems will be dealt by the use of fuzzy logic based techniques. The proposed system has a highly modular design based on the *multi-agent system* philosophy that allows creating very flexible systems able to be physically distributed among different machines. The navi-

gation system developed creates a plan, employing a topological map provided to the robot, consisting of the sequence of doors to transverse and corridors to follow in order to reach a desired room into an indoor environment. The navigation system requires the detection of special landmarks that are placed besides doors in order to indicate their position. The system fuses visual information about the position of the doors (by detecting the landmarks) with range information about the surrounding obstacles (provided by ultrasound sensors) in order to perform the navigation task in indoor environments.

Artificial landmarks have been widely employed for navigation and localisation purposes because of they are relatively easy to detect and distinguish from other elements of the environment. However, they are not an appropriate solution when it is not possible to alter the environment (something that happens frequently). Therefore, we have also designed a visual door-detector able to detect doors themselves (avoiding the use of artificial landmarks). It operates extracting the most relevant image segments and analysing them using a hierarchical classifier based on fuzzy logic. The success of the door-detector depends on two parameters: the true positive fraction (i.e., the success of the system in detecting doors in images that show doors) and the true negative fraction (i.e., the success of the system in detecting the absence of doors in images that do not show doors). The door-detector designed performs well in detecting the doors of our environment using our robot. However, its performance in other environments (with doors of different sizes) or using a different hardware system (a robot with different height or a camera with different focal lenght) might differs. The performance of the door-detector in a specific environment using a specific robot and camera can be maximised by a training procedure (tuning) aimed to learn the characteristics inherent to the operating conditions.

Tuning of fuzzy systems provides an automatic way to increase their performance using a validated data set describing the problem. Automatic tuning of fuzzy systems is a problem that has been dealt in the related literature using several optimisation approaches. When the performance of the system to be tuned is evaluated by a multiobjective function (like ours) the use of evolutionary approaches has shown to be the most appropriate tool. Part of the work of this thesis is devoted to the development and comparison of several evolutionary

multiobjective techniques and their use for tuning our fuzzy door-detector. The evolutionary tuning mechanism designed in this thesis allows our door-detector to be adapted to a wide variety of indoor environments. Thus, the integration of the door-detector as part of the navigation architecture developed provides our robot with the capability of navigating autonomously in fully unmodified environments, since no landmarks are required.

As second goal of this thesis, we are interested in the development of a basis set of perceptual-motor skills that provide robots with basic interaction capabilities. Our aim is that these skills constitute a basic common foundation useful for future robotic applications that require to interact with people. If we desire that humans and robots work together in a future society, it is necessary that humans have the ability to command robots using flexible communication mechanisms. The use of classical communication devices such as mouses, keyboards or screens is insufficient for a massive adaptation of robots in every-day life. Instead, more natural communication mechanisms would be required, e.g., voice, facial gestures, postural gestures, etc. In that sense, vision plays a very important role that allows humans to detect the presence of other humans and the detection of many visual cues involved in the communication. Therefore, providing robots with the ability to detect and track people is an initial step required to enable robots interact naturally with people. People detection and tracking are problems that have been dealt in the related literature. However, the use of stereo vision for that purpose is a very interesting topic that concentrates an important research attention nowadays. Stereo vision brings several advantages when compared to monocular cameras. Firstly, disparity information is more invariable to illumination changes that the colour information provided by a monocular camera. Therefore, systems that employ them can be more robust in real situations. Secondly, it is possible to know the position of the user and thus achieve a better tracking even when partial or total occlusions take place.

In this thesis we develop a people detector and tracker system based on stereo information. The system is able to detect and track simultaneously several people moving in the environment fusing depth information and information about the colour of their clothes. The people detector and tracker system developed is then integrated into a greater multi-agent architecture. That architecture implements a set of perceptual-motor skills that constitutes a

basic common foundation for many mobile robotic applications that require to interact with people. The skills implemented in the system combine visual and range information (using stereo vision and ultrasound sensors) allowing a mobile robot to: (i) detect an user that desires to interact with the robot, (ii) track an user in the environment despite the presence of other people in the surroundings, and (iii) move following an user avoiding collision with the user and obstacles. Altough these skills are useful for many robotic applications that desire to achieve a natural human-robot interaction, it might be necessary the addition of new skills according to the particularities of the desired application. In that sense, the multi-agent philosophy employed constitutes an advantage for an easy adaptation of the architecture to different applications.

We might conclude this summary indicating that the work developed fulfils the goals initially set and points towards interesting future research activities. Particularly, we would like to focus the attention in the possible design and implementation of future robotic systems with full capabilities to interact with people using some of the techniques developed in this work.

Chapter 1

Introduction

"The goal of autonomous mobile robotics is to build physical systems that can move purposefully and without human intervention in unmodified environments".

Alessandro Saffiotti [186].

1.1 Overview and Motivation

Mobile robotics is an ideal field for the development of Artificial Intelligence (AI) techniques. Researchers in robotics see in a robot a double challenge. Firstly, the opportunity to provide a machine with the capability of emulating the human behaviour, perception and reasoning. Secondly, the chance to provide a hardware system with the ability of moving autonomously in a real environment. In summary, it can be said that robotics is one of the places where AI takes contact with the real world to validate its proposals.

A mobile robot, analysed under a very simplistic way, is a machine that must perceive its environment and decide autonomously the actions to perform based on its goals and perceptions. In that sense, there are two crucial aspects to take into account: (i) how to make robots perceive the environment using sensors subject to uncertainty; and (ii) how to decide the appropriate actions to perform among a set of infinite possibilities.

A great research effort has been done since the first works with Skakey [168] at the Stanford Reseach Institute in the mid 60s in order to increase the autonomy level of mobile

robots. The advances obtained in that field have contributed to the first attempts for using mobile robots in a great variety of real-life applications such as: personal robots [20, 68, 130], robotic pets [77], tour guides robots [37, 194], and robotic assistants for disabled or elderly people [101, 178] among others. Little by little, robots start to make an space not only in the market but also in our homes and classes. A prove of it can be seen in the growing number of robots that are appearing for commercial, research and educational purposes (see Fig. 1.1).

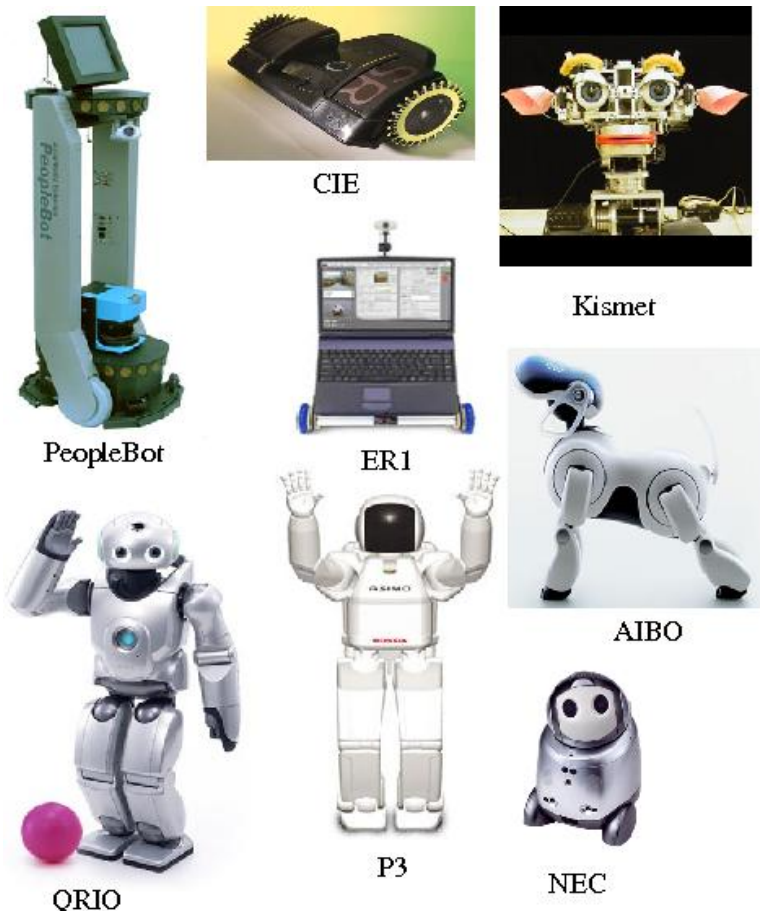


Figure 1.1: Examples of commercial, educational and research robots.

Despite the great advances achieved in the past fifty years, there is still much work to do in order to make robots be present in everyday life. This thesis is our contribution to the research community with the aim of making robots more accessible tools in our living environments.

The goal of our work is twofold. The first goal is the development of techniques that help to increase the autonomy level of mobile robots. Autonomous mobile robots must be able to work autonomously, i.e., without human intervention. This thesis explores the use of soft-computing and computer vision techniques applied to the navigation and perception problems in order to enable robots move autonomously and purposefully. The second goal of this thesis is concerned with the development of perceptual-motor skills to provide robots with basic interaction capabilities. Altough robots must be able to operate autonomously, they are designed to serve humans. Therefore, robots must be able to detect humans and interact with them, instead of considering them as mere obstacles in the environment. Moreover, this communication should be natural. The use of devices like screens, keyboards or mouses for that purpose might be a deterrent for introducing robots in everyday life. Instead, it is preferable the development of communication mechanisms more similar to these employed by humans, like speech, hand gestures, facial expressions, postural gestures, etc. In this thesis, a set of perceptual-motor skills, that constitute a basic common foundation useful for future robotic applications that require to interact naturally with people, is developed.

Regarding the first goal, increasing the autonomy level of robots is a difficult task that involves working on a great variety of research areas. In order to make robots perform real-life tasks autonomously, robot designers must develop very complex systems able to continuously manage sensors and actuators. The different subsystems that inhabit in a robot (artificial vision systems, trajectory planners, hardware controllers, etc) must efficiently share information using appropriate communication mechanisms, be appropriately integrated, coordinated and run in real-time. Meanwhile, the system architecture employed must be flexible enough to adapt to further advances that might be integrated in the future. Therefore, the design and implementation of robots control systems is a work that become harder as the desired tasks to perform become more difficult. Classical control architectures for robots can be classified in: deliberative [25, 168], reactive [14, 34, 146] and hybrid architectures [12, 84]. Despite of these approaches have shown a great ability to overcome some of the requirements mentioned, they still have some drawbacks that forces further development in that area. Flexibility and scalability are two important characteristics that robotic architec-

tures must exhibit, i.e., the facility to include new functionality that might be required in the future and the ability of work properly as the functionality increases. Part of this thesis is devoted to the design of flexible control architectures able to cope with these requirements. In that sense, we rely on the concept of *agent* as the structural element for implementing control architectures. Agents are independent pieces of software that work to accomplish a goal implicit in their design and that are able to communicate with other agents. Agents are the natural extension of the Object Oriented philosophy for distributed systems. Multi-agent architectures are architectures composed of several agents that implement the functionality of the whole system in a distributed way (each agent implement a part of the whole). The main advantages of multi-agent architectures are three. Firstly, the ability to physically distribute the agents in several machines if required. Secondly, they provide enough flexibility to increase and reduce the functionality of the system thus allowing a great adaptability to design changes. Thirdly, multi-agent architectures are very robust because of an agent failure does not necessarily makes the whole system to fail. On the contrary, the remaining agents might take control over the robot in case of partial failure.

A mobile robot must be able to perceive the environment in order to decide the appropriate actions that can lead it towards a desired place. However, sensors are normally subjet to uncertainty making the navigation task a difficult problem because of robots can not rely on their perceptions completely. The most representative case is the use of ultrasound sensors for navigation. Ultrasound sensors have been widely employed as the main navigation sensor because of they provide a very useful information at a very low cost. However, the main deficiency of ultrasound sensors is that the information they provide is submitted to a high uncertainty. An important part of the research on that area focuses on modelling and managing sensors inaccuracy and uncertainty in order to develop robust navigation mechanisms. Fuzzy behaviours are a good solution to the problems mentioned [186]. However, there is an obvious limitation in the performance of ultrasound-based navigation systems that makes them highly biased. In fact, the use of several sources of information seems to be the best choice to dealt with this problem, i.e., the weakness of a sensorial subsystem might be overcome with the strengths of another. Vision is a cheap sensor commonly available in

today's robots that can bring a richful information for autonomous navigation. Part of the work presented in this thesis deals with the combined use of ultrasound sensors and visual information to develop a navigation system based on *visual-landmark-tracking* [58].

Navigation based on *visual-landmark-tracking* is a technique that consists in the visual detection and tracking of landmarks to help the navigation task. Landmarks are representative elements of the environment that can be used for navigation. They can be divided into *artificial* landmarks and *natural* landmarks. The former are human-made elements placed in the environment with the aim of helping the navigation process. The later are representative places of the environment that can be used for navigation (like walls, doors or paintings), although it is not their main purpose. In this thesis, a multi-agent system for controlling the navigation task of a mobile robot in office-like environments is presented. The navigation plan is created using a topological map of the environment and consists of the sequence of doors to transverse and corridors to follow in order to reach a desired position. Fuzzy behaviours are employed to manage the vagueness and uncertainty of the ultrasonic sensor information allowing to navigate safely in the environment. Additionally, visual behaviours locate a required door to cross and fixate it indicating the appropriate direction to reach it. Artificial landmarks have been placed besides the doors of our environment in order to ease the door-detection task. The artificial landmarks designed have been learnt using soft computing techniques and the vision system is able to detect and track them in real-time.

The use of artificial landmarks is an appropriate solution when it is possible to alter the environment. However, this option is not always feasible because of aesthetic or practical reasons. In that case, the ability of a robot to detect doors still being an advantage for navigating and/or map-building purposes. Therefore, we have also designed a door-detector able to detect doors themselves. Doors are found in grey-level images by detecting the borders of their architraves. The proposed door-detector consists in the extraction of the main image segments and their analysis by a hierarchical classifier based on fuzzy logic. It is important to indicate that the success of the door-detector depends on two parameters: the true positive fraction (i.e., the success of the system in detecting doors in images that show doors) and the true negative fraction (i.e., the success of the system in detecting the absence

of doors in images that do not show doors).

The proposed door-detector performs well in detecting the doors of our environment. However, its use on a different environment (with doors of different sizes) or on a different robot (with different camera properties) would require a training phase aimed to adapt the system to the particular operating conditions. Manual tuning of fuzzy systems is a tedious task so that several approaches aimed to automatise that process from a validated data set have been proposed [155]. The fuzzy door-detector designed in this thesis is a multiobjective system because of its performace depends on two conflicting criteria. In these type of problems, the use of evolutionary approaches has shown to be the most appropriate tool. Part of the work of this thesis is devoted to the development and comparison of several multiobjective techniques and their use for tuning our fuzzy door-detector with the objective of making it useful for a wide variety of indoor environments. Finally, the integration of the door-detector designed as part of our navigation architecture provides our robot with the capability of navigating autonomously in fully unmodified environments (since no landmarks are required for the navigation task).

Up to this moment, we have explained the work developed in this thesis aimed to improve the autonomy level of mobile robots. As we have already indicated, the goal of this thesis is twofold. The second goal is the development of a set of basic perceptual-motor skills that enable robots to interact with people naturally. In order to make robots be present in everyday life, it is necessary to achieve a natural and intuitive human-robot interaction (HRI) [31, 32, 33, 38, 73, 137, 192]. The main goal of HRI is to make both robots and humans to communicate with little prior knowledge about each other. In that sense, the use of classical communication devices such as mouse, keyboard or touch screens might be insufficient for a massive introduction of robots in every-day life. Instead, more natural ways of communication (such as voice, facial or postural gestures) are interesting in order to make people comfortable with their use. Examined from the robot's perspective, a crucial aspect to achieve a proper HRI is the awareness of the human presence, i.e., robots must be able to detect and track users in their surroundings instead of considering them as mere obstacles. Part of our work is devoted to the development of basic perceptual-motor skills that provide

robots with a set of primitive interaction capabilities, valid for a wide range of applications that require the natural communication with human users. The skills designed in this work combines stereo vision and ultrasound information in order to allow robots: (i) detect human users; (ii) keep continuous track of users without confusing them with other people in the surroundings, and (iii) follow the users around the environment if necessary. These are skills that would be necessary to create a basic common foundation for many real-life mobile robotic applications, altough there would be many other specific needs depending on the particular application for which the robot is required.

Altough the people detection and tracking topics have been well explored using monocular vision, the problem becomes more complex when the camera is placed on a mobile device such as the robot itself. For that particular problem, stereo vision seems to be a very interesting sensor to employ. The decreasing cost of stereo cameras makes them an appealing sensor to develop sophisticated vision systems that can bring several advantages over monocular vision. Firstly, disparity information is relatively invariable to illumination changes. Therefore, systems that employ stereo vision can be very robust against illumination changes that take place in real scenarios. Secondly, it is possible to know the three-dimensional position of the user and thus achieve a better tracking even when partial or total occlusions occur.

In this work, a visual system able to detect and track people by combining stereo and a dynamic colour model of the people's clothes is presented. It is specifically designed to detect and track users when the stereo camera is placed at under-head positions as required for mobile robots that desire to achieve a comfortable HRI [73]. Then, this people detector and tracker system is employed as part of a multi-agent architecture that provides the three perceptual-motor skills previously indicated. Altough these skills are useful for many robotic applications that desire to achieve a natural human-robot interaction, it might be necessary the addition of new skills according to the particularities of the desired application. In that sense, the multi-agent philosophy employed constitutes an advantage for an easy adaptation of the architecture to different applications.

We might conclude indicating that the work developed fulfils the goals initially set and points towards interesting future research activities. Particularly, we would like to focus

the attention in the possible design and implementation of future robotic systems with full capabilities to interact with people using the techniques developed in this work.

1.2 Structure of this Thesis

Following, we explain the organisation of this thesis by providing a brief overview of the contents of its chapters:

- Chapter 2 provides an overview of the state-of-the-art in the main areas related to this thesis.

- Chapter 3 constitutes the first contribution of this thesis to increase the autonomy level of mobile robots. In that chapter, a multi-agent system (based on fuzzy and visual behaviours) for controlling the navigation task of a mobile robot in office-like environments is presented. The set of agents is structured in a three-layer hybrid architecture. A high level of abstraction plan is created using a topological map of the environment in the *Deliberative* layer. The plan is composed by the sequence of rooms and corridors to transverse and doors to cross in order to reach a desired room. The *Execution and Monitoring* layer translates the plan into a sequence of available skills in order to achieve the desired goal, and monitors the execution of the plan. In the *Control* layer there is a set of agents that implements fuzzy and visual behaviours that run concurrently to guide the robot. Fuzzy behaviours manage the vagueness and uncertainty of the range sensor information allowing to navigate safely in the environment. Visual behaviours locate a required door to cross (by using the artificial landmarks placed besides the doors) and fixate it indicating the appropriate direction to reach it. The system has been validated in numerous experiments in a real office-like environment.

- The navigation system developed in the previous chapter has a main drawback: it can not be used on completely unmodifiable environments, i.e., artificial landmarks must the placed. Chapter 4 presents our approach to visual door-detection in indoor environments, avoiding the use of artificial landmarks, by the use of computer vision

and fuzzy logic. The chapter presents a visual fuzzy system based on a hierarchical structure of three different fuzzy classifiers, whose combined action allows the robot to detect the presence of doors in the images captured by its camera. Doors are found in grey-level images by detecting the borders of their architraves. A variation of the Hough Transform is used in order to extract the main image segments after applying an edge detector. Features like length, direction, or distance between segments are used by a fuzzy system to analyse whether the relationship between them reveals the existence of doors. The system has been designed to detect rectangular doors typical of many indoor environments by the use of expert knowledge. A large database of images containing doors of our building, seen from different angles and distances, has been created in order to test the performance of the system. The system has shown the ability to detect doors under heavy perspective deformations and it is fast enough to be used for real-time applications in a mobile robot.

- In Chapter 5, we tackle the problem of tuning the fuzzy membership functions of the previous visual fuzzy system. Although the global knowledge represented in the knowledge base makes the system perform properly in the door-detection task, its adaptation to the specific conditions of the environment where the robot is operating can significantly improve the classification performance. However, tuning this system is a complex task since two different performance indices are involved in the optimisation process (true positive and false positive detections), thus becoming a multiobjective problem. Hence, in order to automatically put the fuzzy system tuning into effect, different single and multiobjective evolutionary algorithms are considered to optimise the two criteria, and their behaviour in the problem solving is compared. The evolutionary tuning mechanism developed allows our door-detector being adapted to a wide variety of indoor environments.

- Chapter 6 is devoted to the second goal of our work: the development of a set of perceptual-motor skills that provides a basic common foundation for robots that desires to achieve a natural human-robot interaction. The chapter presents a system able to

visually detect and track multiple persons using a stereo camera placed at an under-head position. This camera position is especially appropriated for human-machine applications that requires interact with people or analyse human facial gestures. In an initial phase, the environment is modelled using a height map that will later be used to easily extract the foreground objects and to search people using a face detector. Once a person has been spotted, the system is capable of tracking him while is still looking for more people. Tracking is performed using the Kalman filter to estimate the next position of each person in the environment. Nonetheless, when two or more people become close to each other, information about the colour of their clothes is also employed (whenever they wear clothes of different colour) in order to track them more efficiently. The system has been extensively tested and the low computing time required for the detection and tracking process makes it suitable to be employed in real-time applications.

- Chapter 7 presents an expandable architecture that implements the three basic perceptual-motor skills previously indicated: (i) to detect an interested user who desires to interact with the robot; (ii) to keep track of the user while he/she moves in the environment; and (iii) to follow the user along the environment avoiding possible obstacles in the way. A three-layer multi-agent system architecture is proposed in order to control the robot movement and to enable expandability. Agents in the lower layer provide an abstraction of the real hardware used, thus the system can be easily adapted to different robot platforms. The middle layer implements a set of basic behaviours that allows to control the movement of the robot and to keep visual track of human users (employing techniques developed in the previous chapter). In the upper layer, these basic behaviours are combined in a sequential or concurrent manner in order to implement more complex behaviours named perceptual-motor skills (or simply *skills*). The three skills designed constitute a minimum set of abilities useful for many mobile robotic applications that requires to interact with human users. Besides, the multi-agent approach employed allows the system to be easily expanded with further functionality. The system has been evaluated in different real-life experiments with several users,

achieving good results and real-time performance.

- Finally, Chapter 8 concludes this thesis exposing its main contributions, the publications derived from it, and indicating possible future reseach lines.

Chapter 2

Related Work and State-of-the-art

This chapter is devoted to the study of the state-of-the-art in the areas related to the work developed in this thesis. Firstly, it is described the most important sensors and actuator in nowadays robots. Secondly, the principal approaches employed for representing the environment and for robot localisation are exposed. Thirdly, a general overview of the most relevant control architectures employed in the literature is given. Fourthly, it is shown the main uses of fuzzy logic in robotics stressing the applications involving genetic algorithms for tuning purposes. Fifthly, a general overview of the use of evolutionary algorithms in robotics applications is given. Finally, we made a revision of the main works on vision for robots and an overview of stereo vision applied to people detection and tracking problems.

2.1 Environment Perception

Robots have a sensorial system that enables them to obtain information from the surrounding environment in order to extract meaningful information. There are a wide variety of sensors available, each of one with its particular sources of errors, for measuring different physical properties. Sensors can be classified using two important functional axes into *proprioceptive/extereoceptive* and *passive/active* [195].

Proprioceptive sensors measure internal values of the robot; for example, battery voltage, temperature, wheel load.

Extereoceptive sensors acquire information from the robot's surroundings; for example, distance measurements, light intensity, sound amplitude. Hence, these type of sensors are employed by the robot to obtain meaningful information from the environment.

Passive sensors measure environmental properties by receiving incoming energy; for example, microphones, CCD or CMOS cameras.

Active sensors emits energy, then measure the environmental reaction; examples of these type of sensors are sonar sensors, laser rangefinders.

Following we provide a list of the most usual sensors employed in mobile robotics, as well as an indication of their main deficiencies and limitations:

- *Ultrasound sensors* allow to detect the echo of an ultrasound signal. It is one of the most frequently employed sensors to measure distance. They have a good error precision (1% error in optimal operation conditions) and a operating range varying from 20 cm up to 10 m. It is a very cheap sensor that have several weakness. Firstly, it is difficult to accuratly estimate the exact position of the object that produces the echo because it can be in a cone varying from 15^{o} to 30^{o}. Secondly, the signal does not necessarily have to rebound perpendicular to a surface. Thus, the echo can return perturbed, get lost or even be absorbed. Thirdly, surfaces placed too near the sonar emitter can cause problems in the signal reception.

- *Infrared sensors* emit an infrared light instead of an ultrasound wave, achieving a higher angular precision than ultrasound sensor, and the ability to detect near objects. However, they require a precise calibration and can produce erroneous measures because of several sources of light (specially natural light).

- *Laser rangefinders* work like the two above but emitting a laser signal. The rebounded laser ray is measured to estimate the distance of the surrounding obstacles. The great advantage of this sensor is its high angular precision (approx. 1^{o}) and its low error at large distances.

- *Visual information* is probably one of the most powerful sensors that can be employed on a mobile robot. Vision is the main sense for the human being and is fundamental

for a proper interaction in our daily life. Thus, it has been deeply explored and used in mobile robots in order to provide them with the ability of seeing. A camera consist in an array of cells sensible to a particular spectrum of the light. CCD and CMOS sensors are the devices employed to acquire light information.

A CCD camera comprises photosites, typically arranged in an X-Y matrix of rows and columns. Each photosite, in turn, comprises a photodiode and an adjacent charge holding region, which is shielded from light. The photodiode converts light (photons) into charge (electrons). The number of electrons collected is proportional to the light intensity. Typically, light is collected over the entire imager simultaneously and then transferred to the adjacent charge transfer cells within the columns. Next, the charge is read out: each row of data is moved to a separate horizontal charge transfer register. Charge packets for each row are read out serially and sensed by a charge-to-voltage conversion and amplifier section. This architecture produces a low-noise, high-performance imager. That optimisation, however, makes integrating other electronics onto the silicon impractical. In addition, operating the CCD requires application of several clock signals, clock levels, and bias voltages, complicating system integration and increasing power consumption, overall system size, and cost.

A CMOS imager, on the other hand, is made with standard silicon processes in high-volume foundries. Peripheral electronics, such as digital logic, clock drivers, or analog-to-digital converters, can be readily integrated with the same fabrication process. CMOS imagers can also benefit from process and material improvements made in mainstream semiconductor technology. To achieve these benefits, the CMOS sensor's architecture is arranged more like a memory cell or flat-panel display. Each photosite contains a photodiode that converts light to electrons, a charge-to-voltage conversion section, a reset and select transistor and an amplifier section. Overlaying the entire sensor is a grid of metal interconnects to apply timing and readout signals, and an array of column output signal interconnects. The column lines connect to a set of decode and readout (multiplexing) electronics that are arranged by column outside of the pixel array. This architecture allows the signals from the entire array, from subsections, or even from

a single pixel to be readout by a simple X-Y addressing technique something a CCD can't do.

In Section 2.8, a more detailed introduction to the use of computer vision in robotics is given.

- *Gyroscopes* are devices for measuring or maintaining orientation, based on the principle of conservation of the angular momentum. In physics this is also known as gyroscopic inertia or rigidity in space. The essence of the device is a spinning wheel on an axle. The device, once spinning, tends to resist changes to its orientation due to the angular momentum of the wheel. Gyroscopes can be employed to measure the relative direction of the robot in relation to a reference direction. Nowadays, electronic gyroscopes (based on laser) are cheap and small devices that can be easily mounted on robot platforms.

- *Accelerometers* are devices for measuring acceleration. An accelerometer inherently measures its own motion, in contrast to a device based on remote sensing. Accelerometers are used along with gyroscopes in inertial guidance systems, as well as in many other scientific and engineering systems. One of the most common uses for micro electro-mechanical system (MEMS) accelerometers is in airbag deployment systems for modern automobiles. In this case the accelerometers are used to detect the rapid deceleration of the vehicle to determine when a collision has occurred and the severity of the collision. However, accelerometers are not very accurate devices for localisation and are very sensible to the floor inclination and to the gravitational acceleration.

- *Compass* is a navigational instrument for finding directions on the earth. It consists of a magnetised pointer free to align itself accurately with Earth's magnetic field. A compass provides a known reference direction which is of great assistance in navigation. The cardinal points are north, south, east and west. The main disadvantage of that device is that the earth magnetic field might be affected by electric fields. Thus, the use of compass in indoor environment is not very appropriate.

The reader interested in further details about the characteristic of each kind of sensor is referred to [112, 195].

2.2 Actuators

Actuators are the mechanisms by which robots acts upon an environment. A single robot can contain dozens of different types of actuators, each one chosen to do a specific task. They are normally used for locomotion and object manipulation purposes.

Robots are defined in part by their ability to move. Mobile robots can be categorised according to their locomotion system in: (i) wheeled robot (that use wheels to move), (ii) legged robots (that moves using two or more legs), and (iii) slidding robots (that employs their own body to move).

Regarding object manipulation, it is usually performed by the use of grippers of different size and shape.

Despite the variety of actuators a robot can have, most of them consist in electric motors. Electric motors are actuators that produce motion from electricity by the electromagnetic effect. Electric motors typically operate at a high speed and have small turning power. But most robots need low speed but large power. A collection of gears, called a gearbox, is used to adjust the speed and force of the turning power. However, not all electric motors produce rotational motion. Solenoids are electric motors that produce linear, or in-and-out motion. Solenoids are frequently used in switches that turn things off and on.

Careful adjustments in position are often important in robotics. Special motors called stepper motors turn in precise, incremental "steps", making them ideal for small, repeated adjustments in position. Another electric motor used for positioning is the servo motor. This kind of motor can only turn 90° to the right or left. Servo motors are usually employed in remote-control car, boat, or plane in order to controll the steering.

2.3 Environment Representation

The design of an adequate representation model is necessary when it is desired to perform complex navigation or manipulation tasks. The use of a map of the environment can help the robot to know its current location as well as to create a proper plan for navigating. Three main approaches can be seen in the literature to solve that problem.

- Geometric approaches [64] consist in the use of either complex 3D models or basic cell maps to represent the environment. The latter are the most frequently employed. They represent the environment as a matrix of cells, each one representing an planar region of the environment. Each cell has associated an occupancy value that indicates whether there are obstacles in it. This information is employed by the robot in order to plan a trajectory towards a desired goal avoiding collisions. The main advantage of geometric approaches is that they allow to apply very efficient trajectory planning techniques. However, they usually require the use of an accurate localisation method.

- Topological approaches [120] consist in detecting the different elements of the environment and identifying the relationship between them. They normally represent the environment at a higher degree of abstraction than metric approaches. Topological approaches usually employ connected graphs to represent the environment. Thrun, in Ref. [205], creates a topological map from the region division in a cell map. However, the meaning associated to nodes and arcs in the topological representations is not always the same. Kuipers and Byun, in Ref. [126], use nodes to represent important places from the point of view of the sensory system, and arcs to represent paths between these places. In some cases, the representation includes information about the appropriate way to navigate from one node to another. In Ref. [145], Mataric uses the nodes to represent distinguished places associated to behaviours. Aguirre and González, in Ref. [3], use the arcs to indicate the necessary behaviour to navigate between nodes. The plan is transformed in a sequential set of behaviours to execute (representing the arcs) and the nodes are necessary sub-goals to achieve the plan.

- Finally, other authors [3] have proposed the use of hybrid approaches in order to combine the advantages of both.

2.4 Localisation

One of the most important problems when designing control architectures for mobile robots is the knowledge of their position. An autonomous mobile robot must be aware of its position for the development of certain tasks, specially for navigation purposes.

The accuracy of the localisation method employed depends on the application and on the control architecture used. Whereas certain applications require the use of very accurate localisation systems others need only an approximate location. For example, geometrical approaches normally forces to use an accurate localisation method. However, topological approaches usually need to specify the robot position in the graph (normally less accurate than geometrical approaches).

Localisation is a well explored topic in the related literature and many authors have proposed different techniques, and even special hardware devices, in order to solve the problem. Below, we gives a brief overview of the most famous techniques indicating their drawbacks. The interested reader is referred to [216] for further information.

- *Odometry*: is the most frequently employed localisation method because of its low price and its low computational cost. It is based on the use of simple equations, describing the movement of the robot's platform, in order to transform the movement of the robot into a linear movement. For that purpose, the number of turns of the robot's wheels is measured. The main drawback of this technique is that the error is accumulative. The sources of errors of this technique are diverse but they can be categorised into *systematic* and *non systematic*. The former are caused by imperfections in the robot construction (wheels with different diameter or incorrectly measured). The latter are caused by the interaction of the wheels with the floor (slippage or irregularities in the surface). Therefore, it is difficult to model the real error of the system altough there are several attempts [26, 27].

- *Inertial navigation*: uses gyroscopes and accelerometers to measure the rotations and accelerations of the robot in its movement. The main advantage of this kind of device is that they do not require any external reference. This technique can help to reduce the odometric errors.

- *Compass*: consists in determining the robot orientation in relation to the earth magnetic field. There are different localisation methods that use compass. The most famous is the *flux gate compass* that measures the horizontal component of the earth magnetic field. This is a very advantageous method: low power consumption, low cost, the device employed does not use mobile parts and is not affected by vibrations. Therefore, it is very appropriate method for outdoor robots.

  As previously indicated, the main disadvantage of that positioning system is that the earth magnetic field might be affected by electric fields. Thus, the use of compass in indoor environment is not very appropriate.

- *Active beacons* have been employed since the beginning of navigation. Active beacons can be easily detected and provides a great precision with a low computational cost. There are two main localisation methods that use beacons: trilateration and triangulation.

  Trilateration consists in determining the robot position using the distance to the beacons. For that purpose, they must be placed at known positions. It is common the use of, at least, three emitter beacons and a receptor in the robot. This is the method employed by the Global Positioning System (GPS) explained later.

  Triangulation requires at least three beacons and a rotational receptor in the robot. By measuring the angles observed between the beacons it is possible to calculate the (x, y) position of the robot.

- *Global Positioning System* is a revolutionary technique for navigation in outdoor environments. The GPS is comprised by a set of satelites that emit encoded signals. A receiver in the earth can estimate its position by measuring the time the signal last in

coming to earth. If the distance from the sensor to the three satelites is known, it is possible to estimate the longitude, latitude and height of the receiver. GPS is funded by and controlled by the U. S. Department of Defense (DOD). While there are many thousands of civil users of GPS world-wide, the system was designed for and is operated by the U. S. military.

- *Landmarks* have been widely employed in mobile robotic navigation and localisation. They can be classified into *natural* and *artificial* landmarks.

  Artificial landmarks are known elements that are placed in the environment in order to provide a robot with useful information for self-positioning purposes [58, 132]. Artificial landmarks are human-created artifacts that are placed along the environment. When an artificial landmark is visible, the robot is able to calculate its position in a map. However, while the robot is in a zone not covered by the landmarks, it experiences a cumulative error. Thus, the distance between landmarks should no be excessive. Several designs of landmarks have been proposed depending on the purpose they are used for [117, 191] and even the best arrangement in order to get minimum-error position has been studied [203]. The main disadvantage of that method is the necessity of altering the environment in order to place the landmarks.

  Natural landmarks are environmental elements that can be employed as reference. We refer to walls, doors, corridors, rivers or mountains [69, 138, 221]. The main advantage of using them is that they are non-invasive methods. However, it is more complex the development of appropriate detectors.

- *Map-based localisation*: also known as *map matching*, consists in creating a local map, using the robot's sensors, that is matched against a global map in order to estimate the position of the robot. The global map can be either a CAD model of the environment or a map generated with the robot's sensors. The main advantage of this technique is that the robot gets information about environment without altering it. Besides, algorithms to learn the environment can be employed in order to increase the accuracy in the localisation. The main problems of this technique are that it relies on the sen-

sor readings (that are obviously affected by errors) and that the matching process is a difficult task.

2.5 Control architectures

Controlling a robot is a difficult task that becomes harder as the robot's goals becomes more complex. The most appropriate way of managing the increasing complexity of the control system is to divide the whole functionality in smaller and simpler elements that interact each other. This is commonly named *control architecture* [61].

Several control architectures have appear in the last years and their suitability strongly depends on the application required. There are several types of architectures that can be classified, from a topological point of view, as vertical and horizontal architectures. In horizontal architectures, all the levels have access to the perceptions and to the actuators. In vertical architectures, the lower levels access to perceptions, the information is passed to higher levels where it is processed, and finally, the action commands are passed from the upper layers to the lowers ones. Other authors have classified the different architectures from a functional point of view as: deliberative, reactive (behaviour-based) and hybrid [14, 129].

Next, we describe the characteristics of each type of architecture from a functional point of view:

2.5.1 Deliberative architectures

Deliberative architectures, also known as hierarchical architectures, are based on symbolic artificial intelligence, which is in turn, based on the physical symbol system hypothesis [166]. The essence of this hypothesis can be summarised as follows: a physical symbol system has the necessary and sufficient means for intelligent action.

Deliberative control takes into consideration all of the available sensory information and amalgamates them with all the controller's internal knowledge to create a plan of action. A symbolic model of the world is explicitly represented and decisions are made based on logical reasoning. The control searches through all the possible actions plans until it finds a

suitable one. This search sequence can take a long time and is hence not suitable in situations where the robots are expected to react quickly. Furthermore, there is often a problem in translating the real physical world into an accurate and sufficient symbolic representation for the robot to make meaningful decisions.

The first *intelligent* robot, Shakey [168], was developed at the Stanford Research Institute in the 60s. Its controll architecture was based in the SPA (Sense-Plan-Act) approach [25] (see Fig. 2.1). Sensors were employed to perceive the external world in order to create an precise and complete model of the world. Then, the set of actions required to achieve a desired goal were planned. Each action of the plan was passed to the robot's controller and thus directly executed. Therefore, the actions that the system was able to manipulate were low level actions directly related to the hardware.

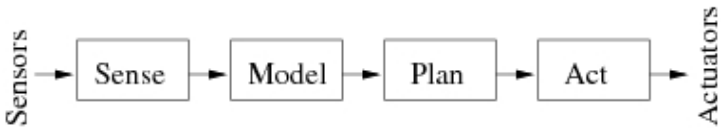


Figure 2.1: Sense-plan-act model.

This is a very appropriate model when the world in perfectly known. Therefore, this model is commonly employed in industrial robots in factories, with magnetic beacons, marked paths, etc. However, when task is to be performed in an unstructured, unpredictable, noisy environment, they fail to succeed. If an unexpected obstacle appears in the path of the robot, it could not be rapidly avoided because the system has to perform the whole sense-plan-act sequence.

2.5.2 Reactive architectures

In sharp contrast to deliberative architectures are reactive architectures, where the perception is tied closely to the effector action. These architectures are also called behaviour-based architectures [14, 34, 146]. The architecture does not entail any kind of symbolic world model and does not use complex reasoning. The reactive control is essentially a reflex mechanism where stimulus-response pairs govern actions. Robots using reactive control respond quickly

to a changing environment without requiring a priori information nor a map. Besides, this type of architectures requires an small amount of memory and does not compute or store representations of the world. The inability to learn over time is perhaps is main drawback of reactive architectures.

Reactive architectures result from the limitations imposed by symbolic AI. System that use that type of architecture do not have a complicated mind, but instead are constructed in a way that allows them to react to a changing environment by their "instincts". As the diction already indicates these architectures are associated with observations from the animal world. E.g. an ant colony consists of different but very simple individuals but the colony itself exhibit more intelligent behaviour than one would expect from the ants seen in isolation (emergence).

The first example of this kind of architecture was the *subsumption architecture* developed by Rodney Brooks [35]. The subsumption architecture proposes a layered design of competing task-accomplishing behaviours. Lower layers exhibit more primitive kinds of behaviour, and have precedence over layers further up the hierarchy. He postulates that complex (and useful) behaviours are not necessarily product of complex control architectures. Rather, complex behaviours can emerge from the fusion of simpler ones. The first robots that employed that architecture were able to go towards a desired direction avoiding obstacles. Other important examples of this type of architecture is the Maes' *action selection* [139, 140] and Arkin's *motor schemas* [13].

Reactive architectures employ a set of behaviours that acts concurrently to achieve independent goals. Each behaviour has direct access to the sensors, and emits signals to the actuators if necessary (see Fig. 2.2). A coordinator module is in charge of deciding the final output to actuators. Depending on the strategy employed, behaviour-based architectures can be classified into:

- *Competitive*: in these architectures, the coordinator selects only one output.

- *Cooperative*: in these architectures, the coordinator creates a composited output based on the individual outputs of the behaviours.

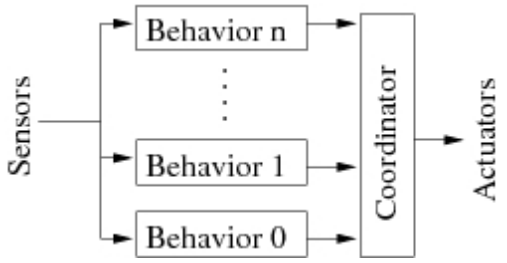


Figure 2.2: Behaviour-based architectures.

In contrast to deliberative architectures, all the layers have direct access both to sensors and actuators. Each layer, analyses the input and decides whether must be activated. Most researchers agree that reactive agents are not well-suited for many kinds of problems. Reactive architectures, such as Brooks' subsumption architecture have some inherent weaknesses. They are inflexible, often requiring completely reworked control system to adapt them to knew environments. They cannot effectively take into account information which is not locally available to their sensors, and they cannot take advantage of previous knowledge in performing a new task. These weaknesses correspond to the strengths of a deliberative architecture.

2.5.3 Hybrid architectures

Hybrid architectures are in the middle of deliberative and reactive architectures. When dealing with real-life problems, there is the need to manage high level information such as a representation of the world, goals and plans, as well as react upon sudden events in the environment. Systems developed using hybrid approaches combine the fast response to environment changes of reactive architectures with the ability to manage high level symbols. Thus, they are more suitable for complex real-life tasks.

Usually, these architectures as structured in three layers (see Fig.2.3). The lower layer is composed by reactive behaviours that establish a direct relation between action and perception. The upper layer is in charge of creating a high level plan according to the goals and beliefs of the robot. This layer is able to manage symbols and reason about the robot's status. Finally, the middle layer usually is the bridge between the upper and lower layers. The

middle layer receives a high-level-of-abstraction plan that decomposes in sub-goals. Then, according to the current status of the plan, this layer coordinates the active behaviours by assigning them appropriate priorities.

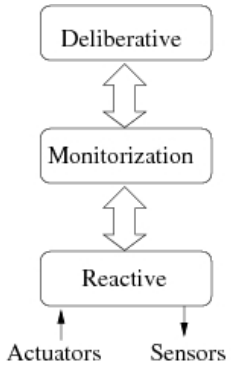


Figure 2.3: Hybrid architectures.

Examples of this approach are Arkin's AuRA [12] and Gat's Atlantis system [84] for JPL's rovers.

2.5.4 Agent-based architectures

The agent theory has become very popular in the last twenty years. Altough the concept of agent is very related to the AI field, its study has not attracted the attention of researches since 80s. Defining the term *agent* is something difficult because its has been commonly employed in many different areas and for many different purposes. For our purposes, an *agent* can be understood as a software process aimed to get or keep a goal implicit in its own design. It can be said that agents must show the following properties [217]:

- *autonomy*: agents must have the capacity to operate independently of humans beings and have control over their actions.
- *social ability*: agents must be able to communicate with other agents or with human beings using some *agent communication language*.
- *reactivity*: agents perceives the environment in which they inhabit (whether it is physical or not) and reacts upon the events that occurs.

- *pro-activeness*: agents are goal-directed.

From a practical point of view, agents can be seen as the natural extension to the object oriented paradigm. They encapsulate functionality, goals and intention and communicate with other agents in a cooperative way. Some of the advantages of using agents for designing complex systems are: (i) the possibility of creating a distributed system comprised by agents running in several computers; (ii) the facility for adding, removing or altering the functionality of the system in an incremental way.

Agent-based architectures have often been employed for coordinating and controlling collections of mobile robots working on completing a task. Collections of two or more mobile robots working together are often referred to as teams or societies of multiple mobile robots, or more concisely *multi-agents*. Robotic teams has a potential use in tasks such as fire extinguishing, rescue and exploratory missions among others. The idea is having a multitude of low cost robots that cooperate to achieve a goal. Each individual robot can be a cheap device whose failure can be overcome by the work of the rest of the group. Thus, teams of robots are more robust against failures. This emerging field falls into the so called *Distributed Artificial Intelligence* (DAI). A famous example of this research effort is the RoboCup competition, where teams of dog robots compete in a soccer match.

Another possible use of an agent set is aimed to build complex and flexible control architectures for a single robot. In this work, we rely on agents to design robot's control architectures. In that sense, we employ the term *multi-agent system* to refer to systems whose functionality is distributed among a set of heterogeneous agents that cooperate in order to accomplish the system goals. The main advantages of multi-agent architectures are: (i) their ability to physically distribute the agents in several machines if required; and (ii) their flexibility to increase and reduce the system's functionality, allowing a great adaptability to design changes.

2.6 Fuzzy logic and Robotics

Fuzzy logic has became a powerful tool in many areas of robotics. The main reasons for that extensive use are that fuzzy logic allows: (i) to manage and propagate uncertainty and vagueness in a very natural way; and (ii) to define expert knowledge using *if-then* rules. In this section we bring a brief introduction to the uses of fuzzy logic in robotics. For further information about this topic, the interested readers is referred to [186].

- Design and coordination of behaviours: because of fuzzy logic is able to manage the vagueness and uncertainty of sensorial systems, it has been widely employed to design behaviours. A representative example of that is the Shaphira architecture. Shaphira [119] is an architecture consisting of a set of fuzzy behaviours (encoded as Mandani rules) able to develop complex tasks. In this architecture, the belief degree of the rule's antedecent is used to calculate its activation level. Aguirre et. al. employ in [1] a similar approach by defining several basic behaviours (*follow-wall*, *navigate-in-corridor*, *avoid-obstable*, etc.) expressed as fuzzy sets. The behaviours are appropriately coordinated according to the robot's context, and the transition between them is soft. Bonarini, in [23], performs the control of the robot by a set of basic behaviours grouped in two classes according the context of applicability. He distinguish *global* behaviours from *context* behaviours. The former are employed to achieve the goals of the robot, like *follow-wall*, while the latter are employed to solve unexpected situations, like *avoid-obstacle*.

- Map building: fuzzy logic has also been successfully employed in the contruction of maps of the environment, using both topological and geometrical approaches. Gasós and Saffiotti, in Ref.[83], propose a technique for building geometric maps using sonar. In an initial phase, the robot creates a global map of the environment that is later updated using its local perceptions. The map is based on the use of fuzzy borders projected on a cell array. Similarly, Aguirre presents in [2] an hybrid map (geometrical and topological) using fuzzy segments created with the sonar sensors. Fuzzy segments define walls of the environment that are employed to plan the navigation of the robot.

Other approaches that employ fuzzy logic to create maps are Refs. [81, 172].

- Localisation: fuzzy logic has been employed to calculate the robot position as a fuzzy location. Instead of representing the position of the robot using probabilistic measures of the uncertainty, some authors have employed a possibility region to estimate the robot location. This approach allows an easy integration with the fuzzy maps previously explained. Examples of this approach can be seen in Refs. [83, 188]

- Perception: Fuzzy logic has also been employed for creating perceptual systems. Howard et al. propose in [103] a method based on fuzzy logic to analyse the characteristics of a terrain. They use the visual system of the robot in order to calculate variables like *transversability and discontinuity*. These variables are then employed by the robot controller to plan the best motion strategy. Le et. al. show in [133] a fuzzy visual system that detect the borders of a road using the images taken from their THMR-III robot.

2.6.1 Genetically Tuned Fuzzy Systems

As previously showed, fuzzy systems have been widely employed for multitude of robotic applications. Probably, one of the reasons that makes fuzzy systems so used in many areas is that they save expert knowledge as rules. Thus, fuzzy systems can be easily created and interpreted by experts. However, sometimes the selection of the proper values of the fuzzy labels is a difficult and tedious task. The optimal configuration of a fuzzy system for a particular context of application might differ from the optimal configuration when the working conditions change. For example, imagine a fuzzy controller that implements the *follow-wall* behaviour. Altough the rules and variables of the behaviour might be appropriate for a wide variety of environments, tuning its labels might improve the controller performace for each particular building.

Tuning of fuzzy systems provides an automatic way to increase their performance using a validated data set describing the problem. Three main tuning approaches can be found in the literature [52]: tuning of the scaling functions that maps the input and output variables into

the ranges in which the fuzzy variables are defined [55, 179, 224]; tuning of the membership functions by moving, stretching or narrowing them [87, 170, 213] ; and tuning of the fuzzy rules modifying the fuzzy labels in the THEN-part of the rules [24]. Although the tuning process has been performed using classical optimisation techniques [155], the use of genetic algorithms (GAs) has become very popular in this field [39, 50, 94, 99, 141]. The term *Genetic Fuzzy System* has been used in the literature to refer to those fuzzy systems that somehow rely on GAs either to define their structure or to adapt some of their parameters [49, 52, 177].

Although a great effort has been done on the optimisation of fuzzy systems using single-objective approaches, there is usually a need in real problems for fulfilling several objectives at the same time. Those problems are referred to as *multiobjective* in the literature (an interesting survey on the topic can be found in [43]). The concept of *Pareto-optimum*, formulated by Vilfredo Pareto [176], constitutes the origin of the research in *Pareto-based* multiobjective optimisation. A solution is called Pareto optimal (or *non-dominated*) if no other solution found is better than it solving at least one problem goal. The term EMO (Evolutionary Multiobjective Optimisation) refers to a set of evolutionary techniques that have been used to solve multiobjective optimisation problems. The main advantage of EMO algorithms, so called *MOEAs*, is the ability of simultaneously searching several non-dominated solutions. MOEAs find different non-dominated solutions to a problem in a single run while several runs of a normal algorithm would be required to obtain similar results.

MOEAs have also been used for Genetic Fuzzy Systems. Much of the work has been focused on the learning of the database or the rule base of the fuzzy system [51, 80, 108, 110]. Nevertheless, there are also approaches to tune the membership functions of the fuzzy system. In [22], a fuzzy system that controls a rocket is tuned using a MOEA. It allowed to simultaneously optimise the rising and settling time, the steady state error and the overshot of the controller. In [42], a fuzzy system to control a MAS (Mass Rapid Transit) is tuned using a multiobjective approach based on Dynamic Evolution. The system was used to control the operation of a train dedicated to the transport of people from one station to another. It was composed of 16 parameters and the results showed that the method was able to optimise the

system according to the objectives of punctuality, energy and passenger comfort.

2.7 Evolutionary algorithms and robotics

The term *evolutionary robotics* [46] refers to the use of Evolutionary Algorithms (EA) in order to create: control architectures, knowledge, minds and even bodies for robots. The idea is representing the control system of a robot as a chromosome subject to the laws of genetics and of natural selection. There are several reasons that support this approach. Firstly, EA have been widely employed in many areas for optimising the parameters and structure of systems. Evolution, both natural and artificial, has the ability to find good solutions to a problem, avoiding getting stuck on local minimas of the search space. Therefore, an EA is able to find a solution near the global optima after an appropriate number of iterations. Secondly, because of its stochastic nature, evolution is able to find a set of different solutions, all of them valid, for a given problem. All these reasons have lead many researchers to change the manual design of intelligent system for evolutionary approaches. The evolutionary philosophy has been used in different areas of robotics employing different soft-computing techniques (fuzzy logic, neural networks, genetic programming, etc).

- Uribe et. al presented in [211] an evolutionary approach for evolving fuzzy controllers. The experiment consisted in evolving the *follow-wall* behaviour by tuning the membership functions of the linguistic variables, preserving the rule base. The input to the controller was a vector indicating the direction of maximum free space and its time variation. The output was the direction, speed and acceleration of the robot's movement. The rule base was comprised by 33 rules whose linguistic variables were created using trapezoidal and triangular functions. The objective of the evolutionary algorithm employed was to adapt the membership functions in order to achieve a controller able to follow the wall at high speed avoiding brusque movements. The fitness of the individuals where evaluated according to an ideal behaviour based on predefined values of speed and acceleration.

- Matellan et. al presented in [147] an approach to learn a fuzzy navigation behaviour. In their work, the robot employed was a Khepera and the environment was a rectangular region, enclosed by artificial walls, with two obstacles in it. The aim was to create a fuzzy behaviour able to navigate in the environment avoiding the obstacles. The number of inputs, output as well as the number of variables and their range were set a priori by the designers. There were two inputs, right and left perception of the infrared sensors, having values in five linguistics labels. Similarly, there were two outputs representing the activation level of the robot's motors. The goal of the genetic algorithm was to learn the rule base to implement the desired behaviour. The results obtained showed that this approach was appropriate for learning rule bases when the domain of inputs and outputs is known.

- Mucientes and Casillas presented in [157] a generic method for evolution of fuzzy behaviours. In their work, they propose the COR methodology to evolve the *follow-wall* behaviour. The main goal of the proposed methodology is preserving the interpretability of the rule bases so that human users could analyse the resulting behaviours. The final behaviour was comprised by 77 rules and was only tested on simulation. However, the behaviour allowed a robot to move safely following walls both with convex and concave corners.

- Floreno and Mondada [70] evolved behaviours that controll the navigation of a Khepera robot. The goal of the experiments were to evolve behaviours that enables the robot to move avoiding obstacles. For that purpose, the behaviours were evolved on real robots placed on a real scenario. The authors define a fitness function that gives high values to behaviours that achieve: high-speed, linear movement and obstacle avoidance. The *brain* of the robot consisted in a neural network with several recurrent connections in the output layer. The evolution process determined the number of neurons in each layer and also the weight of the connections. During the experiments, the authors noticed that the best individuals exhibited a behaviour more intelligent than expected. They observed that good individuals moderated their speed in the proximity

of obstacles.

- Miglino et al. evolved in [149] a simple neural network (using simulation) in order to implement an exploring behaviour. The net was comprised by a pair of neurons in the input layer (one for each sensor), two hidden neurons and two output neurons (one for each motor). Additionally, the network had a recurrent connection in the hidden layer in order to memorise previous states. The simulation environment consisted in a grid map that had to be explored by the robot. The fitness function assigned high values when the number of visited cells was high. During the experiments, the authors noticed that when the robot was placed at the same starting position, the evolved nets tended to find good paths. In order to avoid the over-training, the individuals were forced to start the test from different starting positions. Then, they employed the best individual generated to repeat the experiment in a real scenario. They notice that the behaviour exhibited by the robot in simulation was different from the exhibited in the real-life scenario. They concluded that the reason was the presence of noise in the real sensor. They repeated the evolution introducing noise in the simulation and repeated the real-life experiments obtaining better results.

- Nelson et al. [165] perform a similar experimentation. However, they used sophisticated methods to represent temporal information. The goal of the experiments was the creation of behaviours able to scape from a labyrinth avoiding obstacles. They employed neural networks with several hidden layers and recurrent connections in order to store previous states of the behaviour. The fitness function employed assigned high values for short trajectories and low values when the individuals got stuck in the labyrinth. The best networks had between one and three hiddent layers with five to ten neurons in each layer.

- In [169], Nolfi y Marocco created a recurrent neural network to controll a Khepera equipped with sonar a vision sensors. The working environment was a white square with two black stripes of different sizes placed at opposite sides. The goal of the behaviour was to approach to the larger stripe. The sensor employed to perceive the

stripes was the camera. The best controllers obtained were able to wisely exploit the following structural characteristics of the environment: (i) the angular size of an object varies when the camera approaches to it; and (ii) the time an object is visible while performing a visual scan depends on the size of the object. Therefore, the best individuals exhibit a behaviour consisting in turning over themselves, without performing any translational movement until a black stripe was visible. Then, the individual start an slow translational movement keeping the rotational one. As the longer stripe remains longer in the field of view, the robot approaches slowing to the desired position.

- In [183], Reynolds evolves the corridor-following behaviour using genetic programming. The main contribution of this paper is the use of a realistic models of the sensors and actuators (including noise) to minimise the gap between simulation and real experimentation. The inputs to the system were the sonar values of the simulated robot and the output was the movement command of the robot (among a limited set of possibles movements). The goal was to provide the robot with a *corridor-following* behaviour able to avoid collisions with the walls. Therefore, individuals that collided with walls were automatically removed from the population. The results showed that the best individuals used only the front sonars to guide the robot.

Besides these works, EA have also been successfully employed for soccer robots in Robocup [131, 215], training robotic arms [156], behaviour-coordination [70, 89, 214] and even on legged robots [79, 181].

2.8 Computer Vision

Computer vision is the study and application of methods which allow computers to *understand* image content of multidimensional data in general. The term *understand* means here that specific information is being extracted from the image data for a specific purpose: either for presenting it to a human operator (e.g., if cancerous cells have been detected in a microscopy image), or for controlling some process (e.g., an industry robot or an autonomous

vehicle). The image data that is fed into a computer vision system is often a digital grey-scale or colour image, but can also be in the form of two or more such images (e.g., from a stereo camera pair), a video sequence, or a 3D volume (e.g., from a tomography device). In most practical computer vision applications, the computers are pre-programmed to solve a particular task, but methods based on learning are now becoming increasingly common.

Computer vision can also be described as the complement (but not necessary the opposite) of biological vision. In biological vision and visual perception, real visual systems of humans and various animals are studied, resulting in models of how these systems are implemented in terms of neural processing at various levels. Computer vision, on the other hand, studies and describes technical systems which are implemented in software or hardware, in computers or in digital signal processors.

Even though earlier work exists, it was not until the late 1970's that a more focused study of the field started when computers could manage the processing of large data sets such as images. However, these studies usually originated from various other fields, and consequently there is no standard formulation of the *computer vision problem*. Also, and to an even larger extent, there is no standard formulation of how computer vision problems should be solved. Instead, there exists an abundance of methods for solving various well-defined computer vision tasks, where the methods often are very task specific and seldom can be generalised over a wide range of applications. Many of the methods and applications are still in the state of basic research, but more and more methods have found their way into commercial products, where they often constitute a part of a larger system which can solve complex tasks (e.g., in the area of medical images, or quality control and measurements in industrial processes).

Computer vision is by some seen as a subfield of artificial intelligence where image data is being fed into a system as an alternative to text based input for controlling the behaviour of a system. Some of the learning methods which are used in computer vision are based on learning techniques developed within artificial intelligence.

Since a camera can be seen as a light sensor, there are various methods in computer vision based on correspondences between a physical phenomenon related to light and images of that

phenomenon. For example, it is possible to extract information about motion in fluids and about waves by analysing images of these phenomena. Also, a subfield within computer vision deals with the physical process which given a scene of objects, light sources, and camera lenses forms the image in a camera. Consequently, computer vision can also be seen as an extension of physics.

A third field which plays an important role is neurobiology, specifically the study of the biological vision system. Over the last century, there has been an extensive study of eyes, neurons, and the brain structures devoted to processing of visual stimuli in both humans and various animals. This has led to a coarse, yet complicated, description of how *real* vision systems operate in order to solve certain vision related tasks. These results have led to a subfield within computer vision where artificial systems are designed to mimic the processing and behaviour of biological systems, at different levels of complexity. Also, some of the learning-based methods developed within computer vision have their background in biology.

Yet another field related to computer vision is signal processing. Many existing methods for processing of one-variable signals, typically temporal signals, can be extended in a natural way to processing of two-variable signals or multi-variable signals in computer vision. However, because of the specific nature of images there are many methods developed within computer vision which have no counterpart in the processing of one-variable signals. A distinct character of these methods is the fact that they are non-linear which, together with the multi-dimensionality of the signal, defines a subfield in signal processing as a part of computer vision.

Besides the above mentioned views on computer vision, many of the related research topics can also be studied from a purely mathematical point of view. For example, many methods in computer vision are based on statistics, optimisation or geometry. A significant part of the field is devoted to the implementation aspect of computer vision; how existing methods can be realised in various combinations of software and hardware, or how these methods can be modified in order to gain processing speed without losing too much performance.

Image analysis and image processing are the main fields of computer vision. The distinction between the two is not very clear, i.e., image analysis uses many methods which traditionally belong to image processing. One formal distinction would be to say that image processing deals with transforming images, producing one image from another, or with producing low-level information about an image, such as edges or lines. Neither of these tasks provide, or require, an interpretation about what the image contains in terms of objects or events. Image analysis, on the other hand, uses models and assumptions about the real world depicted in the images to extract information which, e.g., can be used to control actions on objects in a scene. In more advanced systems, these models can be learnt rather than programmed.

2.8.1 Elements of a Computer Vision System

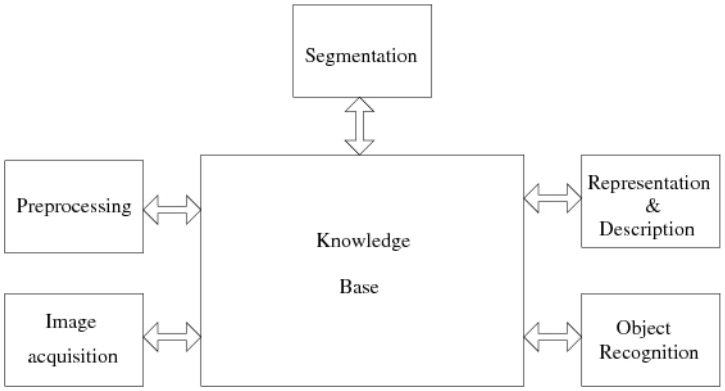


Figure 2.4: Elements of a typical computer vision system.

A typical computer vision system can be divided in the subsystems showed in Fig. 2.4. The central element, labelled as *Knowledge Base*, represents the knowledge about the particular problem to solve through vision. This knowledge may be as simple as detailing regions of an image where the information of interest is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an interrelated list of all major possible defects in a materials inspection problem or an image database containing high-resolution satellite images of a region in con-

nection with change-detection applications.

The *image acquisition* module is in charge of capturing images to fed the system with them. There exists many input devices that can be employed as input to a computer vision system: camera, radar, lidar, tomography system. Cameras are the most frequently employed devices for imagining purposes. There are two major technologies in that sense: CCD and CMOS that where explained in Section 2.1.

In the *preprocessing* step, the image is being treated with low-level operations. This is a process whose input an output are images. The aim of this step is twofold. Firstly, it aims to perform an *image enhancement* and an *image restoration* if needed. The idea behind image enhancement is to bring out details that are obscured, or simply to highlight certain features of interest in an image. A typical example of enhancement is when the contrast of an image is increased. Image restoration also deals with improving the appearance of an image. However, the aim of image restoration is the recovery of an image or removing noise. Typical operations included in these steps are: image sub-sampling, digital filtering (convolutions and correlations), performing an eigentransform on the image (Fourier transform), edge detection or estimation of local orientation, estimating disparity in stereo images and multiresolution analysis.

The *segmentation* step is present in most of the computer vision systems. It consists in partitioning an image into its constituent parts or objects. For example, in the automated inspection of electronic assemblies, interest lies in analysing images of the products with the objective of determining the presence or absence of specific anomalies, such as missing components or broken connection paths. Image segmentation algorithms generally are based on one of two basic properties of intensity values: discontinuity and similarity. In the first category, the approach is to partition the image based on abrupt intensity changes, such as edges in an image. The principal approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria. Thresholding, region growing, region splitting and merging are examples of methods in this category. In general, autonomous segmentation is one of the most difficult tasks in computer vision. A rugged segmentation procedure brings the process a long way towards successful solution

of imagining problems that require objects to be identified individually. On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure.

Representation and description is usually an step that follows segmentation. Its input is usually a raw pixel data, constituting either the boundary of a region, (i.e., the set of pixels separating one image region from another), or all the points in the region itself. There are basically two approaches: (i) represent the region in terms of its external characteristics (its boundary) or (ii) represent it in terms of its internal characteristics (the pixels comprising the region). The next task is to describe the region based on the chosen representation. An external representation is chosen when the primary focus is on shape characteristics. An internal representation is chosen when the primary focus is on regional properties, such as colour or texture. Sometimes, it might be necessary to use both types of representation. In either case, the features selected as descriptor should be as insensitive as possible to variations in size, translation, and rotation. Examples of the first type of descriptors includes chain codes, polygonal approximations, signatures, boundary segments, skeletons and Fourier descriptors among others. Examples of the second type of descriptors include topological descriptors, statistical measures, moments and principal components.

Recognition can be seen, in a wide sense, as the process of assigning a label (e.g, vehicle) to an object (also called *pattern*) based on its descriptors. There exists many techniques that can be employed to detect or classify patterns of different classes, all of them have been covered by the term *machine learning* [150]. Examples of pattern recognition techniques are: neural networks, support vector machines, genetic algorithms, statistical methods, clustering methods, decision trees, etc.

In order to achieve a wider view of the elements involved in typical computer vision systems, and a deep knowledge of their basis, the interest reader is referred to [90] for further reading.

2.8.2 Areas of application of Computer Vision

An interesting way to describe computer vision is in terms of its applications areas. One of the most prominent application fields is medical computer vision or medical image pro-

cessing. This area is characterised by the extraction of information from image data for the purpose of making a medical diagnosis of a patient. Typically image data is in the form of microscopy images [10, 59, 78], X-ray images [206, 208, 225], angiography images [19, 128], ultrasonic images [106, 121, 193], and tomography images [29, 30, 136]. An example of information which can be extracted from such image data is detection of tumours, arteriosclerosis or other malign changes. It can also be measurements of organ dimensions, blood flow, etc. This application area also supports medical research by providing new information, e.g., about the structure of the brain, or about the quality of medical treatments.

A second application area in computer vision is in industry. Here, information is extracted for the purpose of supporting a manufacturing process. One example is quality control where details or final products are being automatically inspected in order to find defects [18, 21, 115, 143, 189]. Another example is measurement of position and orientation of details to be picked up by a robot arm [62, 76, 104, 123, 209].

Military applications are probably one of the largest areas for computer vision, even though only a small part of this work is open to the public. The obvious examples are detection of enemy soldiers or vehicles and guidance of missiles to a designated target. More advanced systems for missile guidance send the missile to an area rather than a specific target, and target selection is made when the missile reaches the area based on locally acquired image data. Modern military concepts, such as *battlefield awareness*, imply that various sensors, including image sensors, provide a rich set of information about a combat scene which can be used to support strategic decisions. In this case, automatic processing of the data is used to reduce complexity and to fuse information from multiple sensors to increase reliability.

One of the newer application areas is autonomous vehicles which ranges from submersibles, land-based vehicles (small robots with wheels, cars or trucks) to aerial vehicles [100, 111, 127, 142, 182, 198, 207, 219]. An unmanned aerial vehicle is often denoted UAV. The level of autonomy ranges from fully autonomous (unmanned) vehicles to vehicles where computer vision based systems support a driver or a pilot in various situations. Fully autonomous vehicles typically use computer vision for navigation, i. e., for knowing

where it is, or for producing a map of its environment (SLAM) and for detecting obstacles. It can also be used for detecting certain task specific events, e.g., a UAV looking for forest fires. Examples of supporting system are obstacle warning systems in cars and systems for autonomous landing of aircraft. Several car manufactures have demonstrated system for autonomous driving of cars, but this technology has still not reached a level where it can be put on the market. There are ample examples of military autonomous vehicles ranging from advanced missiles to UAVs for recon missions or missile guidance. Space exploration is already being made with autonomous vehicles using computer vision, e.g., NASA's Mars Exploration Rover.

2.8.3 Stereo Vision

Stereopsis is the process in visual perception leading to perception of the depth or distance of objects. It comes from two Greek roots, *stereo* meaning solidity, and *opsis* meaning vision or sight. That means it could refer to any sort of visual depth perception, but since about the 1960s it has come to refer to depth perception from binocular vision, requiring two eyes. Prior to then, it was often referred to as "binocular stereopsis".

Depth from stereopsis arises from the slightly different positions each eye occupies on the head, a form of parallax. Stereopsis was discovered by Charles Wheatstone in 1833. He recognised that because each eye views the visual world from slightly different horizontal positions, each eye's image differs from the other. Objects at different distances from the eyes project images in the two eyes that differ in their horizontal positions, giving the depth cue of horizontal disparity, also known as retinal disparity and as binocular disparity. Wheatstone proved that this was an effective depth cue by creating the illusion of depth from flat pictures that differed only in horizontal disparity. To display his pictures separately to the two eyes, Wheatstone invented the stereoscope.

Leonardo da Vinci had also realised that objects at different distances from the eyes project images in the two eyes that differ in their horizontal positions, but had concluded only that this made it impossible for a painter to portray a realistic depiction of the depth in a scene from a single canvas. Leonardo chose for his near object a column with a circular

cross section and for his far object a flat wall. If he had chosen any other near object, he may have discovered horizontal disparity of its features. His column was one of the few objects that projects identical images of itself in the two eyes.

Until about the 1960s, research into stereosis was dedicated to exploring its limits and its relationship to singleness of vision. Researchers included Panum, Hering, Adelbert Ames Jr., and Kenneth N. Ogle.

Also in the 1960s, Bela Julesz invented random-dot stereograms. Unlike previous stereograms, in which each half image showed recognisable objects, each half image of the first random-dot stereograms showed a square matrix of about 10,000 small dots, with each dot having a 50% probability of being black or white. No recognizable objects could be seen in either half image. The two half images of a random-dot stereogram were essentially identical, except that one had a square area of dots shifted horizontally by one or two dot diameters, giving horizontal disparity. The gap left by the shifting was filled in with new random dots, hiding the shifted square. Nevertheless, when the two half images were viewed one to each eye, the square area was almost immediately visible by being closer or farther than the background. Julesz called the square a cyclopean stimulus after the mythical Cyclops who had only one eye. This was because it was as though we have a cyclopean eye inside our brains that can see cyclopean stimuli hidden to each of our actual eyes. Random-dot stereograms highlighted a problem for stereopsis, the correspondence problem. This is that any dot in one half image can realistically be paired with many same-coloured dots in the other half image. Our visual systems clearly solve the correspondence problem, in that we see the intended depth instead of a fog of false matches.

Also in the 1960s, Horace Barlow, Colin Blakemore, and Jack Pettigrew found neurons in the cat visual cortex that had their receptive fields in different horizontal positions in the two eyes. This established the neural basis for stereopsis. Their findings were disputed by David Hubel and Torsten Weisel, although they eventually conceded when they found similar neurons in the monkey visual cortex. In the 1980s, Gian Poggio and others found neurons in the monkey brain that responded to the depth of random-dot stereograms.

Basis of stereo computation

In this Section, we explain briefly the basis of stereo computation. For an updated review of the state of the art, the interest reader is referred to [36].

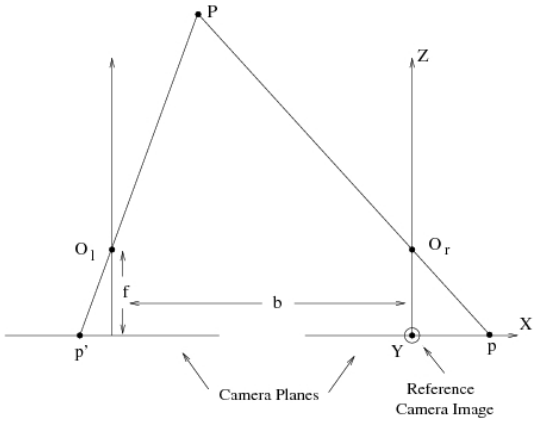


Figure 2.5: Minimal stereo system composed by two cameras

The minimal possible stereo system is composed by a pair of cameras (let us name them left and right cameras) whose optical centres (O_l and O_r) are separated by a distance b. Let us assume, in order to ease the explanation, that both cameras have identical optical characteristics and that their planes are coplanar and collinear (see Fig. 2.5). Let us assume that the centre of the reference system employed is the centre of the right image (named the *reference camera image*).

A point $P = (X, Y, Z)$ in the space projects to two locations ($p = (x, y)$ and $p' = (x', y')$) at the same horizontal scan line in the right and left camera images. The displacement of the projection in one image respect to the other is named *disparity* and the set of all disparities between two images is the so called *disparity map*. Disparities can only be computed for these points that are seen on both images but it is difficult when there is not enough texture or differences in the regions of the image. The points whose disparity can not be calculated are named *unmatched* points. The calculation of the disparity map is called the *correspondence problem*.

Knowing the intrinsic parameters of the stereo system it is possible to reconstruct the three-dimensional structure of the disparity map (*reconstruction problem*). The depth of a

given point can be calculated by triangulation as:

$$Z = \frac{fb}{d} \; ; \; X = \frac{xZ}{f}; \; Y = -\frac{yZ}{f}. \tag{2.1}$$

where f is the focal lenght of the cameras and d is the disparity calculated as $d = p - p'$.

Figure 2.6 shows a scene captured with a stereo system comprised by two cameras separated a distance $b = 12$ cm. Figure 2.6(a) and 2.6(b) show the images captured by the two cameras while Fig. 2.6(c) shows the disparity map calculated. In Fig. 2.6(c), clearer pixels indicate lower values of Z while darker ones represent farther distances. The unmatched points have been coloured in black. Finally, Fig. 2.6(d) shows the three-dimensional reconstruction of the scene.

2.8.4 Vision and Robotics

Vision is probably the most important sensor both for humans and robots. Vision provides richful information that can be of great help for mobile robots and it has been successfully employed in robotics mainly for navigation purposes. Navigation applications that use vision can be divided in two categories: outdoor vision-aided navigation and indoor vision-aided navigation. Altough we are more concern with the latter, a complete survey on both topics can be consulted in [58]. Below, we give a brief overview of the main contributions on indoor vision-aided navigation.

Indoor vision-aided navigation refers to the use of vision with the aim of assisting an autonomous robot in its navigation task in structured indoor environments. In a broad sense, indoor vision-aided navigation can be divided into the three categories explained below:

- *Vision applied to map-based navigation* consists in providing the robot with a map of the environment. These models may contain different degrees of detail (CAD models, occupancy maps, topological maps). The central idea of map-based navigation is to provide to the robot, directly on indirectly, a sequence of landmarks expected to be found during navigation. The task of the vision system is then to search and identify the landmarks observed in the image. Once they are identified, the robot can use the

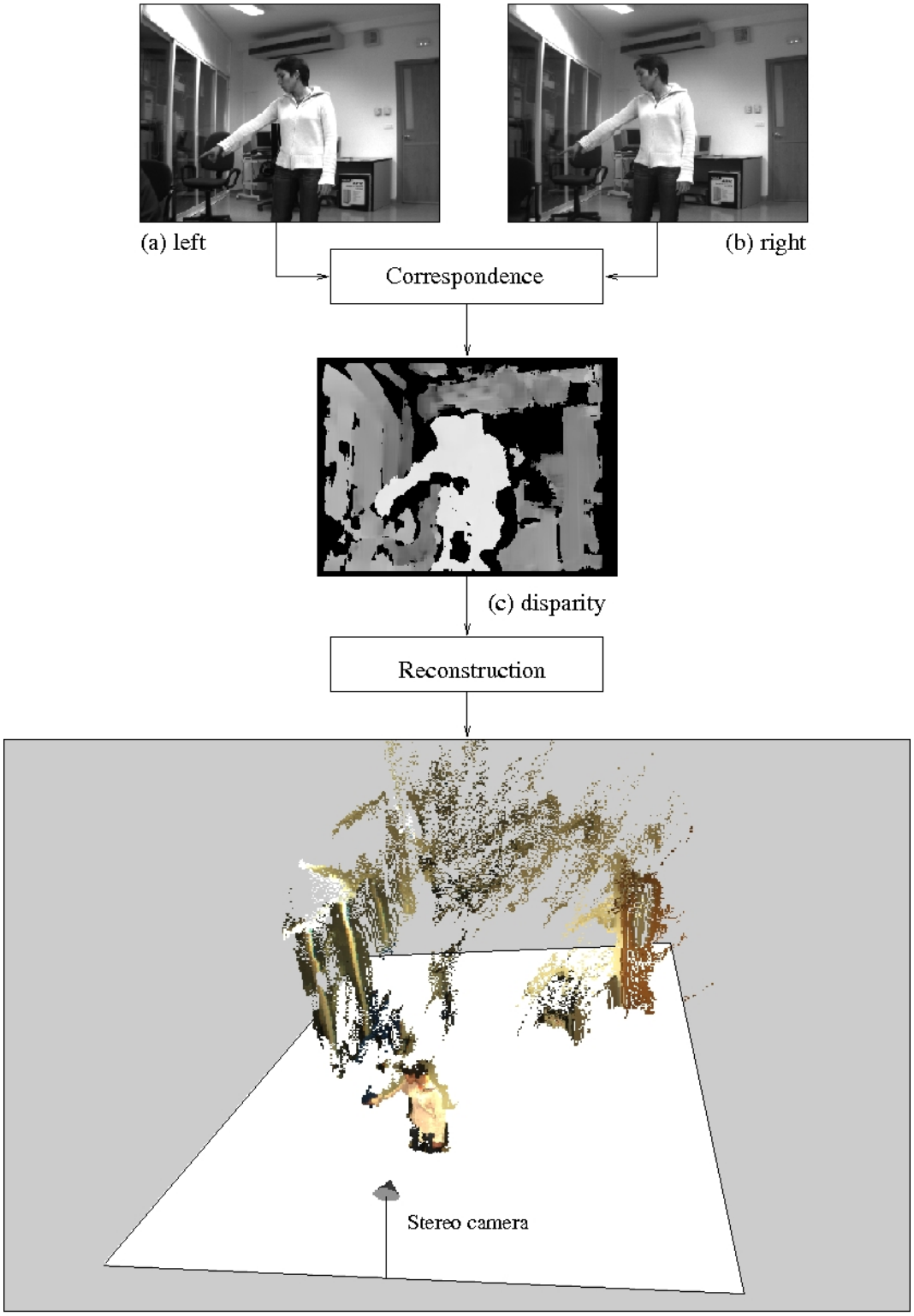


Figure 2.6: (a) Image of the left camera (b) Image of the right camera (c) Disparity map (d) Three-dimensional reconstruction of the scene.

provided map to estimate the robot position by matching the observation against the expectation. Three main approaches can be distinguished in this kind of map-based navigation: *global localisation*, *incremental localisation* and localisation derived from *landmark tracking*.

- *Global localisation* consists in determining the absolute position of the robot in the environment. Sugihara [201] presented one of the pioneering studies in robot self-localisation using single camera images. In that work, some of the localisation problems were pointed out and since both the location of the robot and the identity of the features extracted by the vision system were assumed to be unknown, the problem was reduced to a satisfaction problem. Later, Krotkov [122] extended this work by proposing a model where the effects of observation uncertainty were also analysed.
- *Incremental localisation* assumes that a location of the robot is known approximately, at the beginning of a navigation session, and that the goal of the vision system is to refine the location coordinates. Examples of such system are [148] by Matthies and Shafer, where stereo vision was used for error reduction; the FUZZY-NAV system of Pan et al. [174] that extended NEURO-NAV by incorporating fuzzy logic in a high-level rule-based controller for controlling navigation behaviours of the robot; the system of Horn and Schmidt [102] that describes the localisation system of the mobile robot MACROBE–Mobile and Autonomous Computer-Controlled Robot Experiment–using a 3D-laser-range-camera, etc
- Localisation derived from *landmark tracking* can be done when both the approximate location of the robot if the identity of the landmarks seen in the camera image are known and tracked. The landmarks used can be either artificial or natural. The artificial landmarks vary from circles with a unique bar-code for each landmark [113] to reflecting tapes stretched along the robot path [210]. Landmark tracking in these systems is normally carried out by using simple template matching.

- *Vision applied to map-building navigation.* Model descriptions are nor always easy to generate, especially when it is also required to provide metrical information. Therefore, many researchers have proposed automated or semiautomated robots that could explore their environment and build and internal representation of it. The first attempt at robotic map-building was the Standford Cart by Moravec [153]. The Standford Cart used a single camera to take images spaced along 50 cm slider. Next, an *interested* operator was applied to extract distinctive features in the images. These features were then correlated to generate their $3D$ coordinates. The *world* was represented by the $3D$ coordinates of the features plotted in a grid of two square meter cells. Altought the grid indirectly represented the position of obstacles in the world and was useful for path planning, it did not provide a meaningful model of the environment. For that reason, Moravec and Elfes presented later a new data structure known as occupancy grid [154] and employed sonar to fill it with information about the environment structure. In today's robots, occupancy grids allow measurements from multiple sensors to be incorporated into a single perceived map of the environment managing uncertainties, including stereo vision [158, 185].

  The occupancy-map approach to map learning is to be contrasted with the approaches that create topological representations of space [44, 164]. These representations often have local metric information embedded for node recognition and to facilitate navigational decision making after a map is constructed. One of the major difficulties of topological approaches is the recognition of nodes previously visited. In [205], Thurn has proposed an integrated approach that seeks to combine the best of the occupancy-grid-based approaches and the topology-based approaches. Other notable contributions related to map building are by Ayacha and Faugeras [15] using trinocular vision and Kalman filtering, and by Zhang and Faugeras [223] using $3D$ reconstruction from image sequences.

- *Vision applied to mapless navigation* includes all systems in which navigation is achieved without any prior description of the environment. Mapless navigation refers to robotics

that never use maps. Instead, the needed robot motions are determined by observing and extracting relevant information about the elements in the environment. These elements can be walls, objects such as desks, doorways, etc. Among the approaches that have been tried for that, the most prominent ones are: optical-flow-based [190] and appearace-based [85] techniques.

2.8.5 People detection and tracking using stereo vision

The topic *human-robot interaction* (HRI) has drawn a lot of attention in the last decade. The objective is to create intelligent robots capable of extracting information about the context or about the actions to perform through a natural interaction with users, for example, through their gestures or voice. One fundamental aspect in this sense is people detection and tracking, existing a extensive literature about the topic [86, 134, 197, 218]. People detection techniques are frequently based on the integration of different information sources such as: skin colour, face detectors, motion analysis, etc. Although people detection and tracking with a single camera are well explored topics, the use of stereo technology for these purpose concentrates now an important interest. Stereo vision systems have become a very appealing sensor to develop intelligent systems because of the availability of commercial hardware able to solve low-level problems of stereo processing at relatively low cost.

Stereo vision provides a type of information that brings several advantages when developing human-machine applications. Firstly, stereo information is more invariable to illumination changes than the information provided by a single camera. This is very advantageous factor for the development of background estimation techniques [53, 67, 97]. Furthermore, the possibility to know the distance from the camera to the person is of great help for tracking and gesture analysis purposes.

Darrel et al presented in [54] one of the first works that perform people detection and tracking using stereo vision. They built an interactive display system capable of detecting and tracking several people. Person detection was based on the integration of the information provided by three modules: a skin detector, a face detector and the disparity map provided by a stereo camera. First, independent objects (*blobs*) detected in the disparity map were

treated as candidate to people. Then, the colour of the image was analysed to identify those areas that could belong to skin. Finally, a face detector was applied on selected regions of the camera image. These three items were merged in order to detect and track several people. However, two main drawbacks can be pointed out on their approach. Firstly, the system requires the face of users to be visible in order to appropriately track them. Secondly, as the system relies on a predefined skin colour model, a degradation on the tracking performance might be expected when the illumination conditions differ significantly from the training ones [144]. A dynamic skin colour model [196] could have solved this problem.

Grest and Koch [91] developed a system able to detect and track a single-person using stereo vision. It allowed the user to navigate in a virtual environment by just walking through a room using virtual reality glasses. The user was detected using the face detector proposed by Viola and Jones [212]. Once the user was located, both a colour histogram of the face region and a colour histogram of the chest region were created. A particle filter was employed to estimate the user position in the next image. Stereo information was used to determine the real position of the person into the room. That real position, was employed to calculate the corresponding position in the virtual environment. In their work, the stereo processing was performed using the information gathered by different cameras located at different positions of the room. As in the work of Darrel et. al, the main limitation of this system is that requires the face of the user to be visible in the image in order to perform the tracking.

A very interesting method to locate and track multiple persons in stereo images using occupancy maps was presented by Harville [96]. Before the detection process took place, a model of the environment was created through a sophisticated image analysis method [97]. Once the background model was created, objects that do not belong to it were easily isolated. Then, both an occupancy map and a height map were created. The information from both maps was merged to detect people through the use of simple heuristics. Person tracking was performed using the Kalman filter combined with deformable templates. The stereoscopic system used in his work is located three meters above the ground in a fixed slanting position. The main draw back of his approach is that the simple heuristics employed for people detection may lead the system to incorrectly detect objects whose dimensions are similar to

human beings (as indicated by the author). This is the case of coat placed on a hanger or a big box introduced in the room.

Hayashi et al [98] presented a people detection and tracking system especially designed for video surveillance. The environment was modelled using an occupancy map an the stereo points were projected as *variable voxels* to deal with the stereo errors. People detection is performed in the occupancy map based on a simple heuristic that assumes that a person is a peak in the map whose height is into a normal range. As in the previous case, there can be expected many false positives because of the simplicity of the detection scheme.

Tanawongsuwan presents in [202] the initial steps for designing a robotic agent able to follow a person and recognising his/her gestures. His system employed a basic technique to find the arms and head of a person combining the information provided by a skin colour filter, a movement detector and depth. Once a person was located, Hidden Markov Models were used to recognise his/her gestures among a set of previously learnt ones. Nevertheless, the work does not deal in depth with the detection problem and assumes that there is a person if three skin coloured blobs (corresponding to head and hands) are found.

Chapter 3

A multi-agent system architecture for mobile robot navigation based on fuzzy and visual behaviours

This chapter explains the first contribution of this thesis to achieve our first goal: exploring the use of new mechanisms able to increase the autonomy level of mobile robots. We present a multi-agent system architecture (structured in three layers) that uses a concurrent activation of both visual and fuzzy behaviours to carry out the navigation task of a mobile robot in office-like environments. The higher layer receives orders from an external operator and generates a high level of abstraction plan consisting in the sequence of rooms and corridors to transverse and doors to cross to go to a desired place. This plan is decomposed in the middle layer in a sequence of skills that the robot is able to perform. These skills arise from the interaction of visual and fuzzy behaviours implemented in the lower layer of the architecture. Fuzzy behaviours take as input the information provided by the range sensors. They use fuzzy rules to manipulate the imprecision and vagueness of the sensor data allowing a safe navigation to accomplish each part of the plan. Visual behaviours gather the visual information from a single camera placed at the top of our robot and allow the system to know the location of the doors of the environment and indicate the straight direction to cross them. The camera is placed over a pan-tilt unit (PTU). It allows to move the camera independently

from the movements of the robot giving to vision an active role in our system. In order to ease the door detection, artificial landmarks with numerical information have been placed beside the doors. We assume that all doors are opened, because our robot cannot open them.

The use of artificial landmarks has been successfully used in the literature as a mean to provide extra information to a robot in order to aid the self-positioning process as well as for developing other tasks [58, 132]. Several designs of landmarks have been proposed depending on the purpose they are used for [117, 191] and even the best arrangement in order to get minimum-error position has been studied [203]. We propose a simple artificial landmark that is placed besides the doors to aid the robot to know where the doors of the environment are. The landmark is easy to distinguish from the environment and has digits inside to uniquely identify the door that represents. Fast algorithms have been required in order to achieve real-time visual processing. Neural networks have been trained for either detecting the landmark and for classifying the digits.

Following sections deal with the multi-agent system architecture. First, Section 3.1 briefly describes the hardware and then a general vision of the architecture of the system is given. Following, the agents of the system are explained in detail. In Section 3.2, the *Planner* agent, that plans according to the desired goal, is explained. Then *Monitor* agent, that transform the plan in a sequence of skills that the robot is able to do, is introduced in Section 3.3. Finally, both *Vision* agent and *Navigation* agent are explained in Sections 3.4 and 3.5 respectively. The former develops the visual processing and the latter performs the navigation. Section 3.6 shows the experimental results by an example of the whole system running in a Nomad 200 mobile robot. Finally we offer some final remarks in Section 3.7.

3.1 Multi-agent system architecture

3.1.1 Hardware description

Our robot is a Nomad 200 that has been updated adding new devices. First of all we needed to increase the computational power to perform visual processing in real time. Therefore we

have added a mobile computer Pentium III running at 1.100 MHz that communicates with the robot via Ethernet at 10 Mbps using TCP/IP. There is a computer in the robot that is used only to run a daemon that receives data from the sensors and issues movement commands to the motors. The rest of the system runs in the mobile computer. To perform the visual processing, a pan-tilt unit (PTU) and a Sony EVI-401 video camera have been added. Both are connected to the mobile computer. The PTU can move 139 degrees to each side in the horizontal axis, while in the vertical axis it can move 47 degrees down-side and 31 degrees up-side. The camera is analogical so it has been connected to a PCMCIA frame-grabber obtaining images at a frame rate of 25 fps. In Figure 3.1 there is shown the final appearance of the robot.

Figure 3.1: Robot appearance.

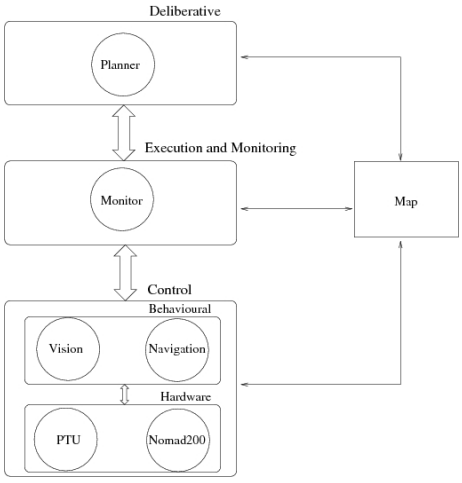


Figure 3.2: Multi-agent system architecture.

3.1.2 Multi-agent system overview

In this work a multi-agent system based on behaviours for controlling the navigation task of a mobile robot is presented. An *agent* is a software process aimed to get or keep a goal implicit in its own design. Our system has been designed as a set of agents organised in a three layer architecture with both reactive and deliberative capabilities in an hybrid architecture [187]. This methodology allows great robustness and an easy expansion of the system either adding

new capabilities to the existing agents or adding new agents to the system. In Figure 3.2 the structure of the system is shown.

We have identified two different navigation tasks. The first one consists in navigating into a room to go to a certain place in it, and the second one consists in travelling from one room to another into the building. The first one has been developed in previous works [1, 3, 4] using fuzzy behaviours to move the robot following walls and a fuzzy perceptual model to build a topological map that contains some metric information of the environment. Using the perceptual model and the topological map the robot can navigate and determines the wall it is following and the corners of the room that are passed.

We want to focus this work in the ability to perform the second navigation task. When the robot must go from one room to another into a building a plan is created. It consists in a sequence of rooms and corridors to transverse and doors to cross to reach the desired room. This sequence is a high level of abstraction plan that turn into sub-goals for the system. For this purpose low precision about the position of the robot is required. The plan is not highly detailed but expressed at a high level of abstraction forcing the robot to trust in the abilities of the system to accomplish those sub-goals. The different agents of the system collaborate distributing the whole navigation task establishing communication between them whenever it is necessary. Both visual and range sensor information are used in order to perceive the environment.

At the higher level is *Planner* agent, a deliberative agent whose main task is to receive the order from an external human operator to go to a certain room in its environment. This agent creates the appropriate navigation plan to safely navigate in the environment. The plan is made using a topological map of the environment that is available for the use of all the agents of the system.

When Planner agent creates the navigation plan, it is passed to the *Execution and Monitoring* layer. Its unique agent, *Monitor*, can develop several *skills* to achieve the sequence of sub-goals of the navigation plan. An *skill* is understood as the ability to accomplish a certain sub-goal and for this purpose it is used the activation of several behaviours either in a sequential or concurrent manner. These behaviours are implemented in the lower level,

the *Execution* level, and *Monitor* agent activates them to accomplish a certain part of the plan while monitors the execution of these behaviours to check a possible error. It is also in charge of the localisation of the robot during its movement on the environment.

The Execution level is divided into two levels: the *Behavioural* level and the *Hardware* level. In the Behavioural level, there are agents that implement behaviours. A *behaviour* can be seen as a relation between the inputs of the sensors and the actuators to achieve a desired goal. This concept regards to a lower level way of acting that directly relates environmental information received from sensor with the immediate actions that the robot takes. In this layer, *Navigation* agent implements several fuzzy behaviours that allow the robot to navigate safely in the environment. It uses information provided by range sensors in order to: avoid obstacles, approach to a door, cross a door and to navigate in a corridor among others abilities. There is also *Vision* agent that implements visual behaviours to process the images captured from the environment. It can detect a desired landmark and move the PTU in order to fixate it indicating the straight direction to the door.

In the Hardware level there are agents that act as wrappers for the real hardware of the system. *Nomad200* agent establishes a communication via TCP/IP with a daemon running in the robot. Any agent that desires to send a command to the robot has to do it through this agent. The agent redirects the command in an appropriate format to the daemon allowing a concurrent use of the robot. The commands that can be sent allow to configure the speed and acceleration of the robot, to obtain the data from the ultrasound and infrared sensors and to move it. *PTU* agent allows the use of the PTU to the rest of the agent community. The main advantage of using this agent is the possibility of queueing the PTU movements in a remote way besides controlling the concurrent use of the PTU. The agent gives the service of moving the PTU and to inform about the exact position of both axes (pan and tilt) by message-passing. The camera has not been wrapped by an agent because the transference of the images on the net would speed down the work of the *Vision* agent. In the following sections all agents except for the hardware ones are explained in detail.

3.2 Planner Agent

It is a deliberative agent that acts as a bridge between an human operator and the middle layer of the system. As we have previously commented we have identified two separated navigation tasks and we focus this work on the ability to navigate from one room to another into an office-like environment. When an human operator commands the robot to go to a certain room, this agent creates the navigation plan as a sequence of rooms and corridors to transverse and doors to cross. This sequence turns out to be sub-goals that are passed to Monitor agent which is able to develop several skills to accomplish them.

To create this navigation plan the agent makes use of a map of the environment that is in a shared representation structure available for all the agents of the system. Formally, the topological map consist in a graph $G = (V, E)$, where $V = \{v_1, ..., v_N\}$ is the set of N nodes, and $E = \{e_{ij} = (v_i, v_j), i = 1...N, j = 1...N\}$, is the set of M edges. The nodes of this graph correspond to distinguished places of the environment and the edges connect pair of this places. A fuzzy perception system [4] is able to classify the distinguished places according to its morphological characteristics in: corners (c), doors (d), hallways (h), end of corridor (ec) and a default object type corresponding to long irregular boundary (i). Then each edge of the graph represents either a wall, a corridor, an edge that crosses a door or a link between an irregular type node and any other kind of node, and it expresses a transition between two distinguished places. The map that can either be manually supplied or autonomously created in an exploration phase. The autonomous creation of the map is based on graph node saturation inspired on [175]. In Figure 3.3 there is the top view of a typical floor and in Figure 3.4 its corresponding topological map is depicted. Each door has an unique number in order to identify it in the map. This number is printed on the landmark placed beside the door and stored in the nodes that represent doors.

Using this map the agent can determine the path from an origin room to a destination one as a sequence of rooms, corridors and doors using the A^* algorithm. This is a high level of abstraction plan since it does not specify the movements of the robot in detail but a sequence of places that the robot has to reach. Lets imagine we wished our robot to go from

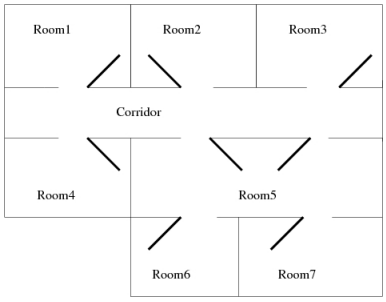


Figure 3.3: Environment.

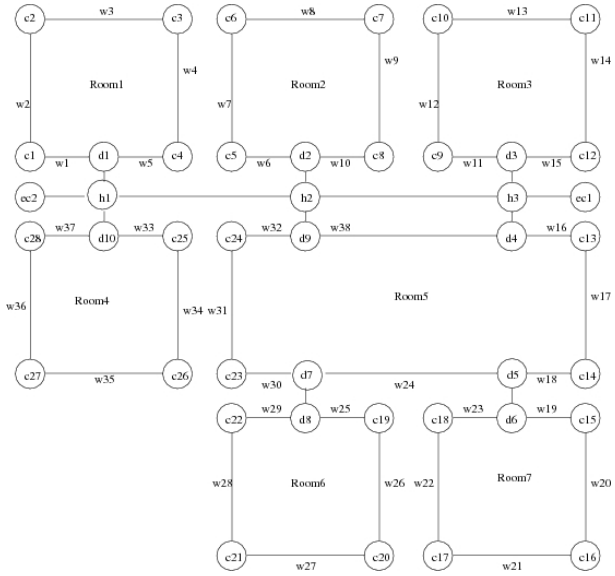


Figure 3.4: Map of the environment.

Room1 to *Room2* in the map shown in Figure 3.4. The plan would consist in the following path *Room1*→ *d1* →*h1*→*h2*→ *d2* →*Room2*. These sub-goals are: find the door d1, cross it, travel along the corridor looking for door d2 and finally cross d2 to enter Room2.

It is possible that during the execution of the navigation plan an error occurs. In that case Monitor agent fixes the error if it is possible. Otherwise, the agent informs about the error to Planner agent that report the incident to the human operator.

3.3 Monitor agent

This agent is mainly in charge of the execution of the appropriate skill to accomplish each part of the plan and also monitors the execution of the behaviours to check if the navigation plan is being achieved. As we explained above, a skill consists in the ability to achieve a certain sub-goal and for this purpose a concurrent or sequential activation of behaviours is used. Skills can be easily added to the system either incorporating new behaviours to the existing agents or new behavioural agents.

Planner agent passes the navigation plan to this agent that transform it in terms of the available skills. At the moment the plan is translated to the four possible skills: *find door in*

room, *find door in corridor*, *approach to door* and *cross door*. Therefore the navigation plan explained above would be translated as: *Find door d1 in room→ Approach door d1→ Cross door d1→ Find door d2 in corridor→ Cross door d2*.

These are the four skills explained more in detail:

a) Find door in room: Every time the robot needs to find a door, the searching method depends on whether the robot is into a room or in a corridor. It is more usual to have enough visibility to distinguish the landmark placed beside the door when the robot is into a room than when it is in a corridor. Therefore we distinguish between the search of the landmark in a room and in a corridor. In any case, Monitor agent leads on Vision agent to detect the required landmark as well as on Navigation agent to move safely in its environment.

In order to find a door in a room, we have opted for searching the landmark with the robot stopped turning over itself (static search). The idea is that the robot turns the turret while the visual behaviour *Landmark detection* (explained in Section 3.4 with the rest of visual behaviours) searches the desired landmark in the images captured. Two conditions are required in order to achieve that, first of all, the landmark must be visible from its current location. And second, the distance from the robot to the landmark must allow to recognise it. Regarding the latter case, we have tested the vision system and adjusted it to work in typical indoor environment rooms. In our experimentation the vision system is able to identify the door at distances ranging from 1 meter to 5 meters. In case of failure the robot starts an exploration process using both visual and range information into the room moving parallel to the walls of the room while rounds it. The camera is pointed towards the wall that the robot follows and analyses the images captured looking for landmarks. This skill is performed by the concurrent use of the fuzzy behaviours *Follow wall* and *Avoid obstacle* and the visual behaviour *Landmark detection*. All fuzzy behaviours are explained later in Section 3.5

A key point to see a landmark is to set the camera inclination correctly to make it appears in the captured images. Lets call α the camera inclination angle, D the distance from the camera to the wall, h_m the distance from the landmark to the floor and h_c the distance from the camera to the floor. Figure 3.5 shows graphically the relation between these variables

showing that α can be calculated using the Equation 3.1. To measure D we use the average of several values provided by the three frontal sonar sensors. Although it is an approximated method submitted to vagueness, it has been tested experimentally obtaining good results for the calculation of α.

$$\alpha = atan(\frac{h_m - h_c}{D}). \tag{3.1}$$

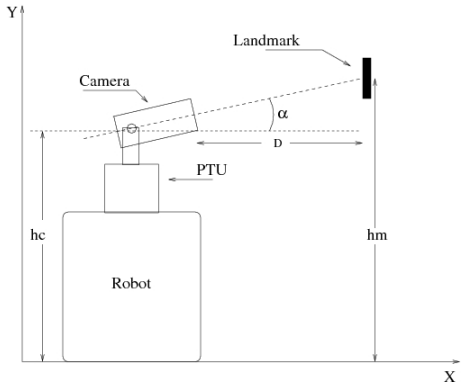


Figure 3.5: Appropriate angle to detect the landmark.

b) Find door in corridor: Whenever the robot leaves a room and enters in a corridor, consults the map to know in what direction it has to go to reach the next room (left or right). The robot orientates its wheels to face the corridor in the correct direction and the turret to point towards the wall where the door is using the fuzzy behaviour *Orientate wheels & turret*. When the robot is correctly placed, a behaviour that moves the robot along the corridor (*Follow corridor*) is activated to maintain the robot centred in the corridor while *Avoid obstacles* allows a safe navigation. At the same time, *Landmark detection* behaviour is activated searching for the landmarks in the appropriate wall.

c) Approach to door: Once the desired door is found, the robot moves towards it. The use of this skill is only necessary when the landmark has been located in a static search, since if it is located either during a search in movement or in a corridor, the robot is already placed

near the door. To achieve this goal, the visual behaviour *Landmark fixation* is activated in order to keep the landmark in the field of vision indicating to the robot the straight way to the door. At the same time the fuzzy behaviours *Go to door* and *Avoid obstacle* are activated in order to lead the robot to the door without colliding with obstacles. It is possible that while approaching to the landmark the robot changes its straight way due to an obstacle. In that case, the Vision agent could loose the landmark from the field of vision loosing the appropriate way to the door when the obstacle is avoided. If so, a visual search using the behaviour *Find lost landmark* is done in the area where the landmark was found last time. In case of not finding it, the searching process in the room starts again.

During the approach, information about the distance from the camera to the landmark is provided by Vision agent. This information is used to decide when the robot is close enough to the door in order to develop the appropriate skill to cross it. The skill *Approach to door* stops when the robot is at an appropriate distance to ease the work of the skill that crosses the door.

d) Cross door: This skill allows the robot to cross a door when it is close enough. The first step is to place the robot in a favourable position to cross the door. For that purpose, *Centre door* moves the robot to place it in front of the door using information gathered from the map and the range sensors. Then, two fuzzy behaviours are used: *Go through door* and *Avoid obstacle*. The first one creates the appropriate path to cross the door detecting the position of its frames while the second one avoids possible obstacles in the way.

Using these skills appropriately, the robot is able to go from its current location to a destination room. While the behaviours are running there is a bidirectional flow of data to inform about the status of each process. For example, Vision agent would inform if the landmark that the robot is approaching to has been lost from the field of vision to make Monitor agent activate the behaviour *Find lost landmark*.

During the execution of the plan, Monitor agent keeps information about its position in the map to re-plan if it is necessary. This agent gets information about the doors that are

passed in the way to the next door in the plan and this information is used to compute the position of the robot. In this way the robot is able to know in which room it is or in which node of the corridor is placed.

Monitor agent is also in charge of detecting possible errors during the execution of the navigation plan. The errors that can be detected by our system can be divided into recoverable and non-recoverable errors. Non-recoverable errors are those that can not be repaired by the system itself, therefore they are reported to the upper layer that informs to the human operator. These errors occurs, for example, when the robot does not reach the door while it is approaching to it because there is an unavoidable obstacle in its way or when a certain landmark is not found in a room. Nevertheless there are errors that can be detected and recovered. If for example the search of the landmark is done in a corridor and it is detected that it has been passed the robot repeats the search process in the opposite direction.

3.4 Vision Agent

Vision agent is in charge of detecting the desired landmark and keep it in the field of vision despite the movements of the robot towards it. The agent can detect the set of landmarks in a captured image and inform if a certain landmark is in it. When the desired landmark needed to accomplish the current part of the plan is identified, the agent can fixate it using the PTU indicating the straight way to the door. This agent can also measure the distance between the landmark and the camera indicating if the robot is close enough to the door to start the appropriate skill to cross it.

This agent implements three behaviours, the first one is called *Landmark detection* and consists in detecting if a certain landmark is present in the image captured at the current position of the robot. The second one is called *Landmark fixation*, it consists in keeping the landmark always in the field of vision (moving the PTU to centre the landmark) despite the movements of the robot. Finally the behaviour *Find lost landmark* searches for the landmark that was lost while the previous behaviour was running.

The landmark employed has been designed taking into account the following principles:

it must be easy to distinguish from the environment, it must include information to uniquely identify the door, and it must be cheap and simple. The landmark finally developed is shown in Figure 3.6. It has an external black border that makes it easy to distinguish from the background and has some numbers inside to identify the door. The landmarks have been designed using a simple text editor and it fits in an A4 paper.

Figure 3.6: Designed landmark.

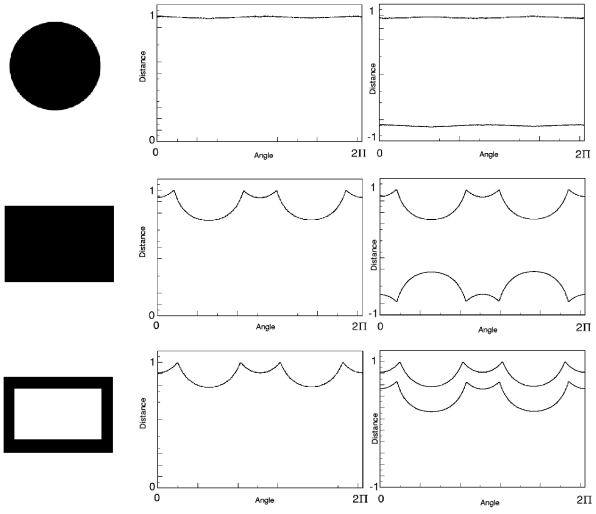


Figure 3.7: Examples of the signature and double signature applied to three shapes.

3.4.1 Landmark detection

The landmark detection is performed in two stages: the detection of the external border and then the determination of the number inside the landmark. The process starts when an image from the environment is captured. A segmentation process is performed to separate the objects that could be a landmark from the rest of the objects of the environment. An automatic thresholding method is needed due to the illumination changes that occurs in real environments. We have opted for the method proposed by Otsu [173] that selects a threshold value based on the illumination in order to minimise the entropy in the result image. This method has shown to give good results in our experiments and is not time-consuming.

Next step consists in detecting the external border of the landmark. Hence, every object

present in the binarized image is analysed to know if is a landmark or not. The first attempt was to use the *signature* as descriptor [90]. It is a function $f(\phi)$ that represents the contour measuring the distance from the centre of the object to the border point that is in a line traced with a certain angle ϕ ranging in $[0, 2\pi]$. However, signature does not allow to detect holes in objects so all rectangular objects have a similar representation. Instead of signature, we have developed a variant of the method that has been denominated *double signature*. It consists in tracing lines form the centre to the external border of the object for angles ϕ in the range $[0, 2\pi]$. Each line is scanned starting from the most external border point that is in the angle ϕ. The scan is performed in direction to the centre, and the distances from the centre to the two first border points found are measured. The distance is considered negative if the point is found after pass the centre in the scanning process. In order to make the representation invariant to scale all the distances are normalised to the maximum one. Figure 3.7 shows three shapes (left column) and either the signature (centre column) and double signature (right column) of them are represented. This method brings two main advantages. First, it has been proved to be more robust in our experiments for detecting the landmarks using several classification methods, and second it allows the description of holes in the objects that it represents.

Double signature distances are stored in an array to represent the objects that appear in the image after segmentation. In order to learn the double signature of the rectangular black border a neural network has been used. The network has been trained using the back-propagation algorithm and the best network found has 76 inputs a hidden layer with 8 neurons and 2 outputs. The training and validation set was created using 135 images where the landmark appears from different points of view. From each image a positive example is taken (the landmark in the image) and the rest of objects are considered negative examples. In this way the total number of patterns extracted is 5281, using the 70% for training and the rest for validation. The neural network obtains a 100% success percentage over the training set and 99.65% in validation.

Once the landmark has been found it is necessary to determine the number that is inside. To do this we analyse each object (greater than a certain number of pixels) into the detected

rectangular border and classify it. Each digit is adjusted to a $16x16$ image and for each one the following features are extracted:

1. *Horizontal and vertical Laplacian*. It consists in creating a function with the number of pixels occupied by the digit in every row and column, then obtaining the Laplacian of this function and escalating it to the range $[-1, 1]$.

2. *Block Sum*. Consist in the creation of $2x2$ blocks and counting the number of pixels occupied by the digit in each block. Finally the value is normalised dividing by 4.

A vector of 74 elements describing the digit is created using these measures. A neural network has been created and trained to classify the digits. In our experiments, digits have always been correctly classified.

3.4.2 Landmark fixation

Once the landmark has been located, it is necessary to fixate it. The objective is to have the landmark always placed into the next image to show the straight direction to the door. This behaviour uses the PTU to centre the landmark in the image. The unique information needed to centre the landmark in the field of vision is the pixel distance from the centre of the landmark in the image to the centre of the image. Then it is necessary to establish the correspondence that leads the PTU to centre the landmark in the image.

We wish to obtain a fast lace of control to avoid loosing the landmark. We assume that the nearest landmark to the centre in the current image is the desired landmark. It is reasonable if the landmark has not been loosed in a previous loop and the control lace is fast enough. In this case the computing time can be reduced by detecting only the external border and not processing the digits. Smaller images can be used for this process because the precision required is less important.

In order to determine how many degrees the PTU must be turn in both axis two regression lines has been calculated. They indicate the necessary angle increment based on the pixel distance from the centre of the image to the pixel that it is desired to be centred. The lines calculated based on real values are:

$$pan\_degrees = -25.99 \cdot \frac{2IncPixX}{Width} + 0.05; \quad tilt\_degrees = 17.63 \cdot \frac{2IncPixY}{Height} + 0.29$$

Where $IncPixX$,$IncPixY$ represent the pixels distance from the centre of the landmark in the image to the centre of the image in both axis X, Y. $Width$ and $Height$ are the width and height of the image respectively.

Distance measurement

Monitor agent needs to know the distance between the robot and the landmark to determine if it is close enough to the door. If so, Monitor activates the appropriate skill to cross the door and this agent stops the fixation process. It is possible to measure the distance using the ultrasound, but it submits the system to the typical problems of these sensors, like false reflections due to the furniture near the door. Eventually we have used vision to perform the measures because it have provided better results than ultrasound in our experimentation.

Assuming that the size of the landmark is known and fixed for all landmarks, we can approach the distance to the door measuring the size in pixels of the vertical projection of the landmark in the image. The closer the mark is to the robot, the bigger the vertical projection is, and vice versa. The horizontal projection is not so appropriate because is more affected by perspective deformations due to the wide range of position in which it can be seen. While the range of angles under which the landmark can be seen in the horizontal axis is wide (when the robot rounds the door), in the vertical axis it is limited by the height of the camera and the height at which the landmark is placed. We have extracted a regression curve by least-squares using real values to approximate the distance D. The curve obtained is:

$$D = 763.02 - 6652.21x + 23403.03x^2 - 29177.50x^3$$

Where $x = \frac{NoPixels}{Height}$, $NoPixels$ is the number of pixels of the vertical projection of the landmark in the image and $Height$ the height of the image. Therefore, x is in the range $[0, 1]$, so the curve is valid for different image sizes.

3.4.3 Find lost landmark

This behaviour is activated when the landmark is lost from the field of vision because of the movements of the robot. In this case, the robot looses the straight direction towards the landmark and, instead of performing a full search in the room, it is preferable to search the landmark in the direction where the robot was looking at. For that, the agent turns the PTU to the left and right side and performs a visual search of it in this area. This process is faster than a full search because the area covered is smaller and usually succeed. But if it does not succeed, a complete visual search in the room is ordered by Monitor agent.

3.5 Navigation Agent

As commented in Section 3.1.2, this agent is in charge of carrying out the navigation of the robot safely in the environment. It relates the perception of the range sensors to the orders which must be sent to the actuators according to the objective that must be executed in each moment. For this reason, it uses different behaviours designed using techniques taken from Fuzzy Control Theory [60].

The use of these fuzzy behaviours allows the robot to control the actuators despite the presence of noise in the data sensors that cause inaccuracy in the measures and uncertainty about the information used. Below, we explain how the behaviour to guide the robot to the door has been designed whereas for the rest of the behaviours only their function is commented.

a) Go to door. This behaviour permits the robot to reach the landmark of the indicated door orientating the wheels and turret in the direction that is indicated by the PTU. When the behaviour is activated by Monitor agent, the desired landmark has already been found and Vision agent makes the PTU point towards it. It is necessary to indicate that the hardware of the robot allows to move the turret independently from the wheels. It gives the robot the possibility of maintaining the turret orientated towards the door when the wheels are turn in order to avoid some obstacle. In this way the work of the Vision agent is helped because less

movements of the PTU are required when avoiding an obstacle. Therefore, two fuzzy rule set have designed. One to orientate the turret and another to set the orientation of the wheels.

- Rule set for orientating the turret. The goal is to keep the turret oriented towards the landmark to ease the work of Vision agent. For this reason, according to the angle of the PTU and the present situation of the turret, the angle that the turret must be turned to remain lined up with the PTU is calculated. This angle, denoted as *Angle Move Turret* (*AMT*), is used as input for the rules to control the *Turret Rotation Speed* (*TRS*). Figure 3.8 shows the labels of *AMT* and Figure 3.9 shows the labels for *TRS*. In Table 3.1(a) the rule set to achieve the desired goal is shown.

- Rule set to orientate the wheels. The idea is similar to the previous one, to maintain the wheels of the robot lined up with the PTU to lead the robot towards the landmark. In this case the variables to control are the *Wheels Rotation Speed* (*WRS*) and the *Translation Speed* (*TS*) of the wheels. Thus, the necessary angle to line up the PTU and the wheels of the robot, *Angle Move Wheels (AMW)*, is calculated and used for controlling the output variables using an appropriate rule set. The labels of *WRS* and *AMW* are similar to the labels of *TRS* and *AMT* respectively and Table 3.1(b) shows the rule set for this variable. In order to ease the alignment, another variable (ΔTS) is used to adjust the variable *TS*. ΔTS is added to *TS* making the robot to decrease its speed up to a certain limit when it is not lined up with the PTU and vice versa. The labels for this variable are shown in Figure 3.10 and the rule set for this variable is in Table 3.1(c).

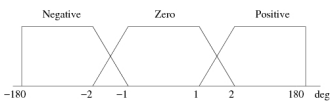


Figure 3.8: Labels of variable *AMT*.

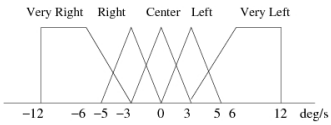


Figure 3.9: Labels of variable *TRS*.

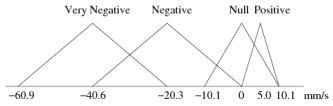


Figure 3.10: Labels of variable ΔTS.

b) Avoid obstacles. This behaviour allows the robot to avoid an obstacle that is placed in its way. The input variables that have been taken into account are: *Frontal Distance*, *Right-*

A M T	TRS
NEG	Right
ZE	Centre
POS	Left

(a)

A M W	WRS
NEG	Right
ZE	Centre
POS	Left

(b)

A M W	ΔTS
NEG	Negative
ZE	Positive
POS	Negative

(c)

Table 3.1: Rules set to control the variables (a) *TRS* (b) *WRS* (c) ΔTS

Left Distance, *Right-Left Difference Distance* (all of them detected by ultrasound) and *TS*. The variables to control are *WRS* and *TS*. The idea that leads the design of this behaviour consists in the fact that if the obstacle is not in front of the robot, it tends to turn a little either to left or right, depending of what is safer. When the obstacle is just in front of the robot, or when it has approached too much to it, the robot turns towards a safe place. In any case, when an obstacle is detected the speed decreases.

c) Follow wall. The objective of this behaviour is to allow the robot to move around a room following the wall outline. The input variables for the behaviour are *Wall Distance* and *Incidence Angle* to the wall while the output variable is *WRS*.

d) Follow corridor. This behaviour allows the robot to go along a corridor, moving forward in the centre of it. The input variables are *Right-Left Distance Difference* in regard to the corridor walls and *Right-Left Incidence Angle Difference*. The aim is to control *WRS* to centre the robot as well as *TS* that increases if the robot is in the centre of the corridor and decreases otherwise.

f) Centre door. This behaviour places the robot in a favourable position to ease the work of *Go through door* behaviour. Crossing a door is a difficult task so the behaviour places the robot in front of the door and orientates it towards the gap to make the trajectory to cross the door as straight as possible. Some times it is possible to detect the gap of the door using

range sensor information. In this case the behaviour can directly orientate the robot to face the door. In some other cases the gap can not be detected using range sensor information and the map must be consulted. It usually happens when the door is in a corridor because the landmark is found either before or after passing the gap of the door. This agent uses as input variables to detect the gap the *Frontal-Lateral Difference Distance* and the output variables are *TS*, *WRS* and *TRS*. If the robot is in a corridor consults the map to determinate if it has to move either forward of backward to find the gap of the door. The inputs variables in this case are the *Frontal-Lateral Difference Distance* to detect the gap and the outputs are *TS* and *WRS*.

g) Go through door. It allows the robot to go trough a door to enter or leave a room. A temporary model of the door has been used with the aim of calculating a straight line that crosses through the gap of the door. The robot takes as reference this trajectory but because this process is not free of noise, it is possible that the trajectory must be readjusted according to the success or failure of the behaviour. The input variables are: *Distance to trajectory* calculated and *Deviation Angle* in relation to the trajectory. The variable *WRS* has been considered as output variable while *TS* decreases to very low values.

h) Orientate wheels & turret. When the robot has to navigate in a corridor it is necessary to orientate the wheels to face it appropriately. Likewise the turret of the robot is oriented to point towards the wall in which the desired landmark is placed using information from the topological map. The inputs are *Angles Move Wheels & Turret* that show how much the robot must be turned to reach the desired position, and the output variables are *WRS* and the Turret Rotation Speed *TRS*.

Further details about the design methodology of the fuzzy behaviours can be seen in [4]

3.6 Experimental Results

This multi-agent system is structured in three levels. It has been implemented and tested using a mobile robot Nomad 200 modernised with a vision system, and placed in the real office-like environment represented in Figure 3.11. We explain the concurrent running of the whole system with a real example. The robot is initially situated in *room 0* and has to get *room 2*, travelling through a corridor. The first step is to supply the robot with the initial and target point. With that information the Planner agent calculates the following path according to the topological map : Room 0 $\rightarrow d0 \rightarrow h2 \rightarrow h3 \rightarrow h4 \rightarrow d2 \rightarrow$ Room 2. These steps represent sub-goals to accomplish in order to achieve the main goal and Monitor translates the plan into the possible skills as : *Find door* $d0$ *in room*$\rightarrow$ *Approach door* $d0 \rightarrow$ *Cross door* $d0 \rightarrow$ *Find door* $d2$ *in corridor* $\rightarrow$ *Cross door d2*. As it can be appreciated, it is a high level of abstraction plan where no trajectory is specified to the robot, but rather a set of sub-goals to reach. The plan is executed in the stages explained below.

a) Find door $d0$ *in room*: According to the path previously obtained, the robot searches for the landmark that identifies the door $d0$ by means of the static search. Figure 3.12 shows two images obtained by the robot in the static search process. In the second image of this Figure the landmark with the digits 00 appears on the left side of the door. In another situation, the landmark could be hidden by an obstacle, and thus be invisible to the robot. In this case, after completing the static search without success (after a 360 degrees turn) the robot would start a search moving parallel to the walls of the room avoiding possible obstacles. Figure 3.13 shows the path followed by the robot in this situation.

b) Approach door $d0$: If the robot finds the landmark 00 in the static search, it approaches to the landmark orientating the wheels and the turret in the direction indicated by the PTU. In Figure 3.14 there is shown the path followed by the robot while approaching to the landmark as well as several images taken in the process. This phase finishes when the robot is at an appropriate distance from the door. It must be noticed that this phase does not take place when the landmark is located by means of the process of searching in movement, since in

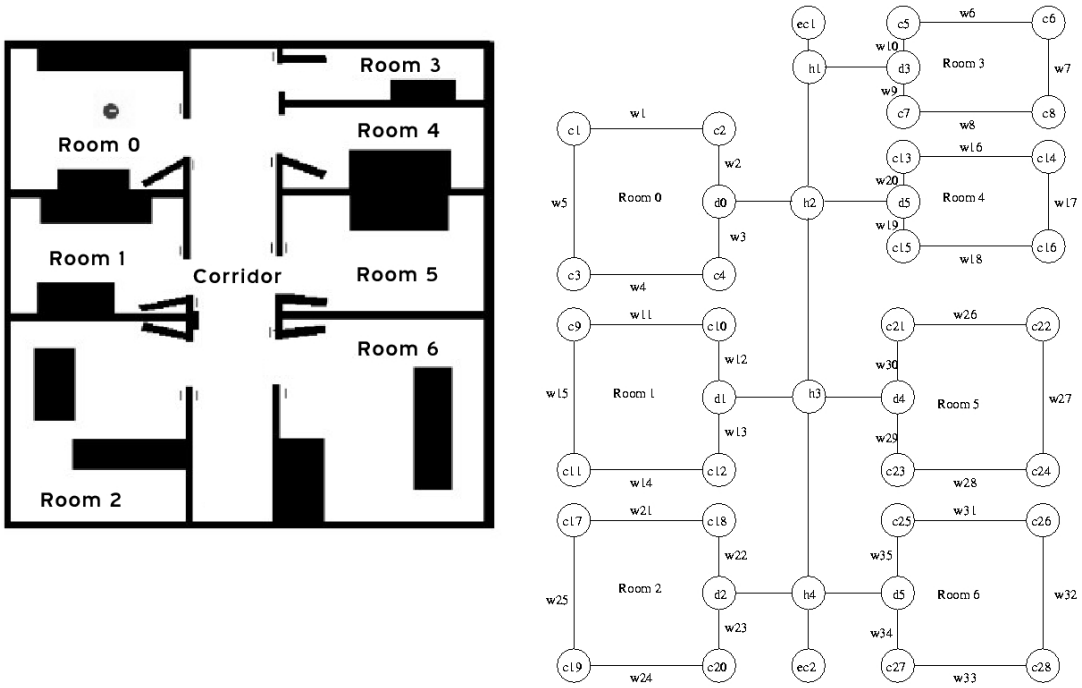


Figure 3.11: Rooms map and topological map of the real environment.

this case the robot approaches the landmark in the searching process.

c) Cross door $d0$: When the robot is placed near the door, *Monitor* agent performs the skill *Cross door*. At first, the robot detects the gap of the door and orientates itself appropriately (using the behaviour *Centre door*) to ease the work of the behaviour *Go through door*. The latter behaviour calculates the appropriate path to cross the door and moves the robot through the gap. The path of the robot and some images taken while the robot crosses the door are shown in Figure 3.15.

d) Find door $d2$ *in corridor*: Once the door has been crossed and the robot is in the corridor, it must navigate lined up in the centre to find the destination room. For this reason, in a first phase the robot orientates the wheels to face the corridor and the turret to facilitate the visual search of the destination door. Then the appropriate behaviours move the robot in

Figure 3.12: Static search of the landmark 00.

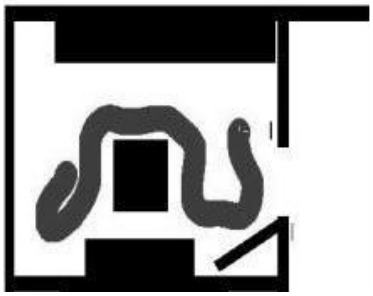


Figure 3.13: Search in movement of the landmark 00.

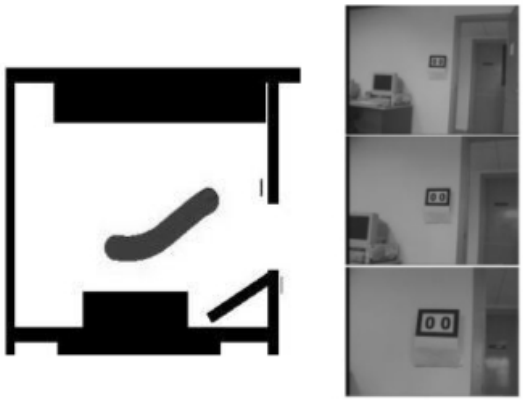


Figure 3.14: Approaching to the landmark 00.

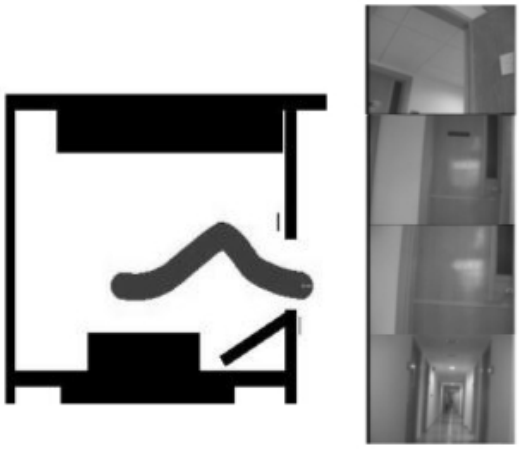


Figure 3.15: Crossing the door $d0$.

the centre of the corridor avoiding possible obstacles while images of the wall are captured and analysed to detect landmark 02. Figure 3.16 shows how the robot searches and finds the landmark of the door $d2$ navigating along the corridor.

e) Cross door $d2$: The skill *Cross door* is executed again and the robot achieve the navigation plan. When the landmark is found the robot moves backwards to place itself in front of the door in order to ease the work of the behaviour that crosses it. The path that the robot has followed and some images taken in the process are shown in Figure 3.17.

The images shown in the previous figures have been taken by the robot in a real execution

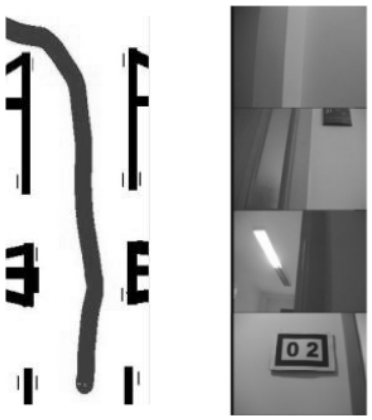


Figure 3.16: Along the corridor searching for door $d2$.

Figure 3.17: Crossing door $d2$.

of the system. This experiments and others have been filmed and can be downloaded at `http://decsai.ugr.es/~salinas/videosrobotmarcas`

3.7 Final Remarks

In this chapter, we have presented a multi-agent system that employs both visual information and range information to control the navigation task of a mobile robot in office-like environments. The system is organised in a three-layer architecture consisting of several agents that share the responsibility to achieve the navigation task. This is performed by means of four skills: a) detect doors in rooms, b) detect doors in corridors, c) lead the robot towards a door and d) cross a door. The system is flexible to the addition of new agents, skills and behaviours and it can be adapted to other types of robots because of the standard hardware employed in this work.

The environment is represented using a topological map where nodes represent distinguished places and arcs join them. Besides each door an artificial landmark has been placed consisting in a rectangular border with digits inside to aid the robot to identify the door. The landmarks designed are easy to make with a normal text editor fitting in an A4 paper and their placement can remain discreetly in the environment. The landmarks and the digits are

recognised using trained neuronal networks achieving high success rates.

A high level of abstraction agent generates the plan and passes it to the middle layer that translate it into a sequence of available skills. A visual agent that uses fast algorithm for image processing is able to maintain the camera focused on the landmark using the PTU to indicate the straight direction to the door. A navigation agent that uses range information implements fuzzy behaviours to manage the underlying vagueness and uncertainty and allows to move the robot safely in the environment. It can guide the robot towards the landmark of the door, avoid the obstacles, follow walls, cross doors and move the robot along a corridor. The experiments carried out in our office-like environment show that the system is able to safely navigate and accomplish the desired navigation plan.

The system employs an appropriate combination of vision and ultrasound sensors. Thus, the weakness of a sensorial subsystem are overcame by the strengths of the other. Altough the proposed system enables the robot to navigate autonomously , its main drawback is the necessity of altering the environment in order to place the artificial landmarks. Thus, a natural extension of this work is the development of a mechanism to detect doors themselves. Next chapter presents a novel approach to solve the door-detection problem in order to provide a higher degree of autonomy to the robot, in unmodified environments.

Finally, we must indicate that the work explained in this chapter has been published in the journal *Robotica* of the *Cambridge University Press* [161].

Chapter 4

Detection of doors using a visual fuzzy system for mobile robots

The use of artificial landmarks is an appropriate solution when it is possible to alter the environment. However, this option is not always feasible because of aesthetic or practical reasons. In that case, the ability of a robot to detect doors still being an advantage for navigating and/or map-building purposes because of doors are natural landmarks present in most indoor environments. This chapter presents a new approach to door-detection in indoor environments using computer vision and fuzzy logic. Doors are found in grey-level images by detecting the borders of their architraves. The architrave of a door is an external border, normally made of wood, that surrounds the door frame. It is comprised by two lateral bars and an upper bar.

In a first phase, edges are extracted using an edge detector [82] and then segments are also extracted using an improved version of the Hough Transform. A fuzzy system analyses features of the segments like length, direction and distance to check if they belong to architraves. The membership functions of the fuzzy system have been designed (based on expert knowledge) to detect rectangular doors under a wide range of situations.

A large database of images containing doors of our building, seen from different angles and distances, has been created to test the performance of the system. It has shown the ability to detect rectangular doors under heavy perspective deformations and it is fast enough to be

used for real-time applications in a mobile robot.

The use of fuzzy logic [222] to solve the problem brings several advantages. Firstly, it allows to transform expert knowledge into linguistic rules that are highly interpretable. Secondly, the knowledge introduced can be adapted to the particularities of an environment if desired by altering the rules. Another advantage of using fuzzy logic is the ability to combine the information provided by several sources of information managing uncertainty and vagueness.

The rest of this paper is structured in the following sections. Section 4.1 gives a brief introduction to the previous works in door-detection. Section 4.2 explains the visual fuzzy system. Section 4.3 shows the experiments carried out. Finally, Section 4.4 exposes some final remarks.

4.1 Previous works in door-detection

Door-detection using artificial vision has been performed using different techniques in the related literature. Cicirelli et. al. present a system based on neural networks for detecting doors [45]. The system is comprised by two neural networks, one detecting the upper corners of doors and another for the lateral and upper bars. Each net analyses a subwindow of size 18x18 pixels (for every pixel in the image) and decides if it belongs either to the corner of a door, to its bar or to none of them. The input of the net is the hue and saturation values of the subwindows. Each net has a total of 648 inputs neurons, a hidden layer and an output layer with a single neuron. The output of the neural networks is analysed to detect if there are present components of doors by an ad-hoc technique. The system is able to detect doors under partial occlusion and from different perspectives, but it has three main drawbacks. First, it requires a high computational effort (3 seconds in analysing an image); secondly, it can not detect fully opened doors; and finally, it is dependent on the colour of the door.

Using a functionality-based approach, a method for generic object recognition is presented in Ref. [118]. Doors are defined as inverted Us that can be crossed by people. A trinocular stereo vision system is used to detect segments in the images of the environment.

Then, segments are analysed to check if they accomplish a set of restrictions according to the doors of their environment. The main disadvantage of that approach is the high cost of the perceptual system.

Stoeter et. al. [200] present a method for detecting doors in corridors. In a first step, an image of the corridor is captured and the most prominent vertical stripes are selected. Doors are detected based on their expected dimensions taking into account the distance and direction of the walls. However, it is not very clear how the vertical stripes are classified in doors. Furthermore, the technique limits the detection of doors to corridors and does not consider deformations caused by changes of perspective.

In the approach developed by Monasterio et. al.[152], a simple technique for detecting doors is employed to aid an autonomous robot to cross them. Door-detection is performed fusing ultrasound and visual information. The visual process consists in detecting the lateral bars of doors by enhancing the main edges in the image. Finally, columns wider than 35 pixels are considered doors. The method does not take into account perspective deformations and it is only applicable when the robot is near the doors. Furthermore, the method has only been tested in a robot at 1 meter distance from the door and with angles not exceeding 30^o.

4.2 Visual Fuzzy Door-Detection

This section explains how doors are detected using a fuzzy system that examines segments in images. Our approach consists in looking for the segments that belongs to the architraves of doors. We have examined many images containing doors and we have identified three main cases of study that are depicted in Fig. 4.1. They have been formally defined by the three complex fuzzy concepts: *Inverted U* (IU), *Complete Architrave* (CA) and *Incomplete Architrave* (IA). Our fuzzy system analyses the membership degree in which the segments of an image belongs to these fuzzy concepts.

The fuzzy concept *Inverted U* is defined as a pair of vertical segments joined in its upper part to a horizontal segment (Fig. 4.1(a)). IU is usually seen when a door is closed and it has a unique colour. Nevertheless, this rectangular configuration of segments is also visible in

other elements of indoor environments (like cupboards or paintings). Therefore, the system defines another two more restrictive fuzzy concepts (CA and IA) that helps to distinguish doors from other objects of the environment.

Opened doors and closed doors whose leaf have a different colour from its architrave often projects a pair of instances of the fuzzy concept IU in the image. One corresponding to the limit between the architrave and the wall, and another caused by the grey-level difference between the architrave and the scene behind it (if its is opened) or its leaf (if it is closed). The fuzzy concept *Complete Architrave* (CA) is introduced in our system to model this situation (Fig. 4.1(b)).

Unfortunately, there are some cases (when the door is opened) that the two borders of an architrave are not visible because the leaf of the door hides one of its lateral bars. In that case it is usually possible to find one IU and *evidences* of the an incomplete IU around the former. This situation is represented by the fuzzy concept *Incomplete Architrave* (IA) (Fig. 4.1(c)).

Figure 4.1 shows the three cases with completely vertical and horizontal segments. However, our aim is to be able to detect architraves under perspective deformations that make them appear with different inclinations and lengths.

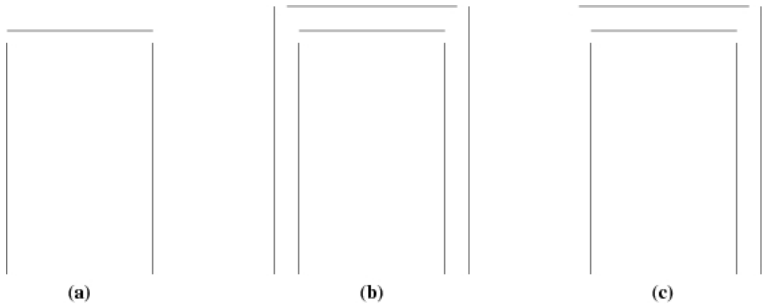


Figure 4.1: (a) IU (b) CA (c) IA

The system proposed in this work performs an incremental search of the above explained concepts among the segments detected in an image. In a first phase, segments are extracted and classified into vertical and horizontal. Then, the system looks for occurrences of the fuzzy concept IU among the segments. Then it checks whether any pair of previously detected IU belongs to the same architrave by employing the fuzzy concept CA. Finally, the

system looks for any occurrence of the fuzzy concept IA among the IUs not classified as CA.

Simpler fuzzy concepts like *vertical segment* or *parallel segments* are employed by the system in order to determine the membership degree in which the detected segments belongs to the three complex fuzzy concepts. Fuzzy logic allows to analyse the segments, taking into account the deformations caused by the perspective projection, and to manage the inaccuracy in the segment extraction.

The process starts by applying an improved version of the Canny edge extractor [82] that is able to integrate multiscale information in order to produce a very robust edge extraction across different illumination conditions. The result is a binary image I where pixels labelled as *true* belongs to edges in the original image. Let us assume for the sake of simplicity that I is a NxN squared image. The segments of the image are extracted from the edge pixels of I. Segment detection is an important aspect in many computer vision applications that has been tackled employing different approaches [41, 40, 75, 93]. As the development of a new technique for segment detection is out of the scope of this paper, we have employed the approach explained in Ref. [75]. The main advantages of this approach are that: (i) the computing time is saved by limiting the voting space to an small range given by the gradient direction of each pixel; and (ii) each pixel votes in its corresponding cell of the Hough space but also in a small window around it. Thus, making the detection more robust against small variations in the main direction of a line. Then, a clustering and repairing process is performed in order extract the s largest segments of the image and fill small gaps in them. As doors are usually tall objects, the segment of their architraves are normally among the s largest segments. In our experiments s is set to 50. Segment extraction usually takes 160 ms in our experiments (using a Pentium IV laptop computer at 2.4 Ghz for images of $320x280$ pixels) and is the most time-consuming step of the detection process.

Let us denote by $S = \{S^0, S^1, ..., S^n\}$ the set of segments extracted from I. Each segment S^i is defined by its two coordinates in the image plane $S^i = \{(p_0^i, p_1^i) \mid p^i = (x, y)\}$. Once the segments are extracted, the system starts its analysis to detect occurrences of the three complex fuzzy concepts.

4.2.1 Candidate segment selection

The number of segments extracted from I is usually very high. However, many of the segments might be irrelevant for our purposes (segments too small or in strange configurations). Therefore, S is analysed to select only these segments that, according to their appearace, have a high possibility of belonging to architraves. In the ideal case, the borders of architraves are projected on the camera plane as completely vertical and horizontal segments. Nevertheless, when architraves are seen from different distances and angles, the dimension and orientation of their bars is altered because of the perspective. Therefore, architraves are not always projected as rectangular objects but still being tall objects whose lateral bars are projected with a large and relatively vertical aspect. Regarding to its upper bar, they are usually projected in a wide range of lengths and orientations around the horizontal direction, and in upper positions of the image.

The fuzzy concepts *Vertical Segment* (VS) and *Horizontal Segment* (HS) are defined to evaluate the degree in which a segment S^i belongs to the lateral or upper part of an architrave respectively. Direction, length and height features of a segment S^i are used to calculate its membership degree to the fuzzy concepts VS and HS.

The first feature, direction, is measured using the linguistic variable $Direction(S^i)$ that has the three possible values (*horizontal, medium* and *vertical*) represented by the fuzzy sets shown in Fig. 4.2(a). The direction of S^i is considered *vertical* when it is perpendicular to the horizontal axis of the image and horizontal in the opposite case. The input value of $Direction(S^i)$ is $directionS(S^i)$ that is calculated as expressed in Eq. 4.1. The function $\arctan$ returns the angle of the segment in the range $[0, \frac{\pi}{2}]$. Then, this value is scaled to the range $[0, 1]$:

$$directionS(S^i) = \frac{2}{\pi} \arctan\left(\frac{|y_1^a - y_0^a|}{|x_1^a - x_0^a|}\right) \tag{4.1}$$

The second feature, length, is measured using the linguistic variable $LengthS(S^i)$ that has the three possible values represented by the fuzzy sets shown in Fig. 4.2(b). The input value of $LengthS(S^i)$ is $length(S^i)$ that represents the length of the segment S^i scaled to

the range $[0, 1]$. The value $length(S^i)$ is calculated by Eq. 4.2:

$$length(S^i) = \frac{dist(p_0^i, p_1^i)}{N\sqrt{2}}, \tag{4.2}$$

where $dist(p_0^i, p_1^i)$ is the Euclidean distance between the extreme points of S^i and the value $N\sqrt{2}$ is the maximum possible distance between two points in the image (the diagonal of the image).

Finally, the height of a segment in the image is measured using the linguistic variable $YPosition(S^i)$. It has the three possible values (*high, medium* and *low*) showed in Fig. 4.2(c). Its input value, $YPos(S^i)$, is the y-component of the segment centre scaled to range $[0, 1]$. It is calculated using Eq. 4.3. Low values of $YPos(S^i)$ indicates that the centre of S^i is in the upper part of the image and vice versa.

$$YPos(S^i) = \frac{y_0^i + y_1^i}{2N}, \tag{4.3}$$

The fuzzy concepts VS and HS are defined using the already explained linguistic variables. The fuzzy sets employed for the fuzzy concept VS are shown in Fig. 4.2(d). They are identical to the fuzzy sets employed for the fuzzy concept HS. The two rule bases shown in Table 4.1 are used to calculate the membership degree in which S^i belongs to the fuzzy concepts VS and HS (scaled in the range $[0, 1]$). Both values are calculated by a fuzzy inference process and its corresponding defuzzification. Let us denote by $VS(S^i)$ and $HS(S^i)$ the membership degree of the segment S^i to the fuzzy concepts VS and HS respectively.

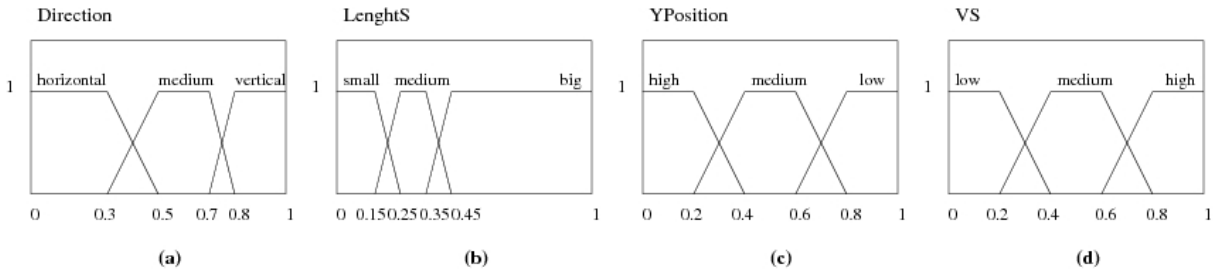


Figure 4.2: Linguistic variables for segment classification

The selection of the segments employed in the following phase is based on the membership degrees $VS(S^i)$ and $HS(S^i)$. Given a segment S^i, it is used in the following phase

IF			THEN	
Direction	LengthS	YPosition	VS(S^i)	HS(S^i)
horizontal	small	high	low	high
horizontal	medium	high	low	high
horizontal	big	high	low	high
medium	small	high	low	medium
medium	medium	high	low	high
medium	big	high	low	high
vertical	small		medium	low
vertical	medium		high	low
vertical	big		high	low

Table 4.1: Rule bases for classification of segments in vertical or horizontal

if either $VS(S^i)$ or $HS(S^i)$ exceed a certain threshold α_1. In this case, S^i is classified as vertical segment if $VS(S^i) > HS(S^i)$ and as horizontal segment otherwise. Let us denote the set of vertical segments selected for the following phase as $V = \{V^0, ..., V^n / VS(S^i) > \alpha_1 \wedge VS(S^i) > HS(S^i)\}$ and the set of horizontal segments as $H = \{H^0, ..., H^n / HS(S^i) > \alpha_1 \wedge VS(S^i) \leq HS(S^i)\}$.

Figure 4.3 shows an example of the classification process previously explained. Figure 4.3(a) shows an image with an architrave in it while Figure 4.3(b) shows all the segments detected in the former figure. Figures 4.3(c) and 4.3(d) show the vertical and horizontal segments selected for $\alpha_1 = 0.4$ while Fig. 4.3(e) and 4.3(f) show the vertical and horizontal segments selected for $\alpha_1 = 0.7$. Please notice that higher values of α_1 reduces the number of segments thus decreasing the computing time required for the following phases. However, high values of α_1 could also reject segments belonging to architraves. Therefore, the proper selection of α_1 is a non trivial problem that must be experimentally solved.

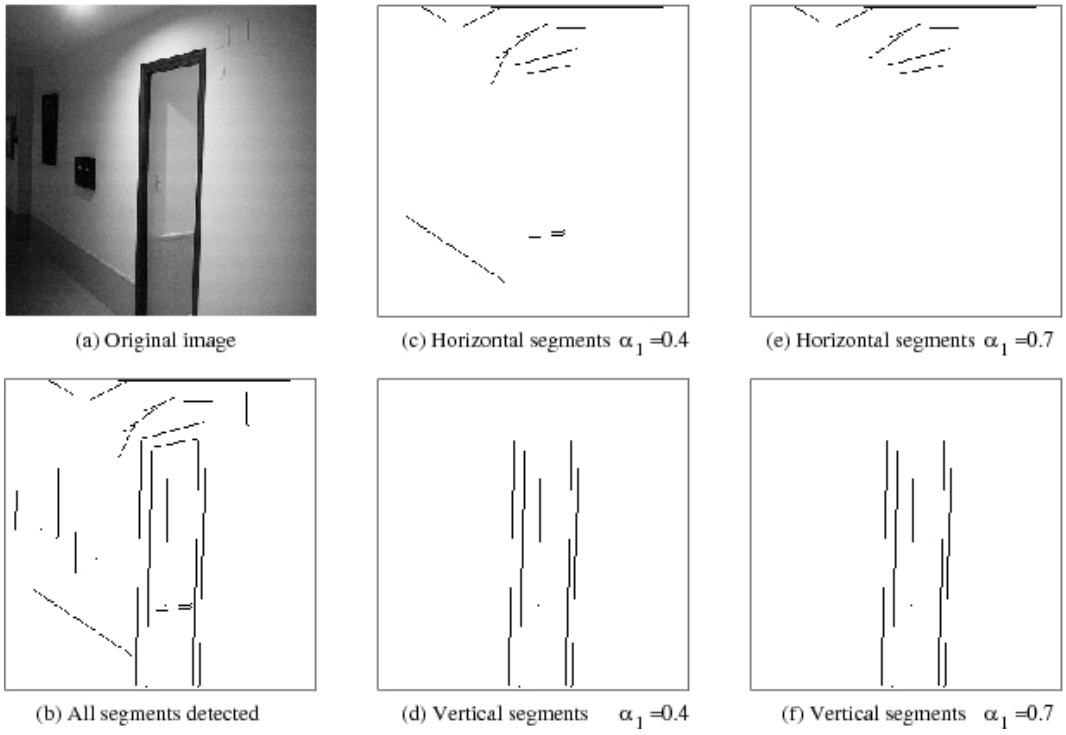


Figure 4.3: Segment classification for different values of α_1

4.2.2 Inverted U

The next step consists in detecting occurrences of the complex fuzzy concept IU among the segments selected in the previous phase. The fuzzy concept IU represents the edge configuration observed when a closed door, whose leaf and architrave have the same colour, is projected in an image (Fig. 4.1(a)). In order to find occurrences of this fuzzy concept, all the possible combinations of two vertical segments with a vertical one are analysed. The fuzzy concepts VS and HS have already been calculated at this point. The new fuzzy concept, *Edges Cohesion* (EC), measures the degree in which the arrangement of three segments is similar to the projection of the external or internal border of an architrave. Finally, IU is calculated by aggregating: (i) the membership degree of the three segments to the fuzzy concept EC and (ii) the individual membership degree of each segment to the fuzzy concepts VS and HS.

The process starts selecting a pair of vertical segments $\{V^j, V^k\} \in V$ for each horizontal segment $H^i \in H$. Let us denote the three segments by $F^i = \{L^i, Sup^i, R^i\}$, being $L^i \in V$

the leftmost vertical segment, $Sup^i \in H$ the horizontal segment and $R^i \in V$ the rightmost vertical segment. If F^i is part of an architrave, the upper extremes of the vertical segments should be very close to the extremes of the horizontal segment. Furthermore, the vertical segments should not be too close to each other. Both distances should be modelled taking into account that there can be errors in the segment extraction that makes the extremes of the segments appear slightly separated. It must also be considered that the distance between the vertical segments varies depending on the point of view under which doors are seen. The two linguistic variables $SegDistVH(F^i)$ and $SegDistVV(F^i)$ are defined to measure these distances. $SegDistVH(F^i)$ measures the distance between the vertical segments and the horizontal one while $SegDistVV(F^i)$ measures the distance between the vertical segments. Let us denote by Sup^i_l the leftmost extreme point of the segment Sup^i and by Sup^i_r the rightmost one. Likewise, let us denote by L^i_u the uppermost extreme point of L^i and by R^i_u the rightmost segment of R^i (see Fig. 4.4(a)).

The linguistic variable $SegDistVH(F^i)$ has the five possible values represented by the fuzzy sets shown in Fig. 4.5(a). Its input value, $distVH(F^i)$, is calculated using Eq. 4.4. This equation calculates the maximum of the distances between the upper extreme points of each vertical segment and the horizontal one. The value is scaled to the range $[0, 1]$, where the value 0 indicates that both segments are joined and the value 1 indicates the maximum separation. The maximum operator has been employed to force a small distance between both extreme points. These distances are shown in Fig. 4.4(b).

$$distVH(F^i) = \frac{max\{dist(Sup^i_l, L^i_u), dist(Sup^i_r, R^i_u)\}}{N\sqrt{2}} \tag{4.4}$$

The linguistic variable $SegDistVV(F^i)$ is used to evaluate the distance between the vertical segments. The variable has the five possible values represented in Fig. 4.5(b). Its input value $distVV(F^i)$, that is calculated using Eq. 4.5, escalates the distance between the upper points of both segments to the range $[0, 1]$. This distance is also shown in Fig. 4.4(b).

$$distVV(F^i) = \frac{dist(L^i_u, R^i_u)}{N\sqrt{2}} \tag{4.5}$$

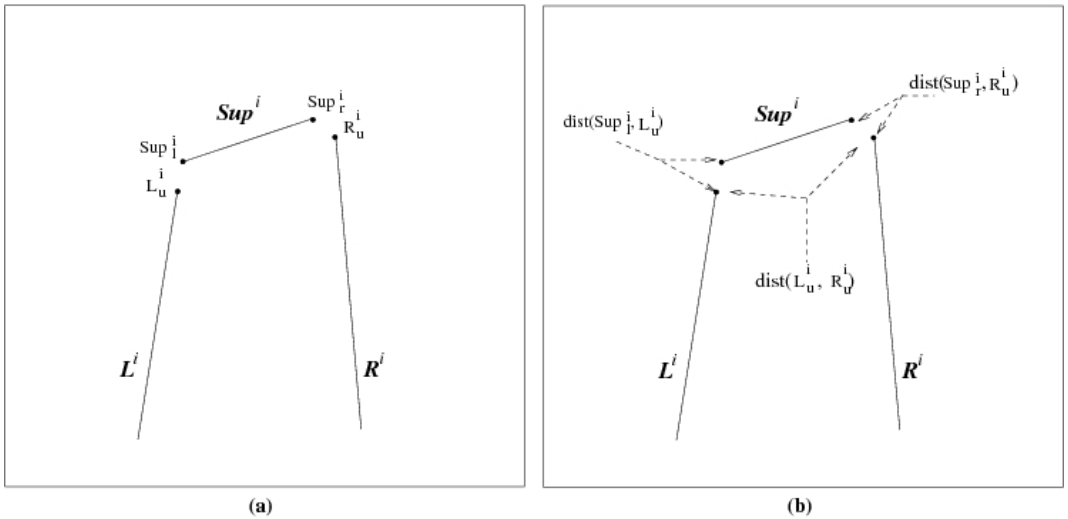


Figure 4.4: (a) Points used to calculate $SegDistVV(F^i)$ and $SegDistVH(F^i)$ (b) Distances used to calculate $distVV(F^i)$ and $distVH(F^i)$

The membership degree $EC(F^i) \in [0, 1]$ of F^i to the fuzzy concept IU is calculated using the rule base of Table 4.2 by a fuzzy inference process and its corresponding defuzzification. The fuzzy sets related to the possible values of the concept EC are shown in Fig. 4.5(c).

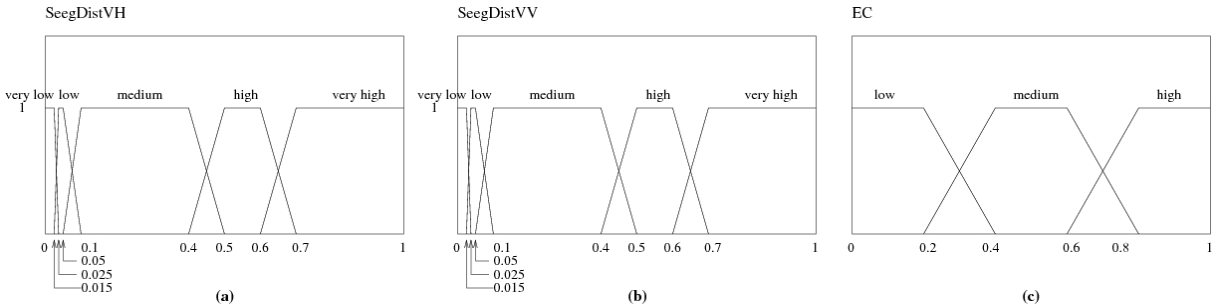


Figure 4.5: Linguistic variables employed for detecting the fuzzy concept IU

$EC(F^i)$ indicates that the separation between the segments of F^i is appropriate to be part of an architrave, but it does not take into account the corresponding membership degree of each individual segment to the concepts VS and HS. Finally, the membership degree $IU(F^i) \in [0, 1]$ of F^i to the fuzzy concept IU is calculated as expressed in Eq. 4.6. $IU(F^i)$ combines the membership degree of F^i to the fuzzy concept EC with the individual mem-

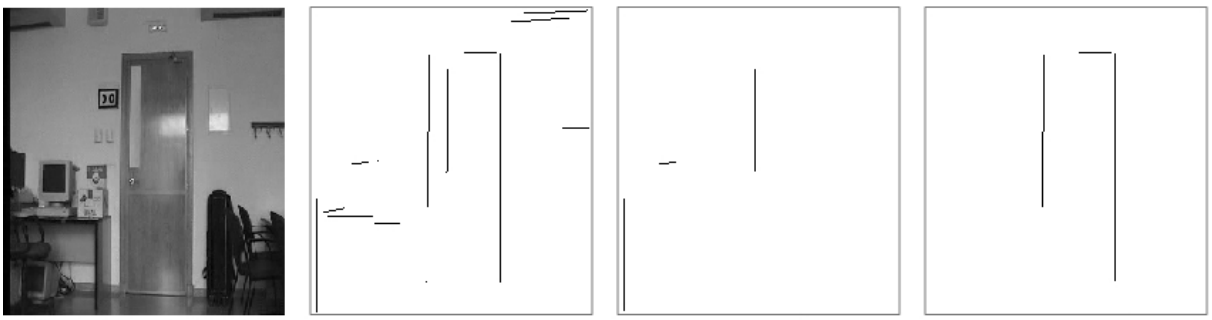
$SegDistVV(F^i)$	$SegDistVH(F^i)$				
	VL	L	M	H	VH
VL	L	L	L	L	L
L	H	M	M	L	L
M	H	H	M	L	L
H	H	M	L	L	L
VH	M	L	L	L	L

Table 4.2: Rule base for linguistic variable $EC(F^i)$

bership degree of the three segments to the fuzzy concepts VS and HS.

$$IU(F^i) = min\{EC(F^i), VS(L^i), HS(Sup^i), VS(R^i)\} \tag{4.6}$$

Only the F^i whose membership degree $IU(F^i)$ exceeds the threshold α_2 are used in the following phase. Let us denote this set as $F = \{F^0, ..., F^n \mid IU(F^i) > \alpha_2\}$. Figure 4.6 shows an image and the IU detected for different values of α_2. As in the previous case, the selection of the parameter α_2 is a critical issue that must be tackled based on experimentation.



(a) Original image (b) Segments selected α_1 =0.6 (c) IU detected for IU(F^i) in [0,0.7] (d) IU detected for IU(F^i) in [0.7,1]

Figure 4.6: Occurrences of the fuzzy concept IU detected in a image

It must be noticed that the fuzzy concept IU is detected for any rectangular object whose dimensions are similar to the dimensions of doors (like cupboards or paintings). Therefore, the system defines another two fuzzy concepts (CA and ICA) that helps to distinguish doors from other objects of the environment.

4.2.3 Complete Architrave

The segment configuration depicted in Fig. 4.1(b) is often produced by doors whose leaf and architrave have different colours and by opened doors seen from the side where its leaf is not visible. The system is able to recognise these cases by the complex fuzzy concept *Complete Architrave* CA. Detecting occurrences of that fuzzy concept can be of great help in mobile robotic applications because it is usually a more powerful indicator of the presence of doors than the fuzzy concept IU. The fuzzy concept CA indicates the degree in which F^i and F^j (detected in the previous phase) belong to the same architrave. CA is calculated based on: (i) the membership degree of F^i and F^j to the new fuzzy concept *Inverted U Similarity* (IUS) and (ii) the particular membership degree of F^i and F^j to the fuzzy concept IU. The process for detecting occurrences of the fuzzy concept CA is explained below.

If F^i and F^j belong to the same door their segments should appear parallel and relatively near. The fuzzy concept *Inverted U Similarity* (IUS) is introduced to evaluate the degree in which F^i and F^j are parallel and near. It is calculated using the linguistic variables $IUDist(F^i, F^j)$ and $Paralellism(F^i, F^j)$. The linguistic variable $IUDist(F^i, F^j)$ (whose five possible values are showed in Fig. 4.7(a)) measures the distance between F^i and F^j. Its input value $distF(F^i, F^j)$, that is calculated using Eq. 4.7, measures the distance between F^i and F^j as the maximum distance (scaled to the range $[0, 1]$) between its analogous segments.

$$distF(F^i, F^j) = max\{minDist(L^i, L^j), minDist(Sup^i, Sup^j), minDist(R^i, R^j)\} \tag{4.7}$$

The function $minDist(S^i, S^j)$ (see Eq. 4.8) is the minimum distance (scaled to the range $[0, 1]$) between S^i and S^j .

$$minDist(S^i, S^j) = \frac{min\{dist(p_0^i, p_0^j), dist(p_0^i, p_1^j), dist(p_1^i, p_0^j), dist(p_1^i, p_1^j)\}}{N\sqrt{2}} \tag{4.8}$$

The linguistic variable $Paralellism(F^i, F^j)$ (whose possible values are shown in Figure 4.7(b)) measures the degree of parallelism between F^i and F^j. Its input value, $parallelismF(F^i, F^j)$, is calculated using Eq. 4.9.

$$parallelismF(F^i, F^j) = min\{|cos(L^i_\phi - L^j_\phi)|, |cos(Sup^i_\phi - Sup^j_\phi)|, |cos(R^i_\phi - R^j_\phi)|\}, \quad (4.9)$$

where:

$$S^i_\phi = \arctan\left(\frac{y^i_1 - y^i_0}{x^i_1 - x^i_0}\right) \quad (4.10)$$

The term $cos(S^a_\phi - S^b_\phi)$ of Eq. 4.9 measures the parallelism betwen S^a and S^b in the range $[0, 1]$. If cosine is 0 means that the segments form an angle of $\frac{\pi}{2}$ radians and if it is 1 means that there is no difference in the angle between them. Therefore, Eq. 4.9 expresses the minimum degree of the parallelism between the analogous segments of F^i and F^j.

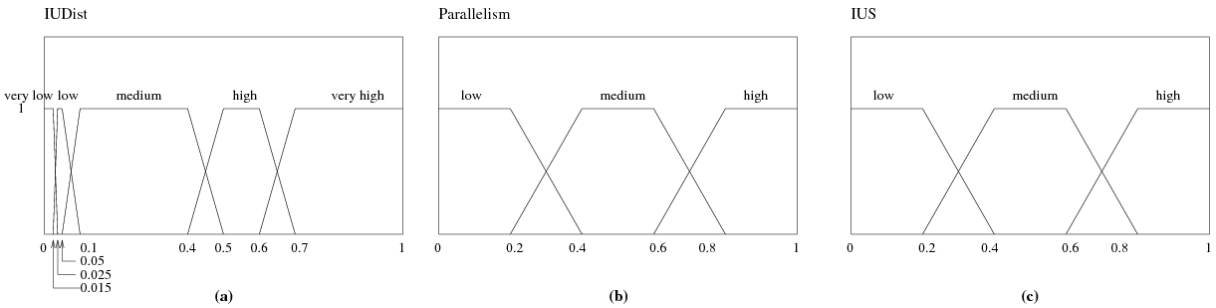


Figure 4.7: Linguistic variables for detecting complete doorframes

Using these fuzzy variables, $IUS(F^i, F^j) \in [0, 1]$ is defined (see Fig. 4.7(c)) to express the membership degree of F^i and F^j to the fuzzy concept IUS. The concept IUS expresses the degree in which F^i and F^j are parallel and near, taking into account its distance and parallelism. The rule base of Table 4.3 is used for that purpose. As in previous cases, the value is calculated by a fuzzy inference process and its corresponding defuzzification.

Finally, the membership degree $CA(F^i, F^j) \in [0, 1]$ of F^i and F^j to the fuzzy concept CA is calculated using Eq. 4.11. CA takes into account both the individual membership degree of each F^i and F^j to the concept IU and the membership of the pair to the concept IUS. If the membership degree $CA(F^i, F^j)$ exceeds a certain threshold α_3, the system consider that F^i and F^j belong to the same door. The appropriate value for α_3 is an important issue that must be selected based on experimentation. Figures 4.8(a), 4.8(c) and 4.8(e) show several images of doors. Below, Figures 4.8(b), 4.8(d) and 4.8(f) show the complete architraves detected.

$Paralellism(F^i, F^j)$	$IUDist(F^i, F^j)$				
	VL	L	M	H	VH
L	L	L	L	L	L
M	M	M	M	M	L
H	H	H	H	M	L

Table 4.3: Rule base for linguistic variable $IUS(F^i, F^j)$

$$CA(F^i, F^j) = min\{IUS(F^i, F^j), IU(F^i), IU(F^j)\} \tag{4.11}$$

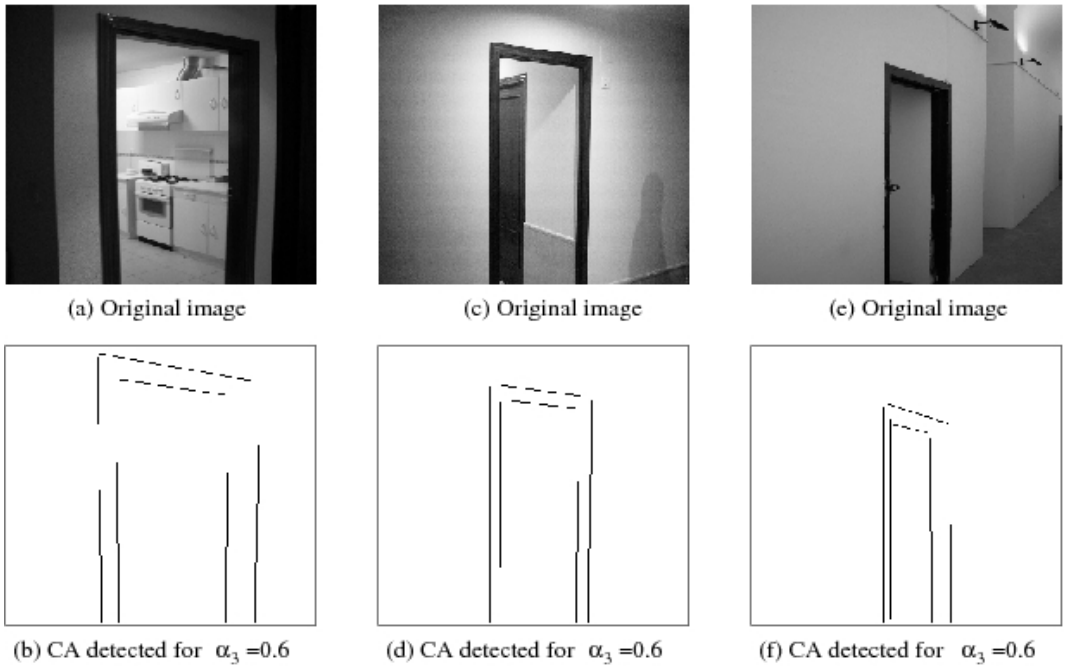


Figure 4.8: Complete Architraves detected in images

4.2.4 Incomplete Architraves

In some situations, the two borders of architraves can not be seen because there is a hidden segment. It usually happens when a door is open and the outer border of the architrave is hidden by the leaf. In that case, it is usual to find evidences of the incomplete outer border

around the inner border. This situation has been considered by the fuzzy concept *Incomplete Architrave* (IA) (Fig. 4.1(c)). The evidence is a pair of connected segments (one vertical and one horizontal) that is near and relatively parallel to a detected IU. Let us call the pair of segments *junction*.

The first step to detect occurrences of the fuzzy concept IA consists in detecting junctions in the image. Then, the distance and parallelism between each possible junction and IU pair is analysed in order to check whether it belongs to the same door. The next two sections explains this process. Section 4.2.5 explains how junctions are detected in images and Sect. 4.2.6 explains how the analysis of the distance and parallelism is performed.

4.2.5 Junction Detection

The fuzzy concept *Junction* (JC) is introduced in order to detect the junctions of an image. JC expresses that a pair of segments (one vertical and one horizontal) are joined in its upper part. The vertical and horizontal concepts have been previously computed (VS and HS). The new fuzzy concept *Junction Cohesion* (JCC) is defined to evaluate whether the distance between the extreme points of the segments is appropriate as explained below.

The system analyses if any segment $S_l \in V$ (that does not belong to any IU) is joined to any remaining segment $S_h \in H$ by measuring the distance between them. Let us denote a junction by $J^i = \{S_l, S_h / S_l \in V \wedge S_h \in H\}$. The linguistic variable $DistJ(J^i)$ (whose fuzzy sets are identical to those shown in Fig. 4.7(a)) is employed in order to measure the distance between the segments. Its input value is $minDist(S^i, S^j)$ as was already defined in Eq. 4.8.

The membership degree $JCC(J^i) \in [0, 1]$ of J^i to the fuzzy concept JCC is calculated using the rule base of Table 4.4 by a fuzzy inference process and its corresponding defuzzification. The fuzzy sets related to the possible values of JCC are identical to those shown in Fig. 4.7(c). Finally, the membership degree $JC(J^i) \in [0, 1]$ of J^i to the fuzzy concept JC is calculated using Eq. 4.12. The fuzzy concept JC takes into account: (i) the membership degree of each segment to the previously defined concepts HS and VS; (ii) the membership degree of J^i to the fuzzy concept JCC. If $JC(J^i)$ exceeds a threshold α_4 then J^i is consid-

ered for the next phase. Let us denote by $J = \{J^0, ..., J^n \, / J^i = \{S_l^i, S_h^i\} \wedge JC(J^i) > \alpha_4\}$ the set of junctions selected for the next phase. The value α_4 must be appropriately selected to avoid rejecting interesting junctions and to avoid accepting as valid many invalid segment configurations.

IF	THEN
$DistJ(J^i)$	$JCC(J^i)$
VL	H
L	M
M	L
H	L
VH	L

Table 4.4: Rule base for linguistic variable $JCC(J^i)$

$$JC(J^i) = min\{JCC(J^i), V(S_l), H(S_h)\} \tag{4.12}$$

Figure 4.9(a) shows a door whose leaf hides the inner border of the architrave. Figure 4.9(b) shows all the segments selected for $\alpha_1 = 0.6$ while Fig. 4.9(c) shows the IU edge detected for $\alpha_2 = 0.6$. Notice that only two of the segments belonging to the outer border of the architrave are visible. Figures 4.9(d), 4.9(e) and 4.9(f) show the junctions detected for different values of $JC(J^i)$.

In the following section it is explained how the system analyses the IU and junctions to find occurrences of the complex fuzzy concept IA.

4.2.6 Inverted U and Junction Analysis

The next step consists in analysing if there is a J^i that is part of an architrave. If so, J^i must be near an parallel to a F^j previously detected. The fuzzy concept *Incomplete Architrave* (IA) expresses that a junction and an IU are part of the same door. Both J^i and F^j have

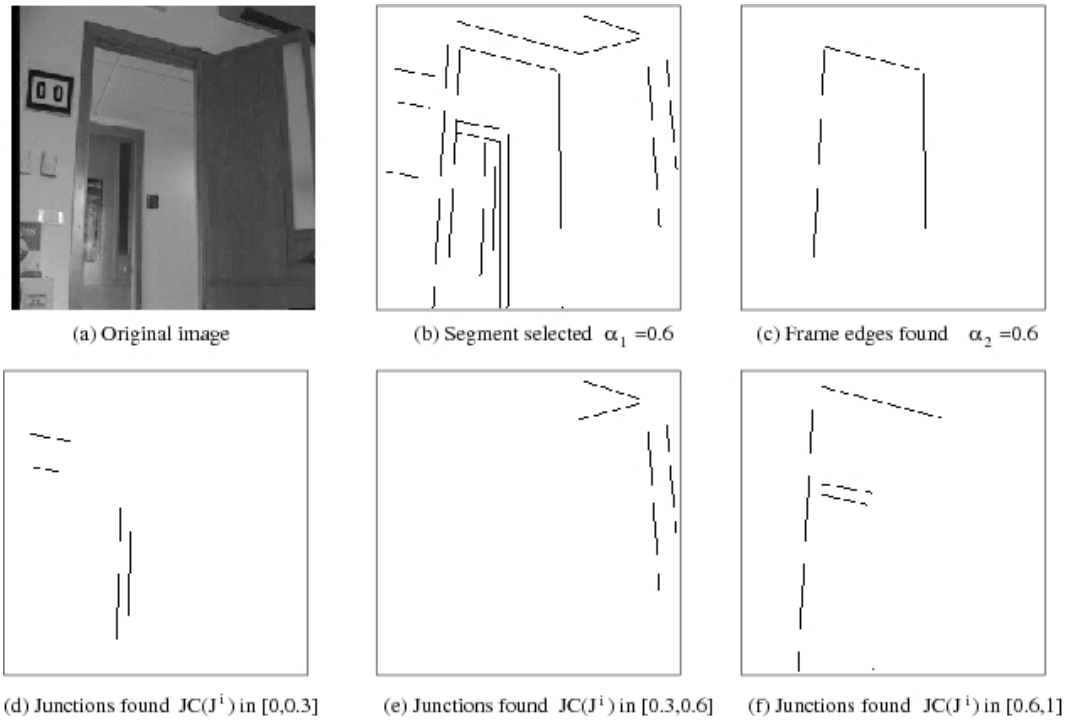


Figure 4.9: Examples of junctions detected for different membership degrees

been previously detected. The new fuzzy concept, *Inverted-U and Junction Cohesion* (IJC), evaluates if J^i and F^j are near and parallel as explained below.

When analysing a junction it can be observed that its vertical segment can be either at the left or at the right side of its horizontal segment. In the former case, the junction should belong to the left side of the architrave, thus being placed at the left side of an existing F^j. Analogously, if the horizontal segment is at the right side of the vertical one, J^i should be at the right side of the architrave. Besides, if J^i is part of the same door as F^j its segments must be parallels. The linguistic variable $JParallelism(J^i, F^j)$ (whose fuzzy labels are identical to those shown in Fig. 4.7(b)) is introduced to measure the parallelism between J^i and F^j. Its input value $parallelismJ(J^i, F^j) \in [0, 1]$, that is calculated using Eq. 4.13, measures the parallelism between J^i and F^j. If $parallelismJ(J^i, F^j)$ is 1 it means that J^i and F^j are parallel and if it is 0 means that they are perpendicular. $S^i_{l\phi}$ and $S^i_{v\phi}$ are the angles of the horizontal and vertical segments of J^i, calculated using Eq. 4.10.

$$parallelismJ(J^i, F^j) = \begin{cases} max\{|cos(S^i_{h\phi} - Sup^j_\phi)|, |cos(S^i_{l\phi} - L^j_\phi)|\} & if\ S^i_l\ is\ at\ left\ side\ of\ S^i_h \\ max\{|cos(S^i_{h\phi} - Sup^j_\phi)|, |cos(S^i_{l\phi} - R^j_\phi)|\} & otherwise \end{cases} \tag{4.13}$$

If J^i belongs to the same door than F^j the distance between them must be relatively small. The linguistic variable $DistJF(J^i, F^j)$ (whose fuzzy labels are identical to those shown in Fig. 4.7(a)) is used in order to measure the distance between J^i and F^j. Its input value $distJ(J^i, F^j) \in [0, 1]$ is calculated using Eq. 4.14. Low values of $distJ(J^i, F^j)$ indicates short distances between J^i and F^j and vice versa.

$$distJ(J^i, F^j) = \begin{cases} \frac{max\{minDist(S^i_h, Sup^j), minDist(S^i_l, L^j)\}}{2\sqrt{N}} & if\ S^i_l\ is\ at\ left\ side\ of\ S^i_h \\ \frac{max\{minDist(S^i_h, Sup^j), minDist(S^i_l, R^j)\}}{2\sqrt{N}} & otherwise \end{cases} \tag{4.14}$$

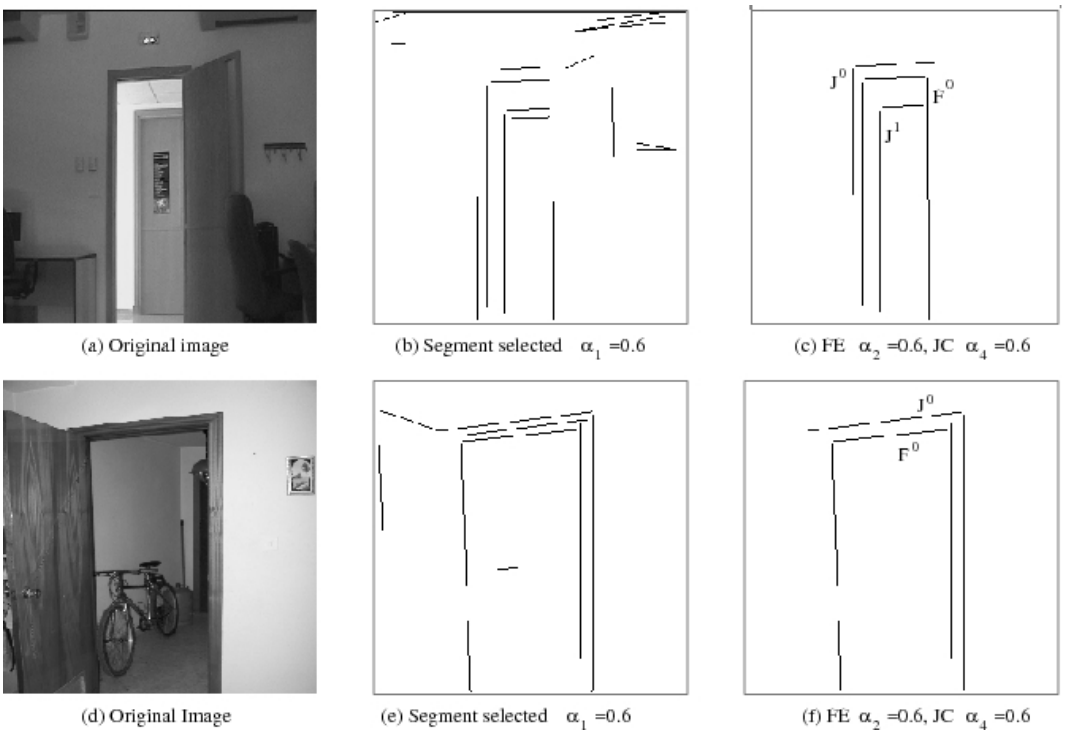
Using these variables, the membership degree $FJC(J^i, F^j) \in [0, 1]$ indicates the possibility that J^i and F^j belong to the same door. $FJC(J^i, F^j)$ is calculated by a fuzzy inference process, using the rule base of Table 4.5, and its corresponding defuzzification. Finally, $IA(J^i, F^j) \in [0, 1]$ (Eq. 4.15) indicates the possibility that F^i and J^i belong to the same door, taking into account the individual membership degree of each element. As in previous cases, if $IA(J^i, F^j)$ exceeds a threshold value α_5, the system considers that J^i and F^j belong to the same door. The appropriate value for α_5 must be appropriately selected to achieve an optimal performance.

$JParallelism(J^i, F^j)$	$DistJF(J^i, F^j)$				
	VL	L	M	H	VH
L	L	L	L	L	L
M	M	M	M	M	L
H	H	H	H	M	L

Table 4.5: Rule base for linguistic variable $FJC(J^i, F^j)$

$$FEE(J^i, F^j) = min\{FJC(J^i, F^j), FE(F^i), JC(J^i)\} \quad (4.15)$$

Figure 4.10 shows two images with doors in the left side. The two images in the middle show all the segments detected for $\alpha_1 = 0.6$. Finally, the right side shows the junctions and occurrences of IU detected in each image (labelled as J^i and F^j respectively). Figure 4.10(c) shows two junctions for a value $\alpha_4 = 0.6$ but only J^0 belongs to the door represented by F^0. For that particular case, the value $IA(J^0, F^0) = 0.82$ and $IA(J^1, F^0) = 0.3$. Thus, the system indicates that the first case is more possible. Figure 4.10(f) shows only one junction J^0 and one IU F^0. For that case, the value $IA(J^0, F^0)$ is equal to 0.74.



(a) Original image (b) Segment selected α_1 =0.6 (c) FE α_2 =0.6, JC α_4 =0.6

(d) Original Image (e) Segment selected α_1 =0.6 (f) FE α_2 =0.6, JC α_4 =0.6

Figure 4.10: Images with incomplete architraves

4.3 Experimental Results

This section shows the results obtained from the experimentation performed to test our fuzzy system. A database with images of our building has been created in order to check its per-

formance

The database contains several images of the three fuzzy concepts that the system is able to detect (IU, CA and IA) as well as images without doors. The images have been captured using the camera of our robot with a resolution of $320x280$ pixels. Both the robot and the PTU where the camera is placed have been manually controlled to capture the images. The images containing doors have been captured by placing the robot at distances ranging from 0.5 m to 4 m from the doors and in angles varying from -60^{o} to 60^{o} around them. We have taken care that both the two lateral bars and the upper bar of the architrave are visible in the images. The illumination conditions have not been artificially altered in our experiments. Some doors are illuminated by natural light coming from windows while others are illuminated by artificial lights. The database contains a total of: (i) 130 images of closed doors (to evaluate the fuzzy concept IU) (ii) 130 images of doors where the two borders of the architrave are visible (in order to test the fuzzy concept CA) and (iii) 130 images of doors where the inner border of the architrave is completely visible but the outer one is partially occluded (in order to test the fuzzy concept IA). Figure 4.11 shows some of these images. Finally, the database contains another 130 images without doors in them, of different places of our building. Although these images do not contain doors, some of them contains rectangular elements that could be confused with doors. Figure 4.12 shows some of these images.

It should be noticed that in order to detect an instance of the fuzzy concept CA, the system firstly has to detect two instances of the fuzzy concept IU. Similarly, to detect an instance of the fuzzy concept IA the system has to detect an instance of the fuzzy concept IU. Thus, an image that shows a CA also contains at least two IU and images that show an IA also contains at least one IU. Therefore, the images employed to evaluate the performance of the system when detecting the fuzzy concepts CA and IA have been also employed to evaluate the performance of the system when detecting the fuzzy concept IU. The total number of instances of the fuzzy concept IU in the database is 520. For each image, all the segments must be extracted and those that belong to architraves manually labelled. For that purpose we have created a specific application that helps in this process.

The fuzzy system receives as input a set of segments detected in an image and returns those that belong to one of the three fuzzy concepts (IU, CA and IA). Therefore, it is necessary to know the segments that really belong to doors in order to evaluate the success of the fuzzy system. All the segments in each image of the database have been extracted and those belonging to doors have been manually labelled. The time required to extract the segments in each image is approximately 160 ms in a Pentium IV (2.4 Ghz) laptop computer.

As indicated along the explanation of the fuzzy system, the overall performance strongly depends on the five α parameters. These parameters indicate the membership degree in which the segments of an image belongs to the fuzzy concepts analysed by the fuzzy system. When high values are employed, the system becomes very strict so that it is more sensible to the uncertainty in the segment extraction and to perspective deformations. However, low values force the system to accept erroneous configuration of segments as valid instances of the fuzzy concepts (as shown in Fig. 4.6(c)).

Figure 4.11: Images of the database containing doors

The system has been evaluated on our database by means of using the same v value for

all the α_i parameters repeating the evaluation with the four different values of v: $0.3, 0.5, 0.7$ and 0.9. Obviously, an exhaustive analysis of the performance would require to test all the possible values combinations for the different α_i parameters. However, even assuming that each α_i parameter can have only the four possible values previously indicated, there is a the total of 1024 possible combination to test. Nevertheless, an appropriate analysis of the optimal parameter configuration for the α_i parameters is out of the scope of this chapter and will be dealt in the next one.

Table 4.6 shows the success and error rate of the fuzzy system when detecting the three different fuzzy concepts IU, CA and IA. The measures employed in the table are TPF and μ_{FP}. The true positive fraction (TPF) employed in Table 4.6 is the success fraction of the system when detecting a fuzzy concept (IU, CA or IA). It is calculated as the number of correct detections of a fuzzy concept divided by the number of instances of that fuzzy concept in the database. Therefore, TPF is in the range $[0, 1]$, where 0 means that the system is not able to detect any correct instance of a fuzzy concept and 1 means optimal detection. The measure μ_{FP} is the average number of false positives detected per image.

v	IU		CA		IA	
	TPF	μ_{FP}	TPF	μ_{FP}	TPF	μ_{FP}
0.3	0.727	1.234	0.818	0.170	0.842	0.177
0.5	0.706	0.047	0.621	0.013	0.642	0.013
0.7	0.648	0.027	0.424	0.002	0.442	0.002
0.9	0.000	0.000	0.000	0.000	0.000	0.000

Table 4.6: Success of the fuzzy system in detecting the fuzzy concepts FEE, CDF and FEE on unseen data

As expected, the system obtains higher detection rates for low values of v but also a higher number of false positives. As v increases, the number of false positives decreases as well as the TPF. We consider that the system has an appropriate trade-off between the two measures for $v = 0.5$.

Figure 4.13(a) show some of the correct detections of that fuzzy system. The figure shows

Figure 4.12: Images of the database without doors

several images of the database where the output segments returned by the system have been superimposed in white. Figure 4.13(b) shows some false positives detected by the system. Finally, we want to mention that the average time employed for the system in analysing the segments of an image is approximately $10ms$ in our tests.

4.4 Final Remarks

This chapter has presented a visual fuzzy system able to detect doors with architraves in grey-level images. The system can be used for robotic tasks such as map-building, navigation and positioning. The system is comprised by several fuzzy systems that analyse the segments extracted from images in order to find the borders of doors architraves. For that purpose, the system defines three complex fuzzy concepts that represent three possible views of the doors architraves. They are: *Inverted U* (IU), *Complete Architrave* (CA) and *Incomplete Architrave* (IA). The first fuzzy concept represents a closed door whose leaf and architrave have the same colour. Thus, it is only visible the outer border of the architrave. The second fuzzy concept represents a closed door whose leaf and architrave have different colours or an opened door whose leaf is not visible. Thus, both the inner and outer borders of the

(a) Correct Detections

(b) False detections

Figure 4.13: Examples of correct detections and false positives of the fuzzy system

architrave are visible. Finally, the third fuzzy concept represents an opened door whose leaf hides the external border of its architrave. In that case, it is visible the inner border and part of the outer one.

Because of our visual system is mounted on a pan-tilt unit, the projection of the doors

in an image is affected by perspective deformations that make the borders of the architraves appear inclined (instead of completely vertical or horizontal as they really are). The system is able to manage these deformations and detect doors under a wide range of points of view. Whatever the image dimensions the system works due to its parameters are independent from size. A large database containing images of our environment has been created in order to test the system. The results shown that the proposed system achieves an appropriate level of performace and that can be used for real time applications.

The use of fuzzy logic for designing the system brings several advantages: (i) it allows to define and to manage fuzzy concepts like *Vertical Segment*, *Horizontal Segment* or *Parallelism* in a natural way; (ii) it helps to manipulate the ambiguity and uncertainty in the segment extraction; and (iii) it allows to create the system as a set of rules extracted from expert knowledge that future users can understand and even alter if required.

The system contains a multitude of parameters that have been selected based on expert knowledge. However, the most appropriate configuration for the fuzzy labels and for the rest of the parameters of the system is an application-dependent problem. Altough the system has been tested on images of our environment obtaining good performance, the optimal set of parameters can vary from one environment to another (different doors sizes and heights) and from one camera to another (different focal lenght or camera height). Therefore, it seems to be appropriate the use of any automatic tuning mechanisms for adjusting the system parameters in order to achieve optimal performance on a particular environment. Classical tuning approaches [155] does not seem to be appropriated for our complex fuzzy system for two major reasons. Firstly, the proposed system is comprised by several fuzzy systems highly inter-dependent. Secondly, the maximisation of the fuzzy system success is a multi-objective problem, i.e., maximising the TPF measure and minimising the μ_{FP} measure at the same time. Next chapter deals with that problem and proposes a novel approach for automatic tuning of fuzzy systems employing evolutionary algorithms that are capable of dealing with the difficulties indicated.

Finally, we must indicate that the work explained in this chapter has been accepted for publication in the journal *Autonomous Robots* of *Springer Netherlands* [160].

Chapter 5

Automatic tuning of fuzzy systems using evolutionary algorithms: single-objective vs. multiobjective

Manual tuning of the fuzzy system designed in the previous chapter is a tedious task that should be repeated in case of translating it to other working environment (with different doors' dimensions, camera height or focal lenght). An additional problem of tuning our fuzzy visual system is that its performance is evaluated by a multiobjective function. Although the global goal of the fuzzy visual system can be enunciated as *"to detect the doors present in the images captured by the robot's camera"*, this imply both (a) detect doors in the images where doors are present, and (b) not to indicate the false presence of a door in an image without any door. The former case is evaluated using the *True Positive Fraction* (TPF) measure and the latter by the μ_{FP} measure (explained in Sect. 4.3). The two objectives are independent and in conflict, but both must be optimised in the tuning process. For these reasons, we consider interesting to develop an automatic mechanism for tuning the fuzzy visual system employing training images captured at the particular environment in which it is going to be used. The goal is to increase the level of generality of the fuzzy system so it can operate in different environments.

In this chapter we propose a methodology to automatise the tuning of the fuzzy visual

system using an evolutionary approach. We decided to use evolutionary algorithms (EAs) [16] for tuning the system instead of classical tuning approaches [155] because it is possible to consider the whole fuzzy visual system as a unique system to optimise. Therefore, it is only necessary to define one error function to evaluate the performance of the whole system, instead of a different error function for each one of the five composing fuzzy systems. Besides, the multiobjective nature of the problem can be managed in a proper way by means of multiobjective EAs (MOEAs) [56, 47].

Within the optimisation field using EAs, we find the possibility to choose among single-objective algorithms, involving the aggregation of the different objectives into an scalar single-objective function, and multiobjective algorithms, that considers the joint optimisation of all of them [56, 47]. In this work we have tested both approaches. On the one hand, we have used two single-objective EAs: a generational Genetic Algorithm [88] and the CHC algorithm [65]. On the other hand, we have tested three MOEAs: SPEA [228], SPEA2 [227] and NSGA-II [57]. All of them are run under similar conditions and their results compared.

The remainder of the paper is structured as follows: Section 5.1 explains the basis of the multiobjective optimisation. Sections 5.2 and 5.3 describe the basis of the single-objective and multiobjective EAs used, respectively. Section 5.4 shows the coding scheme, genetic operators and fitness functions employed for our tuning approach. Section 5.5 shows the experiments carried out and Section 5.6 gives some final remarks.

5.1 Evolutionary Multiobjective Optimisation

Generally speaking, multiobjective problems can be formulated as: *"finding the best solution* $\vec{x} = [x_1, ..., x_n]^T$ *to a problem achieving the maximum value of the objective function* $\vec{f}(\vec{x}) = [f_1, ..., f_n]^T$*"* . The term $\vec{x}$ refers to a vector solution, $\vec{f}(\vec{x})$ refers to the objective function that evaluates $\vec{x}$ according to some criteria and let us denote the set of objectives to evaluate by O.

When using single-objective algorithms to solve the problem, it is usual to transform the different objectives functions into an real scalar value (*plain-aggregating* approach [56]) in

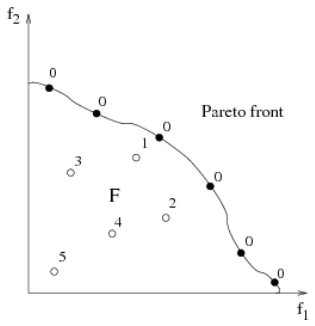


Figure 5.1: An example of ranking several solutions in EMO.

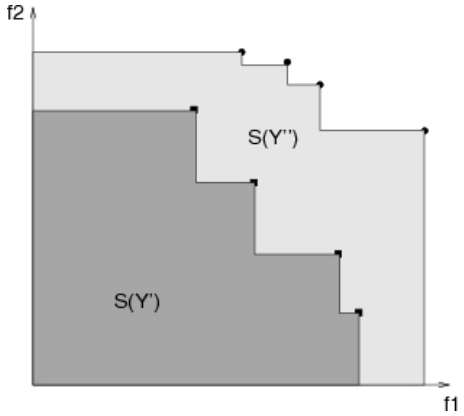


Figure 5.2: Example of metric $\mathcal{S}$ applied to two Pareto fronts

the following way:

$$\vec{f}(\vec{x}) = \beta_1 f_1 + \beta_2 f_2 + \cdots + \beta_n f_n$$

where

$$\sum_{i=1..n} \beta_i = 1$$

The different β_i values independently weight one objective against the others. This approach, from the multiobjective point of view, consist in directing the search process to only one direction of the multiobjective space.

Nevertheless, it is also possible to employ a *Pareto-based* approach to solve the problem. In that approach, the goal of the optimisation process is to find the ideal vector $\vec{x}^i$ that optimises all the elements of the objective function. The optimisation concept depends on the way the problem is formulated. It could be either a minimisation or a maximisation of $\vec{f}(\vec{x})$. If the optimisation process is a maximisation one, a solution $\vec{x}^i$ of the set of found solutions F *dominates* another solution $\vec{x}^j$ if:

$$\forall_{k \in O} f_k(\vec{x}^i) \geq f_k(\vec{x}^j) \wedge \exists_k : f_k(\vec{x}^i) > f_k(\vec{x}^j)$$

The set of non-dominated solutions found is denominated *Pareto set*. Hence, Pareto-based techniques allow us to find a set of non-dominated solutions, each one with a different trade-off of the objectives, thus avoiding to use an importance factor β_i for each objective.

Once a set of non-dominated solutions is given by the multiobjective technique, they can be analysed and it is possible to select a unique solution with the most suitable trade-off at the light of the results.

EAs have been employed to solve multiobjective problems using plain-aggregating approaches as well as both *population-based non-Pareto* approaches and *Pareto-based* approaches. This framework has been referred in the literature by the term *EMO* (Evolutionary Multiobjective Optimisation) and much work has been done since then in very different research areas [56, 47, 107]. These algorithms are also referred by the term MOEAs and have experimented a great advance with the introduction of the *Pareto-based fitness* concept [74]. The idea consists in ranking the individuals of the population according to its proximity to the Pareto-optimum.

Pareto-based approaches can be divided into two different groups: first and second generation [47]. The difference between them is in the use of elitism. First generation algorithms are non-elitist multiobjective algorithms like Niched Pareto Genetic Algorithm (NPGA), Non-dominated Sorting Genetic Algorithm (NSGA) and Multiple-Objective Genetic Algorithm (MOGA). On the other hand, second-generation algorithms employ elitist approaches in order to speed up the performance of the algorithm. Examples of this group are Strength Pareto Evolutionary Algorithm (SPEA) and SPEA2, NSGA-II and NPGA2. For more information about all these algorithms, the interested reader is referred to [56, 47].

MOEAs are able to optimise $\vec{f}(\vec{x})$ ranking each individual according to its proximity to the Pareto front, i.e., non-dominated solutions have the highest rank and the rest of the solutions are ranked according to some criteria. In Figure 5.1, there is depicted a typical situation in a multiobjective optimisation search. The filled circles represent the individuals of the population that belong to the Pareto set and the empty circles represent dominated solutions of the discovered space F. The numbers under each solution represent a possible ranking employed by a MOEA. While the elements of the Pareto set are ranked with the lowest values, the rest of solutions are ranked with higher values

5.1.1 Measuring the performance of MOEAs: EMO metrics

In multiobjective optimisation, the definition of quality is more complex than for single-objective optimisation problems. The multiobjective optimisation process involves several objectives itself:

- The distance of the resulting Pareto front to the Pareto-optimal front must be minimised.
- A good distribution of the solutions found is desirable. The assessment of this criterion might be based on a certain distance metric.
- The extent of the obtained Pareto front must be maximised.

Several quantitative metrics have been proposed in the literature to formalise the above definition (or parts of it) [56, 47, 226]. Those employed in this paper are defined below.

Given a set of pairwise non-dominated decision vectors $X' \subseteq X$, a neighbourhood parameter $\sigma > 0$ (to be chosen appropriately), and a distance metric $\| \cdot \|$:

1. The function $\mathcal{M}_1$ gives the average distance to the Pareto-optimal set $(\bar{X}) \subseteq X$:

$$\mathcal{M}_1(X') := \frac{1}{|X'|} \sum_{a' \in X'} min\{\|a' - \bar{a}\|; \bar{a} \in \bar{X}\} \tag{5.1}$$

2. The function $\mathcal{M}_2$ takes the distribution in combination with the number of non-dominated solutions found into account:

$$\mathcal{M}_2(X') := \frac{1}{|X' - 1|} \sum_{a' \in X'} |\{b' \in X'; \|a' - b'\| > \sigma\}| \tag{5.2}$$

3. The function $\mathcal{M}_3$ considers the extent of the front described by X':

$$\mathcal{M}_3(X') := \sqrt{\sum_{i=1}^{m} max\{\|a'_i - b'_i\|; a', b' \in X'\}} \tag{5.3}$$

Analogously, [226] defines three metrics $\mathcal{M}_1^*$, $\mathcal{M}_2^*$, and $\mathcal{M}_3^*$ on the objective space. Let $Y', \bar{Y} \subseteq Y$ be the sets of objective vectors that correspond to X' and $\bar{X}$, respectively, and $\sigma^* > 0$ and $\|\cdot\|^*$ as before:

$$\mathcal{M}_1^*(Y') := \frac{1}{|Y'|} \sum_{p' \in Y'} min\{\|p' - \bar{p}\|^*; \bar{p} \in \bar{Y}\} \tag{5.4}$$

$$\mathcal{M}_2^*(Y') := \frac{1}{|Y' - 1|} \sum_{p' \in Y'} |\{q' \in Y'; \|p' - q'\|^* > \sigma^*\}| \tag{5.5}$$

$$\mathcal{M}_3^*(Y') := \sqrt{\sum_{i=1}^{n} max\{\|p'_i - q'_i\|^*; p', q' \in Y'\}} \tag{5.6}$$

Additionally to the latter, in [228] Zitzler et al. proposed the metric $\mathcal{S}(Y')$, also called size of the space covered, that measures the volume enclosed by the Pareto front Y'. In a maximisation problem where the optimal objective values are equal to 1, $\mathcal{S}(Y') = 1$ means that the algorithm has reached the optimal solution and lower values for this metric indicates a lower quality of the Pareto front. In our case, as there are only two objectives, $\mathcal{S}(Y')$ measures the area covered by the Pareto front by adding the areas covered by each individual point. In Figure 5.2, the areas covered by two possible Pareto fronts are shown. It can be noticed that the outer Pareto front (Y'') dominates all the solutions of the inner one (Y'), thus the area $S(Y'') > S(Y')$.

Finally, the $\mathcal{C}$ metric [228] is introduced in order to evaluate the dominance of one Pareto set over another. This metric is used to evaluate the degree in which the solutions of a Pareto set cover the solutions of another. Given two Pareto sets Y' and Y'', the function $\mathcal{C}$ can be calculated using Equation 5.7. When $\mathcal{C}(Y', Y'') = 1$ it means that all solutions in Y'' are dominated by solutions in Y'. The value $\mathcal{C}(Y', Y'') = 0$ means that none of the solutions of Y'' are dominated by the set Y'. It is important to point out that this function is not conmutative, therefore it is necessary to calculate both $\mathcal{C}(Y', Y'')$ and $\mathcal{C}(Y'', Y')$.

$$\mathcal{C}(Y', Y'') = \frac{|\{p'' \in Y''; \exists\, p' \in Y' : p' \; dominates \; p''\}|}{|Y''|} \tag{5.7}$$

5.2 Single-Objective approaches for tuning the fuzzy visual system

The two single-objective EAs considered are described in the next two subsections.

5.2.1 Generational GA

Generational Genetic Algorithm(N,p_{select},$p_{crossover}$):

1. $N_{select} = N * p_{select}$
2. $N_{crossover} = N * p_{crossover}$
3. $N_{mutate} = N * (1 - p_{select} - p_{crossover})$
4. $P_{t=0}$=Create Initial Population of N individuals
5. While not achieved the termination condition
   - 5.1 Select N_{select} individuals using binary tournament and copy them to P'
   - 5.2 Select $2 * N_{crossover}$ individuals using binary tournament and cross over them. Add the $N_{crossover}$ offsprings to P'
   - 5.3 Select N_{mutate} individuals and mutate them. Add the mutated individuals to P'
   - 5.4 Assign $P_{t+1} = P'$ and empty P'
   - 5.5 t=t+1

Figure 5.3: Generational GA

EAs have proved to be a useful tool for solving a broad class of difficult optimisation problems [17]. Among EAs, GAs are a particular instance based on the natural evolution concept. In that area, the generational GA (GGA) [88] is the classic exponent of this type. A GGA starts with random population that is evolved each iteration using selection, crossover and mutation operators, and the offspring population directly substitutes the parent one for

the next generation. Each individual is ranked with a fitness accordingly to its goodness in the problem solving.

In our implementation, the selection operator chooses a set of individuals to be part of the following generation without any modification. This operator introduces the selective pressure concept, making the good individuals to more probably be in the next generation. In our case, we have used a binary tournament that involves randomly choosing two individuals of the current population and selecting the one with the best fitness for the next generation population.

The crossover operator exploits the idea that from the combination of good individuals (parents) it is possible to generate good new individuals (offsprings). The parents are selected using binary tournament and the resulting offsprings are used for the next population.

Finally, the mutation operator works on an individual by randomly altering it. The use of this operator allows to explore undiscovered parts of the search space and prevents the algorithm to get stuck in local minima.

Figure 5.3 outlines the GGA operation. In order to run the algorithm, it is necessary to specify the number of individuals of the population N and the percentage of those individuals that are to be selected, crossed over and mutated. At each iteration, a new population P' is created using the aforementioned operators over the individuals of the current population P_t. The current population is entirely replaced by the new one.

5.2.2 CHC

The CHC algorithm [65] is an evolutionary approach that introduces an appropriate trade-off between diversity and convergence. For that purpose, it uses a high selective pressure based on an elitist scheme in combination with a highly disruptive crossover and a re-start when the population is stagnated. It is based on four distinguishing components:

- HUX crossover operator. The original algorithm was designed to be used with binary coding and this crossover operator ensures the obtaining of the most diverse offsprings from their parents.

CHC:

1. P_0=Create initial population
2. D_{mean}=Calculate mean distance of the population
3. D_{max}=Calculate maximum distance between individuals
4. $Decrement = DecFactor * D_{max}$
5. While not achieved the termination condition
   - 5.1 Pair up randomly the individuals of the population to be used as parents of the new offspring population
   - 5.2 Create a new offspring population $Child_i$ with M individuals $(M \leq N)$ using the crossover operator. If the distance between the two parents is smaller than D_{mean} do not generate the offspring
   - 5.3 Replace the individuals of P_i with the individuals of $Child_i$ that are better than them
   - 5.4 If no new offspring are generated $D_{mean} = D_{mean} - Decrement$
   - 5.5 If $D_{mean} < 0$ re-start keeping the best individual and recalculate D_{mean} and $Decrement$

Figure 5.4: CHC evolutionary algorithm

- Incest prevention. Two parents are not crossed over if they are too similar. This ensures diversity.

- Elitist selection. The new population is composed of the best individuals taken among the parents and the offsprings of each generation.

- Re-start. When the population reaches an stagnated state, it is re-started keeping the best individual.

The algorithm pseudo-code is shown in Figure 5.4. In a first step, the population is created using a perturbation operator over the initial chromosome. In our case, the initial chromosome is created from the original definition of the fuzzy visual system previously explained (based on expert knowledge). Then, the algorithm measures the mean distance of the population (D_{mean}) in order to estimate when two individuals are too close to be crossed and in this way avoiding a possible incest. Incest prevention forces to cross over separated elements causing an exploration of the area covered by the individuals of the population. D_{mean} is decremented each time the incest prevention mechanism makes impossible to generate any new offspring in one iteration. *DecFactor* allows us to select the degree of convergence of the algorithm, the smaller it is, the greater the level of convergence allowed and *vice versa*. When D_{mean} is below zero, the population is re-started keeping the best individual found. The new population is generated using the perturbation operator on the best individual with probability 35%. It means that only the 35% of the chromosome is altered.

5.3 MultiObjective approaches for tuning the fuzzy visual system

The three MOEAs used are reviewed in the next three subsections.

SPEA:

1. P=Initial Population
2. P^e=Non-dominated solutions of P
3. While not achieved the termination condition
   - 3.1 Assign a fitness value to the elements of P and P^e
   - 3.2 Select N individuals from $(P \cup P^e)$ by binary tournament
   - 3.3 Apply crossover and mutation
   - 3.4 Copy the non-dominated solutions in P^e and remove the dominated solutions
   - 3.5 If $|P^e| > N^e$ then reduce P^e using clustering

Figure 5.5: SPEA procedure

5.3.1 SPEA

The SPEA algorithm [228] is a Pareto-based MOEA that employs an elitist scheme. In order to maintain the elitism, the algorithm keeps an external population P^e with the non-dominated solutions found since the start of the run. The outline of the algorithm can be seen in Figure 5.5.

The algorithm starts creating a random population P of N elements. The non-dominated solutions of P are stored in the external elitist population P^e that is updated each generation.

A fitness value is assigned to all the elements of P and P^e. The fitness of the elements from P is calculated using a different criteria than the elements of P^e. An individual $\vec{x}^i$ from P^e is evaluated using Equation 5.8. The value n_i is the number of solutions in P dominated by $\vec{x}$. Equation 5.8 assigns low values to those non-dominated solutions that do not dominate other solutions in order to enforce the search in this area that is not appropriately covered by the population. On the other hand, an individual $\vec{x}^j$ from P is evaluated using Equation 5.9. This equation gives low values to those individuals that are weakly dominated.

SPEA Clustering:

1. Assign each element to a cluster
2. While $|P^e| > N^e$ do
   - 2.1 For each pair of clusters
     - i. Calculate the distance between clusters, like the average distance among all their elements

$$D_{ij} = \frac{1}{|c_1| \cdot |c_2|} \cdot \sum_{i_1 \in c_1, i_2 \in c_2} \|i_1 - i_2\|$$

   - 2.2 Merge the two clusters with minimum distance between them
3. Choose a solution of each cluster, that with minimum average distance to the remaining cluster elements

Figure 5.6: Clustering in SPEA

$$s(\vec{x}^i) = \frac{n_i}{N+1} \tag{5.8}$$

$$s(\vec{x}^j) = 1 + \sum_{\vec{x}^i \in P^e \text{ and } \vec{x}^i \text{ dominates } \vec{x}^j} s(\vec{x}^i) \tag{5.9}$$

Once a fitness value is calculated for all the individuals, those that are going to be mutated and crossed over are selected using binary tournament among all the elements of $P \cup P^e$. After applying the variation operators, the non-dominated solutions existing in the new population are copied to P^e, removing the dominated and duplicated ones. Therefore, the new elitist population is comprised by the best non-dominated solutions found so far, including new and old elitist solutions.

In order to limit the growth of the elitist population, a clustering algorithm is employed if the number of non-dominated solutions found is greater that a threshold value N^e. The clustering algorithm (showed in Figure 5.6) selects the solutions closer to the centre of each cluster.

SPEA2:

1. Set t=0 and P_t=Initial Population.
2. Assign a fitness value to the elements of P_t
3. Copy all the non-dominated solutions of P_t in P_t^e. If $|P_t^e| > N^e$ then reduce P_t^e by means of the truncation operator. If $|P_t^e| < N^e$ then fill P_t^e with the best dominated individuals of P_t
4. If the termination condition is reached stop
5. Create a mating pool P^m by binary tournament selection with replacement on P^e
6. Use crossover and mutation operators to create the new population P_{t+1}.
7. Set $t = t + 1$ and go to step 2

Figure 5.7: SPEA2 procedure

5.3.2 SPEA2

SPEA2 [227] is a MOEA that uses a Pareto-based elitist approach but it is modified to eliminate the potential weaknesses of SPEA. SPEA2 mainly differs from SPEA in three points:

- The fitness assignment scheme is modified to take into account both the number of individuals that an individual dominates and is dominated by. The problem appears in SPEA when P^e has only one non-dominated solution. In that case, all the dominated solutions have the same fitness and the algorithm behaves as a random search algorithm.
- In order to avoid the grouping of the individuals of the population (P and P^e), a density estimation value is introduced in the fitness function. This value assigns a better fitness to those solutions that are far from other solutions, thus enforcing an extended Pareto front.
- The management of the external population is modified. In SPEA2, P^e has fixed size

and if the number of non-dominated solutions is not enough to fill P^e then the best dominated solutions from P are copied to P^e. On the other hand, if the number of non-dominated solutions exceeds the limit, a truncation operator is employed instead of the clustering technique of SPEA.

The outline of the algorithm is shown in Figure 5.7. It starts creating the initial random population P_0. All the elements are ranked according to the fitness assignment scheme, that in contrast to SPEA, is the same for all the elements in P and P^e.

The fitness value of an element $\vec{x}^i$ is based on the sum of two values, the raw fitness $R(\vec{x}^i)$ and a density estimation $D(\vec{x}^i)$. The raw fitness (calculated using Equation 5.10) is determined based on the strength of its dominators $S(\vec{x}^i)$. The function $S(\vec{x}^i)$ is a strength value that indicates the number of individuals that $\vec{x}^i$ dominates as showed in Equation 5.11.

$$R(\vec{x}^i) = \sum_{\vec{x}^j \in \{P_t \cup P_t^e\} \wedge\ \vec{x}^j\ dominates\ \vec{x}^i} S(\vec{x}^j) \tag{5.10}$$

$$S(\vec{x}^i) = |\{\vec{x}^j | \vec{x}^j \in \{P_t \cup P_t^e\} \wedge\ \vec{x}^i\ dominates\ \vec{x}^j\}| \tag{5.11}$$

When most of the solutions are non-dominated, the raw fitness might be the same for all of them and this value might be useless. Therefore, a density estimator measure $D(\vec{x}^i)$ is calculated using Equation 5.12. The value σ_i^k is the distance of $\vec{x}^i$ to the k-th nearest neighbour in the objective space. $D(\vec{x}^i)$ assigns a better fitness value to those individuals that are isolated, thus enforcing an extended Pareto front.

$$D(\vec{x}^i) = \frac{1}{\sigma_i^k + 2} \tag{5.12}$$

A truncation operator is used to reduce the number of non-dominated solutions in P^e keeping the extent of the Pareto front. The individual with the minimum distance to other individual is the first selected for removal; if several individuals are in the same situation, the second smallest distance is considered and so forth.

NSGA-II:

1. P_0=Initial Population.
2. Fast-nondominated-sorting(P_0)
3. $t = 0$
4. While the termination condition is not reached
   - 4.1 Use crossover and mutation operators to create the new population Q_t
   - 4.2 $R_t = P_t \bigcup Q_t$
   - 4.3 $\mathcal{F}$=fast-nondominated-sorting(R_t)
   - 4.4 $P_{t+1} = 0$ and $i = 1$
   - 4.5 While $|P_{t+1}| \neq N$
     - i. If $|P_{t+1}| + |\mathcal{F}_i| < N$ add all the solutions from $\mathcal{F}_i$ into P_{t+1}
     - ii. Else Sort($\mathcal{F}_i$) and insert the first solutions of $\mathcal{F}_i$ into P_{t+1} to fill it
     - iii. $i = i + 1$
   - 4.6 $t = t + 1$

Figure 5.8: NSGA-II procedure

5.3.3 NSGA-II

The NSGA-II MOEA [57] appears to solve the weaknesses of its predecessor NSGA [199]. NSGA-II is a Pareto-based evolutionary algorithm that employs an elitist approach (one of the drawbacks of NSGA is that it does not use elitism). In contrast to SPEA and SPEA2, NSGA-II does not maintain an external population of non-dominated solutions but the new population is comprised by the best individuals of the parent and offspring populations (like CHC).

The ranking scheme employed consists in sorting the individuals of a population in different Pareto fronts (or levels) $\mathcal{F}_i$. The first level ($\mathcal{F}_1$) is comprised by the non-dominated individuals of the population. Once the individuals of the first level have been found, they are

temporarily removed. The second level ($\mathcal{F}_2$) is comprised by the remaining nondominated individuals. This process is repeated until no individual remains in the population. This ranking scheme is also used in its predecessor NSGA and is called *Nondominated Sorting* (naming the algorithm). Nevertheless, the original proposal is computationally inefficient ($O(MN^3)$) and the authors provides NSGA-II with a faster version called *Fast Nondominated Sorting*. Once the different levels $\mathcal{F}_i$ have been found, their individuals are ranked with a fitness according to the level they belong to. The individuals from $\mathcal{F}_1$ are ranked with lower values than the solutions of $\mathcal{F}_2$. Thus, minimisation of fitness is assumed.

In order to enforce spread Pareto fronts, a second sorting mechanism is employed among the individuals of a level $\mathcal{F}_i$. For that purpose, NSGA-II uses a density estimation value to assign a better ranking to those individuals that are far from other solutions. For each individual in a level $\mathcal{F}_i$, the mean distance to the two nearest individuals is computed. The higher this distance, the more isolated the individual and the lower the density around it. Therefore, those individuals with lower density value are better ranked within a level $\mathcal{F}_i$.

In Figure 5.8 the algorithm is outlined. It starts creating an initial random parent population $P_{t=0}$ (with N individuals) that is ranked using the Fast Nondominated Sorting. Then, an offspring population Q_t (of size N) is generated using the mutation and crossover operators over the individuals of P_t selected using binary tournament. The individuals of P_t and Q_t are merged into R_t and the Fast Nondominated Sorting is performed. The next generation P_{t+1} is comprised by the N best individuals of R_t. Since R_t contains all the individuals of P_t, elitism is ensured. If the first level of R_t ($\mathcal{F}_1$) has less than N individuals, the best individuals of $\mathcal{F}_2$ are used to fill P_{t+1}. If it is not possible to accommodate all the individuals of a level $\mathcal{F}_i$ into P_{t+1}, the density estimation value is used to select the best ones.

5.4 Coding Scheme, Genetic Operators and Objective Functions

In this section it is explained first how the fuzzy visual system has been encoded to be adapted by the EAs. Then, the basis of the genetic operators employed are introduced. Finally, the

objective functions used to measure the performance of an individual are also described.

5.4.1 Coding Scheme

In our approach, the whole fuzzy visual system is represented using a single chromosome composed of the joining of its membership function parameters. Each fuzzy variable is encoded based on the crossing points of its membership functions and the separation between them using a real coding scheme.

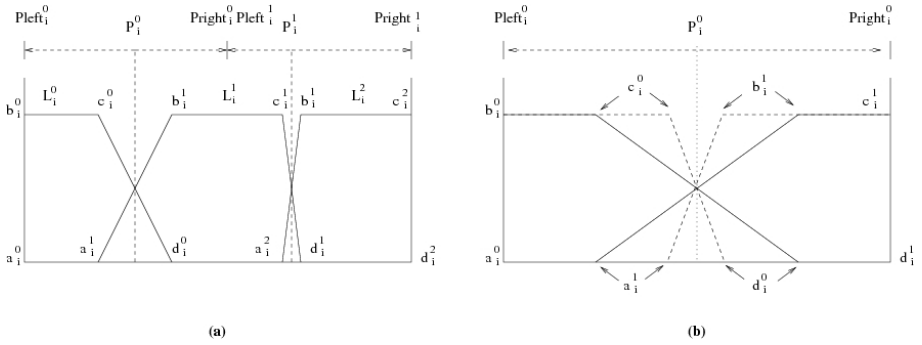


Figure 5.9: (a) Representation of a variable with 3 labels. (b) Representation of a variable for different values of s_i^j

We shall denote the set of variables of the fuzzy visual system as $Z = \{Z_0, ..., Z_m\}$ and let us denote the set of membership functions of variable Z_i as $L_i = \{L_i^0, ..., L_i^{n+1}\}$. As our system is entirely composed of trapezoidal functions, a membership function can be defined by four parameters as in Equation 5.13, where $[Left_i, Right_i]$ is the range of the input variable Z_i.

$$L_i^j = [a_i^j\ b_i^j\ c_i^j\ d_i^j]\ /\ a_i^j, b_i^j, c_i^j, d_i^j \in [Left_i, Right_i] \tag{5.13}$$

In order to reduce the search space, we limit the set of possible configurations forcing the membership functions to cross at level 0.5. This approach has been previously employed in [116] with triangular membership functions. In our case, the use of this constraint is forced to be accomplished by the following restriction:

$$a_i^{j+1} = c_i^j\ \wedge\ b_i^{j+1} = d_i^j$$

Consequently, the crossing point P_i^j of two consecutive labels L_i^j and L_i^{j+1} is calculated as expressed in Equation 5.14:

$$P_i^j = \frac{c_i^j + d_i^j}{2} = \frac{a_i^{j+1} + b_i^{j+1}}{2} \tag{5.14}$$

Therefore, for each variable Z_i, there is a set of n crossing points $P_i = \{P_i^0, .., P_i^n\}$ that have to be learnt by the EA to obtain the maximum performance. A range of allowed positions $[Pleft_i^j, Pright_i^j]$ is defined for each crossing point P_i^j instead of allowing EAs freely select any value. This range is calculated using Equations 5.15 and 5.16 and it limits the definition interval of each crossing point to an interval around its position in the initial fuzzy system created by the expert. This restriction has two purposes. On the one hand, it limits the search space of each crossing point to a range that the expert considered appropriate when he created the knowledge base. On the other hand, it avoids both the relative displacement and the overlap of the membership functions given by the initial system, thus preserving the initial meaning given by the expert to each label. In Figure 5.9(a), it can be seen an example of a fuzzy variable Z_i with three membership functions $\{L_i^0, L_i^1, L_i^2\}$. The variable has two crossing points $\{P_i^0, P_i^1\}$. As it can be seen, the search space for each P_i^j is independent, thus avoiding both the overlap and the relative displacement between the membership functions. The range of each crossing point is calculated at the start of the process and remains the same during the evolution of the population.

$$Pleft_i^j = \begin{cases} Left_i & if\ j = 0 \\ \frac{P_i^j + P_i^{j-1}}{2} & otherwise \end{cases} \tag{5.15}$$

$$Pright_i^j = \begin{cases} Right_i & if\ j = n \\ \frac{P_i^j + P_i^{j+1}}{2} & otherwise \end{cases} \tag{5.16}$$

In order to increase the capability for adjusting the membership functions, a parameter that indicates the separation between them is used. It represents the separation of the upper points (c_i^j and b_i^{j+1}) of two consecutive membership functions L_i^j and L_i^{j+1} to its crossing point P_i^j. We shall call this parameter $s_i^j \in [0, 1]$ and it will be calculated as expressed in Equation 5.17.

$$s_i^j = \frac{P_i^j - c_i^j}{min\{P_i^j - Pleft_i^j, Pright_i^j - P_i^j\}} \tag{5.17}$$

When $s_i^j = 0$, there is no separation between c_i^j and b_i^{j+1}, i.e., $c_i^j = b_i^{j+1}$. On the other hand, if $s_i^j = 1$, the separation between c_i^j and b_i^{j+1} is the maximum allowed by the limits $[Pleft_i^j, Pright_i^j]$ of the crossing point P_i^j. To clarify the utility of this parameter, Figure 5.9(b) shows the example of a crossing point P_i^0 and the corresponding membership functions for two different values of s_i^0. While the membership functions represented by the solid lines correspond to $s_i^0 = 0.5$, the membership functions represented by dashed lines correspond to $s_i^0 = 0.25$. This parameter can be viewed as the degree of fuzzification, i.e., when the parameter s_i^j is 0 it corresponds to an interval discretization with no fuzziness at all [109].

Tuning of trapezoidal membership functions using real coding was also performed in [99], but the approach explained in this work reduces the number of parameters employed for each variable and thus the search space for the EA.

Hence, a complete fuzzy visual system is represented by a chromosome $\vec{x}^i$. It is encoded by joining the set of membership function definition parameters and the five threshold values $\{\alpha_1, \ldots, \alpha_5\} \in [0, 1]$ as in Equation 5.18. A total number of 75 parameters is thus required to represent the whole fuzzy visual system definition. In order to ease the notation, we will also denote the elements of each $\vec{x}^i$ by $(x_0^i, ..., x_{74}^i)$.

$$\vec{x}^i = (\alpha_1, \ldots, \alpha_5, P_0^0, s_0^0, \ldots, P_0^n, s_0^n, \ldots, P_n^0, s_n^0, \ldots, P_n^m, s_n^m) \tag{5.18}$$

5.4.2 Genetic Operators

Both a crossover and a mutation operator are required to apply the EAs selected for the tuning process. BLX_α [66] has been selected as crossover operator. This operator works by generating a random real value in an extended range (given by a parameter α) of the parents. Let us suppose that we want to cross over two chromosomes $\vec{x}^a = (x_0^a, ..., x_{74}^a)$ and $\vec{x}^b = (x_0^b, ..., x_{74}^b)$ to obtain an offspring $\vec{x}^c = (x_0^c, ..., x_{74}^c)$. The BLX_α operator generates

for each x_i^c a random value in a extended range $[BLX_{Inf}^i, BLX_{Sup}^i]$ given by the parent values x_a^i and x_b^i as shown in Figure 5.10. Nevertheless, in order to keep the coherence in the values of the offspring, the range $[BLX_{Inf}^i, BLX_{Sup}^i]$ can not exceed the range imposed by the coding scheme for each element x_c^i. It means that if x_c^i is either an $\{\alpha_1, \ldots, \alpha_5\}$ or a s_k^j parameter then $[BLX_{Inf}^i, BLX_{Sup}^i]$ must not exceed the range $[0, 1]$. Similarly, if x_c^i is a crossing point P_j^k then $[BLX_{Inf}^i, BLX_{Sup}^i]$ must not exceed the range $[Pleft_j^k, Pright_j^k]$.

Figure 5.10: Operation mode of the $BLX\alpha$ operator.

The mutation operator has been implemented by randomly selecting an element of $\vec{x}^a$ and assigning a random number in the range of possible values. In the case of the CHC algorithm, the perturbation operator has been implemented by randomly altering the 35% of an individual.

5.4.3 Objective Functions

In order to evaluate the performance of the solutions generated, a set of validated patterns is required. These patterns must be images of the environment where the system is going to work, containing scenes with and without doors. Let us denote by $I = \{I^0, ..., I^n\}$ the images in the image set I.

The measures employed to evaluate the system in the previous chapter (TPF and μ_{FP}) give an intuitive idea of the overall performance. They express the success and errors in terms of the number of instances of the fuzzy concepts that are correctly and incorrectly detected. However, they discard information about the individual segments thus avoiding information that might be important for tuning. Therefore, the two new metrics employed as fitness functions, f_1 and f_2, are defined to express the success of the system with a higher level of detail. They are explained below.

Let us denote all the segments extracted from an image I^i by IS^i (*Image Segments*) and

those that belong to architraves by $DS^i \in IS^i$ (*Door Segments*). When IS^i is passed to a fuzzy visual system $\vec{x}$, it returns only the set of segments that considers belonging to a door. Let us denote them by SS^i (*Solution Segments*). If the system correctly classifies all the segments, then $SS^i = DS^i$. Otherwise, there may be a subset of SS^i with segments that actually belong to the architrave, and there may be another subset of SS^i with segments that have been incorrectly considered as belonging to an architrave. Let us denote the former subset by $CC^i = \{SS^i \bigcap DS^i\}$ (*Correctly Classified*) and the latter by $IC^i = \{SS^i - DS^i\}$ (*Incorrectly Classified*).

In our problem there are two different objectives to be optimised. As first objective, it is desired to achieve a maximum positive detection (TPF), i.e., we want to detect all the segments that belong to architraves. It can be measured using the function $f_1(\vec{x}) \in [0, 1]$ defined in Equation 5.19 that is to be maximised. This function is 1 when $\vec{x}$ correctly returns all the segments of I that really belong to architraves. On the other hand, if it is 0, it means that $\vec{x}$ does not return any correct segment at all.

$$f_1(\vec{x}) = \frac{1}{n} \sum_{i=0}^{n} \frac{|CC^i|}{|DS^i|} \tag{5.19}$$

As second objective, it is desired that the fuzzy visual system rejects all those segments that are in the image but do not belong to architraves (TNF). This can be measured using the function $f_2(\vec{x}) \in [0, 1]$ defined in Equation 5.20. This function is 1 when no incorrect segments are returned by $\vec{x}$ and it is 0 in the opposite case.

$$f_2(\vec{x}) = \frac{1}{n} \sum_{i=0}^{n} \left(1 - \frac{|IC^i|}{|IS^i - DS^i|}\right) \tag{5.20}$$

When using single-objective approaches, we must combine the two objective functions into a single scalar function. Therefore, we employ Equation 5.21 that uses a parameter λ to independently weight the importance of f_1 and f_2. The use of λ directs the search towards a single direction of the multiobjective space. This operation can be seen referred in the literature as *plain aggregating approach* [56]. In order to obtain a set of solutions with different trade-offs between their objectives, it is necessary to run the single-objective

algorithm for different values of λ.

$$f(\vec{x}) = \lambda f_1(\vec{x}) + (1 - \lambda) f_2(\vec{x}) \tag{5.21}$$

5.5 Experimental Results

The aim of our experimentation is to tune the fuzzy visual system jointly optimising the two goals previously explained, f_1 and f_2. Therefore, we have employed two single-objective EAs: GGA and CHC, and the three MOEAs: SPEA, SPEA2 and NSGA-II.

The database of images explained in the previous chapter has been employed for the tuning purposes. It has been split into two subsets. The first one is to be used in the evolution process to evaluate the individuals generated (training set) that contains the 75% of the whole patterns. The other set (test set) has the rest of the patterns and is used to test the final solutions generated by the algorithms.

In order to be able to compare the solutions of the different algorithms, we have set almost the same conditions for the execution of all them. The same initial fuzzy system ($\vec{x}_0$) has been used for all the algorithms. Its labels have been uniformly distributed covering the whole range of the fuzzy variables. The individual $\vec{x}_0$ has the following fitness values $f_1(\vec{x}_0) = 0.841$ and $f_2(\vec{x}_0) = 0.884$.

Both GGA and CHC have been run 10 times using the fitness function showed in Equation 5.21 for the values $\lambda = [0, 0.11, 0.22, .., 0.88, 1]$ in order to emulate the behaviour of the MOEAs. The size of the population has been set to 100 and the number of iterations to 300. For the GGA, the percentage of individuals employed for the selection, crossover and mutation operators are 0.4, 0.5 and 0.1, respectively. In CHC, the $Decrement$ factor has been experimentally set to 0.08.

Either SPEA, SPEA2 and NSGA-II have been run 10 times using a population of 100 individuals. For SPEA and SPEA2, the number of individuals of the elitist population P^e has been set to 25. The total number of iterations employed for the three algorithms are 300 and the objective functions showed in Equations 5.19 and 5.20 have been used. All

Algorithm	#p	$\sigma_{\#p}$	$\mathcal{M}_2^*$	$\sigma_{\mathcal{M}_2^*}$	$\mathcal{M}_3^*$	$\sigma_{\mathcal{M}_3^*}$	$\mathcal{S}$	$\sigma_{\mathcal{S}}$
GGA	3	-	0	-	0.227	-	0.9971	-
CHC	5	-	2	-	0.910	-	0.9624	-
SPEA	19.1	2.26	2.635	0.795	0.503	0.053	0.9930	0.0071
SPEA2	17.4	2.80	3.066	0.783	0.516	0.070	0.9871	0.0120
NSGA-II	29.9	3.46	4.259	1.878	0.514	0.038	0.9972	0.0029

Table 5.1: Quality metrics values of the Pareto fronts obtained by GGA, CHC, SPEA, SPEA2 and NSGA-II

the algorithms uses BLX_α with $\alpha = 0.3$ as crossover operator and the random mutation operator.

We have used five different metrics for the comparison of the algorithms: four individual metrics, $\#p$, $\mathcal{M}_2^*$, $\mathcal{M}_3^*$, and $\mathcal{S}$; and one comparison metric, $\mathcal{C}$ (see Sect. 5.1.1 for information about the metrics). Metric $\#p$ is the number of non-dominated solutions in the final Pareto front. $\mathcal{M}_2^*$ measures the distribution of non-dominated solutions while $\mathcal{M}_3^*$ measures the extent of the Pareto front. The reason of using $\mathcal{M}_2^*$ and $\mathcal{M}_3^*$ (instead of $\mathcal{M}_2$ and $\mathcal{M}_3$) comes from the fact that we are interested in that the fuzzy visual systems learnt are well distributed in the objective space, to be able to obtain several fuzzy visual systems with different TPF-TNF trade-offs. $\mathcal{S}$ measures the area covered by the Pareto front. Finally, the comparison metric $\mathcal{C}$ measures the dominance of the solutions of a Pareto front over those of another.

Notice that, as our problem is composed of just two objectives, $\mathcal{M}_3^*$ corresponds to the distance between the objective vectors of the two outer solutions (hence, the maximum possible value is $\sqrt{2} = 1.4142$).

These metrics are appropriate to measure the quality of Pareto fronts. Nevertheless, they can not be applied to the results of the single-objective EAs. Since it is of our interest to compare all the algorithms, the 10 solutions obtained from the 10 runs of each single-objective EA have been gathered to create an aggregated Pareto front for each one. The four metrics previously indicated are also applied to these aggregated Pareto sets.

Table 5.1 shows the metrics results for the ten runs of the three MOEAs and for the

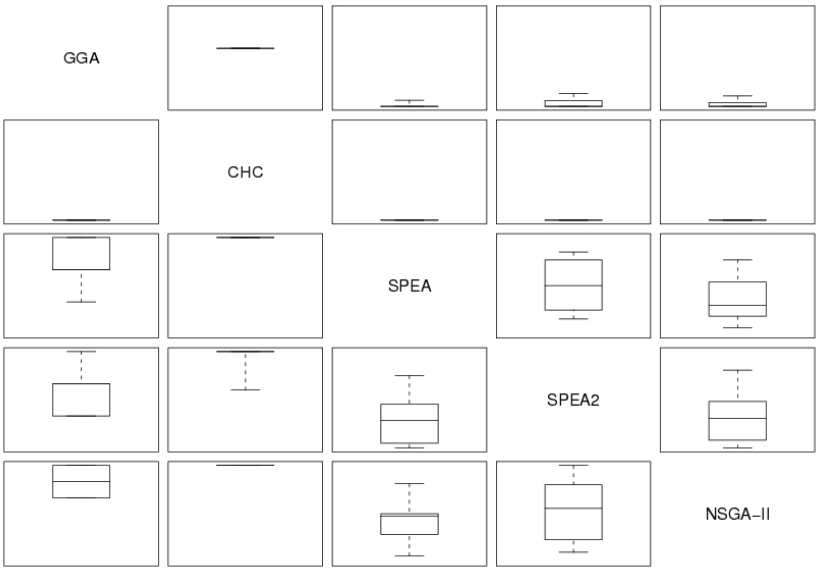


Figure 5.11: $\mathcal{C}$ metric values for the different runs of GGA, CHC, SPEA, SPEA2, and NSGA-II

aggregated Pareto fronts of GGA and CHC. From left to right, the columns show the values of the abovementioned metrics and their corresponding standard deviations in the 10 runs of the algorithms. In the case of GGA and CHC, the standard deviations can not be calculated because there is only one Pareto set. Parameter σ^* of $\mathcal{M}_2^*$ has been set to $0.1\sqrt{2}$, i.e., the ten percent of the maximum possible distance. Figure 5.11 graphically shows the values of the $\mathcal{C}$ metric for the different Pareto sets obtained in the form of box-plots.

As usually done in the experimental comparison of MOEAs in the specialised literature (see, for example, [226]), we have created a global Pareto front for each MOEA by aggregating the Pareto sets found in each of the 10 runs performed and removing the dominated solutions. Figure 5.12(a) shows the best non-dominated solutions found by SPEA in the different runs. Similarly, Figures 5.12(b) and 5.13(a) shows the best non dominated solutions found by SPEA2 and NSGA-II. Finally, Figure 5.13(b) depicts the aggregated Pareto sets from the 10 runs of GGA and CHC (those used to calculate the metrics). Note that in Figure 5.13(b) the range of the variables represented in the graph is different from the ranges of the other three graphs to be able to represent all the solutions.

In view of the results obtained, several conclusions can be drawn. On the one hand, it can

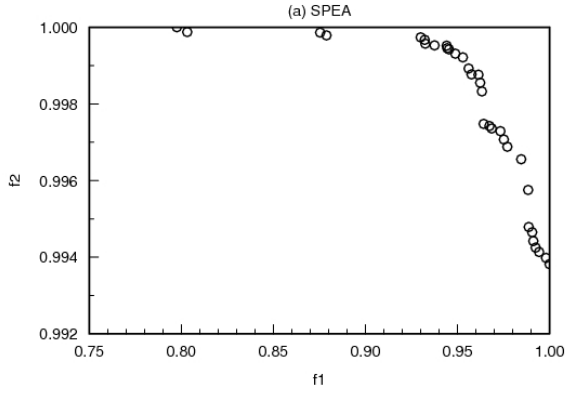


Figure 5.12: Aggregated Pareto fronts obtained by (a) SPEA and (b) SPEA2

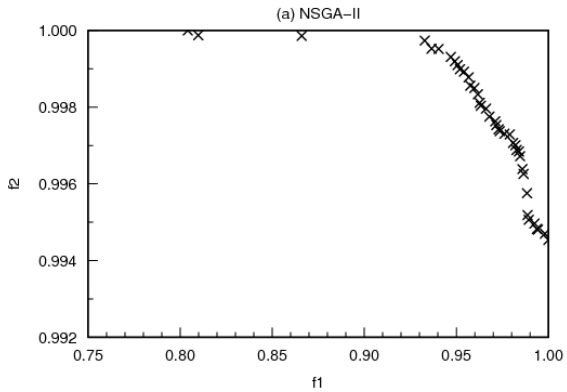


Figure 5.13: Aggregated Pareto fronts obtained by (a) NSGA-II and (b) GGA and CHC

be seen how the three MOEAs clearly outperform the two single-objective EAs considered. As expected, the direct application of a MOEA seems to be a most promising technique to obtain a Pareto set of non dominated solutions for our tuning problem than the consideration of repeated runs of a single-objective EA with different weights for the aggregated fitness function. The direct comparison between both single-objective approaches, GGA and CHC, shows better results of the latter in three of the four individual metrics: the number of solutions #p, the distribution metric $\mathcal{M}_2^*$, and the extension metric $\mathcal{M}_3^*$ (see Table 5.1). However, the results associated to the $\mathcal{M}_3^*$ metric can be confusing if analysed in isolation. Notice that CHC gets the best value of the five algorithms considered in this metric, with a significant difference (almost the double) with respect to the remainder. If we have a look to Figure

5.13, we recognise that the reason why the Pareto fronts generated by CHC are so spread in the problem space is because they have not converged properly to the real non dominated solutions, which are actually obtained by the remaining algorithms. Besides, GGA outperforms CHC in the remaining individual metric, the area $\mathcal{S}$, and in the comparison metric $\mathcal{C}$. Notice from the top left part of Figure 5.11 that, while no solution of CHC dominates the GGA ones in any case, the Pareto sets generated by GGA dominate CHC ones to a high degree.

Focusing the analysis on the three MOEAs, it can be first noticed how SPEA and SPEA2 perform in a similar way, although SPEA seems to be a little bit more adapted to our fuzzy visual system tuning problem than its improved version SPEA2. While the latter only outperforms the former in the distribution metric $\mathcal{M}_2^*$ (3.066 versus 2.635) and also, although very slightly, in the extension metric $\mathcal{M}_3^*$ (0.516 versus 0.503) (see Table 5.1), SPEA gets better values in the remaining three metrics. It founds more non dominated solutions (19.1 versus 17.4 in average), gets a slightly better value in the area metric $\mathcal{S}$ (0.9930 versus 0.9871), and the solutions in its Pareto sets clearly dominate those in SPEA2 ones in view of the boxplots of Figure 5.11.

The global best choice seems to be NSGA-II. Although the $\mathcal{C}$ metric show that the Pareto sets generated by this MOEA are of similar quality to those of SPEA, the former ones get the best results in every individual metric but in $\mathcal{M}_2^*$, in which CHC obtains the (false) best value as mentioned before. Besides, NSGA-II takes advantage of applying elitism in the main population, without needing an external one, to generate the largest Pareto sets with an average of 29.9 solutions. In Figure 5.13, it can be seen how this fact results in a significantly better distributed aggregated Pareto front, with a lesser number of gaps than SPEA and SPEA2 ones.

In order to verify that the solutions generated do not overfit the training data, we have evaluated their accuracy on the images of the test set. The solutions evaluated are those depicted in the Pareto fronts of Figures 5.12 and 5.13. Table 5.2 shows in columns for all the algorithms, from left to right: the minimum, maximum, average and standard deviation of the objective functions over the patterns of the test set. As it can be noticed, the results show

that the solutions generated are valid and do not overfit the training data.

Algorithm	$min(f_1)$	$min(f_2)$	$max(f_1)$	$max(f_2)$	μ_{f_1}	μ_{f_2}	σ_{f_1}	σ_{f_2}
GGA	0.9834	0.9865	1	0.9961	0.9944	0.9897	0.0073	0.0042
CHC	0.7107	0.8962	0.9834	1	0.9245	0.9615	0.0855	0.0305
SPEA	0.9049	0.9837	1	1	0.9826	0.9949	0.0138	0.0035
SPEA2	0.9132	0.9879	1	1	0.9836	0.9954	0.0146	0.0028
NSGA-II	0.9132	0.9900	1	1	0.9898	0.9955	0.0133	0.0027

Table 5.2: Results of the best individuals of the algorithms over the test set

As previously commented, the measures TPF and μ_{FP} employed in the previous chapter give a more intuitive idea performance of the system than f_1 and f_2. We have selected a tuned individual for evaluating its performance employing the TPF and μ_{FP} measures. The individual selected was generated by the NSGA-II algorithm and has the following fitness values $f_1 = 0.984$ and $f_2 = 0.989$. Table 5.3 shows the success of this individuals in both the trainind and test images for the three fuzzy concepts. As it can be observed, there is an important improvement in the system performance compared to the untuned version (Table 4.6). Therefore, the tuning process is highly recommended before exploiting the system.

Image set	IU		CA		IA	
	TPF	μ_{FP}	TPF	μ_{FP}	TPF	μ_{FP}
Training	0.982	0.015	0.965	0.010	0.866	0.049
Test	0.977	0.017	0.954	0.013	0.814	0.062

Table 5.3: Success of the a fuzzy system generated by the NSGA-II algorithm in detecting the fuzzy concepts IU, CA and IA

5.6 Final Remarks

In this paper, an evolutionary methodology able to tune the hierarchical fuzzy visual system employed door for detection has been proposed with the aim of adapting the system structure

to the specific characteristics of the environment where the mobile robot operates. Since the system performace is measured by two different conflicting criteria, the positive and negative rates of door detection, the tuning task becomes a multiobjective problem. Two different single-objective and three different multiobjective EAs have been considered for the problem solving, showing the better performance of the latter over the former by means of a sound experimental study.

Up to this point, the first goal established at the beginning of this thesis can be considered to be achieved. Chapter 3 has presented a multi-agent architecture for navigating in indoor environments where artificial landmarks are placed besides doors in order to indicate their position. As the use of artificial landmarks is not always possible, in Chap. 4 we designed a door-detector that eliminates the need of using them. Instead, doors are directly detected by detecting their architraves. Finally, this chapter has presented an evolutionary tuning mechanism that allows our door-detector to be adapted to a wide variety of indoor environments. Thus, our robot is capable now of navigating autonomously in fully unmodified environments, since no artificial landmarks are required to locate doors. The rest of this thesis is devoted to our second goal: developing a basis set of perceptual-motor skills that provide robots with basic interaction capabilities.

Finally, we must indicate that the work explained in this chapter has been accepted for publication in the journal *IEEE Transactions of Fuzzy Systems* [159].

Chapter 6

People Detection and Tracking using Stereo Vision and Color

The three previous chapters have been devoted to the development of techniques aimed to increase the autonomy level of mobile robotics. As we indicated at the beginning of this thesis, the goal of our work is twofold. The second goal is the development of perceptual-motor skills that enable robots to interact with people naturally. In that sense, people detection and tracking are important capabilities for future robots that have to achieve a natural human-robot interaction (HRI). A system able to detect and track multiple persons is presented in this chapter. It is specially designed for situations in which the camera must be placed at an under-head position. The position of the camera is a problem-dependent issue related with the purpose of the system. Several authors have mounted their cameras in the ceiling to perform people detection in Ambient Intelligence domains [95, 197]. Others have used slating cameras in overhead positions [11, 96, 98] to achieve the same functionality. Nevertheless, in most of the works that seek interacting with people, the position of the camera is usually lower than them [54, 167, 202]. This approach is mainly supported by two facts. On one hand, this camera configuration allows to see the faces and arms of the people and thus, be able to analyse their facial expressions and gestures. On the other hand, studies in the human-robot field reveals that people tends to feel threaten by big robots [73]. However, the main drawback of low camera positions is that occlusions between people are more frequent

that in elevated ones.

Most of people detection and tracking systems have an initial phase where a background model is created. It is employed to easily detect the moving objects in the environment (foreground). Techniques usually employed for that purpose consist in creating a background image (pixel by pixel) using several images of the scene, if possible, without motion elements in them [86, 96, 134, 197]. The simplest approach is to use the average of the sequence in each pixel as the background value. Others authors have used the median value and even Kalman filters to perform the update process. In this work, a height map of the environment (built using stereo information) is employed as background model. This technique has been previously employed by Darrell et al [53] to estimate the trajectory of moving people in a room. The background model is created in their work using several stereo cameras placed at different locations of the room. Height maps bring several advantages over traditional techniques to create background models. Firstly, because of stereo information is used instead of intensity values, the background model created is more invariant to sudden illumination changes. Secondly, the background model can be simultaneously created and employed by different devices placed at different locations. In fact, height maps have been widely used in mobile robotics to describe the environment and plan trajectories on them [37, 63, 185, 204]. Once the height map of the environment is created, the foreground is modelled as an occupancy map that registers the position of the moving objects in the environment. The creation of the height map is not always the best choice, i.e, in case of mobile robots with imprecise localisation methods. In these cases, the whole set of points captured by the stereo system can be employed to create the foreground map. The main disadvantage of this approach is that is slower than using a background model. However, the techniques to analyse the foreground are the same than these explained here.

There can be found mainly two approaches for people detection when using stereo vision in the related literature. The first one is considering a person as an object in an occupancy map with sufficient weight [95, 96, 98]. This approach is commonly used when the camera is placed at elevated positions. As previously commented in Sect. 2.8.5, the main problem of that approach is that objects with dimensions similar to human beings that enters in the

scene can be incorrectly detected as people. The second approach consists in looking for faces in the camera image [54, 91, 167]. This approach seems to be more appropriate when low camera positions are employed. However, if no additional information is employed, this approach is sensible to the false positives of the face detector. The system proposed in this work combines these two approaches to avoid the drawbacks of each one them. An object detected in the foreground occupancy map is considered as person if it has appropriate dimensions (human being dimensions) and if it is detected a face on it. To speed up computation, the face detector is only applied on selected regions of the image where it seems possible to find faces.

When a foreground object is identified as a person, the system starts to track him in the occupancy map. While tracking him, the system is able to keep tracking the rest of people previously detected and is still looking for new people. To track each person, the system combines information about his position (predicted using the Kalman filter [92]) with information about the colour of his clothes. Both pieces of information are dynamically combined in the following way. If a person is far from others, relying on the prediction of his position is safe. However, if the person is close to others, the prediction about his future position is not reliable because he could change his trajectory to avoid a collision or to interact with other people. In that case, information about the colour of his clothes is employed to enhance the tracking process whenever this information is relevant, i.e., only if the colours of his clothes differs from the colours of other's people clothes. It is important to remark that the face detector is only employed to detect people. However, once a person is detected, the tracking process does not employ the face detector. Thus, the person does not need to look at the camera to be tracked.

The remainder of this chapter is structured as follows. Section 6.1 explains the basis of the background modelling and foreground extraction techniques. In Sect. 6.2 it is shown how people detection and tracking are performed. Section 6.3 presents the experimentation carried out and Section 6.4 gives some final remarks.

6.1 Environment Map Building

In this section, the basis of the stereo processing, background modelling and foreground extraction are presented. The section is structured in three parts. Subsection 6.1.1 explains the basis of stereo calculation and how the 3D points captured by the stereo camera are translated to another reference system more appropriate for our purposes. Then, Subsection 6.1.2 explains the technique employed to create the background height map using the translated 3D points. Finally, Subsect. 6.1.3 explains how the height map is used to extract the foreground objects.

6.1.1 Stereo Processing

A commercial stereo camera [180] has been employed in this work. It can capture two images from slightly different positions (stereo pair) that are transferred to the computer to calculate a *disparity image* I_d containing the points matched in both images. Knowing the extrinsic and intrinsic parameters of the stereo camera it is possible to reconstruct the three-dimensional position p_{cam} of a pixel (u, v) in I_d. Let us denote by $P_{cam} = \{p^0_{cam}, \ldots, p^{np-1}_{cam} |\ p^i_{cam} = (X^i_{cam}, Y^i_{cam}, Z^i_{cam})^T\}$ the set of three dimensional points captured by the camera that are calculated using Eq. 6.1. Where, f is the focal length of the cameras, b is the baseline distance between the cameras and d the disparity value of the pixel (u, v) in the image I_d.

$$\begin{aligned} Z_{cam} &= \tfrac{fb}{d} \\ X_{cam} &= \tfrac{uZ_{cam}}{f} \\ Y_{cam} &= \tfrac{vZ_{cam}}{f} \end{aligned} \tag{6.1}$$

The three-dimensional positions p^i_{cam} calculated by Eq. 6.1 (affected by typical stereo errors [151, 184]) are referred to the stereo camera reference system. In our case it is centred at the right camera. However, this reference system may be changed from one application to another, i.e., the stereo camera can be placed at different positions and with different orientations. Hence, it is preferable for our purposes to translate the position of the points

captured to a *"world"* reference system placed at ground level and parallel to it. Knowing the position and orientation of the camera in relation to the floor plane, it is possible to calculate the linear transformation matrix T that translates the points p^i_{cam} into $p^i_w = (X^i_w, Y^i_w, Z^i_w)^T$ using Eq. 6.2. For more information about three-dimensional transformations the interested reader is referred to [71].

$$p_w = T p_{cam}. \tag{6.2}$$

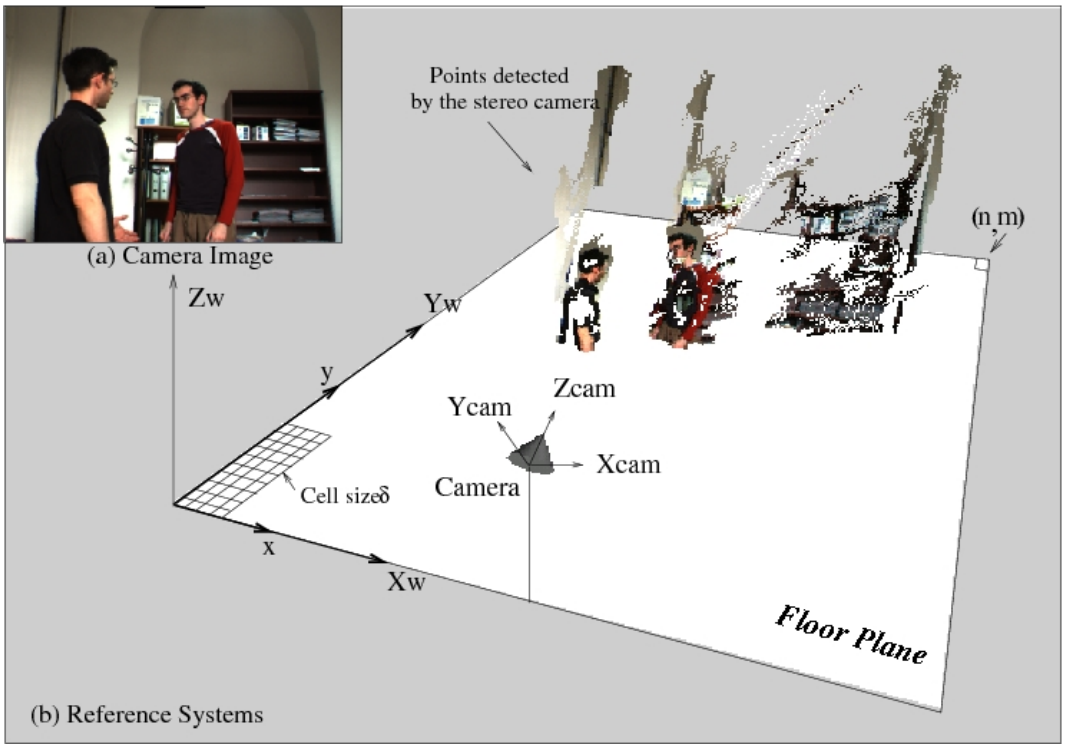


Figure 6.1: (a) Image of the right camera captured with the stereo system. (b) Three-dimensional reconstruction of the scene showing the reference systems employed.

Figure 6.1(a) shows an example of an scene captured with our stereo camera (the image corresponds to the right camera). Figure 6.1(b) shows the three-dimensional reconstruction of the scene captured using the points detected by the stereo camera. The *"world"* and camera reference systems have been superimposed in the Figure 6.1(b).

As it can be seen in Figure 6.1(b), the number of points acquired by an stereo camera can be very high (they are usually referred to as point cloud). For that reason, many authors

perform a reduction of the amount of information by orthogonally projecting them into a 2D plan-view map [95, 96, 98]. This decision is also supported by the fact that people do not tend to be overlapped in the floor plane as much as they are in the original captured images. Therefore, the detection and tracking process is more reliable in the 2D projection.

A plan-view map divides a region of the floor plane into a set of nxm cells of fixed size δ. In this work, the cell $(x, y) = (0, 0)$ coindices with the *"world"* positions $(0, 0, Z_w)$ (see Fig. 6.1(b)). Hence, the cell (x^i, y^i) in which a three-dimensional point p_w^i is projected can be calculated as:

$$x^i = (X_w^i/\delta)\ ;\ y^i = (Y_w^i/\delta) \tag{6.3}$$

Every time a stereo pair is captured, the set of points that are projected on each cell is calculated as:

$$P_{(x,y)} = \{i \mid x^i = x \wedge y^i = y \wedge Z_w^i \in [h_{min}, h_{max}]\}. \tag{6.4}$$

Where $[h_{min}, h_{max}]$ is a height range that has two purposes. On one hand, the superior limit h_{max} avoids using points from the ceiling or from objects hanging from it (i.e. lamps). On the other hand, the inferior limit h_{min} excludes from the process low points that could not be relevant for a particular application (floor points or even the legs of people) and thus helps to reduce the computing time. The height range $[h_{min}, h_{max}]$ should be such that, at least, the head and shoulders of the people to detect should fit in it. The rest of points p_w^i whose projection is outside the limits of the plan view map are not considered.

The selection of δ must be made taking into account several aspects. A high value helps to decrease the computational effort and the memory used. The side effect is that it causes a loss of precision in the estimation of the position. On the other hand, a low value increases the precision (up to the limit imposed by the errors of the stereo computation [151, 184]) but also the computational requirements. Ismail et al. in [95] propose the use of $\delta = 0.2$ cm while Harville in [96] uses $\delta \in [2, 4]$ cm. We have selected a value of $\delta = 1$ cm which according to our experimentation is an adequate balance between both requirements (precision and memory).

6.1.2 Background Modelling

Because people can be considered movable elements in the environment, it is very helpful to separate the points that belong to the environment (background) from those that do not (foreground). Our approach for background modelling differs from others in that it is based on the creation of a geometrical height map of the environment $\hat{\mathcal{H}}$ [53], instead of directly modelling the intensity values from the camera image. $\hat{\mathcal{H}}$ is a plan-view map that indicates in each cell $\hat{\mathcal{H}}_{(x,y)}$ the maximum height of the points projected in it. We might think of $\hat{\mathcal{H}}$ as a representation of the surface of the environment over which foreground objects move. To avoid including as part of the background objects momentarily passing by, $\hat{\mathcal{H}}$ is created aggregating several instantaneous height maps $\mathcal{H}^t$ with a robust estimator as the median:

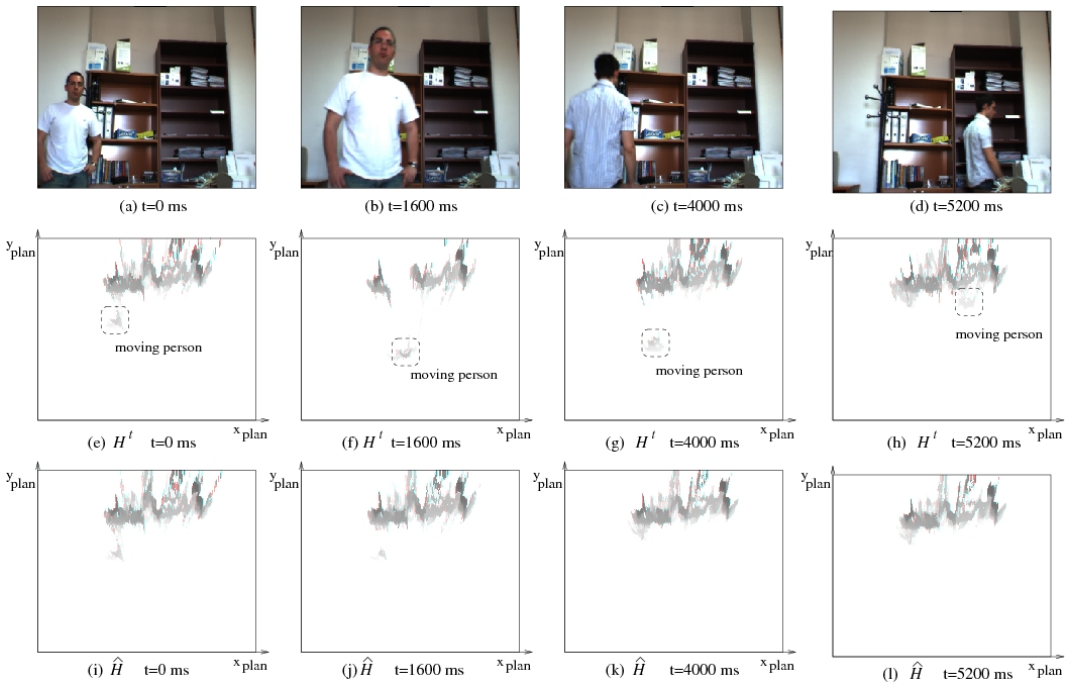


Figure 6.2: Creation of the height map. Upper row (a,b,c,d) shows the images in instants $\{0, 1600, 4000, 5200\}$ ms. Central row (e,f,g,h) shows the instantaneous height maps $\mathcal{H}^t$ for each one of the upper images. Lower row (i,j,k,l) shows the evolution of the height map $\hat{\mathcal{H}}$ created as the median of the height maps $\mathcal{H}^t$ created until that moment

$$\hat{\mathcal{H}}_{(x,y)} = median(\mathcal{H}^{t=t_0}_{(x,y)}, \ldots, \mathcal{H}^{t=t_0+\Delta t}_{(x,y)}). \tag{6.5}$$

Each cell of An instantaneous $\mathcal{H}^t$ is calculated as:

$$\mathcal{H}^t_{(x,y)} = \begin{cases} max(Z^j_w \mid j \in P_{(x,y)}) & if\ P_{(x,y)} \neq \emptyset \\ h_{min} & if\ P_{(x,y)} = \emptyset \end{cases} \tag{6.6}$$

Figure 6.2 shows the evolution of the height map $\hat{\mathcal{H}}$ from a set of 13 instantaneous height maps $\mathcal{H}^t$ captured at time intervals of $\Delta t = 400$ ms. Due to space reasons, the Fig. 6.2 shows only the status of the map at the time instants $t = \{0, 1600, 4000, 5200\}$ ms. Figures 6.2(a-d) (upper row) show the images captured by the right camera. Figures 6.2(e-h) (middle row) are the corresponding instantaneous height maps $\mathcal{H}^t$. Dark areas represent the highest zones and white areas represent the lowest ones h_{min}. Finally, Figs. 6.2(i-l) (lower row) show the evolution of the height map $\hat{\mathcal{H}}$ as more instantaneous height maps are employed to calculate it. Notice that $\hat{\mathcal{H}}$ has been created in the presence of people moving in the environment (their positions has been marked in the instantaneous height maps). As it can be seen, at the beginning $\mathcal{H}^{t=0} = \hat{\mathcal{H}}$ and thus the moving person appears in the height map. But as the time goes and more instantaneous height maps are employed to create $\hat{\mathcal{H}}$, it tends to truly represent the motionless characteristics of the environment. To create these maps we have used $h_{min} = 0.5$ m and $h_{max} = 2.1$ m.

$\hat{\mathcal{H}}$ can be periodically updated in order to dynamically be adapted to the changes in the environment. Nonetheless, to avoid old data influence in the update process and to save memory, the number of instantaneous height maps $\mathcal{H}^t$ employed to create $\hat{\mathcal{H}}$ must be limited to the more recent ones. According to our experimentation, $\hat{\mathcal{H}}$ can be appropriately updated using the last 10 instantaneous height maps. The frequency employed to update the height map should be smaller than the employed to detect and track people. In this way, it is possible to avoid including as part of the background people momentarily standing by or other moving objects. The update frequency employed in our experiments has been set to 0.1 Hz.

6.1.3 Foreground Extraction

The foreground points present in each captured stereo pair can be easily isolated using the height map $\hat{\mathcal{H}}$. To model the foreground we have employed a plan view-map $\mathcal{O}$ that is called *occupancy map*. $\mathcal{O}$ registers in each cell $\mathcal{O}_{(x,y)}$ the amount of points belonging to the foreground that are projected in it. Lets denote by

$$F_{(x,y)} = \{i \mid i \in P_{(x,y)} \wedge Z_w^i > \hat{\mathcal{H}}_{(x,y)}\},$$

to the set of points detected in a stereo pair, projected on the cell (x, y) and whose height is above the height indicated in $\hat{\mathcal{H}}_{(x,y)}$, i.e., the foreground points detected over the background surface that represents $\hat{\mathcal{H}}$. Each cell of the occupancy map is calculated as:

$$O_{(x,y)} = \sum_{j \in F_{(x,y)}} \frac{(Z_{cam}^j)^2}{f^2} \tag{6.7}$$

The idea is that each foreground point increments the cell in with it is projected by a value proportional to the surface that it occupies in the real scene [96]. Points closer to the camera correspond to small surfaces and vice versa. If the same increment is employed for every cell, the same object would have a lower sum of the areas the farther it is located from the camera. This scale in the increment value will compensate the difference in size of the objects observed according to their distance to the camera. Figure 6.3(b) shows the occupancy map $\mathcal{O}$ of the scene in the Figure 6.3(a) using the height map $\hat{\mathcal{H}}$ from Fig. 6.2(l). The darker values represent the areas with higher occupancy density. The image has been manually retouched to make the occupied areas visible. As it can be seen, there are small dark dots in the upper area of Fig. 6.3(b) that are caused by errors of the stereo correlation process. However, the person that stands in the scene is clearly projected in the occupancy map as a connected group of cell with high occupancy level.

The next step in our processing, is to identify the different objects present in $\mathcal{O}$ that could correspond to human beings. For that purpose, $\mathcal{O}$ is processed with a closing operator in order to link possible discontinuities in the objects caused by the errors in the stereo calculation. Then, objects are detected as groups of connected cells. Those objects whose area is similar to the area of a human being and whose sum of cells (occupancy level of

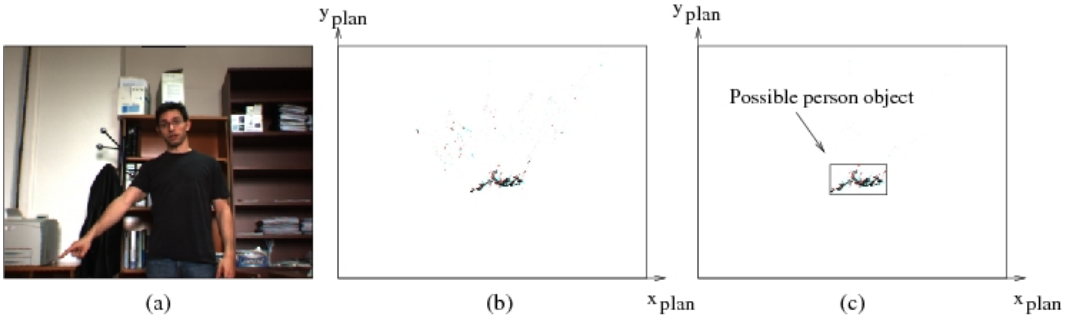


Figure 6.3: (a) Right image of the pair in an instant, the environment with an object not in background. (b) Occupancy map $\mathcal{O}$ corresponding to the environment. (c) Framed information corresponding to the object, detected using $\mathcal{O}$

the object) is above a threshold θ_{occ} are employed in the next phase for people detection and tracking. This test is performed in a flexible way so that it is possible to deal with the stereo errors and partial occlusions. Figure 6.3(c) shows the unique object detected in the occupancy map of Fig. 6.3(b) after the above mentioned process.

6.2 People Detection and Tracking

People detection and tracking are performed as separate processes. Every time a new scene is captured, the system must decide first, which one of the objects detected in $\mathcal{O}$ corresponds to each one of the people that are being tracked (an assignment problem). Then, the system applies a face detector on the remaining objects in order to detect new people. Notice that the face detector is only applied on those objects that have not been detected as people yet. That approach allows to manage false negatives detections of the face detector, i.e., if the face detector fails in detecting a person once, he could be detected in the next image.

Tracking people in that work consist in: (i) predicting their future positions according to its past movement using the Kalman filter [92] and, (ii) solving the assignment problem using the predicted positions as well as information about the colour of the objects. Therefore, a colour model of each object detected in $\mathcal{O}$ is created to be compared with the colour model of each person. The colour model of a person is he one created when he was first de-

tected that is updated each time the person is tracked. Both sources of information (position and colour) are combined dynamically to achieve a robust assignment in the following way. When a person is far from others, the system gives more importance to the prediction about its position. However, when a person is near others the system uses also information about the colour of his clothes to enhance the tracking.

In the next subsections these processes are explained in detail. Subsection 6.2.1 explains how the colour model of each object is created. Later, in Subsect. 6.2.2 it is shown how people detection is performed. And finally, Subsect. 6.2.3 explains how these pieces of information are fused to perform the tracking.

6.2.1 Color Modelling

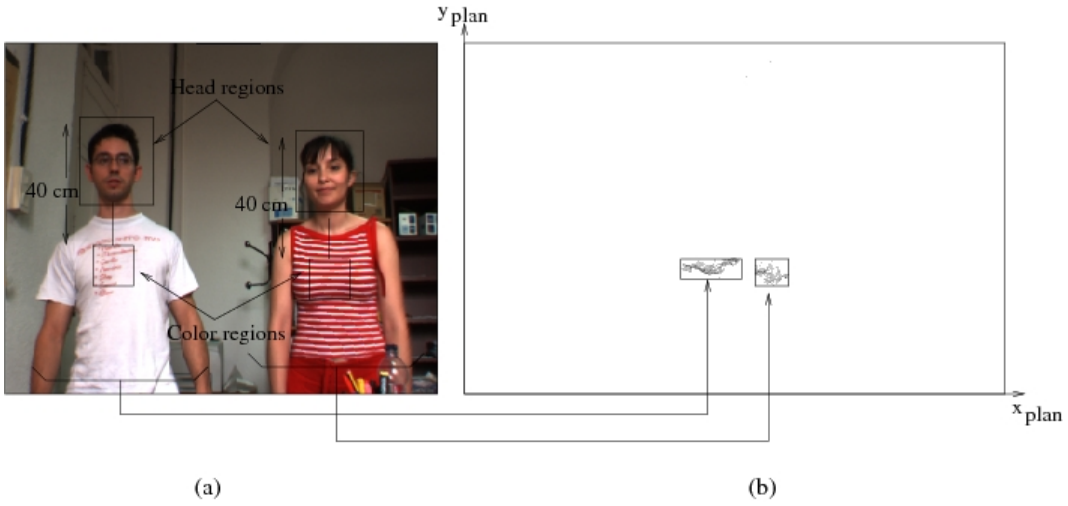


Figure 6.4: (a) Scene with two objects in it. The bottom black boxes are the regions used for creating their corresponding colour models (b) Occupancy map $\mathcal{O}$ of the scene

A colour model $\hat{q}$ of each object is created to assist the tracking process. The colour model aims to capture information about the colour of the clothes of the people in the scene. Thus, the pixels around what it should be the chest of a person are employed. For that purpose, the position of the chest p_{chest} in the camera image is estimated as 40 cm below the top of the head region. The head region is roughly estimated using the centre of mass of the

object in $\mathcal{O}$ and the disparity image I_d. The colour model $\hat{q}$ is created using the pixels in a region around p_{chest} whose size varies accordingly to the distance of the object to the camera. When the object is far from the camera, the region employed is smaller to avoid including pixels from the background and it becomes bigger when the object is near to the camera.

Figure 6.4(a) shows an image where there are two objects in the environment. The regions employed to create the colour models are depicted as boxes in the Fig. 6.4(a). Figure 6.4(b) shows the occupancy map of the scene in Fig. 6.4(a).

The colour model $\hat{q}$ of each object is modelled as an histogram using technique described by Comaniciu et al [48]. The HSV space [72] has been selected to represent colour information. The histogram $\hat{q}$ is comprised by $n_h n_s$ bins for the hue and saturation. However, as chromatic information is not reliable when the value component is too small or too big, pixels on this situation are not used to describe the chromaticity. Because these "colour-free" pixels might have important information, the histogram is also populated with n_v bins to capture its illuminance information. The resulting histogram is composed by $m = n_h n_s + n_v$ bins.

Let $\{x_i^*\}_{i=1...n}$ be the locations of the pixels employed to create the colour model. We define a function $b : \Re^2 \rightarrow \{1...m\}$ which associates to the pixel at location x_i^* the index $b(x_i^*)$ of the histogram bin corresponding to the colour of that pixel. The colour density distribution for each bin $\hat{q}_u$ of the region x^* is calculated in the following way:

$$\hat{q}_u = K \sum_{i=1}^{n} w(x_i^*) \kappa[b(x_i^*) - u]. \tag{6.8}$$

The weighting function w gives more relevance to pixels near the central point of the region x^* (p_{chest}) and thus reduces the influence of background pixels that might be incorrectly included. Function κ represents the Kronecker delta function. Finally, K is a normalisation constant calculated by imposing the condition $\sum_{u=1}^{m} \hat{q}_u = 1.$, from where

$$K = \frac{1}{\sum_{i=1}^{n} w(x_i^*)} \tag{6.9}$$

since the summation of the Kronecker delta functions is equal to 1.

Once the colour model $\hat{q}$ of an object is created, it can be compared with other colour

model $\hat{p}$ using the Bhattacharyya coefficient [6, 114]. In the case of the continuous distributions it is defined as:

$$\rho(q,p) = \int \sqrt{q(u)p(u)}du, \tag{6.10}$$

and for the discrete distribution of our colour models it can be expressed as:

$$\rho(\hat{q},\hat{p}) = \sum_{u=1}^{m} \sqrt{\hat{q}_u \hat{p}_u}. \tag{6.11}$$

The value $\rho(\hat{q},\hat{p})$ gives a measure of the similarity of two colour models in the range $[0,1]$ where 1 means that both colour models are identical and it decreases as the they differ. Figure 6.5(a) shows three people (objects) in the scene. The boxes on each person indicates the regions employed to create their colour models. Figure 6.5(b) shows as a table the values $\rho(\hat{q},\hat{p})$ for each pair of colour models using a total of $m = 30$ bins.

(a)

	0	1	2
0	1	0.339	0.359
1	0.339	1	0.094
2	0.359	0.094	1

(b)

Figure 6.5: (a) Scene with three objects in it. (b)$\rho(\hat{q},\hat{p})$ of the colour models of the objects

6.2.2 People Detection

Our approach for people detection consists in analysing if an object detected in $\mathcal{O}$ shows a face in the camera image. As previously indicated, the detection process is performed only on the remaining objects after tracking of known people has been completed.

Face detection is a process that can be time consuming if applied on the entire image, thus, it is only applied these on regions of the camera image where the head of each object

should be (head region). As the human head has an average width and height, the system analyses first if the head region of an object has similar dimensions. If the object does not pass this test, the face detector is not applied on it. This test is performed in a flexible manner so that it can handle stereo errors and people with different morphological characteristics can pass it. If the object passes the test, the corresponding region in the image analysed to detect if it contains a face. This reduction the search region where to apply the face detector brings two main advantages. First, it reduces the computational time as smaller regions are analysed. Second, it reduces the number of false positives as stated in [124].

The face detector provided by the OpenCv's Library [105] has been selected to detect if there is any face in the head region of the objects. It is not in the scope of this work to develop face detection techniques since there is plenty literature about it [220]. The face detector employed is based on the face detector of Viola and Jones [212] which was later improved by Lienhart [135]. The implementation is trained to detect both frontal and lateral views of human faces and works on grey level images.

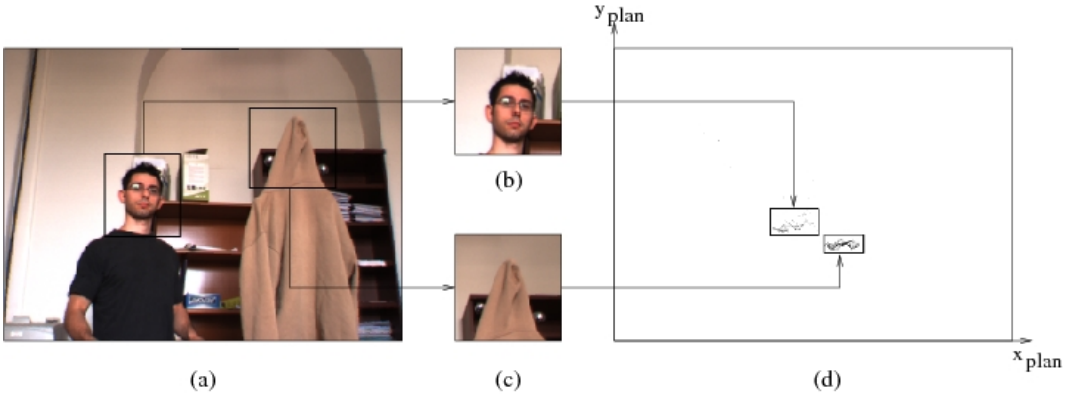


Figure 6.6: Example of people detection (a) Image captured by the camera (b-c) Images of the upper part of the objects detected. (d) Occupancy map of the scene.

Figure 6.6(a) shows a scene where there is a person that has entered and has hanged his coat. Figure 6.6(d) shows the occupancy map $\mathcal{O}$ of that scene. As it can be noticed, two objects are detected, the person and the coat. Using the procedure explained before, it is detected that the size of the upper part of the two objects are similar to human's heads. Hence, the regions of the image that should contain their faces (Figs. 6.6(b) and 6.6(c)) are

processed by the face detector indicating that there is a face only in Fig. 6.6(b).

6.2.3 People Tracking

Once an object has been identified as a person by the procedure explained before, it is necessary to keep track of him in the following images. The tracking problem can be seen as an assignment problem, i.e., relate a person that is being tracked with an object currently detected in $\mathcal{O}$. The problem has been solved using the Kuhn's well-known Hungarian Method for solving optimal assignment problems [125].

Let us denote by $\Upsilon = (\Phi, \Pi)$ the set of n objects detected in $\mathcal{O}$. The set

$$\Phi = \{obj^1, ..., obj^n \mid obj^i = (x^i_{obj}, y^i_{obj})\}$$

denotes the location of their centre of masses in the plan-view map, and $\Pi = \{\hat{p}^1, ..., \hat{p}^n\}$ their corresponding colour models, being $\hat{p}^i$ the colour model of the object obj^i.

Let also denote by $\Psi = (\varphi, \Omega)$ the set of m people detected and being tracked. The set $\varphi = \{s^1 ... s^m \mid s^j = (x^j_p, y^j_p, v^j_x, v^j_y)\}$, denotes their positions and velocities, and $\Omega = (\hat{q}^1, ..., \hat{q}^m)$, their colour models created when their faces were detected.

The assignment problem consists in determining the optimal assignment of currently detected objects Υ, to the people being tracked Ψ. In order to use the Hungarian method, it is necessary to calculate the probability value (or cost) of assigning the object obj^i to the person s^j. In this work, this probability value is calculated accordingly to two features. The first one is the difference between the position of the object and the predicted position for the person. The second one is the similitude between the colour models of the object and the person by Eq. 6.11.

Kalman filter is employed to predict the new position $s^j_{pred} = (x^j_p, y^j_p)$ of each person in the plan-view maps using a linear model of his movement:

$$x(t+1) = x(t) + v_x t;\ y(t+1) = y(t) + v_y t,$$

where v_x and v_y are the velocities of the person in the plan-view map. Although it is a basic movement model, it is able to successfully predict the position of people when images

are captured at short time intervals. Figure 6.7 shows the tracking results of a real sequence of 9 seconds captured at 7 Hz where a person walks 6.93 meters at steady speed. The solid line represents the observed path of a person moving in the environment while the dashed line represents the predictions of the Kalman filter. As it can be observed, the prediction model is able to estimate the trajectory of the person with a low error rate. Figure 6.8 shows the estimation errors. Notice that the maximum error does not exceed 15 cm, and it occurs when the person is turning.

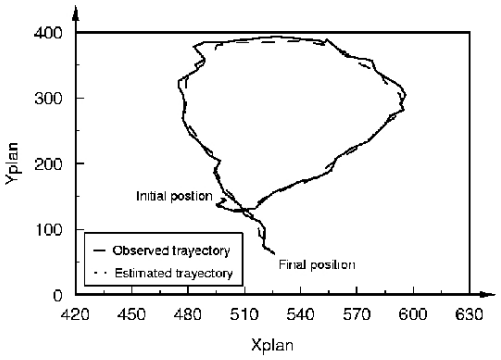


Figure 6.7: Example of trajectory of a person. Lines show both the observed positions and the positions estimated by the Kalman filter.

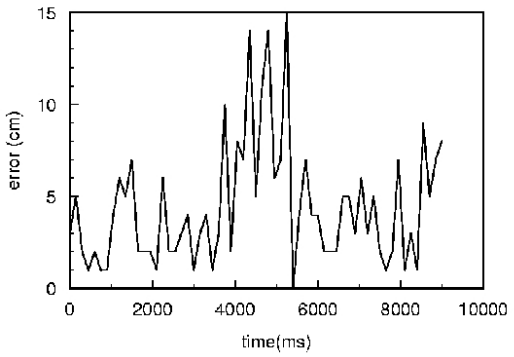


Figure 6.8: Errors in the predictions while tracking a person

In order to combine both pieces of information (colour and position) into a single probability value, Eq. 6.12 has been employed. It calculates the probability value of the object obj^i to be assigned to the person s^j in the range $[0, 1]$. The Eq. 6.12 assigns values close to 1 to indicate a high probability and vice versa. The first term of this equation measures the distance between the position of the object obj^i (x_o^i, y_o^i) and the prediction for the person s^j (x_p^j, y_p^j). Values close to 1 indicate that the object is near the predicted position of the person. The parameters σ_x^j and σ_y^j are a measure of the uncertainty associated to the position predicted for s^j and are given by the Kalman filter prior uncertainty matrix. They decrease when the person does not move and increase when the person moves (in proportion to the speed) or when the person can not be tracked in a image (indicating that his position has a

higher uncertainty).

The second term of the Eq. 6.12 compares the colour models of the object and the person via Eq. 6.11. Finally, the parameter $\alpha_m^j \in [0,1]$ is used to weight independently the importance of each term.

$$S(o^i, \hat{p}^i, s_{pred}^j, \hat{q}^j) = (1-\alpha_m^j) e^{-\left(\frac{(x_o^i - x_p^j)^2}{2(\sigma_x^j)^2} + \frac{(y_o^i - y_p^j)^2}{2(\sigma_y^j)^2}\right)} + \alpha_m^j \rho(\hat{q}^i, \hat{p}^j) \tag{6.12}$$

When a person is far from others, the errors in the predicted position are less important because they are less likely to cause an error in the assignment process. Hence, a low value for α_m^j can be selected. However, relying only on position information is inappropriate when an person becomes close to others. In that case, the person could change its trajectory to avoid a collision or start an interaction with another (think of two people that approach each other to shake hands). Therefore, their predicted positions would be erroneous and even confuse the system, i.e., the identity of a person would be incorrectly assigned to the object corresponding to another person. In these cases, it is desirable a higher value for α_m^j. Nevertheless, if two people are wearing clothes of similar colours, using colour information is also unreliable. To couple with all these situations, α_m^j is dynamically calculated for each person as:

$$\alpha_m^j = (1 - \rho(\hat{q}^j, \hat{q}^k)) * e^{-\left(\frac{(d_x^k)^2}{2(\sigma_x^j)^2} + \frac{(d_y^k)^2}{2(\sigma_y^j)^2}\right)}. \tag{6.13}$$

The pair (d_x^k, d_y^k) represents the displacement in the plan-view map from the person s^j to the nearest person in the scene s^k whose colour model is $\hat{q}^k$. If their colour models are similar then the term $(1 - \rho(\hat{q}^j, \hat{q}^k))$ tends to 0 and thus decreases α_m^j so the tracking of s^j is mostly based on his predicted position. However, if the similitude between their colour models is low, α_m^j is more affected by the exponential term. The exponential term tends to have low values when the person s^j is far from the nearest person in the scene s^k (so tracking is based mostly on position). Nevertheless, it tends to increase as the s^j is nearer to s^k. It also increases as the uncertainty about the position of s^j (σ_x^j, σ_y^j) increases, reflecting that his predicted position is not reliable.

Figure 6.9 shows the trajectories of two people (s^0 and s^1) in a real test and Fig. 6.10 the evolution of α_m^0 and α_m^1 (in the particular case of only two people in the scene $\alpha_m^0 = \alpha_m^1$). In that test, the similitude between the colour models of both people was $\rho(\hat{q}^0, \hat{q}^1) = 0.12$. When s^0 and s^1 begin to move, they are far from each other so α_m^0 and α_m^1 are high and the tracking is mostly based on information about the predictions of their positions. While they approach each other, the distance decreases so α_m^0 and α_m^1 increase and the tracking is more based on colour information.

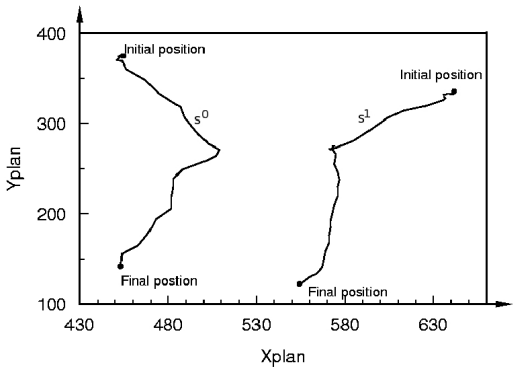


Figure 6.9: Trajectory of two people (s^0 and s^1) moving within the environment.

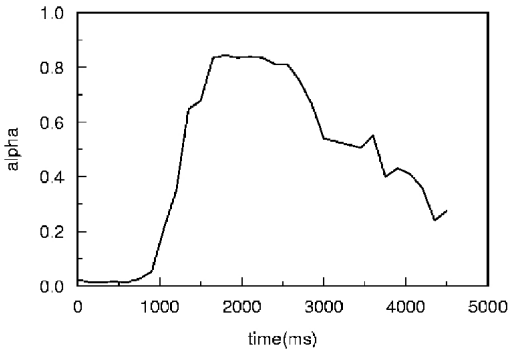


Figure 6.10: Values of the parameter α_m^0 and α_m^1 for the people s^0 and s^1 while they are moving

In order to obtain a robust tracking system it must deal with occlusions. If the person is partially visible, it will be detected as an object if the sum of its cells in $\mathcal{O}$ is higher than the threshold θ_{occ} (previously explained in Sect. 6.1.2). The value θ_{occ} is selected so that a person could be projected as an object in case of partial occlusion and dealing with stereo errors. However, if a person s^j is very occluded, it is not extracted any object from $\mathcal{O}$ for that person. In that situation the system must take care of not to assign an incorrect object obj^i to that person. For that reason, the system only consider valid an assignment when the probability value, given by Eq. 6.12, overcomes a threshold value θ_{assig}. If it does not happen for the person s^j, the system keep predicting his position and still looking for it for a maximum number of times. If the person s^j remain unseen for too long time, the system assumes that the person has definitively left the scene and deletes him from Ψ.

When the assignment problem has been solved, both the Kalman filter and the colour model of each *"assigned"* person are updated using the information of its corresponding objects. The colour model $\hat{q}^j$ of the person s^j is updated using the colour model $\hat{p}^i$ of its corresponding object obj^i as:

$$\hat{q}_u^j = (1 - \alpha_{cu})\hat{q}_u^j + \alpha_u \hat{p}_u^i, \tag{6.14}$$

where α_{cu} weights the contribution of the new observation to the update process [171].

6.3 Experimentation

During the explanation of the model we have shown examples of its performance. A broader experimentation has been done to test detection and tracking of different people under different illumination conditions and different distances from the vision system. To perform the stereo process we have used $320x240$ sized images and sub-pixel interpolation to enhance the precision in the stereo calculation. The operation frequency of our system is about 10 Hz on a 3.2 Ghz Pentium IV laptop computer running with Linux.

The face detector used has been configured to detect people at distances ranging from $0,5$ to $2,5$ meters. Although it could be configured for detecting people at larger distances, we have noticed that it substantially increments the time required to analyse an image. However, once a person has been located, it can be tracked up to distances of 5 meters. At higher distances, the errors of the stereo system employed are so high that people can not be correctly tracked.

A set of 18 colour-with-depth video sequence, captured at 7 Hz, have been recorded in order to test the performance of the tracking process and the influence of colour in it. The total time of all the sequences sum $10'23''$ and they were recorded for camera heights ranging between $0,5$ m and 1.2 m. Some of them were recorded using an stereo camera of $f = 6$ mm and the others using an stereo camera of $f = 4$ mm. The number of people in each sequence is different and it varies from 2 to 4. In the sequences, people perform several types of interactions: walk at different distances, shake hands, cross their paths, jump, run, embrace

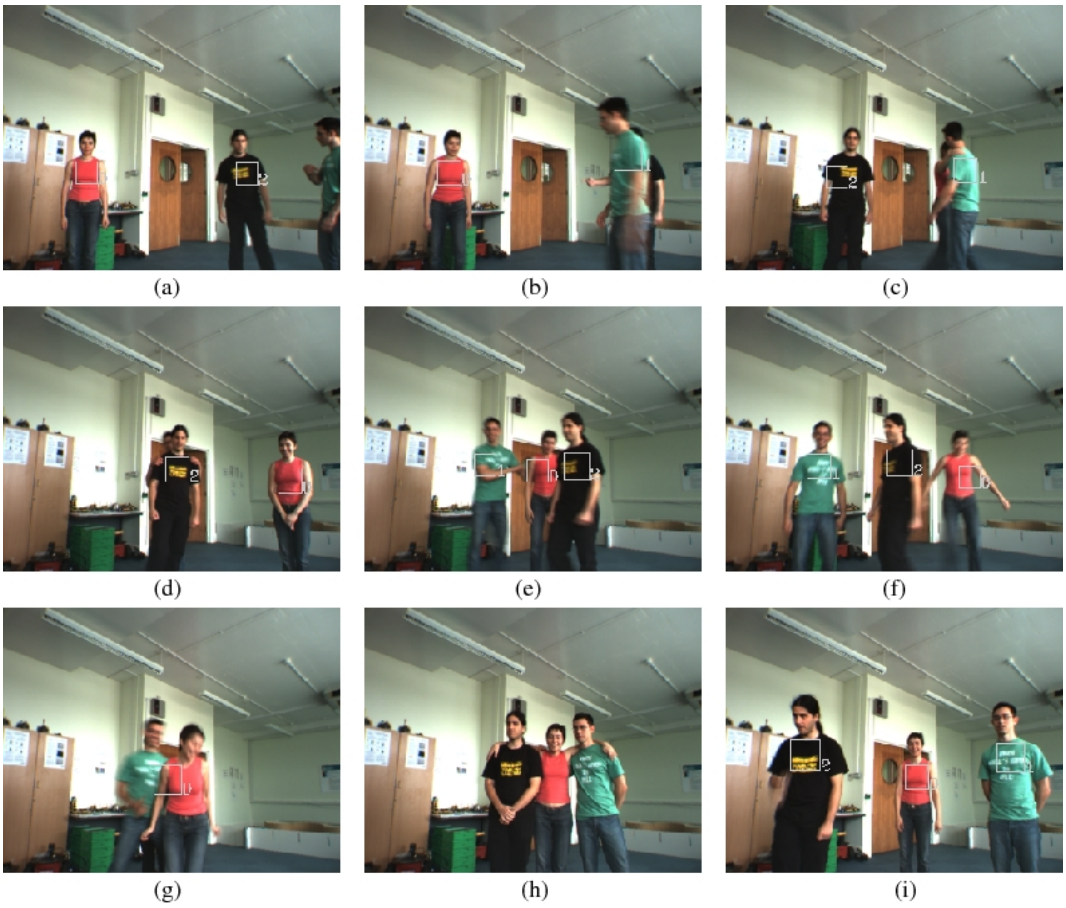


Figure 6.11: Images of a sequence employed to test the system.

each other and even quickly swap their positions trying to confuse the system.

To evaluate the success of the system in tracking people, we have manually count the number of potentially *"conflicting"* situations that takes place in each video sequence. A conflicting situation is considered when: (i) most of the body of a person is out of the camera image, (ii) a person is almost totally occluded by another person and (iii) two or more people collide, embrace or touch each other. Figure 6.11 shows some images of one of the video sequences. An example of the conflicting situation (i) can be seen in Fig. 6.11(a). Examples of the conflicting situation (ii) can be observed in Figs 6.11(b,c,g), and examples of the conflicting situation (iii) can be seen in Figs 6.11(e,h).

Each video sequence has been processes twice: one time using colour information and a second time without using it (i.e., setting $\alpha_m^j = 0$ for all the people). The processed video se-

#people	f	#conflicts	#nc	#c
2	6 mm	30	86%	100%
3	6 mm	17	58%	82%
3	4 mm	52	69%	100%
4	4 mm	13	50%	100%

Table 6.1: Success of the tracking system when using and not using colour

quences can be downloaded in *avi* format at *http://decsai.ugr.es/∼salinas/humanrobot.htm*. After processing them, the results have been examined and we have count the number of times that the system was able to successfully detect the object associated to each person when a conflicting situation had finished. Table 6.1 summarises the results obtained. Column *#people* indicates the number of people in the test, f the focal length of the camera used and *#conflicts* the number of all the conflicting situations counted in these video sequences. Column *#nc* indicates the success of the system in tracking the people in the conflicting situations without using colour information. Finally, column *#c* indicates the success of the system when colour information is employed.

At the light of the results showed in Table 6.1, it can be pointed out that the use of colour information is a powerful clue when tracking people. We have also observed that for the $f = 6$ mm stereo camera, more than 3 people makes the tracking process unreliable because of the excessive occlusions that occurs when people is near the camera. Similarly, the maximum number of people for appropriate tracking with the $f = 4$ mm camera is 4. It can be observed that as the number of people increases, the system is more prone to fail (specially when colour information is not employed). It must also be mentioned, that no false positives where detected on the video sequences. Thus, the combination of face detection and object detection in the occupancy map seems to be a very appropriate method for accurate people detection.

Additionally, we have observed that an important advantage of using colour is that despite the system can incorrectly assigns a person to the object of another person in a image, the error can be corrected in the next image if both people are wearing clothes of different

colours. This is because the colour comparison is continuously done and the correct assignment can be done in the next image (while the confused people still close). This correction is not possible when tracking is only based on position information. Nevertheless, if the colours of the clothes of the people in the scene are similar, the system mostly use information about their predicted positions and it is more likely to fail when people are close each other.

6.4 Final Remarks

We have presented a system able to detect and track multiple people using an stereo camera placed at under-head positions. The method proposed is especially indicated for applications that require analysing the user gestures and facial expression because of the position of the camera.

The system uses a height map built using depth information to model the environment that can be created and updated even in the presence of moving people in it. The background model obtained is more robust to sudden illumination changes than approaches based on intensity values because of the use of depth information [53].

Each time a new stereo pair is captured, an occupancy map that registers the position of the foreground objects is created. Foreground objects with dimensions similar to human beings are considered as potential candidates to people and a colour model of each one of them is created.

Then, the system performs the tracking of the already known people. Tracking is considered as an assignment problem, i.e., assign known people to the objects detected in the occupancy map. To calculate the best assignment scheme, the Hungarian Method [125] has been employed. For that purpose it is necessary to calculate a probability value of each object detected to be a known person. This probability value is calculated combining information about a colour model of each person and its predicted position using the Kalman filter. The combination is dynamically done is the following way. When a person is far from others, the probability value is mostly based on his predicted position. Nevertheless, when he comes

closer to others, position information becomes unreliable. Thus, the system takes into account information about the colour of his clothes to prevent confusing him with others. The degree of confidence assigned to colour information when tracking a person is based on the similarity between the colours of his clothes and the colours of the clothes of the people in his surroundings. Once the assignment problem is solved, both the Kalman filter and the colour models of the tracked people are updated.

The remaining objects not belonging to any known person are examined using a face detector. The aim is to detect the new people that enters in the scene. Instead of employing the face detector on the entire camera image, it is only applied on selected regions where is more probable to find the face in each object. This technique allows to greatly reduce the computing time required and helps to avoid false positives of the face detector [124].

The system has been extensively tested on 18 colour-with-depth video sequences (summing a total time of $10'23''$) where several people move freely in the environment. The video sequences have been processed twice, with and without colour, in order to analyse the influence of colour in the tracking process. The results show that the use of colour allows to greatly reduce the number of errors of the tracking system. The proposed system is able to track multiple-persons up to distances of 5 meters with very high success rate without using complex three dimensional models to describe the scene. Besides, the time required to perform the detection and tracking is only 40 ms, what induces as to think that it could be suitable for real time applications. It is also important to remark that no false detections were registered in the tests performed. Finally, we must indicate that part of the work developed in this chapter has been published in [163].

The next chapter deals with the problem of creating perceptual-motor skills to enable mobile robots to interact with human users. The chapter develops a multi-agent architecture using a philosophy very similar to the one employed in Chapter 3 but improved to allow a higher degree of reactivity upon events that takes place while interacting with human users. The module in charge of detecting and tracking users is based on the techniques developed in this chapter.

Chapter 7

Multi-agent based system for human-robot interaction using stereo vision

Part of the work presented in this chapter has been developed in collaboration with the Intelligent Mobile Robots and Creative Computing Research Group of the De MontFort University of Leicester. The chapter presents a multi-agent-based system that provides a basis set of perceptual-motor skills (or simply skills) useful for many mobile robotic applications that require to interact with human users. The skills designed constitute a first step for eliminating the use of conventional communication devices (e.g., touch screens, mouses) and replace them by natural interaction mechanisms (e.g., speech, facial gestures, postural gestures, etc). In that sense, the skills allow mobile robots to detect, track and follow human users, combining visual information (provided by an stereo system) with ultrasound information.

The system is divided in three layers and is based on the concept of building complex behaviours (skills) based on the combination of simpler behaviours. Agents in the lower layer wrap the real hardware, thus the system can be easily ported to different robot platforms. Agents in the middle layer implement basic behaviours that are able to control the movement of the robot and keep visual track of human users. The visual system is based on the work presented in the previous chapter. Finally, agents in the upper layer employ the behaviours

of the middle layer in a concurrent or sequential manner in order to implement skills. The skills were designed to allow the detection and tracking of a user without confusing him/her with other people and to navigate safely following him/her in indoor environments avoiding obstacles. The design of the system allows an easy expansion and/or modification to cope with ever changing environments and variety of applications. The system developed has been extensively tested with different users achieving very good results and real-time performance. Our detection and tracking visual approach has proved to: track users reliably while the robot is in movement, keep focused on the user despite of the presence of other people in its surroundings and be robust against complex illumination conditions. Applications such as personal robot assistants, robotic pets or guide robots might find useful the system presented in this paper.

The remainder of the chapter is as follows. Section 7.1 gives a description of the hardware employed and an overview of system architecture. Sections 7.2, 7.3 and 7.4 explains in detail the lower, middle and upper layers of the system respectively. Section 7.5 shows the results of the proposed architecture in several real experiments. Finally Sect. 7.6 draws some conclusions.

7.1 Multi-agent system architecture

7.1.1 Hardware description

The system presented has been developed to run both on the Nomad 200 and Peoplebot robots. Both of them are equipped with a ring of 16 sonars sensors that are employed to measure the distance to the surrounding obstacles. Altough both robots have in-board computers, we have opted for running the system on a laptop computer in order to have standard computational power for both platforms. The laptop employed is a Pentium IV running at 3.2 Gh that communicates with the computer of the robot via Ethernet using TCP/IP. The computer of the robot is used only to run a daemon that informs about the data from the sensors and manage the movement commands to the motors. All the agents of our system

run in the laptop computer under Linux OS.

In order to perform the visual processing, a pan-tilt unit (PTU) (model *PTU-D46-17* of *Direct Perceptions*) and an *Bumblebee* stereo camera (of *Point Grey Research*) have been used. The PTU is able to move 139º in the horizontal axis (both to the left and right side), 47º down-side and 31º up-side. This device allows to move the camera independently from the movement of the robot. The stereo camera connects to the firewire port of the laptop computer and is able to send images at 15 fps. The camera does not perform the stereo processing itself, it is done in the laptop computer via software.

Figure 7.1 shows the appearance of the robots for which the system is currently operative.

Figure 7.1: Robots for which the system is operative.

7.1.2 Architecture overview

The three layers architecture proposed for our system is shown in Fig. 7.2. The rounded rectangles represent agents. An *agent* can be seen in this work as an independent software process aimed to get or keep a goal implicit in its own design, and that has the capability to communicate with other agents. The lower layer contains agents that act as wrappers of the real hardware in order to abstract its particularities and allow for an easy portability. The

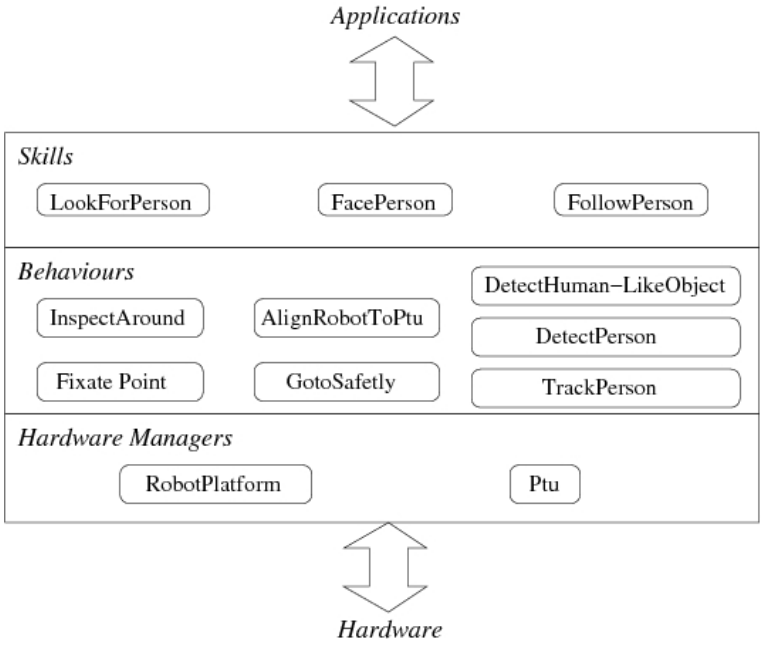


Figure 7.2: System Architecture.

middle layer is comprised of a set of agents that implement behaviours that make use of the services provided by the agents in the lower layer. Finally, the third layer is comprised of agents that employ the behaviours of the middle layer in a concurrent or sequential manner in order to implement complex behaviours (named *skills*). The skills implemented provide the base for a multitude of mobile robot applications that require to interact with human users. It is important to remark at this point that the aim of this work is to provide a set of skills that can be employed in real applications but not a particular application itself.

The *Hardware Managers* layer is comprised of agents that act as wrappers for the particular hardware of the system. *RobotPlatform* agent is an abstraction of the actual robot employed. It provides a concurrent use of the hardware platform to the agents that desire sending movement commands or requesting the sensors values. This agent makes the system independent on the kind of robot platform employed. Thus, the system can be run in different robots ,as far as they accomplish a minimum set requirements, by just changing the implementation of this agent. *Ptu* agent wraps the particularities of the PTU employed and allows the rest of the agent community to use it. The camera has not been wrapped by an

agent in this work because the transference of the images on the net would speed down the performance of the whole system. Instead, agents that requires the use of the camera can access the images through the mechanisms provided by the operative system.

The *Behaviour* layer contains a set of agents that implement simple behaviours. A behaviour can be seen as the capability of performing an action based on the current perceptions [14, 35]. Depending on the type of behaviour, the action can be a physical action like moving the PTU or an internal action or decision. And depending on the problem, the behaviours could be developed using a rule-based fuzzy system, neural networks or any other kind of desired technique. A total of seven agents have been employed in the architecture proprosed in this work. They are: *FixatePoint*, *InspectAround*, *GotoSafely*, *AlignRobotToPtu*, *DetectHuman-LikeObject*, *DetectPerson* and *TrackPerson*. These are explained in detail in Sect. 7.3.

The layer named *Skills* is comprised of a set of agents that develop complex behaviours based on the concurrent or sequential execution of the simple behaviours in the middle layer. These complex behaviours have been denominated *skills* and there is a total of three in our architecture: *LookForPerson*, *FacePerson* and *FollowPerson*. They are explained in detail in Sect. 7.4.

Finally, applications should be placed on the top of the third layer employing the skills provided to solve the person detection and tracking problem. Applications can be created combining the skills provided (in a sequential or concurrent manner) without needing to worry about the details that takes place at the lower levels. A wide range of mobile robotic application can take advantage of the skills proposed in this work.

7.1.3 Agent modes of communication

Each agent of our system has an unique name that identifies it. Agents can establish communication with other agents through TCP/IP protocol in two ways.

The first type of communication is when an agent knows the name of the agent that desire to communicate with and thus calls it directly to its corresponding port.

Besides, agents can also perform an *event-based* communication. An *event* is a anything

that happens, especially something important or unusual. Agents can be designed to detect events and inform to other interested agents about its occurrence by *emitting events*. An agent X that desires to receive information about the occurrence of an event emitted by the agent Y must register itself as *event-listener* of that event by notifying it to Y (on run time). Then, whenever the agent Y detects the event, it is emitted to all the registered *event listeners*. Two main advantages can be achieved using this *event-based* philosophy. On one hand, very re-usable agents can be designed because it is not necessary to know at design time the *event listener* of an agent. Hence, new agents added in posterior phases of the design can listen to previously implemented agents without any modification on the latter. On the other hand, as a long as a particular event does not occur, the *event-listeners* that reacts upon it can be waiting without performing any action thus the system saves resources to perform other tasks.

7.2 Hardware Managers

The *Hardware Manager* agents constitute the lower level of abstraction into our architecture. They provide basic hadware services to the upper layers hidding the particularities of the real platform employed. The main advantage of using this layer is that the system can be easily adapted to be fully operative on several hardware platforms, as far as the employed hardware accomplish a minimum set of requirements. The two agents in this layer are explained in detail.

a) RobotPlatform: This agent wraps the set of minimum services that the real robot provides. They are: (i) setting the translational and rotational speed of the robot, (ii) getting the current position of the robot using an odometric system and (iii) return the values of the sonar sensors. It must be indicated that the behaviours and skills developed in this work does not requires the use of the position of the robot to operate thus a localisation method has not been included as part of the architecture. However, a localisation agent could be included if required [5].

RobotPlatform agent allows the concurrent use of the robot to the rest of agents that desire both sending commands to it and requesting the sensor values. The agent communicates with the real robot at fixed time interval. It is called *frequency operation* of the agent. At each communication with the real robot, the agent gets information about the current status of the sensors and about the position, and sends the pending motion commands. During the period that the agent is not communicating with the real robot, it is listening to the requests of the agents in the upper layers. Petitions asking about the position of the robot or about the status of the sonar sensors are immediately answered using the last available information. However, petitions to move the robot are collected to be sent the next time that the agent communicates with the real robot. The petitions of movement must be sent with a priority value and only this with higher priority is sent to the real robot. When there are several conflicting petitions with equal priority, only the last one received is sent to the real robot and the rest are discarded.

If after few time intervals the agent has not received any new movement command, the robot is stopped. It avoids that old movement commands (sent by agent that crashes) remains being indefinitely executed compromising the safety of the robot and the people around it. Therefore, the behaviours that control the robot must work in a closed loop providing movement commands at the operation frequency of this agent. However, it is not a duty of this agent to detect and avoid possible obstacles while executing movement petitions. This is a task that must be solved in the upper layer.

One of the advantages of setting a frequency operation is that agents can adapt their own frequency operation and avoid to overload the network with continuous transmission of commands. Moreover, they leave free computing resources so that more agents can be concurrently employed in the same processor. In our experimentation, we have found that 6 Hz is an appropriate frequency operation for the *RobotPlatform* agent.

b) Ptu agent is an abstraction of the pan-tilt unit employed to move the camera independently from the movement of the robot. The agent provides concurrent access of the pan-tilt unit hardware to other agents in the system. It has been designed using the same philosophy employed for the *RobotPlatform*. The usual operation frequency for this agent is 6 Hz in our

experiments.

As previously commented, the camera has not been wrapped by any agent. This is because sending the stereo images through the net would slow down the operation frequency of the whole system. Instead, the camera is directly accessed by the agents that require the images.

7.3 Behaviours

The *Behaviours* layer is comprised of a set of agents that implements behaviours which in the upper layer are combined together to create more complex ones (named *skills*). This layer is composed by a total of seven behaviours (see Fig. 7.2). Once a behaviour is commanded to start, its execution does not stop either until (i) its goal is achieved (if it has any), (ii) it is detected a failure that makes impossible to continue or (iii) it is commanded to stop. The seven behaviours currently available in our system are:

a) InspectAround: This behaviour moves slowly the PTU around an indicated direction. It is employed when the system is actively looking for a person to interact with. As this is an endless process, it must be ordered to start and stop its execution and it will be done by the agents in the upper layer.

b) FixatePoint: This is a general-purpose behaviour that leads the PTU towards a three-dimensional point in the space so that this point becomes the centre of the following images captured by the camera. The behaviour is not directly called to perform the action but it responds to *tracked* events. The event *tracked* is an special kind of event that is emitted in our current system only by behaviour *TrackPerson* when the person that is being tracked has been detected. This behaviour is used to keep visual track of the person while he/she moves or while the robot moves. However, new behaviours that emit this kind of event could be added to track any kind of object. As in the previous case, this behaviour must be ordered to start and to stop listening to *tracked* events by an agent in the upper layer.

c) AlignRobotToPtu: This behaviour is employed in our system to avoid a person that is being tracked getting away from the field of vision of the PTU. Hence, when the angle

between the robot and the PTU increases, this behaviour is in charge of turning the robot to the appropriate direction in order to reduce it, thus making possible to track a person that moves around the robot. The behaviour is implemented as a rule-based fuzzy system that takes as input the angle between the horizontal axis of the PTU and the heading direction of the robot and generates as output the rotational speed that the robot must turn. It is an auto-regulated behaviour that adapts its operation frequency to the operation frequency of the agent *RobotPlatform*. It means that if the operation frequency of the latter is 6 Hz, this agent performs its operations at time intervals of $1/6$ s and if they are finished within that period of time, the agent waits the remaining free time in an idle state without consuming resources. As in the previous cases, an agent in the upper layer must command agent this behaviour to stop and start its execution.

d) DetectHuman-LikeObject, *DetectPerson* and *TrackPerson*: These three behaviour are related to the analysis of the images captured by the stereo camera to detect and track the presence of human users. The first one is in charge of detecting the presence of an object that, according to its dimensions, could belong to a human being (named *human-like* object). The second behaviour examines if any of the human-like objects detected really correspond to an human being by applying a face detector. Finally, the third tracks a detected person by combining information about its position (using the Kalman filter) and information about the colour of his/her clothes. A detailed explanation of this abilities is given in Sect. 7.3.

e) GotoSafely: This behaviour allows the robot to go from its current location to an indicated position (relative to its current location) avoiding the obstacles in the path. It is employed in our architecture by the *FollowPerson* skill to move the robot to follow a person (explained later in Sect. 7.4). The method employed to safely navigate is the *virtual forces fields* (vff) [28]. This technique models the desired position to go as an attractive force while the objects around the robot acts as repulsive forces. The sum all of these forces is a resultant force that indicates the most appropriate direction in each moment. In our case, the repulsive force are calculated employing the sonar readings and the attractive force is represented by the position of the person (calculated by the *TrackPerson* behaviour). The appropriate direction and speed for the robot are recalculated using vff at a frequency of 6 Hz in our

system, i.e., each time that new data from the sonar sensors of the robot is available and that the position of the person is updated.

People detection and tracking

The ability of detecting and tracking people is fundamental in robotic systems that desire to achieve a natural human-robot interaction. They are achieved in our architecture by combining information from stereo vision and colour (using the system developed in the previous Chapter) and are implemented as three separate behaviours *DetectHuman-LikeObject*, *DetectPerson* and *TrackPerson*.

The first behaviour is the base for the other two and it consists in detecting in the image captured by the camera, objects that according to their dimensions could be human beings (named human-like objects). The second behaviour employs the first one and it consists in analysing the detected human-like objects using a face detector in order to check if they really are human-beings. Finally, the third behaviour employs the first one and it consists in keeping track in the images captured of a specific human-like object (normally the user with which the robot is interacting). All these behaviours operates at a given frequency (normally at 6 Hz) to avoid employing all the available computing resources. The reason to split the detection and tracking into three different agents is that it provides more independence. It would allow to modify the technique employed to detect people without altering the tracking method and vice versa. Next, the basis of each behaviours are explained in detail.

Stereo processing and detection of human-like objects

As was explained in detail in the previous Chapter, our stereo camera captures two images from slightly different positions (stereo pair) that are transferred to the computer to calculate a *disparity image*. Using the knowledge about the intrinsic parameters of the stereo camera, the three-dimensional position p^i_{cam} of the points matched is calculated. As the camera reference system is affected by the PTU movements, it is preferable for our purposes to translate the position of the points captured to a *"robot"* reference system placed at the centre of the

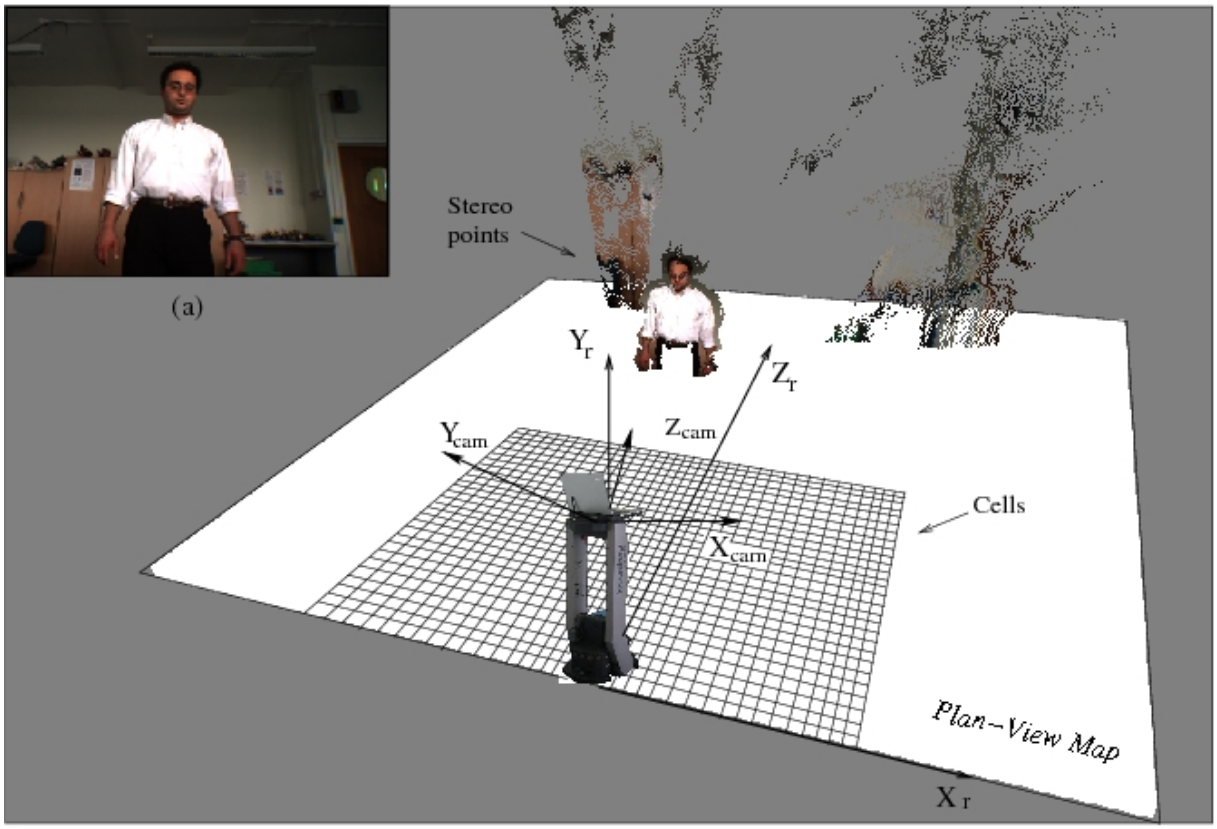


Figure 7.3: (a) Image of the right camera captured with the stereo system. (b) Three-dimensional reconstruction of the scene showing the reference systems employed.

robot at the ground level in the heading direction of the robot Z_r (see Fig. 7.3(b)). Knowing the position and orientation of the camera (pan and tilt angles), it is possible to calculate the linear transformation matrix T that translates the points p^i_{cam} into $p^i_r = (X^i_r, Y^i_r, Z^i_r)^T$ using Eq. 6.2.

Figure 7.3(a) shows an example of scene captured with our stereo camera (the image corresponds to the right camera). Figure 7.3(b) shows the three-dimensional reconstruction of the scene captured using the points detected by the stereo camera. The *"robot"* and camera reference systems have been superimposed in Fig. 7.3(b).

As indicated in the previous Chapter, the number of points captured by our stereo system is very high. In order to perform a reduction of the amount of information, the points captured

by the camera are orthogonally projected into a 2D plan-view map $\mathcal{O}$ (*occupancy map*). The plan-view map $\mathcal{O}$ divides a region of the environment into a set of cells of fixed size δ (see Fig. 7.3(b)). The cell (x^i, y^i) in which a three-dimensional point p_r^i is projected can be calculated as indicated in Eq. 6.3.

The occupancy map employed in this application differs from the defined in the previous chapter. This time, the environment has not been modelled by any height map. The main reason is that, in order to keep a background model when the robot is moving, it is necessary an accurate localisation method. We preffer to develop an architecture able to detect, track and follow people avoiding an accurate localisation mechanisms so that the system developed can be easily adapted to a wide range of hardware platforms.

Therefore, the occupancy map employed for this application registers in each cell $\mathcal{O}_{(x,y)}$ the amount of points that are projected in it, including background points. The set of points that projects on each cell of $\mathcal{O}_{(x,y)}$ is calculated as:

$$P_{(x,y)} = \{i \mid x^i = x \wedge y^i = y \wedge Z_r^i \in [h_{min}, h_{max}]\},$$

where the limit $[h_{min}, h_{max}]$ is employed to reduce the whole amount of points to an interesting height range. Finally, each cell of the occupancy map is calculated as:

$$\mathcal{O}_{(x,y)} = \sum_{j \in P_{(x,y)}} \frac{(Z_{cam}^j)^2}{f^2}. \tag{7.1}$$

We keep the idea, given in the previous chapter, that each detected point increments the cell in which it is projected by a value proportional to the surface that it occupies in the real scene [96]. Figure 7.4(b) shows the occupancy map $\mathcal{O}$ of the scene in the Figure 7.4(a). The darker values represent the areas with higher occupancy density.

The next step in our processing, is to identify the different objects present in $\mathcal{O}$ that could correspond to human beings (human-like objects). For that purpose, $\mathcal{O}$ is processed with a closing operator in order to link possible discontinuities in the objects caused by the errors in the stereo calculation. Then, objects are detected as groups of connected cells. Those objects whose area are similar to the area of a human being and whose sum of cells

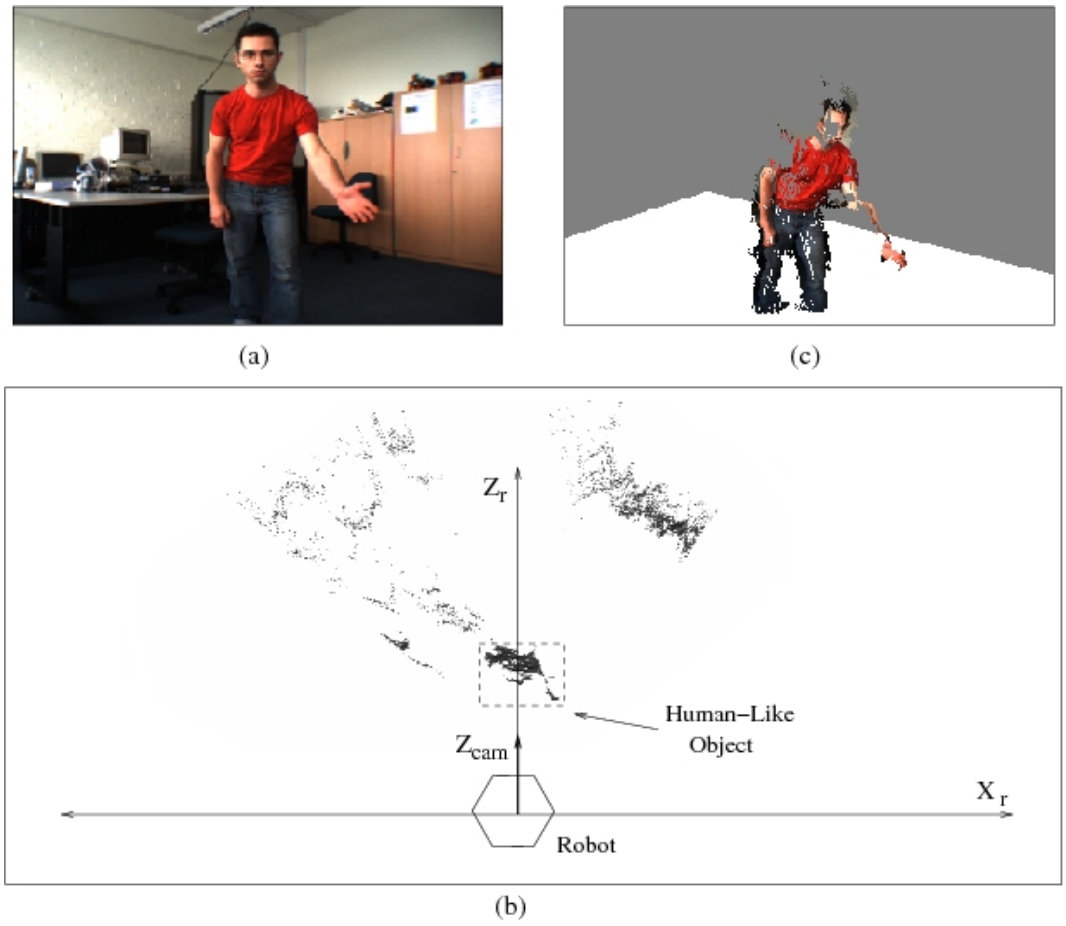


Figure 7.4: (a) Right image captured with the stereo camera. (b) Occupancy map $\mathcal{O}$ corresponding to the image (a). (c) Three dimensional reconstruction of the human-like object detected

(occupancy level of the object) is above a threshold θ_{occ} are considered human-like objects. This test is performed in a flexible way so that it is possible to deal with the stereo errors and partial occlusions. In Fig. 7.4(b) there is an example of human-like object that has been detected using the above mentioned process (its three-dimensional reconstruction can be seen Fig. 7.4(c)).

Once that the previous processing has been finished, the agent emits an event indicating the position of the human-like objects that have been detected.

Detect person

The approach employed to detect a person in this application consists in detecting if any of the human-like objects found in $\mathcal{O}$ shows a face in the camera image (as in the previous chapter). Therefore, it is necessary to detect first the human-like objects present in $\mathcal{O}$ using the previously explained method. As the human head has an average width and height, the system analyses first if the upper part of a human-like object has similar dimensions. If the object does not pass this test, the face detector is not applied on it. Face detection is only applied on these regions of the camera image where the head of each object should be (head region). Once that a face has been detected on a human-like object, a colour model of it is created [48]. The colour information will be employed to assist the tracking process in order to avoid confusions with other people in the environment.

As there can be detected several people at the same time in an image, it is necessary to select one of them as potential user. In this application we have opted for selecting the nearest person to the robot as potential user. Therefore, when one or more people are detected, the agent stores information about the position and colour model of the nearest one and emits an event indicating the detection. Then, it is the responsibility of the agents of the upper layer to decide what to do with that information.

Person tracking

This behaviour consists in tracking the person that has been detected by the previous method in the following images captured. It is a precondition for employing this behaviour that a person has been detected first using the *DetectPerson* behaviour.

Person tracking is tackled as an assignment problem each time a new stereo pair is captured, i.e., deciding which one the human-like objects detected in $\mathcal{O}$ (using the *DetectHuman-LikeObject* ability) correspond to the person that is being tracked. The method employed for tracking is the explained in the previous chapter, i.e., position and colour information are fused to assign a probability value to each human-like object detected in $\mathcal{O}$ indicating its probability to be the person being tracked. On one hand, a prediction of the future position

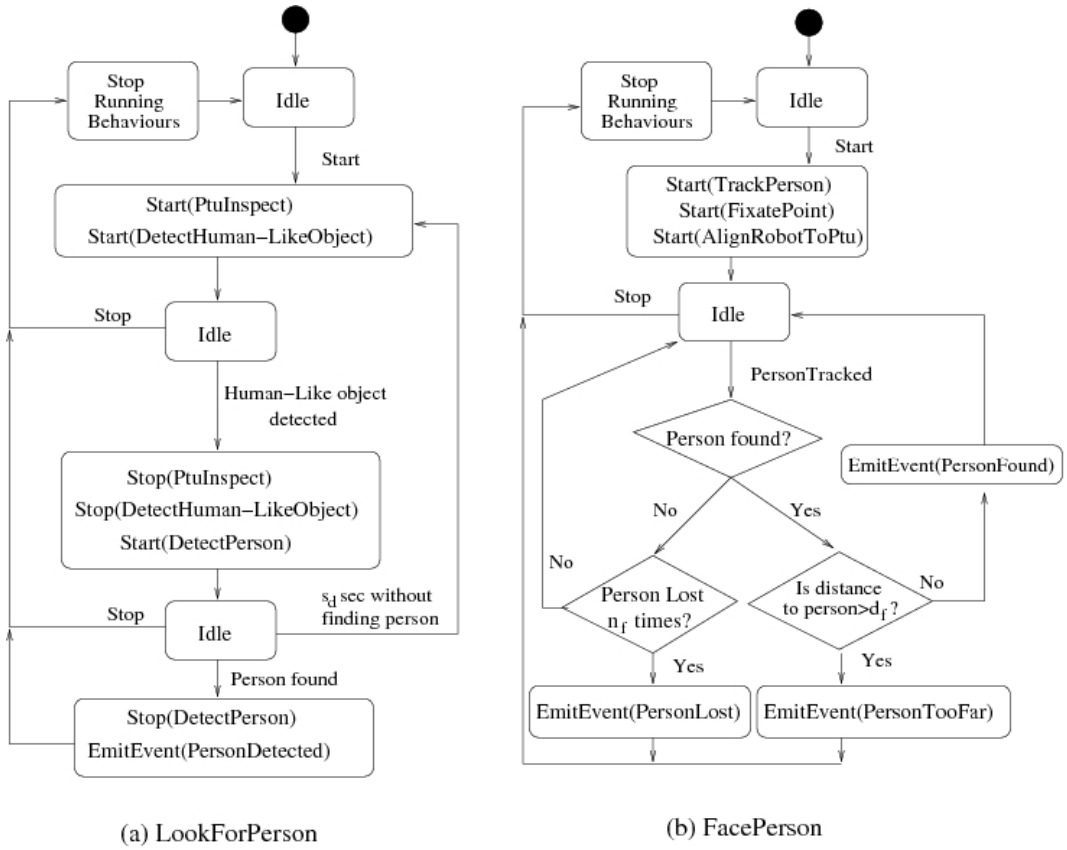


Figure 7.5: (a) Diagram of *LookForPerson* skill. (b) Diagram of *FacePerson* skill.

of the person in tracking is calculated using the Kalman filter [92]. The nearer a human-like object is from the position estimated for the person being tracked, the higher probability it will have to be him/her. On the other hand, colour information is employed to achieve a more robust tracking. The more similar the colour model of a human-like object is to the colour model of the person in tracking, the higher probability it will have to be him/her. Both probabilities are combined so that when the person in tracking is near others, colour information can help to distinguish him/her. The human-like object with highest probability is considered to be the person in tracking if its probability value exceeds a certain threshold. In that case, the Kalman filter is updated with the new observations and also the colour model of the person is updated so that it can adapt to the illumination changes that take place. Please notice that to perform the tracking it is not necessary the face of the person to track to be visible.

One of the drawbacks of the proposed method is that people wearing clothes of similar colours would be confused when they are near each other. This problem could be solved in future works with the use of additional information.

After that process, the agent emits a *tracked* event to inform about the results of the tracking. If the person has been successfully detected, the message contains the position of the person and a prediction of its future position. However, if the person is not tracked the *tracked* event is also emitted but indicating only the predicted position for the person and also indicating the uncertainty associated with that prediction (given by the prior uncertainty matrix of the Kalman filter). In that way, even if the person is lost momentarily, the system can continue working with the predictions until the person is found again. Nevertheless, if the person remains unseen for too long and the uncertainty about its position becomes very high, it would be necessary to restart the detection process. However, this decision is not taken at this level but at higher ones.

7.4 Skills

The layer named *Skills* is composed by a set of agents that develop particular *skills* based on the concurrent or sequential execution of the previously explained behaviours. The execution of an skill implies a perceptual-motor coordination, i.e., a skill coordinates perception and movement in order to react upon the user actions by an adequate activation/deactivation of behaviours in the lower layer. The proposed architecture is designed in such a way that the developer of new applications that requires the robot to interact with people must only use the agents of this layer and avoid any communication with agents at lower layers. The idea is to abstract the developer of applications of the processes that take place at lower levels providing a clear interface to interact with the agents that implement the skills. Each skill implemented can be commanded to stop and start its execution by other agents in the application layer. Besides, each one of them emits particular events to inform the applications about its status. The three skills currently implemented in our system are shown in Fig. 7.2.

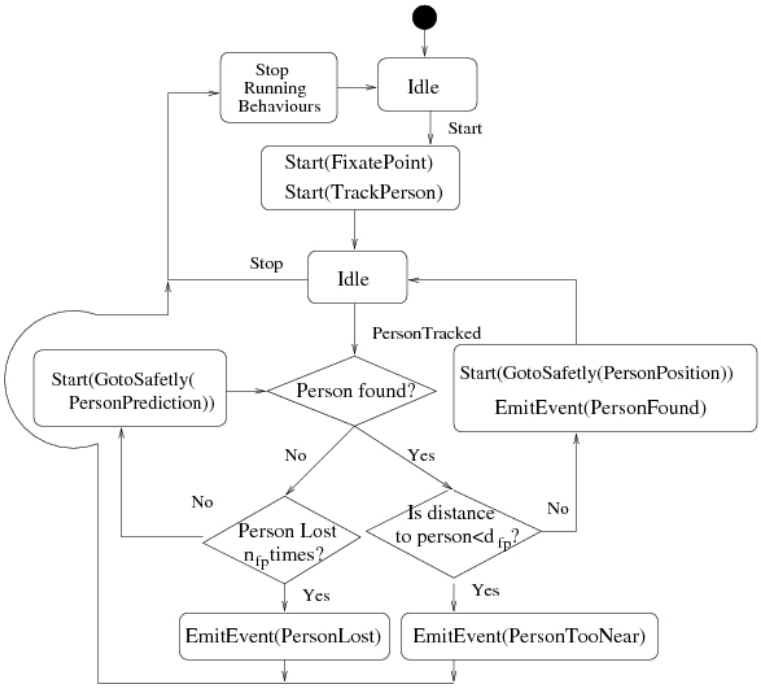


Figure 7.6: Diagram of *FollowPerson* skill

a) LookForPerson: This skill is employed when it is desired that the robot starts an interaction with a user. This skill makes the robot stand in its current position moving the PTU performing an horizontal scan until a human-like object appears in its proximity. Then, the robot analyses the human-like object looking for a face. If so, the agent emits an event indicating that a person has been found and stops. If no face is found in the human-like object after s_d seconds, the robots returns to the previous task of moving the PTU looking for human-like objects. The parameter s_d is a configurable parameter that can be set by the applications.

The diagram of the skill is shown in Fig. 7.5(a). This skill is achieved through the concurrent execution of the behaviours *InspectAround* and *DetectHuman-LikeObject* and *DetectPerson*. At the beginning of its execution, the skill is in an idle state waiting for receiving the *Start* command from other agent in the application layer. When it is received, the skill starts the behaviours *InspectAround* that moves the PTU to scan the environment and the behaviour *DetectHuman-LikeObject* that examines the stereo images to detect human-like objects. After that, the skill waits in a idle state until a human-like object is found in the

proximity of the robot. This skill register itself as event listener of the events emitted by the agents *DetectHuman-LikeObject* and *DetectPerson* so it can know when a human-like or a person has been detected. When the agent *DetectHuman-LikeObject* emits an event indicating that a human-like object has been detected, this skill stops the running behaviours and starts the *DetectPerson* behaviour. This makes the PTU stop, pointing in the direction where the human-like object was found, and starts analysing if it has a face. The skill waits in this state a maximum number of seconds s_d to receive the event from behaviour *DetectPerson* indicating that a person has been found and stops its execution. If a person is found, this skill emits an event indicating it to the upper applications. If no person is found after s_d seconds, this behaviour re-start the execution of the behaviours *InspectAround* and *DetectHuman-LikeObject* to continue looking for human-like objects in the proximity of the robot.

It is important to remark at this point that this skill can be stopped at any point if an agent in the application layer sends it an *Stop* message. In that case, the *LookForPerson* skill stops all the behaviours that might be currently running and goes to the initial idle state.

b) FacePerson: This skill is employed once a person has been detected near the robot. Its aim is to keep track of the person while he/she walks around the robot at near distances even if he/she looks away from the robot. For that purpose, the robot keeps track of the person moving the PTU to keep always him/her in the centre of the images captured. Besides, when the horizontal angle of the PTU is not null (i.e., the person is not in front of the robot), the robot turns over itself to the direction of the person. This skill can stop its execution for two reasons. The first one is that the person being tracked can not be seen for a maximum number of times n_f (i.e., a tracking failure). As commented in Sect. 7.3, if the person is not correctly tracked, it is possible to know its estimated position given by the Kalman filter. This skill relies on that prediction for a maximum number of samples. However, if the person remains untracked for more that n_f samples, the skill considers that the predictions are no longer reliable and stops its actions. The second reason that makes this skill stop, is that the distance between the person and the robot becomes greater than d_f. Whether the robot stops because of the first condition or because of the second, appropriate events are emitted

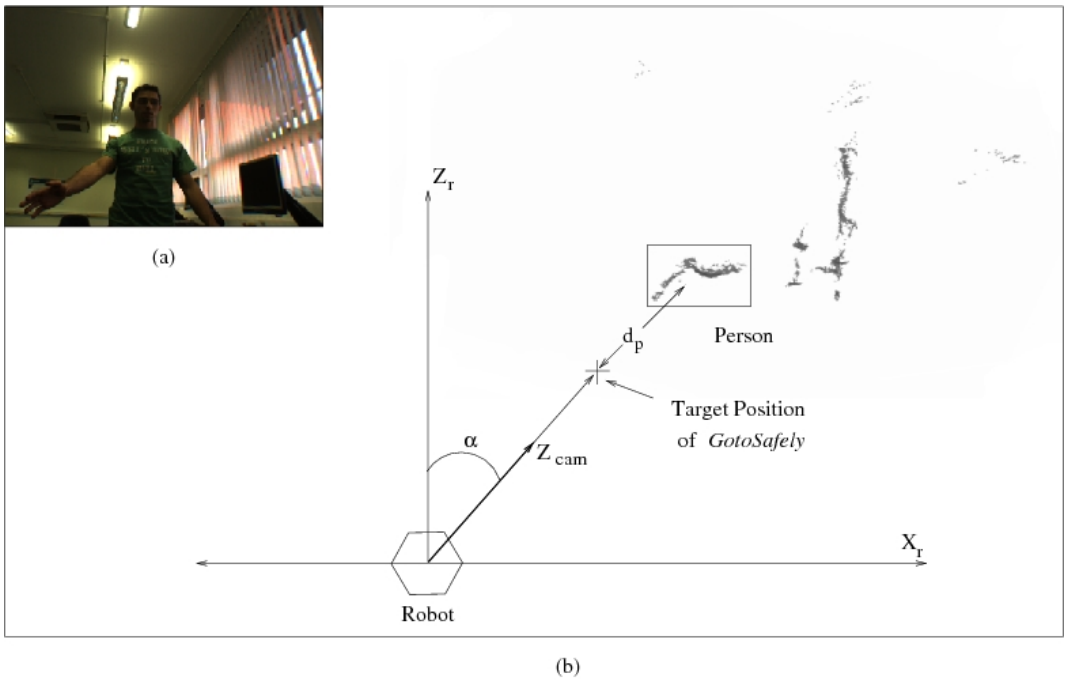


Figure 7.7: (a) Image taken with the camera of the robot (b) Scheme of the scene where the image was captured

to keep informed the applications in the upper layer. Besides, both parameters n_f and d_f can be configured by the applications on run-time to fit the desired requirements.

The diagram of this skill is shown in Fig. 7.5(b). When this skill is commanded to start by an application, it starts the behaviours *TrackPerson*, *FixatePoint* and *AlignRobotToPtu*. The first one, is employed to keep track of the person that has been detected in a previous phase by the *LookForPerson* skill. The second one, *FixatePoint*, is commanded to listen to the *tracked* events sent by *TrackPerson* so that whenever the person is tracked, the PTU is directed towards its direction. Besides, the *AlignRobotToPtu* behaviour is continuously analysing the angular difference α between the heading direction of the robot and the horizontal angle of the PTU. Figure 7.7(b) shows the orthogonal view of the scene captured in Fig. 7.7(a) where the parameter α has been depicted on it. This behaviour makes the robot to spin around itself to make $\alpha = 0$. The result, is that the robot turns to the same direction that the person being tracked is moving to. While these behaviours are running, the skill is listening to the *tracked* events sent by *TrackPerson* to detect if the tracking has failed more than

n_f consecutive samples or if the person has gone away a distance greater then d_f. In either case, this skill emits particular events to inform the applications and stop its execution and the execution of the active behaviours. However, whenever a person has been successfully tracked, it emits an event to keep the applications informed.

c) FollowPerson: This skill aims to make the robot follow the person with which it is interacting avoiding possible obstacles. When it is started, the robot tracks the person using the PTU to keep him/her always in the centre of the image while commands the robot move towards the direction of the person. The target position for the robot is placed d_{fp} meters from the person in the straight line that joins both the robot and the person (see Fig.7.7(b)). This target position is updated every time the new position of the person is acquired using the tracking information (6 Hz in our experiments). The skill can stop its execution for three reasons: (i) it reaches the target position, (ii) the person can not be tracked for a maximum number of samples n_{fp} or (iii) it is commanded to stop by an application. This skill emits events to the applications to indicate the current state of the process. Both the parameter d_{fp} and n_{fp} can be configured by the applications.

The diagram of this skill can be seen in Fig. 7.6(a). This skill is achieved by the concurrent execution of the three basic behaviours *TrackPerson*, *FixatePoint* and *GotoSafely*. At the beginning of the execution of *FollowPerson*, it waits in an idle state until the *Start* command is received. Then, it starts the behaviours *TrackPerson* and *FixatePoint* to keep track of the person moving the PTU so that he/she remains always in the centre of the images captured. The skill is event listener of the *TrackPerson* behaviour so it receives the *tracked* events emitted. Whenever an new *tracked* event is received, it analyses if the person has been successfully tracked. If so, the distance from the person to the robot is calculated and if it is below the threshold d_{fp}, the skill stops its execution and emits an event to inform the applications above it. If the distance from the person is greater than d_{fp} the behaviour *GotoSafely* is commanded to go to a position placed in the straight line that joins the robot and the person at a distance d_{fp} away from the person (see Fig.7.7(b)). If the person can not be tracked the predicted position for the person is employed to calculate the new target position. However, if the person remains untracked more than n_{fp} samples and its prediction

becomes unreliable, this skill stops its executions and emits an event to inform to the upper applications.

7.5 Experimental Results

A total of five set of experiments have been performed in order to test the proposed system. The first set tests the capacity of the behaviour *DetectHuman-LikeObject* to detect people as human-like objects at different distances. The second set evaluates the performance of the behaviour *DetectPerson* to detect several people at different distances. These two set of experiments evaluate the basic behaviours employed by the *LookForPerson* skill. A third set of experiments has been performed in order to evaluate the process of detecting a person using the *LookForPerson* skill. These tests evaluate whether the integration of perception and motion is appropriate in order to achieve the goal established for the *LookForPerson* skill. Finally, the fourth and fifth set of experiments evaluate the *FacePerson* and *FollowPerson* skills respectively.

Experiments for *DetectHuman-LikeObject* behaviour

The aim of this set of experiments is to evaluate the performance of the *DetectHuman-LikeObject* behaviour. As detecting human-like objects constitutes the first step for both people detection and tracking behaviours, the success of this behaviour sets the maximum performance expected for the others.

The experiments evaluate the success of the behaviour in detecting people moving in front of the camera at different distances as human-like objects. To perform the experiment, a person was instructed to walk naturally for approximately 30 seconds in the field of view of the camera keeping a fixed distance while a colour-with depth video sequence was being recorded.

The video sequence was recorded at 15 fps and during which both the camera and the robot were still. In addition, the person was instructed to avoid looking at the camera all the time but also to appear in lateral and backwards positions. The environment was prepared

so that only the human-like object, who was present as experiment subject, was the walking person. The experiment was repeated for the same person at different distances and for a total of four different people. The total recording time was 8 minutes summing 7365 frames.

The videos were processed off-line by the *DetectHuman-LikeObject* behaviour. Whenever the behaviour detected the presence of a human like object, it was considered a success. Otherwise, it was considered as a fail. The success rate of the behaviour in correctly detecting a walking person as a human-like object in relation to its distance to the camera are shown in the Table 7.1.

d (cm)	frames analysed	success
100	1043	95.3%
150	1121	94.1%
200	1035	94.2%
250	1050	85.3%
300	1037	83.2%
350	1058	80.1%
400	1021	68.4%

Table 7.1: Success of the *DetectHuman-Like* behaviour

As it could be expected, there is a degradation in the success of the behaviours as the person is located farther from the camera. However, the detection of a human-like object by the behaviour can be done with an acceptable degree of success up to distances of 350 cm. This behaviour is the base not only for the *DetectPerson* behaviour but also for the *TrackPerson*. Thus, it can be expected that the latter behaviour could be able to track people up to distances of 350 cm.

Experiments for *DetectPerson* behaviour

The purpose of the experiments is to evaluate the performance of the *DetectPerson* behaviour and are very similar to the previous. A total of four people were instructed to walk keeping a fixed distance to the camera, but this time always looking at it. Sequences of 15 seconds were

recorded for the same person at different distances. The 4356 frames acquired were analysed off-line in the following way. Each time a human-like object with a face was detected in a frame, it was considered as a success, otherwise fail. The Table 7.2 shows the results of these experiments.

d (cm)	frames analysed	success
100	838	95.2%
150	842	83.4%
200	843	70.8%
250	944	14.7%
300	889	0.0%

Table 7.2: Success of the *DetectPerson* behaviour

As it can be noticed there is a sharp cut in the performance between the distances 200 cm and 250 cm. This is because the method employed for detecting faces (Viola and Jones [212]) requires the faces to have a minimum size in the image. In our system, the face detector is configured to detect faces that fit in a squared window of at least 15 pixels. Faces smaller than this size will not be detected. Altough this limit can be changed, the smaller it is the higher computational effort is required to analyse the images. Analysing the results of the experiment in Table 7.2, it can be noticed that at distances farther than 200 cm, the size of the people's faces in the images begin to be too small to be detected. However, a distance of 200 cm seems to be a reasonable maximum limit for detecting people that desire to interact with the robot.

Experiments for *LookForPerson* skill

The two previous set of experiments evaluate the two main behaviours used by the *LookForPerson* skill. However, this third experiment is designed to evaluate the whole skill. It evaluates if the robot is able to successfully detect a person that wants to start an interaction with it and how long is required. For that purpose, the robot is commanded to start the execution of the *LookForPerson* skill. Then, a person approaches the robot and tries to be detected

by it. The time that elapsed between the person being detected as a human-like object at distances nearer than 2 m and he/she is detected as a person, is measured. An excessive amount of time used by the robot to detect a user could make the user feel frustrated and avoiding future interactions.

The experiment was repeated 10 times for the same person starting from different positions (always in a cone of 120 degrees around the heading direction of the robot) and for a total of four different people. Table 7.3 shows the results of the experiment. Each row shows the results for the ten tries of each person. The first column names the person. The second column indicates the average time that was required by the robot to detect the person in the ten tries. Finally, the third and fourth column indicate the minimum and maximum times respectively employed for the ten tries of each person.

Person	avrg time (ms)	min time (ms)	max time (ms)
1	515	139	1623
2	677	134	1682
3	429	112	1886
4	512	143	1782

Table 7.3: Times employed by the *LookForPerson* skill in detecting potential users

As it can be noticed, the average time necessary to be detected by the proposed skill is around half a second. We consider that this is an appropriate performance for the skill and that potential users should not feel frustrated having to wait for too long to be detected.

Experiments for *FacePerson* skill

This fourth set of experiments aim to evaluate the performace of the *FacePerson* skill to keep track of a person moving in the presence of other people. As explained in Sect. 7.4, the behaviours *TrackPerson*, *FixatePoint* and *AlignRobotToPtu* are run concurrently to develop that skill. This experiment does not only evaluates the performace of each one of the three behaviours but also the coordination between them.

The experiments consist of a person (named $P0$) approaching the robot, being detected

using the *LookForPerson* skill. Once it has been detected, the skill *FacePerson* starts and the person begins to move following the trajectory described in Fig. 7.8. In the same environment there are two other people ($P1$ and $P2$) that can distract the robot from tracking $P0$. The trajectory is such that $P0$ has to pass in front of $P1$ and behind $P2$, i.e., $P0$ can not be seen for a period of time by the camera of the robot. The lenght of the path was approximately of 6.5 m.

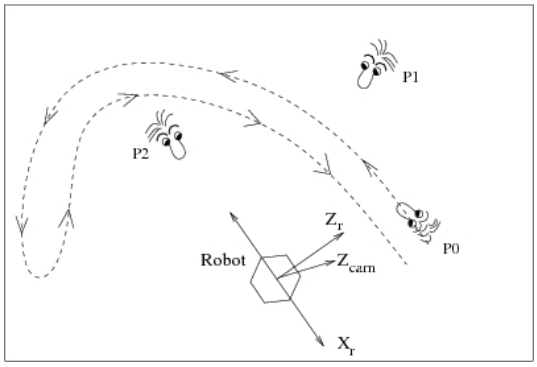


Figure 7.8: Scheme of the trajectory drawn by the person for the fourth set of experiments

The experiment was repeated five times for four different people. An experiment was considered successful if the robot was able to keep track of $P0$ while he/she was moving along the specified trajectory without confusing him/her with one of the other people in the room. The experiment was considered to fail if the robot confused the person $P0$ with another or lost track of him/her more than 10 consecutive frames. For these experiments, the behaviour *TrackPerson* and the hardware managers *RobotPlatform* and *Ptu* were configured to run at 6 Hz.

Figure 7.9 shows images taken from the camera of the robot while performing the experiment. Initially, the person $P0$ looks at the camera to be detected by the robot. $P0$ is then detected and the rectangle depicted on each image indicates the region employed to analyse the colour of his/her clothes. As $P0$ moves along the specified trajectory, the other people ($P1$ and $P2$) appear in the images. As it can be seen, the system is able to keep track of the person even while moving behind $P2$ (see images 4-5 and 8-9 of Fig. 7.9).

Table 7.4 summarises the results of the experiments. Each row indicates the results of the

five times the experiment was repeated for each person. The first column indicates the person. The second column indicates the number of times that the experiment was successfull (out of the maximum value of five). The third column indicates the average time employed by the person to complete the experiments (only for successful experiments). The fourth column indicates the total average number of frames analysed in each experiment by the *TrackPerson* behaviour and in the last column there are indicated the percentage of them in which the person was successfully tracked. The rest of the frames correspond to tracking failures where the user could not be located.

Person	#succ.	time (s)	#frames	track.
1	4	25	131	83.0%
2	4	29	151	80.3%
3	4	26	136	94.8%
4	5	22	113	92.9%

Table 7.4: Success of the *FacePerson* skill

As it can be noticed, the experiment failed one time for all the people except for the last one. The cause of the failure in the two first cases was because $P0$ and $P1$ wore clothes with very similar colours and as they became too near the robot confused them. The reason of the failure in the third case was because the person being tracked remained more than 10 consecutive frames behind person $P1$ so the robot lost track of him/her for too long.

The results obtained show that the proposed skill is able to coordinate the three basic behaviours appropriately to keep track of the person $P0$ without confusing him/her with the rest of the people the majority of the times that the experiment was repeated. However, the system becomes confused when the user $P0$ is wearing clothes with similar colours to other people and he/she moves near them. In future works we will try to find a solution to this problem with the use of additional information to describe each person.

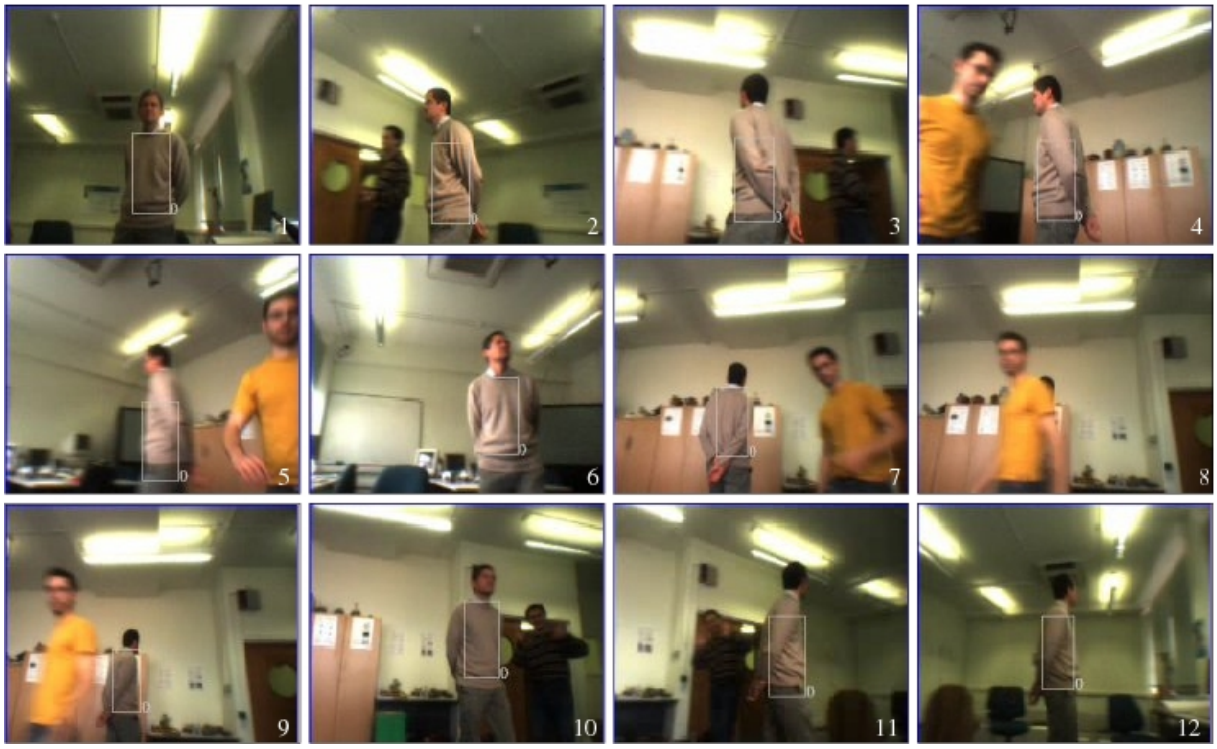


Figure 7.9: Images captured from the camera of the robot while performing the fourth set of experiments

Experiments for *FollowPerson* skill

This set of experiments are designed to evaluate the success of the *FollowPerson* skill in moving the robot while following a person. As in the previous case, the experiments do not only test the operation of the individual behaviours employed but also the performance of their combination. As explained in Sect. 7.4, this skill is achieved through the combination of the behaviours *FixatePoint*, *TrackPerson* and *GotoSafely*. The simple application showed in Fig. 7.10 was build to test this skill.

Initially the robot starts waiting for a user to interact with it using the *LookForPerson* skill. Once it is detected at distances nearer to 80 cm, the *LookForPerson* skill emits the event *PersonDetected* and the application starts the *FacePerson* skill. *FacePerson* is configured to work up to distances of 1 m, i.e., when the person goes away at further distances, the skill emits the event *PersonTooFar* and stops. Then, the application starts the *FollowPerson*

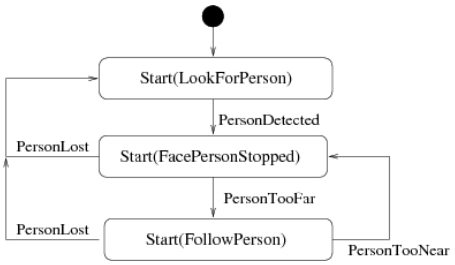


Figure 7.10: Scheme of the application employed for the fifth set of experiments

skill to approach to the person up to a distance of 80 cm. When this is achieved and the *FollowPerson* skill emits the event *PersonTooNear*, the application activates the *FacePerson* skill again. In case that the person is lost more than 10 consecutive frames by either the *FollowPerson* or *FacePerson* skills, the application activates the *LookForPerson* skill again.

The experiments carried out consisted of a person $P0$ approaching the robot to start an interaction and he/she should make the robot follow him/her around our laboratory along a path of approximately 8 m. During the experiment, another person $P1$ was in the laboratory moving around $P0$ and even crossing between the robot and the person. This was to test if the robot got confused and followed $P1$ instead of $P0$. Person $P0$ was instructed to regain the attention of the robot and continue the experiment if it lost track of him/her for more than 10 consecutive frames. For these experiments, the behaviour *TrackPerson* and the hardware managers *RobotPlatform* and *Ptu* were configured to run at 6 Hz.

Figure 7.11 shows images taken from an external camera while the experiment was performed. The images show how the person to be followed ($P0$) starts approaching the robot to get detected (images 1-2 of Fig. 7.11). Then, $P0$ start moving towards the opposite side of the room and the robot follows him/her (images 3-7). As it can be seen, $P1$ moves near $P0$ and even crosses between the robot and $P0$. However, the robot is able to keep track of $P0$. When the robot has reached the end of the room, $P0$ comes back to the initial position while the robot continues following (images 8-12). Again, while $P0$ is being followed, $P1$ is moving as well, and even crossing between $P0$ and the robot.

The experiments were repeated four times for four different people. It was considered as success if $P0$ was able to lead the robot from one side of the lab to the other and then

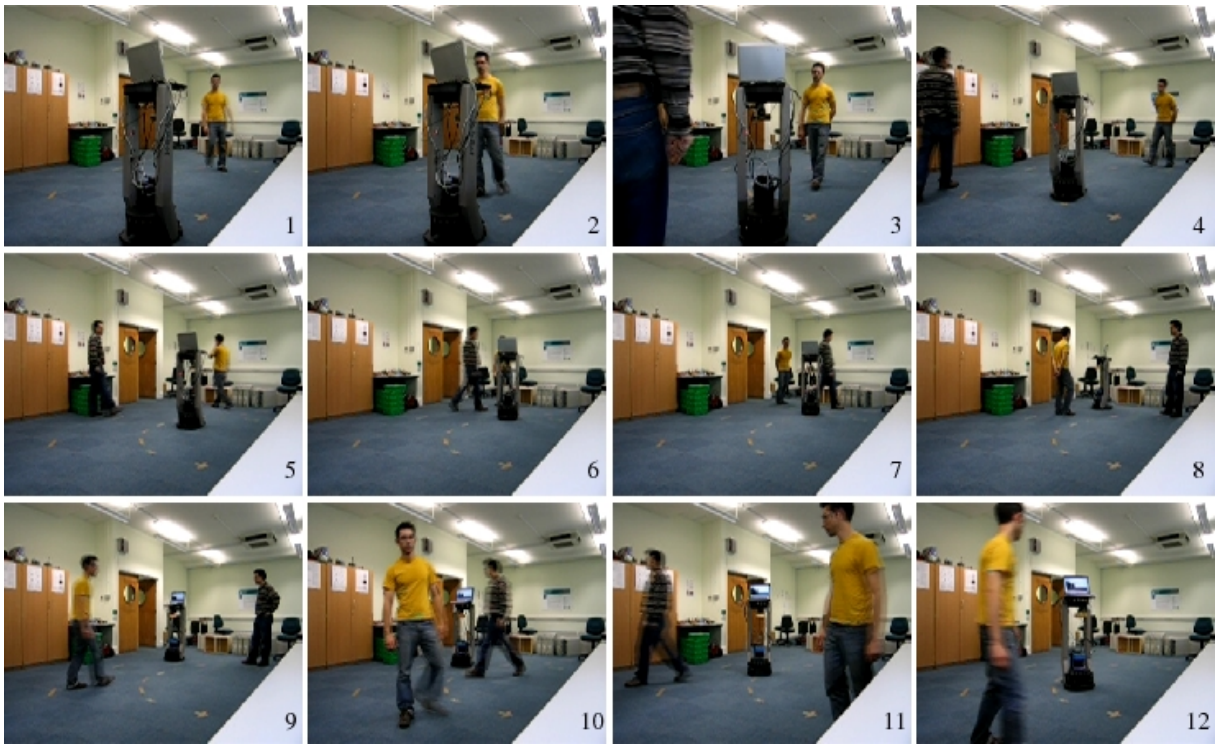


Figure 7.11: Images captured from an external camera while performing the fifth set of experiments

go back without being confused with $P1$. The results of the experiments can be seen in Table. 7.5. The first column indicates the person that performed the experiment. The second indicates the number of times that the experiment was successful. The third column indicates the average time employed to complete the experiment in those that were successful. The fourth column represents the average number of frames analysed in each successful experiment and the fifth column the percentage of these frames in which the person $P0$ was successfully tracked. Finally, the sixth column indicates the average number of times that the skill lost track of $P0$ more than 10 consecutive frames and the application had to restart the *LookForPerson* skill.

A selection of videos that show some of the tests performed are publicly available at *http://decsai.ugr.es/∼salinas/humanrobotvideos/*.

Person	#succ.	time (s)	#frames	track.	restart
1	4	42	252	85.0%	1
2	4	59	360	74.2%	2
3	4	55	310	92.4%	2
4	4	67	387	86.7%	2

Table 7.5: Success of the *FollowPerson* skill

7.6 Final Remarks

In this chapter we have presented an expandable agent-based system that implements a set of perceptual-motor skills useful for many human-robot applications. The system architecture is structure into three layers abstracting hardware for portability, providing a pool of behaviours for basic operations and a layer of fused behaviours namely perceptual-motor skills. Skills constitutes high level components useful for many mobile robotic applications that requires to establish human robot interaction. The skills designed allow the detection and tracking of an user without confusing him/her with other people and to safely navigate following him/her in indoor environments avoiding the possible obstacles in the way.

Human detection and tracking is based on fusing stereo-vision and colour information. The use of stereo vision allows to overcome some of the drawbacks of monocular vision. It is more robust against hard illumination conditions and provides distance information that is employed to know the position of the user. The vision system has demonstrated the ability to detect people with a high success rate (up to distances of 2 m) and to track users handling partial and total occlusion caused by other people in the surroundings of the robot. The proposed system is able to achieve real-time performance running on a single laptop computer. For that purpose, the agents of the system work at a fixed frequency operation that allows an efficient sharing of the computing resources. Finally, we must indicate that part of the work developed in this chapter has been published in [162].

With the development of this architecture we have fulfilled the second goal set at the beginning of this thesis. The next chapter gives a brief summary of the work explained in this thesis and indicates the publications derived from it.

Chapter 8

Conclusions and Future Work

Along the different chapters of this thesis, we have exposed particular conclusions about each one of the works presented. In this final chapter we aim to join all this remarks under a more global view and comment the main characteristics of our work stressing its contributions. As we indicated at the beginning of this thesis, the goal of our work is twofold. Firstly, the development of techniques to increase the autonomy level of mobile robots. Chapters 3, 4 and 5 are dedicated to this goal. Secondly, the development of a basis set of perceptual-motor skills that provide robots with basic interaction capabilities. Chapters 6 and 7 constitute our work aimed to fulfil the second goal.

Chapter 3 presents a multi-agent system able to control the navigation task of our mobile robot. The navigation task is based on the use of artificial landmarks placed besides doors to indicate their position. Visual information is employed to detect and track the artificial landmarks and ultrasound sensors are employed to detect the surroundings obstacles. The system is structured in a three-layer hybrid architecture. The upper layer plays the role of a deliberative planner that creates an appropriate plan as a sequence of sub-goals to lead the robot from its current position to a desired one. The plan consists of a sequence of doors to cross and corridors to follow and is generated using a topological map of the environment provided to the robot. We have opted for a topological map because it allows to represent the environment at a high level of abstraction (as a graph), thus avoiding to employ accurate localisation mechanisms. The middle layer is in charge of monitoring the execution of the

plan. It receives the high level plan and decomposes it accordingly to the skills available in the lower layer. An skill is the concurrent or sequential activation of behaviours in the lower layer. The available skills are: a) detect doors in rooms, b) detect doors in corridors, c) lead the robot towards a door and d) cross a door. The middle layer coordinates the appropriate activation of skills to accomplish the plan. Finally, the lower layer is comprised by reactive behaviours used for implementing the skills, and agents that wrap the real hardware allowing an easy portability of the whole system to other platforms. The most remarkable agents in this layer are *Vision* and *Navigation*. The latter implements a set of fuzzy behaviours that employ ultrasound information to safely navigate in the environment. The former implements visual behaviours that detect and track the artificial landmarks. In order to detect landmarks, an special object descriptor has been developed (named *double signature*), and a neural network was trained to detect them in the images captured.

The use of a multi-agent philosophy is motivated by the advantages that this approach provides. Firstly, it allows to create systems easily expandable through the addition of new agents. Secondly, multi-agent systems allow to distribute the agents over a network of computers. Thus, the computation limitations of traditional systems might be overcome.

The main drawback of the navigation approach described in Chap. 3 is that requires the use of artificial landmarks. However, altering the environment to place landmarks is not always possible. Instead, the use of natural landmarks is preferable. This has lead us to develop, in Chap. 4, a visual fuzzy system able to detect doors themselves. Apart from its use in our navigation system, the potential uses of this door-detector are localisation and map-building tasks. Door are found in images by our system through the detection of the their architraves. The system is designed to detect rectangular doors typical of many indoor environments. It is comprised by a hierarchy of fuzzy classifiers that analyse the segments extracted from images. Automatic segment extraction is performed through a sophisticated and robust method based on the Hough Transform. Then, the relationship between the segments is analysed in order to detect instances of three complex fuzzy concepts. These concepts refer to possible ways in which doors architraves can be projected on the camera of our robot. It is important to remark that because of our camera is mounted on a pan-tilt

unit, the projection of the doors architraves is strongly affected by perspective deformations. Thus, their borders are not projected as rectangular elements but highly inclined. The use of fuzzy logic has helped to overcome this problem besides bringing additional advantages: (i) it allows to define and to manage fuzzy concepts like *Vertical Segment*, *Horizontal Segment* or *Parallelism* in a natural way; (ii) it helps to manipulate the ambiguity and uncertainty in the segment extraction; and (iii) it allows to create the system as a set of rules extracted from expert knowledge that future users can understand and even alter if required. The results show that the proposed visual fuzzy system is able to detect doors at high success rates and low false detection rates. This system constitutes an important contribution in order to increase the autonomy level of autonomous robots since enable them to detect a very important landmark present in many indoor environment.

The visual fuzzy system developed is based on expert knowledge and achieves good performance. However, an improvement can be expected when it is tuned to the particular environment in which it is going to be used. Manual tuning of fuzzy systems is a tedious task that should be repeated in case of translating them to different working environments (with different doors dimensions or camera height). An additional problem of tuning our fuzzy visual system is that its performance is evaluated by a multiobjective function. In Chap. 5, we deal with the problem of tuning the fuzzy membership functions of our visual fuzzy system and we develop an evolutionary methodology for that purpose. Single and multiobjective evolutionary algorithms are considered to optimise the two criteria, and their behaviour in the problem solving is compared. The methodology designed allows to encode a whole fuzzy system as a single chromosome, bringing the advantage of reducing the number of parameters compared to other coding schemes available in the literature. Two different single-objective and three different multiobjective EAs have been considered for the problem solving, showing the better performance of the latter over the former by means of a sound experimental study. With the development of this tuning methodology, the door-detection system of the previous chapter can be adapted to different environments thus making our navigation system more general and applicable in many situations. We might conclude then, that the work developed in Chapters 3, 4 and 5 fulfils our first initial goal: the development

of techniques to increase the autonomy level of mobile robots.

The second goal set at the beginning of this thesis is the development of a basis set of perceptual-motor skills that provide robots with basic interaction capabilities. Our aim is that these skills constitute a basic common foundation useful for future robotic applications that require to interact with people. We consider that the use of classical interaction devices like mouses, touch screens or keyboards are not appropriate methods for a natural communication. Instead, we think that in order to make robots be present in everyday life, it is important that humans can command robots by means of natural interaction methods, i.e., gestures, voice, visual tracking, etc. In that sense, we have developed skills that combine stereo vision and ultrasound information to enable a mobile robot to detect, track and follow users in indoor environments.

In Chap. 6, a system able to detect and track multiple persons is presented. It is specially designed for situations in which the camera must be placed at an under-head position (typical in the case of robots that must interact with people). The system builds, in a initial phase, a background map of the environment as the aggregation of height maps. The purpose of this background model is to remove points, detected by the stereo system, that belong to static objects and thus speeding up the subsequent processing. Foreground objects are detected (using the background model) on a occupancy map that registers the volume of the elements moving in the scene. The objects found are analysed in order to detect these whose dimensions are similar to human beings. Following, the upper part of these objects are analysed using a face detector to decide whether they belong to people. Our people detection approach combines stereo information and a face detector in order to reduce the number of false positive detections. Whenever a new person is detected, the system stores information about his/her clothes and start a tracking process using the Kalman filter. Our approach of combining position and colour information has proved to be more powerful than the use of position information alone. The use of colour information allows to keep track of the people reducing the number of errors of the tracking system. The proposed system is able to track multiple persons up to distances of 5 meters with a very high success rate without using complex three dimensional models to describe the scene. Besides, the system can be used

for real-time applications.

Finally, Chap. 7 presents a multi-agent system that provides a set perceptual-motor skills that provides mobile robots with basic interaction capabilities. The multi-agent system is structured in a three-layer architecture. Agents in the lower layer provide an abstraction of the hardware employed making the system highly portable to other platforms. The middle layer detect and track users, and implements a set of behaviours that controll the movement of the robot while interacts with him/her. The vision module is an adaptation of the system developed in the previous chapter. Finally, the third layer implements skills highly reusable by combining in a concurrent and sequential manner the behaviours of the lower layer. The skills designed allow the robot: (i) to detect a user who desires to interact with the robot; (ii) to keep track of the user while he/she moves in the environment; and (iii) to follow the user along the environment avoiding possible obstacles in the way. The three skills constitute a minimum set of abilities useful for many mobile robotic applications that requires to interact with human users. Besides, the multi-agent approach employed allows the system to be easily expanded with further functionality. The system has been evaluated in different real-life experiments with several users, achieving good results and real-time performance. With the development of this multi-agent system we achieve the second goal initially proposed.

As indicated along the different chapters of this thesis, most of the work presented here has already been published in relevant international journals, as well as international and national conferences. Following, we provide a detailed list of all of them.

8.1 Publications

Following, we indicate the publications derived from our work that can be summarized as: 5 papers in international journals, 4 of them indexed by Science Citation Index (SCI), 4 papers in international conferences and 5 in national conferences.

- **Papers in International Journals**
  - Muñoz-Salinas R., Aguirre E., García-Silvente M. and Gómez M. "A Multi-agent

system architecture for mobile robot navigation based on fuzzy and visual behaviors". Robotica 23: 689-699, 2005. *This Journal is indexed by Science Citation Index. This paper is related to part of the contents of Chapter 3.*

- Muñoz-Salinas R. , Aguirre E. and García-Silvente M. "Detection of doors using a genetic visual fuzzy system for mobile robots". Aceptado para su publicación en Autonomous Robots. *This Journal is indexed by Science Citation Index. This paper is related to part of the contents of Chapter 4.*
- Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Door-detection using computer vision and fuzzy logic", WSEAS Transactions on Systems, 10(3): 3047-3052, 2004. *This paper is related to part of the contents of Chapter 4.*
- Muñoz-Salinas R., Aguirre E., Cordón O., García-Silvente M. "Automatic tuning of a fuzzy visual system using evolutionary algorithms: single-objective vs. multiobjective approaches". Aceptado para su publicación en IEEE Transactions on Fuzzy Systems. *This Journal is indexed by Science Citation Index. This paper is related to part of the contents of Chapter 5.*
- Muñoz-Salinas R., Aguirre E, García-Silvente M. and Gonzalez A. "People detection and tracking through stereo vision for human-robot interaction". Lectures Notes on Artificial Intelligence 3789: 337-346, 2005. *This Journal is indexed by Science Citation Index. This paper is related to part of the contents of Chapter 6.*

- **Papers on International Conferences**

  - Aguirre E., García-Silvente M., Gómez M., Muñoz-Salinas R. and Ruiz C. "A multi-agent system based on active vision and ultrasounds applied to fuzzy behavior based navigation". En 10th International Symposium on Robotics and Applications (ISORA 2004), in CD-Rom, Sevilla (Spain), 2004. *This paper is related to part of the contents of Chapter 3.*
  - Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Door-detection using computer vision and fuzzy logic". En 6th WSEAS International Confer-

ence on Mathematical Methods & Computational Techniques in Electrical Engineering, En CD-Rom, Atenas (Greek), 2004. *This paper is related to part of the contents of Chapter 4*.

- Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Tuning a visual fuzzy system using SPEA2". En 1st International Workshop on Genetic Fuzzy Systems (GFS'2005), pages 37-43, Granada (Spain), 2005. *This paper is related to part of the contents of Chapter 5*.
- Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "A fuzzy system for visual detection of interest in human-robot interaction". In 2nd International Conference on Machine Intelligence (ACIDCA-ICMI'2005), pages 574-581, Tunisia, 2005. *This paper is related to part of the contents of Chapter 7*.

- **Papers on National Conferences**

  - Aguirre E., Gómez M., Muñoz-Salinas R. and Ruiz C. "Un Sistema Multi-Agente que emplea Visión Activa y Ultrasonidos aplicado a Navegación con Comportamientos Difusos". In IV Workshop de Agentes Físicos (WAF'2003), pages 63 - 74, Alicante (Spain), 2003. *This paper is related to part of the contents of Chapter 3*.
  - Aguirre E., García-Silvente M., González A. and Muñoz-Salinas R. "Detección de puertas mediante visión y lógica difusa". In XII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2004), pages 389-394, Jaén (Spain), 2004. *This paper is related to part of the contents of Chapter 4*.
  - Aguirre E., García-Silvente M., González A. and Muñoz-Salinas R. "Detección de puertas mediante información multisensorial y su aplicación a la navegación de robots móviles". In V Workshop en Agentes Físicos (WAF'2004), pages 33-42, Girona (Spain), 2004. *This paper is related to part of the contents of Chapter 4*.

- Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Detección y seguimiento de personas usando visión estéreo". In VI Workshop de Agentes Físicos (WAF'2005), pages 51 - 58, Granada (Spain), 2005. *This paper is related to part of the contents of Chapter 6.*
- Muñoz-Salinas R., Aguirre E., García-Silvente M., González A. "Un Sistema Visual Difuso para la Detección de Interés en la Interacción Robot-Persona" Por aparecer en el VII Workshop en Agentes Físicos, Las Palmas de Gran Canaria (España), Abril 2006. *This paper is related to part of the contents of Chapter 7.*

- **Technical Reports**

  - Muñoz-Salinas R., Aguirre E. and García-Silvente M. "People Detection and Tracking using Stereo Vision and Color".Technical Report DECSAI-SI-2005-04. *This paper is related to part of the contents of Chapter 6.*
  - Muñoz-Salinas R., Aguirre E., García-Silvente M., Ayes A. and Gongora M."Multi-agent system for human-robot interaction using stereo vision".Technical Report DECSAI-SI-2005-05. *This paper is related to part of the contents of Chapter 7.*

8.2 Future Work

Finally, we would like to expose possible future works derived from ours. We mainly stress the possibility of still researching on human-robot interaction techniques in order to expand the system created. We point towards the use of additional sources of information in order to create multimodal interaction interfaces (e.g., sound, laser rangefinders, etc), and the development of gesture analysis techniques. The multi-agent architecture developed would allow an easy integration of this new functionality into the system.

Conclusiones y Trabajo Futuro

A lo largo de los diferentes capítulos de esta memoria hemos ido exponiendo una serie conclusiones acerca del trabajo presentado en ellos. En este capítulo finalizamos la memoria dando una visión conjunta de las conclusiones que pueden extraerse del trabajo realizado y haciendo especial hincapie en las contribuciones del mismo. Tal y como se indicó al comienzo de la memoria, el objetivo de este trabajo de investigación es doble. En primer lugar, el desarrollo de técnicas para incrementar el nivel de autonomía de robots móviles. Los Capítulos 3, 4 y 5 están dedicados a este primer objetivo. El segundo objetivo es el desarrollo de un conjunto base de habilidades percepto-motoras que doten a los robots con capacidades primitivas de interacción natural. Los trabajos presentados en los Capítulos 6 y 7 son nuestra contribución a este segundo objetivo.

El Capítulo 3 presenta una arquitectura híbrida multiagente para el control de la tarea de navegación autónoma de nuestro robot móvil. La tarea de navegación se basa en el uso de marcas artificiales colocadas junto a las puertas para indicar su posición. La información visual proporcionada por nuestra cámara es utilizada para detectar y realizar un seguimiento visual de las marcas artificiales. Por otro lado, sensores de ultrasonido son utilizados para detectar obstáculos en el camino. El sistema emplea una arquitectura híbrida de tres nieveles. El nivel superior es un planificador deliberativo que crea el plan adecuado para ir desde la posición actual del robot a una posición deseada. Este plan consiste en la secuencia de habitaciones y pasillos a recorrer así como las puertas a cruzar para llegar a una habitación objetivo. El plan es generado haciendo uso de un mapa topológico de entorno que es previamente proporcionado al robot. La razón de utilizar mapas topológicos es que permiten representar el entorno a un alto nivel de abstracción (en forma de grafo) y así podemos evitar

el uso de mecanismos de localización precisos. La capa intermedia se encarga de monitorizar la ejecución del plan. Esta capa recibe el plan desde la capa superior y lo descompone en una serie de habilidades que se desarrollan en la capa inferior. Un habilidad se entiende como la ejecución secuencial o concurrente de un conjunto de comportamientos que están implementados en la capa inferior. Las habilidades que han sido diseñadas permiten al robot: a) detectar las puertas que hay en habitaciones, b) detectar puertas en pasillos, c) guiar el robot hacia una puerta; y d) cruzar una puerta. La capa intermedia se encarga pues de coordinar de forma adecuada la activación de estás habilidades para conseguir la ejecución del plan. Finalmente, la capa más inferior de la arquitectura se compone de una serie de comportamientos reactivos, que se usan para implementar las habilidades anteriormente mencionadas, y de agentes encargados de abstraer el hardware del robot para así permitir una fácil portabilidad a otras plataformas. Los agentes más destacados en esta capa son los agentes *Vision* y *Navigation*. El último implementa una serie de comportamientos difusos para, utilizando información de ultrasonidos, permitir la navegación segura del robot. El agente *Vision* implementa comportamientos visuales para la detección y seguimiento de marcas artificiales. Para poder detectar las marcas, hemos diseñado un nuevo descriptor visual que hemos denominado *firma doble*, y una red neuronal ha sido apropiadamente entrenada para reconocerlas en las imágenes capturadas por el robot.

El uso de una filosofía multiagente está motivada por las múltiples ventajas que aporta. En primer lugar, permite la creación de sistemas fácilmente expansibles mediante la adición de nuevos agentes. En segundo lugar, la capacidad de los sistemas multiagente de ser distribuidos sobre una red de computadoras evitando así la limitación de potencia de cálculo de las arquitecturas tradicionales.

La principal desventaja del enfoque empleado para la navegación descrito en el Capítulo 3 es que requiere de marcas artificiales. Sin embargo la modificación del entorno no es siempre posible y por tanto sería preferible el uso de marcas naturales. Este problema nos llevó al desarrollo, en el Capítulo 4, de un sistema visual difuso capaz de detectar las puertas sin necesidad de usar ninguna marca. Aparte de la utilidad que este detector de puertas tiene para nuestro sistema de navegación (ya que permite prescindir de marcas), los usos poten-

ciales también comprenden tareas de localización y de construcción autónoma de mapas. El detector es capaz de encontrar puertas en imágenes mediante la detección de sus marcos. El sistema ha sido diseñado para detectar los bordes rectangulares típicos de las puertas que se encuentran en entornos de interior. El detector se compone de una jerarquía de clasificadores difusos que analizan los segmentos extraidos de las imágenes capturadas por el robot. La extracción de estos segmentos se realiza de forma automática mediante el uso de un sofisticado y robusto método basado en la Transformada de Hough. Después, la relación entre estos segmentos es analizada para tratar de detectar entre ellos ocurrencias de tres conceptos difusos que han sido definidos. Estos conceptos difusos hacen referencia a posibles situaciones en que los marcos de las puertas pueden proyectarse en la cámara de un robot móvil. Es importante hacer notar que la cámara de nuestro robot está colocada sobre una cabeza robótica, y por ello, la proyección de las puertas en las imágenes capturadas están fuertemente afectadas por deformaciones debidas a la perspectiva. Esto hace que los bordes de las puertas no se proyecten como elementos rectangulares sino como segmentos altamente inclinados. El uso de lógica difusa ha permitido superar este problema así como proporcionar otra serie de ventajas: (i) nos ha permitido definir y manejar conceptos difusos como *Segmento Vertical*, *Segmento Horizontal* o *Paralelismo* de una forma natural; (ii) nos ha permitido manipular la ambigüedad e incertidumbre en la extracción de segmentos; y (iii) hemos podido crear un sistema como una serie de reglas extraidas del conocimiento experto que usuarios futuros del mismo puedan entender e incluso modificar si fuese necesario. Las pruebas realizadas con nuestro sistema demuestran que éste es capaz de detectar puertas obteniendo un bajo ratio de falsos negativos y un alto porcentaje de verdaderos positivos. Este sistema constituye una interesante aportación de nuestro trabajo a incrementar la autonomía de robots móviles dado que permite la detección de una marca natural de gran relevancia en entornos de interior.

El sistema visual difuso desarrollado en el capítulo anterior ha sido diseñado empleando conocimiento experto y obtiene buenos resultados de clasificación. Sin embargo, es posible esperar una mejora de su rendimiento si lo ajustamos a las condiciones particulares del entorno en que va a ser utilizado. El ajuste manual de sistemas difusos es una tarea tediosa que debe ser repetida en caso de trasladar el sistema a otro entorno de trabajo (con puertas

de diferentes dimensiones o altura de cámara distinta). Un problema adicional para ajustar nuestro sistema difuso, es que su rendimiento es evaluado por una función multiobjetivo. En el Capítulo 5 tratamos el problema de ajustar las etiquetas difusas de nuestro sistema visual difuso y para ello desarrollamos una metodología de ajuste basada en algorimos evolutivos. La metodología diseñada permite la codificación de todo un sistema difuso como un único cromosoma, aportando la ventaja de reducir de forma apreciable el número de parámetros respecto de otros esquemas de codificación presentes en la literatura. Hemos considerado tanto algoritmos evolutivos monoobjetivo como algoritmos multiobjetivo para optimizar los dos criterios de evaluación de nuestro sistema y hemos realizado una comparativa de sus resultados. En particular, hemos empleado dos algoritmos monoobjetivo distintos y tres algoritmos multiobjetivo, mostrando los últimos un mejor comportamiento en la solución de nuestro problema. Con el desarrollo de este mecanismo de ajuste, el sistema visual difuso diseñado para la detección de puertas puede ser apropiadamente adaptado para su uso en diversos entornos reales y así hacer nuestro sistema de navegación más general y aplicable en diferentes condiciones. Debemos concluir indicando que el trabajo desarrollado en los Capítulos 3, 4 y 5 nos permiten cumplir con el primer objetivo planteado.

El segundo objetivo fijado al inicio de esta memoria era el desarrollo de una base de habilidades percepto-motoras que proporcionen a los robots con primitivas de interacción natural. Nuestro deseo es que esas habilidades constituyan un conjunto aplicable a un amplio rango de aplicaciones robóticas que requieran interactuar con humanos. Consideramos que el uso de mecanismos clásicos de interacción como ratón, pantalla táctil o teclados no son métodos apropiados para una comunicación natural. Si deseamos que los robots se integren como parte de nuestra vida diaria, es importante que los humanos los podamos dirigir mediante el uso de mecanismos de interacción más naturales, es decir, gestos, voz, expresiones faciales, etc. En este sentido, hemos desarrollado un conjunto de habilidades que combinan información visual estéreo con información de ultrasonidos para permitir a los robots móviles: detectar, seguir visualmente y acompañar a usuarios humanos en entornos de interior.

En el Capítulo 6, presentamos un sistema visual capaz de detectar y seguir a varias personas de forma simultánea. El sistema ha sido especialmente diseñado para situaciones en

que la cámara debe estar colocada en posiciones bajo la cabeza (típico en el caso de utilizar robots que deben interactuar con las personas). El sistema construye en una fase inicial un mapa del entorno mediante la agregación de mapas de altura. El propósito de este modelado del entorno es la eliminación de puntos tridimensionales detectados por el sistema estéreo que pertenecen a objetos inmóviles y de esta manera acelerar el proceso de detección y seguimiento de personas. Después, los objetos que no pertenecen al entorno son rápidamente detectados mediante la utilización de mapas de ocupación encargados de registrar el volumen de los objetos móviles presentes en la escena. Los objetos encontrados son analizados para determinar cuáles de ellos tienen dimesiones similares a los del ser humano. Después, aquellos objetos con dimensiones de humano son analizados con un detector de caras para decidir si se corresponden a personas. Nuestra técnica para realizar la detección de personas combina información estéreo con un detector de caras permitiendo reducir el número de falsos positivos. Cada vez que una nueva persona es detectada, el sistema almacena información acerca del color de su ropa y comienza un proceso de seguimiento utilizando el filtro de Kalman. Nuestro enfoque de combinar información sobre la posición e información de color ha demostrado ser más robusto que el uso exclusivo de información de posición. El uso de color permite seguir visualmente a las personas reduciendo el número de errores del sistema. El sistema que hemos diseñado es capaz de realizar el seguimiento simultáneo de múltiples personas a distancias de hasta 5 metros con un alto porcentaje de éxito y sin la necesidad de emplear complejos modelos tridimensionales. Además, la latencia del sistema los hace apropiado para su uso en aplicaciones de tiempo real.

Finalmente, el Capítulo 7 presenta un sistema multiagente que implementa un conjunto de habilidades percepto-motoras que dotan a robots móviles con capacidades básicas de interacción. El sistema se estructura en una arquitectura de tres niveles. Los agentes de la capa inferior proporcionan una abstracción de hardware empleado haciendo que el sistema sea altamente potable a otras plataformas. La capa intermedia detecta y realiza el seguimiento de los usuarios, e implementa una serie de comportamiento para controlar el movimiento del robot mientras se realiza la interacción. El módulo de visión empleado para la detección y seguimiento de usuarios es una adaptación de sistema desarrollado en el capítulo anterior.

Finalmente, el nivel superior implementa habilidades mediante la combinación secuencial o concurrente de los comportamientos de la capa intermedia. El conjunto de habilidades que el sistema es capaz de realizar son: (i) detectar la presencia de un usuario interesado en interactuar con el robot; (ii) realizar un seguimiento visual del usuario mientras este se mueve en el entorno; y (iii) seguir al usuario por el entorno evitando colisionar con los obstáculos presentes. Las tres habilidades desarrolladas constituyen un conjunto mínimo de gran utilidad para una amplia variedad de aplicaciones robóticas que requieran interactuar con humanos. Además, el uso de la arquitectura multiagente dota al sistema de una alta capacidad para expandirse con nueva funcionalidad. El sistema ha sido evaluado en numerosos experimentos reales con distintas personas demostrando tener un gran robustez y un rendimiento en tiempo real. De esta manera, se cumple con el segundo objetivo planteado en esta memoria.

Tal y como se ha indicado a los largo de los distintos capítulos de esta memoria, la mayoría del trabajo realizado ha sido publicado tanto en prestigiosas revistas de ámbito internacional, como el congresos internacionales y nacionales. A continuación se expone una lista detallada de las publicación realizadas.

Publicaciones

A continuación se indican las publicaciones a las que ha dado lugar el presente trabajo y que en resumen son: 5 artículos en revistas internacionales, 4 de las cuales están indexadas por el Science Citation Index (SCI), 4 comunicaciones en congresos internacionales y 5 en congresos nacionales.

- **Artículos en Revistas Internacionales**

  - Muñoz-Salinas R., Aguirre E., García-Silvente M. and Gómez M. "A Multi-agent system architecture for mobile robot navigation based on fuzzy and visual behaviors". Robotica 23: 689-699, 2005. *Esta revista está indexada en el SCI. Este trabajo está relacionado con el contenido del Capítulo 3.*

  - Muñoz-Salinas R. , Aguirre E. and García-Silvente M. "Detection of doors using

a genetic visual fuzzy system for mobile robots". Aceptado para su publicación en Autonomous Robots. *Esta revista está indexada en el SCI. Este trabajo está relacionado con el contenido del Capítulo 4.*

- Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Door-detection using computer vision and fuzzy logic", WSEAS Transactions on Systems, 10(3): 3047-3052, 2004. *Este trabajo está relacionado con el contenido del Capítulo 4.*
- Muñoz-Salinas R., Aguirre E., Cordón O., García-Silvente M. "Automatic tuning of a fuzzy visual system using evolutionary algorithms: single-objective vs. multiobjective approaches". Aceptado para su publicación en IEEE Transactions on Fuzzy Systems. *Esta revista está indexada en el SCI. Este trabajo está relacionado con el contenido del Capítulo 5.*
- Muñoz-Salinas R., Aguirre E, García-Silvente M. and Gonzalez A. "People detection and tracking through stereo vision for human-robot interaction". Lectures Notes on Artificial Intelligence 3789: 337-346, 2005. *Esta revista está indexada en el SCI. Este trabajo está relacionado con el contenido del Capítulo 6.*

- **Artículos en Conferencias Internacionales**

  - Aguirre E., García-Silvente M., Gómez M., Muñoz-Salinas R. and Ruiz C. "A multi-agent system based on active vision and ultrasounds applied to fuzzy behavior based navigation". En 10th International Symposium on Robotics and Applications (ISORA 2004), in CD-Rom, Sevilla (Spain), 2004. *Este trabajo está relacionado con el contenido del Capítulo 3.*
  - Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Door-detection using computer vision and fuzzy logic". En 6th WSEAS International Conference on Mathematical Methods & Computational Techniques in Electrical Engineering, En CD-Rom, Atenas (Greek), 2004. *Este trabajo está relacionado con el contenido del Capítulo 4.*
  - Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Tuning a visual fuzzy system using SPEA2". En 1st International Workshop on Genetic

Fuzzy Systems (GFS'2005), pages 37-43, Granada (Spain), 2005. *Este trabajo está relacionado con el contenido del Capítulo 5*.

- Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "A fuzzy system for visual detection of interest in human-robot interaction". En 2nd International Conference on Machine Intelligence (ACIDCA-ICMI'2005), pages 574-581, Tunisia, 2005. *Este trabajo está relacionado con el contenido del Capítulo 7*.

- **Artículos en Conferencias Nacionales**

  - Aguirre E., Gómez M., Muñoz-Salinas R. and Ruiz C. "Un Sistema Multi-Agente que emplea Visión Activa y Ultrasonidos aplicado a Navegación con Comportamientos Difusos". En IV Workshop de Agentes Físicos (WAF'2003), pages 63 - 74, Alicante (Spain), 2003. *Este trabajo está relacionado con el contenido del Capítulo 3*.
  - Aguirre E., García-Silvente M., González A. and Muñoz-Salinas R. "Detección de puertas mediante visión y lógica difusa". En XII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2004), pages 389-394, Jaén (Spain), 2004. *Este trabajo está relacionado con el contenido del capítulo 4*.
  - Aguirre E., García-Silvente M., González A. and Muñoz-Salinas R. "Detección de puertas mediante información multisensorial y su aplicación a la navegación de robots móviles". En V Workshop en Agentes Físicos (WAF'2004), pages 33-42, Girona (Spain), 2004. *Este trabajo está relacionado con el contenido del Capítulo 4*.
  - Muñoz-Salinas R., Aguirre E., García-Silvente M. and González A. "Detección y seguimiento de personas usando visión estéreo". En VI Workshop de Agentes Físicos (WAF'2005), pages 51 - 58, Granada (Spain), 2005. *Este trabajo está relacionado con el contenido del Capítulo 6*.
  - Muñoz-Salinas R., Aguirre E., García-Silvente M., González A. "Un Sistema Visual Difuso para la Detección de Interés en la Interacción Robot-Persona" Por

aparecer en el VII Workshop en Agentes Físicos, Las Palmas de Gran Canaria (España), Abril 2006. *Este trabajo está relacionado con el contenido del Capítulo 7.*

- **Memorias técnicas de trabajo**

  - Muñoz-Salinas R., Aguirre E. and García-Silvente M. "People Detection and Tracking using Stereo Vision and Color".Technical Report DECSAI-SI-2005-04. *Este trabajo está relacionado con el contenido del Capítulo 6.*
  - Muñoz-Salinas R., Aguirre E., García-Silvente M., Ayes A. and Gongora M."Multiagent system for human-robot interaction using stereo vision".Technical Report DECSAI-SI-2005-05. *Este trabajo está relacionado con el contenido del Capítulo 7.*

Trabajo Futuro

Para finalizar esta memoria de tesis, nos gustaría indicar los posibles trabajos futuros que pueden derivarse éste. Principalmente destacamos el posible desarrollo de técnicas de interacción humano-robot a partir de la base creada en este trabajo. Nos estamos refiriendo a la utilización incorporación de interfaces multimodales (sonido, láser, etc), y el desarrollo de técnicas de análisis gestual. El uso de la arquitectura multiagente empleada permitirá una adecuada integración de estos nuevos módulos en el sistema.

Bibliography

[1] E. Aguirre and A. González. Fuzzy behaviors for mobile robot navigation: Design, coordination and fusion. *International Journal of Approximate Reasoning*, 25:255–289, 2000.

[2] E. Aguirre and A. González. Integrating topological and geometrical world modeling for mobile robot navigation. In *I Workshop Hispano-Luso de Agentes Físicos*, pages 167–181, Tarragona, 2000. (In spanish).

[3] E. Aguirre and A. González. Integrating fuzzy topological maps and fuzzy geometric maps for behavior-based robots. *International Journal of Intelligent Systems*, 17(3):333–368, 2002.

[4] E. Aguirre and A. González. A fuzzy perceptual model for ultrasound sensors applied to intelligent navigation of mobile robots. *Applied Intelligence*, 19(3):171–187, 2003.

[5] E. Aguirre, A. González, and R. Muñoz-Salinas. Mobile robot map-based localization using approximate locations and the extended kalman filter. In *10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2004)*, pages 191 – 198, 2004.

[6] F. Aherne, N. Thacker, and P. Rockett. The Bhattacharyya Metric as an Absolute Similarity Measure for Frecuency Coded Data. *Kybernetica*, 32:1–7, 1997.

[7] James S. Albus. RCS: A reference model architecture for intelligent control. *j-COMPUTER*, 25(5):56–59, May 1992.

[8] James S. Albus. 4-d/rcs reference model architecture for unmanned ground vehicles. In *Proceedings os SPIE*, volume 3693, Orlando, FL, abril 1999.

[9] James S. Albus and Fred G. Proctor. A reference model architecture for hybrid control systems. In *Proceedings of the International Federation of Automatic Control*, San Francisco, CA, 1996.

[10] W.J. Alford, R.D VanderNeut, and V.J. Zaleckas. Laser scanning microscopy. *Proceedings of the IEEEs*, 70:641 – 651, 1982.

[11] AA. Argyrs and MI. Lourakis. Three-dimensional tracking of multiple skin-colored regions by a moving stereoscopic system. *Applied Optics*, 43:366–378, 2004.

[12] Ronald C. Arkin. Path planning for a vision-based autonomous robot. In *Proc. of the SPIE Conference on Mobile Robots*, pages 240–249, 1986.

[13] Ronald C. Arkin. Motor schema based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, 1989.

[14] Ronald C. Arkin. *Behavior-Based Robotics*. The MIT Press, 1998.

[15] N. Ayache and O.D. Faugeras. Mantaining Representation of the Environment of a Mobile Robot. *IEEE Trans. Robotics and Automation*, 5:804–819, 1989.

[16] T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press, 1997.

[17] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *Evolutionary Computation*, 1(1):3–17, 1993.

[18] M. Barth, D. Hirayama, G. Beni, and S. Hackwood. A color vision inspection system for integrated circuit manufacturing. *IEEE Journal on Semiconductor Manufacturing*, 5:290 – 301, 1992.

[19] Y. Bentoutou and N. Taleb. Automatic extraction of control points for digital subtraction angiography image enhancement. *IEEE Transactions on Nuclear Science*, 52:238 – 246, 2005.

[20] R. Bischoff and V. Graefe. Hermes - a versatile personal robotic assistant. *Proceedings of the IEEE*, 92:1759 – 1779, 2004.

[21] W.E. Blanz, J.L.C. Sanz, and E.B Hinkle. Image analysis methods for solderball inspection in integrated circuit manufacturing. *IEEE Journal of Robotics and Automation*, 4:129 – 139, 1988.

[22] A. L. Blumel, E. J. Hughes, and B. A. White. Fuzzy autopilot design using a multiobjective evolutionary algorithm. In *Congress on Evolutionary Computation*, pages 54–61, 2000.

[23] A. Bonarini and F. Basso. Learning to compose fuzzy behaviors for autonomous agents. *International Journal on Approximate Reasoning*, 17(4):409–432, 1997.

[24] P. P. Bonissone, P. S. Khedar, and Y. Chen. Genetic algorithms for automated tuning of fuzzy controllers: A transportation application. In *Fifth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96)*, pages 674–680, 1996.

[25] R.P. Bonnaso, D. Kortenkamp, and R. Murphy. Mobile robots. A proving ground for artificial intelligence. In *Artificial Intelligence and Mobile Robots*, pages 3–18. AAAI Press, 1998.

[26] J. Borenstein, H.R. Everett, and L. Feng. *Where am I?. Systems and Methods for Mobile Robot Positioning*. J. Borenstein, 1996.

[27] J. Borenstein and L. Feng. UMBmark: A benchmark test for measuring dead reckoning errors in mobile robots. In *SPIE Conference on Mobile Robots*, 1995.

[28] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[29] D. Boulfelfel, R.M. Rangayyan, L.J. Hahn, and R. Kloiber. Preconstruction restoration of myocardial single photon emission computed tomography images. *IEEE Transactions on Medical Imaging*, 11:336 – 341, 1992.

[30] D. Boulfelfel, R.M. Rangayyan, L.J. Hahn, and R. Kloiber. Three-dimensional restoration of single photon emission computed tomography images. *IEEE Transactions on Nuclear Science*, 41:1746 – 1754, 1994.

[31] Cynthia Breazeal. An ethological and emotional basis for human-robot interaction. *Robotics and Autonomous Systems*, 42:191–201, 2003.

[32] Cynthia Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42:167–175, 2003.

[33] Cyntia Breazeal. Social interactions in HRI: the robot view. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34:181 – 186, 2004.

[34] R.A. Brooks. A layered intelligent control system for a mobile robot. In *Third International Symposium of Robotics Research*, pages 1–8, Gouvieux, France, 1985.

[35] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.

[36] M.Z. Brown, D. Burschka, and G.D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:993 – 1008, 2003.

[37] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thurn. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 144:3–55, 1999.

[38] J.L. Burke, R.R Murphy, E. Rogers, V.J. Lumelsky, and J. Scholtz. Final report for the darpa/nsf interdisciplinary study on human-robot interaction. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34:103 – 112, 2004.

[39] J. Casillas, O. Cordón, M. J. del Jesus, and F. Herrera. Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems*, 13(1):13–29, 2005.

[40] J. Cha, R.H. Cofer, and S.P. Kozaitis. Extended hough transform for linear feature detection. *To appear in Pattern Recognitiong (Elservier)*, 2005.

[41] T.S. Chan and R.K.K Yip. Line detection algorithm. In *13th International Conference on Pattern Recognition*, volume 2, pages 25–29, 1996.

[42] C. S. Chang, D. Y. Xu, and H. B. Quek. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. In *IEE Proc. Electric Power App*, volume 145, pages 577–583, 1999.

[43] V. Changkon and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. North-Holland, 1983.

[44] H.I. Christensen, N.O. Kirkeby, S. Kristensen, and L. Knudsen. Model-Driven Vision for In-Door Navigation. *Robotics and Autonomous Systems*, 12:199–207, 1994.

[45] G. Cicirelli, T. D'orazio, and A. Distante. Target recognition by component for mobile robot navigation. *Journal of Experimental and Theoretical Artificial Intelligence*, 15(3):281–297, 2003.

[46] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behaviour*, 2:73–110, 1993.

[47] C. A. Coello, D. A. Van Veldhuizen, and G. B. Lamant. *Evolutionary Algorithms for Solving Multi-objective Problems*. Kluwer Academic Publishers, 2002.

[48] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR'2000)*, volume 2, pages 142–151, 2000.

[49] O. Cordón, F. Gomide and. F. Herrera, F. Hoffman, and L. Magdalena. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy sets and systems*, 141:5–31, 2004.

[50] O. Cordón and F. Herrera. A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases. *International Journal of Approximate Reasoning*, 17(4):369–407, 1997.

[51] O. Cordón, F. Herrera, M. J. del Jesus, and P. Villar. A multiobjective genetic algorithm for feature selection and granularity learning in fuzzy-rule based classification systems. In *Joint 9th IFSA World Congress-20th NAFIPS International Conference*, volume 3, pages 1253–1258, 2001.

[52] O. Cordón, F. Herrera, F. Hoffman, and L. Magdalena. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knwoledge Bases*. World Scientific Publishing, 2001.

[53] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb. Plan-view trajectory estimation with dense stereo background models. In *Eighth IEEE International Conference on Computer Vision (ICCV 2001)*, volume 2, pages 628 – 635, 2001.

[54] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated Person Tracking Using Stereo, Color, and Pattern Detection. *Int. Journ. Computer Vision*, 37:175–185, 2000.

[55] W. C. Daugherity, B. Rathakrishnan, and J. Yen. Performance evaluation of a self-tuning fuzzy controller. In *First IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'92)*, pages 387–397, 1992.

[56] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[57] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[58] G.N. DeSouza and A.C. Kak. Vision for Mobile Robot Navigation: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:237–267, 2002.

[59] A. Dima, M. Scholz, and K. Obermayer. Automatic segmentation and skeletonization of neurons from confocal microscopy images based on the 3-d wavelet transform. *IEEE Transactions on Image Processing*, 11:790 – 801, 2002.

[60] D. Driankov, H. Hellerdoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Kluwer Academic Publishers, 1993.

[61] Hansye S. Dulimarta and Anil K. Jain. A client/server control architecture for robot navigation. *Pattern Recognition*, 29(8):1259–1284, 1996.

[62] Y. Edan, D. Rogozin, and T. Flash G.E. Miles. Robotic melon harvesting. *IEEE Transactions on Robotics and Automation*, 16:325 – 333, 2000.

[63] C. Eldershaw and M. Yim. Motion planning of legged vehicles in an unstructured environment. In *IEEE International Conference on Robotics and Automation (ICRA'2001)*, volume 4, pages 3383 – 3389, 2001.

[64] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Magazine, Special Issue on Autonomous Intelligent Machines*, 22(6):46–57, 1989.

[65] L.J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *First Workshop on Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann, 1991.

[66] L.J. Eshelman and D. Schaffer. Real-coded genetic algorithms and interval-schemata. In *Second Workshop on Foundations of Genetic Algorithms*, pages 187–202. Morgan Kaufmann, 1993.

[67] C. Eveland, K. Konolige, and R.C. Bolles. Background Modelling for Segmentatation of Vide-Rate Stereo Sequences. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 266–271, 1998.

[68] E. Falcone, R. Gockley, E. Porter, and I. Nourbakhsh. The Personal Rover Project: The comprehensive design of a domestic personal robot. *Robotics and Autonomous Systems*, 42:245–258, 2003.

[69] Zhao Feng-Ji, Guo Hai-Jiao, and K. Abe. Mobile robot localization using two sonar sensors and one natural landmark. In *37th SICE Annual Conference (SICE '98)*, pages 29–31, 1998.

[70] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transaction on Systems, Man, and Cybernetics, Part B*, 26:396–407, 1996.

[71] J.D. Foley, A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*. Addison Wesley, 1990.

[72] J.D. Foley and A. van Dam. *Fundamentals of Interactive Computer Graphics*. Addison Wesley, 1982.

[73] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42:143–166, 2003.

[74] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization. In *Proc. of the Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[75] G.L. Foresti. A real-time hough-based method for segment detection in complex multisensor images. *Real-Time Imaging*, 6:93–111, 2000.

[76] G.L. Foresti and F.A Pellegrino. Automatic visual recognition of deformable objects for grasping and manipulation. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34:325 – 333, 2004.

[77] M. Fujita and H. Kitano. Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robots*, 5:7–18, 1998.

[78] K. Fukuzawa, T. Ohkubo, J. Kishigami, and Y. Koshimoto. Kerr effect microscopy imaging of recorded domains on perpendicular magnetic recording media. *IEEE Transactions on Magnetics*, 28:3420 – 3422, 1992.

[79] J. Furusho, S. Akihito, S. Masamichi, and K. Eichi. Realization of bounce gair in a quadruped robot with articular-joint-type legs. In *IEEE Int. Conf. on Robotics and Automation*, pages 679–702, 1995.

[80] L. Gacôgne. Multiple objective optimization of fuzzy rules for obstacles avoiding by an evolution algorithm with adaptive operators. In *Proc. of the Fifth International Mendel Conference on Soft Computing (Mendel'99)*, pages 236–242, 1999.

[81] M.C. García-Alegre, A. Ribeiro, and J.M. Cañas. A Cartographer Robot: Merging and Interpreting Maps in Unknown Environments. In *Proc. of the Third IFAC Symposium on Intelligent Autonomous Vehicles*, pages 730–734, Madrid, Spain, 1998.

[82] M. García-Silvente, J. Fdez-Valdivia, J.A. García, and A. Garrido. A new edge detector integrating scale-spectrum information. *Image and Vision Computing*, 15(12):913–923, 1997.

[83] J. Gasós and A. Saffiotti. Using fuzzy sets to represent uncertain spatial knowledge in autonomous robots. *Spatial Cognition and Computation*, 1(3):205–226, 1999.

[84] E. Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots*. PhD thesis, Virginia Polytechnic Institute, 1991.

[85] P. Gaussier, C. Joulain, S. Zrehen, and A. Revel. Visual Navigation in an Open Environment without Map. In *IEEE Int. Conf. Intelligent Robots and Systems*, pages 545–550, 1997.

[86] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding: CVIU*, 73(1):82–98, 1999.

[87] P. Y. Glorennec. Adaptive fuzzy control. In *Fourth International Fuzzy Systems Association World Congress (IFSA'91)*, pages 33–36, 1991.

[88] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learning. *Addison-Wesley, Reading, MA*, 1989.

[89] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behaviour. *Adaptive Behaviour*, 5:317–342, 1997.

[90] R. González. *Digital Image processing*. Addison-Wesley Iberoamericana, SA, 1996.

[91] D. Grest and R. Koch. Realtime multi-camera person tracking for immersive environments. In *IEEE 6th Workshop on Multimedia Signal Processing*, pages 387–390, 2004.

[92] M.S. Grewal and A.P. Andrews. *Kalman Filtering. Theory and Practice*. Prentice Hall, 1993.

[93] N. Guil, J. Villalba, and E.L. Zapata. A fast hough transform for segment detection. *IEEE Transactions on Image Processing*, 11:1541 – 1548, 1995.

[94] H. B. Gurocak. A genetic-algorithm-based method for tuning fuzzy logic controllers. *Fuzzy Sets and Systems*, 108(1):39–47, 1999.

[95] I. Haritaoglu, D. Beymer, and M. Flickner. Ghost 3d: detecting body posture and parts using stereo. In *Workshop on Motion and Video Computing*, pages 175 – 180, 2002.

[96] M. Harville. Stereo person tracking with adaptive plan-view templates of height and occupancy statistics. *Image and Vision Computing*, 2:127–142, 2004.

[97] M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 3–11, 2001.

[98] K. Hayashi, M. Hashimoto, K. Sumi, and K. Sasakawa. Multiple-person tracker with a fixed slanting stereo camera. In *6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 681–686, 2004.

[99] F. Herrera, M. Lozano, and J.L. Verdegay. Tuning of fuzzy controllers by genetic algorithms. *International Journal of Approximate Reasoning*, 12:299–315, 1995.

[100] S. R. Herwitz, L. F. Johnson, S. E. Dunagan, R. G. Higgins, D. V. Sullivan, J. Zheng, B. M. Lobitz, J. G. Leung, B. A. Gallmeyer, and M. Aoyagi et al. Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. *Computers and Electronics in Agriculture*, 44:49–61, 2004.

[101] P. Hoppenot and E. Colle. Localization and control of a rehabilitation mobile robot by close human-machine cooperation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9:181 – 190, 2001.

[102] J. Horn and G. Schmidt. Continous Localization of a Mobile Robot Based in 3D-Laser-Range-Data. *Robotics and Autonomous Systems*, 14:99–118, 1995.

[103] A. Howard, H. Seraji, and E. Tunstel. A Rule-Based Fuzzy Traversability Index for Mobile Robot Navigation. In *IEEE International Conference on Robotics and Automation*, pages 3067–3071, 2001.

[104] I.W. Hunter, S. Lafontaine, P.M.F. Nielsen, P.J. Hunter, and J.M. Hollerbach. Manipulation and dynamic mechanical testing of microscopic objects using a tele-micro-robot system. *IEEE Control Systems Magazine*, 10:3 – 9, 1990.

[105] Intel. OpenCV: Open source Computer Vision library. *http://www.intel.com/research/mrl/opencv/*.

[106] D. Iraca, L. Landini, and L. Verrazzani. Power spectrum equalization for ultrasonic image restoration. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 36:216 – 222, 1989.

[107] H. Ishibuchi and S. Kaige. Implementation of simple multiobjective memetic algorithms and its application to knapsack problems. *International Journal of Hybrid Intelligent Systems*, 1(1):22–35, 2004.

[108] H. Ishibuchi, T. Murata, and I. B. Turksen. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Sets and Systems*, 89(2):135–150, 1997.

[109] H. Ishibuchi, T. Nakashima, and M. Nii. *Classification and Modeling with Linguistic Information Granules: Advances Approaches to Linguistic Data Mining*. Springer, 2004.

[110] H. Ishibuchi and T. Yamamoto. Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers. In *Proc. of Genetic and Evolutionary Computation (GECCO 2003), Part I*, pages 1077–1088, 2003.

[111] E.N. Johnson, A.A. Proctor, H. Jincheol, and A.R. Tannenbaum. Visual search automation for unmanned aerial vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, 41:219 – 232, 2005.

[112] J.L. Jones and A. M. Flynn. *Mobile Robots. Inspiration to Implementation*. A K Peters, 1993.

[113] M.R. Kabuka and A.E. Arenas. Position Verification of a Mobile Robot Using Standart Pattern. *IEEE J. Robotics and Automation*, 3:505–516, 1987.

[114] T. Kailath. The Divergence and Bhattacharyya Distance Measures in Signal Selection. *IEEE Transactions on Communication Technology*, 15:52 – 60, 1967.

[115] A.R. Kalukin and V. Sankaran. Three-dimensional visualization of multilayered assemblies using x-ray laminography. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A*, 20:8 – 17, 1997.

[116] C. Karr. Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):26–33, 1991.

[117] R. Katsuki, J. Ota, T. Mizuta, T. Kito, T. Arai, T. Ueyama, and T. Nishiyama. Design of an artificial mark to determine 3d pose by monocular vision. In *IEEE International Conference on Robotics and Automation, 2003. Proceedings ICRA '03*, volume 1(14-19), pages 995–1000, 2003.

[118] D. Kim and R. Nevatia. Recognition and localization of generic objects for indoor navigation using functionality. *Image and Vision Computing*, 16(11):729–743, 1998.

[119] K. Konolige and K. Myers. *In Artificial Intelligence and Mobile Robots:Case Studies of Successful Robot Systems*. The MIT Press, 1997.

[120] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proc. of the Twelfth National Conf. on AI (AAAI-94)*, pages 979–984, Menlo Park, Calif., 1994.

[121] C. Kotropoulos, X. Magnisalis, I. Pitas, and M.G. Strintzis. Nonlinear ultrasonic image processing based on signal-adaptive filters and self-organizing neural networks. *IEEE Transactions on Image Processing*, 3:65 – 77, 1994.

[122] E. Krotkov. Mobile Robot Localization Using a Simgle Image. In *Int. Conf. on Robotics and automation*, number 978-983, 1988.

[123] A. Krupa, J. Gangloff, C. Doignon, M.F. de Mathelin, G. Morel, J. Leroyand L. Soler, and J. Marescaux. Autonomous 3-d positioning of surgical instruments in robotized laparoscopic surgery using visual servoing. *IEEE Transactions on Robotics and Automation*, 19:842 – 853, 2003.

[124] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and Robust Face Finding via Local Context. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003.

[125] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[126] B. Kuipers and Y.T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[127] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, G. Yanlin, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt. Aerial video surveillance and exploitation. *Proceedings of the IEEE*, 89:1518 – 1539, 2001.

[128] K.S. Kump, G.M. Saidel, and D.L. Wilson. Comparison of algorithms for combining x-ray angiography images. *IEEE Transactions on Medical Imaging*, 20:742 – 750, 2001.

[129] Blake A. Langland, Otto L. Jansky, Joseph S. Byrd, and Robert O. Pettus. The integration of dissimilar control architectures for mobile robot applications. *Journal of robotic systems*, 14(4):251–262, 1997.

[130] S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, and A. Klein. Training personal robots using natural language instruction. *IEEE Intelligent Systems and Their Applications*, 16:38 – 45, 2001.

[131] C. Lazarus and H. Hu. Evolving goalkeeper behaviours for simulated soccer competition. In *3th IASTED Int. Conf. on Artificial Intelligence and Applications (AIA 2003)*, 2004.

[132] Hao Li and S.X. Yang. A behavior-based mobile robot with a visual landmark-recognition system. *IEEE/ASME Transactions on Mechatronics*, 8:390–400, 2003.

[133] W. Li, X. Jiang, and Y. Wang. Road recognition for vision navigation of an autonomous vehicle by fuzzy reasoning. *Fuzzy Sets and Systems*, 93:275–280, 1998.

[134] W. Liang, H. Weiming, and L. Tieniu. Recent developments in human motion analysis. *Pattern Recognition*, 36:585–601, 2003.

[135] R. Lienhart and J. Maydt. An Extended Set of Haar-Like Features for rapid Object detection. In *IEEE Conf. on Image Processing*, pages 900–903, 2002.

[136] B. Lipinski, H. Herzog, E. Rota Kops, W. Oberschelp, and H.W. Muller-Gartner. Expectation maximization reconstruction of positron emission tomography images using

anatomical magnetic resonance information. *IEEE Transactions on Medical Imaging*, 16:129 – 136, 1997.

[137] H.H. Lund. Modern artificial intelligence for human-robot interaction. *Proceedings of the IEEE*, 92:1821 – 1838, 2004.

[138] R. Madhavan, H. Durrant-Whyte, and G. Dissanayake. Natural landmark-based autonomous navigation using curvature scale space. In *IEEE International Conference on Robotics and Automation (ICRA '02)*, volume 4, pages 3936 – 3941, 2002.

[139] P. Maes and R. Brooks. The dynamics of action selection. In *11th International Joint Conference on Artificial Intelligence (IJCA'89)*, pages 991–997, 1989.

[140] P. Maes and R. Brooks. Learning to coordinate behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI'90)*, pages 796–802, Boston, MA, 1990.

[141] L. Magdalena. Adapting the gain of an FLC with genetic algorithms. *International journal of Approximate reasoning*, 17(4):327–349, 1997.

[142] R. Mahony and T. Hamel. Image-based visual servo control of aerial robotic systems using linear image features. *IEEE Transactions on Robotics*, 21:227 – 239, 2005.

[143] M. Marefat and R.L. Kashyap. Image interpretation and object recognition in manufacturing. *IEEE Control Systems Magazine*, 11:8 – 17, 1991.

[144] B. Martinkauppi, M. Soriano, and M. Pietikainen. Detection of skin color under changing illumination: a comparative study. In *12th International Conference on Image Analysis and Processing*, pages 652 – 657, 2003.

[145] M. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

[146] M. Mataric, Matthew Williamson, John Demiris, and Aswath Mohan. Behavior-based primitives for articulated control. In *Proceedings, From Animals to Animats 5,*

Fifth International Conference on Simulation of Adaptive Behavior (SAB-98), Zurich, Switzerland, agosto 1998.

[147] V. Matellan, J. M. Molina, and J. Sanz. Learning fuzzy reactive behaviours in autonomous robots. In *4th workshop on Learning Robots*, 1995.

[148] L. Matthies and S.A. Shafer. Error Modelling in Stereo Vision. *Robotics and Automation*, 3:239–248, 1987.

[149] O. Miglino, C. Nafasi, and C. Taylor. Selection for Wandering Behaviour in a Small Robot. Technical Report UCLA-CRSP-94-01, Department of Cognitive Science, UCLA, 1994.

[150] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[151] R. Mohan, G. Medioni, and R. Nevatia. Stereo error detection, correction, and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:113 – 120, 1989.

[152] I. Monasterio, E. Lazkano, I. Rañó, and B. Sierra. Learning to traverse door using visual information. *Mathematics and Computer in Simulation*, 60:347–356, 2002.

[153] H.P. Moravec. The Standford Cart and the CMU Rover. *Proc. IEEE*, 71:872–884, 1983.

[154] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 116–121, 1985.

[155] F.J. Moreno-Velo, I. Baturone, R. Senhadji, and S. Sanchez-Solano. Tuning complex fuzzy systems by supervised learning algorithms. In *The 12th IEEE International Conference on Fuzzy Systems*, volume 1, pages 226–231, 2003.

[156] D.E. Moriarty and R. Miikkulainen. Evolving obstacle avoidance behaviour in a robot arm. In *4th Int. Conf. on Simulation of Adaptive Behaviour (SAB96)*, pages 468–475.

[157] M. Mucientes and J. Casillas. Obtaining a fuzzy controller with high interpretability in mobile robots navigation. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004)*, pages 1637–1642, 2004.

[158] D. Murray and J.J. Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8:161–171, 2000.

[159] R. Muñoz-Salinas, E. Aguirre, O. Cordón, and M. García-Silvente. Automatic tuning of a fuzzy visual system using evolutionary algorithms: single-objective vs. multiobjective approaches. *To Appear in IEEE Transactions of Fuzzy Systems*, 2006.

[160] R. Muñoz-Salinas, E. Aguirre, and M. García-Silvente. Door-detection using artificial vision and fuzzy logic. *To Appear in Autonomous Robots*, 2006.

[161] R. Muñoz-Salinas, E. Aguirre, M. García-Silvente, and M. Gómez. A multi-agent system architecture for mobile robot navigation based on fuzzy and visual behaviours. *Robotica*, 23:689–699, 2005.

[162] R. Muñoz-Salinas, E. Aguirre, M. García-Silvente, and A. González. A fuzzy system for visual detection of interest in human-robot interaction. In *2nd International Conference on Machine Intelligence (ACIDCA-ICMI'2005)*, pages 574–581, 2005.

[163] R. Muñoz-Salinas, E. Aguirre, M. García-Silvente, and A. González. People detection and tracking through stereo vision for human-robot interaction. *Lectures Notes on Artifficial Intelligence*, (3789):337–346, 2005.

[164] E. Chown nad S. Kaplan and D. Kortenkamp. Prototypes, Location and Associative Networks (plan): Towards a Unified Theory of Cognitive Mapping. *Cognitive Science*, 19:1–51, 1995.

[165] A.L. Nelson, E. Grant, J.M. Galeotti, and S. Rhody. Maze exploration behaviours using an integrated evolutionary robotics environment. *Robotics and Autonomous Systems*, 46:135–150, 2004.

[166] A. Newell and H.A. Simon. Computer science as empirical enquiry. *Communications of the ACM*, 19:113–126, 1976.

[167] K. Nickel, E. Seemann, and R. Stiefelhagen. 3D-Tracking of Head and Hands for Pointing Gesture Recognition in a Human-Robot Interaction Scenario. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FGR'04)*, pages 565–570, 2004.

[168] N.J. Nilsson. A mobile automaton: An application of AI techniques. In *Proc. of the First Int. Joint Conf. on Artificial Intelligence*, pages 509–520, San Francisco, USA, 1969.

[169] S. Nolfi and D. Marocco. Evolving Robots Able to Visually Discriminate Between Objects with Different Size. *International Journal of Robotics and Automation*, 17:163–170, 2002.

[170] H. Nomura, H. Hayashi, and N. Wakami. A self-tuning method of fuzzy control by descendent method. In *Fourth International Fuzzy Systems Association World Congress (IFSA'91)*, pages 155–158, 1991.

[171] K. Nummiaro, E. Koller-Meier, and L.V Gool. An Adaptive Color-Based Particle Filter. *Image and Vision Computing*, 21(1):99–110, 2003.

[172] G. Oriolo, G. Ulivi, and M. Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics*, 28(3):316–333, 1998.

[173] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions On-Systems, Man, and Cybernetics*, SMC-9(1):66–69, 1979.

[174] J. Pan, D.J., A. Kosaka, and A.C. Kak. FUZZY-NAV: A Vision-Based Robot Navigation Architecture Using Fuzzy Inference for Uncertainty-Reasoning. In *IEEE World Congress Neural Networks*, volume 2, pages 602–607, 1995.

[175] P. Panaite and A. Pecl. Exploring unknown undirected graphs. *Journal of Algorihtms*, 33(2):281–295, 1999.

[176] V. Pareto. *Cours D'Economie Politique*. Rouge, Lousanne, Switzerland, 1896-7.

[177] D. Park, A. Kandel, and G. Langholz. Genetic-based new fuzzy reasoning models with applications to fuzzy control. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):39–47, 1994.

[178] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42:271–281, 2003.

[179] T. J. Procyk and E. H. Mamdani. A linguistic self-organizing process controller. *Automatica*, 15(1):15–30, 1979.

[180] PtGrey. Bumblebee. *http://www.ptgrey.com/products/bumblebee/index.html*.

[181] M. Raibert. *Legged robots that balance*. MIT Press, 1986.

[182] C. Rekeczky, I. Szatmari, D. Balya, G. Timar, and A. Zarandy. Cellular multiadaptive analogic architecture: a computational framework for uav applications. *IEEE Transactions on Circuits and Systems I*, 51:864 – 884, 2004.

[183] C.W. Reynolds. Evolution of Corridor Following Behaviour in a Noisy World. In *3rd Int. Conf. on Simulation of Adaptive Behaviour (SAB94)*, pages 402–410, 1994.

[184] J.J. Rodriguez and J.K. Aggarwal. Stochastic analysis of stereo quantization error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:467 – 470, 1990.

[185] K. Sabe, M. Fukuchi, J.-S-Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara. Obstacle avoidance and path planning for humanoid robots using stereo vision. In *IEEE International Conference on Robotics and Automation (ICRA'04)*, volume 1, pages 592 – 597, 2004.

[186] A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, 1:180–197, 1997.

[187] A. Saffiotti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76:481–526, 1995.

[188] A. Saffiotti and L.P. Wesley. Perception-based self-localization using fuzzy locations. In M. van Lambalgen L. Dorst and F. Voorbraak, editors, *Reasoning with Uncertainty in Robotics.LNAI*, pages 368–385, Berlin, DE, 1996. Springer-Verlag.

[189] V. Sankaran, A.R. Kalukin, , and R.P. Kraft. Improvements to x-ray laminography for automated inspection of solder joints. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C*, 21:148 – 154, 1998.

[190] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent Stereo in Autonomous Navigation: from Bees to Robots. *Int. J. Computer Vision*, 14:159–177, 1995.

[191] D. Scharstein and A. J. Briggs. Real-time recognition of self-similar landmarks. *Image and Vision Computing*, 19:763–772, 2001.

[192] K. Severinson-Eklundh, A. Green, and H. Hüttenrauch. Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, 42:223–234, 2003.

[193] K.K. Shung and M. Zippuro. Ultrasonic transducers and arrays. *IEEE Engineering in Medicine and Biology Magazine*, 15:20 – 30, 1996.

[194] R. Siegwart, K. O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, and M. Meisser. Robox at expo.02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42:203–222, 2003.

[195] R. Siegwart and I.R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[196] L. Sigal, S. Sclaroff, and V. Athitsos. Skin color-based video segmentation under time-varying illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:862 – 877, 2004.

[197] L. Snidaro, C. Micheloni, and C. Chiavedale. Video security for ambient intelligence. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35:133 – 144, 2005.

[198] J.L. Solka, D.J. Marchette, B.C. Wallet, V.L. Irwin, and G.W. Rogers. Identification of man-made regions in unmanned aerial vehicle imagery and videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:852 – 857, 1998.

[199] N. Srinivas and K. Deb. Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2:221–248, 1995.

[200] S. A. Stoeter, F. L. Mauff, and N. P. Papanikolopoulos. Real-time door detection in cluttered enviroments. *Proceeding of the 15th IEEE International Symposium on Intelligent Control (ISIC 200)*, pages 187–191, 2000.

[201] K. Sugihara. Some Location Problems for Robot Navigation Using a Single Camera. *Computer Vision, Graphics, and Image Processing*, 42:112–129, 1988.

[202] R. Tanawongsuwan. Robust Tracking of People by a Mobile Robotic Agent. Technical Report GIT-GVU-99-19, Georgia Tech University, 1999.

[203] K. Tashiro, J. Ota, Y.C. Lin, and T. Arai. Design of the optimal arrangement of artificial landmarks. *IEEE International Conference on Robotics and Automation*, pages 407–413, 1995.

[204] S. Thompson and S. Kagami. Stereo vision terrain modeling for non-planar mobile robot mapping and navigation. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 6, pages 5392 – 5397, 2004.

[205] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[206] S.V. Tipnis, V.V. Nagarkar, V. Gaysinskiy, S.R. Muller, and I. Shestakova. High-speed x-ray imaging camera for time-resolved diffraction studies. *IEEE Transactions on Nuclear Science*, 49:2415 – 2419, 2002.

[207] S. Todorovic and M.C.Nechyba. A vision system for intelligent mission profiles of micro air vehicles. *IEEE Transactions on Vehicular Technology*, 53:1713 – 1725, 2004.

[208] D. Tomazevic, B. Likar, and T. Slivnikand F. Pernus. 3-d/2-d registration of ct and mr to x-ray images. *IEEE Transactions on Medical Imaging*, 22:1407 – 1416, 2003.

[209] M.M. Trivedi, C. Chen, and S.B. Marapane. A vision system for robotic inspection and manipulation. *Computer*, 22:91 – 97, 1989.

[210] T. Tsumura. Survey on Automated Guided Vehicle in Japanese Factory. In *Proc. Int. Conf. Robotics and Automation*, pages 1329–1334, 1986.

[211] J. Uribe, J. Mujika, and R. Braunstingl. Controlador fuzzy optimizado mediante algoritmos genéticos para el seguimiento de paredes en un robot móvil. In *ESTYLF*, pages 211–216, 1995.

[212] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[213] P. S. Vishnupad and Y. C. Shin. Adaptive tuning of fuzzy membership functions for non-linear optimization using gradient descendent method. *Journal of Intelligent and Fuzzy Systems*, 7:13–25, 1999.

[214] M. Wahde and H. Sandholt. Evolving Complex Behaviors on Autonomous Robots. In *7th UK Mechatronics Forum International Conference (Mechatronics 2000)*, 2000.

[215] M. Walker. Evolution of a Robotic Soccer Player. *Research Letters in the Information and Mathematical Sciences*, 3:15–23, 2002.

[216] J. Wiley. Movile robot navigation using artificial landmarks. *Journal of Robotc Systems*, 14(2):93–106, 1997.

[217] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10:115–152, 1995.

[218] C.R. Wren, A. Azarbayejani, T. Darrell, Trevor, and A.P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785, 1997.

[219] O.A. Yakimenko, I.I. Kaminer, W.J. Lentz, and P.A. Ghyzel. Unmanned aircraft navigation for shipboard landing using infrared vision. *IEEE Transactions on Aerospace and Electronic Systems*, 38:1181 – 1200, 2002.

[220] M.H. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[221] D.C.K. Yuen and B.A. MacDonald. Vision-based localization algorithm based on landmark matching, triangulation, reconstruction, and comparison. *IEEE Transactions on Robotics*, 21:217 – 226, 2005.

[222] L.A. Zadeh. The concept of linguistic variable and its applications to approximate reasoning. *Parte I Information Sciences vol. 8, pag. 199-249, Parte II Information Sciences vol. 8, pag. 301-357, Parte III Information Sciences vol. 9, pag. 43-80*, 1975.

[223] Z. Zhang and O. Faugeras. A 3D World Model Builder with a Mobile Robot. *Int. J. Robotics Research*, 11:269–285, 1992.

[224] L. Zheng. A practical guide to tune proportional and integral (PI) like fuzzy controllers. In *First IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'92)*, pages 633–640, 1992.

[225] J. Zhou and M. Abdel-Mottaleb. A content-based system for human identification based on bitewing dental x-ray images. *Pattern Recognition*, 38:2132–2142, 2005.

[226] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[227] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Proc. of Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN2001)*, pages 95–100, 2001.

[228] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.