UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

Tesis Doctoral

SISTEMA DE COORDINACIÓN MULTIAGENTE BASADO EN COMPORTAMIENTOS APRENDIDOS

Programa de doctorado: Ingeniería de Telecomunicación

Autor: José Manuel Peula Palacios

Directora: Dra. Cristina Urdiales García

MÁLAGA, 2015

Publicaciones y
Divulgación Científica

AUTOR: José Manuel Peula Palacios

http://orcid.org/0000-0002-3798-5195

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

Dª CRISTINA URDIALES GARCÍA, PROFESORA TITULAR DEL DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA DE LA UNIVERSIDAD DE MÁLAGA

CERTIFICA:

Que D. José Manuel Peula Palacios, Ingeniero de Telecomunicación, ha realizado en el Departamento de Tecnología Electrónica de la Universidad de Málaga, bajo mi dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

"SISTEMA DE COORDINACIÓN MULTIAGENTE BASADO EN COMPORTAMIENTOS APRENDIDOS"

Revisado el presente trabajo, estimo que puede ser presentado al Tribunal que ha de juzgarlo.

Y para que conste a efectos de lo establecido por la normativa de la Universidad de Málaga, **AUTORIZO la presentación de esta Tesis en la Universidad de Málaga.**

Málaga, a 22 de octubre de 2015

Fdo. Cristina Urdiales García

Profesora Titular del Departamento de Tecnología Electrónica

A los que han estado,
a los que están
y a los que estarán

Agradecimientos

Dice el refrán que *de bien nacíos es ser agradecíos*. Aunque en general no suelo hacer caso a los refranes, pero creo que la ocasión merece hacer una excepción.

Así pues, en primer lugar querría agradecer a mi familia (papa, mama, Salvi, Chelo, Virgi y Alicia) por estar ahí y, aunque no se si en algún momento han tenido claro que estaba haciendo ni para qué servía, pero gracias por interesaros y apoyarme. Especialmente querría agradecer a Chelo por ser tan *insistente* y tener, aún más ganas que yo, de que acabase la tesis.

Por otro lado, querría mencionar de forma especial y obligatoria a Cristina, a quien debo un múltiple agradecimiento por su ayuda con el proyecto fin de carrera, el diploma de estudios avanzados y la presente tesis. Gracias por tus locas y descabelladas ideas que, aunque no siempre llevasen a buen puerto, pero al menos nos echábamos unas risas en el intento. Gracias también por estar siempre ingeniando formas de solucionar los problemas y por apoyarme hasta el final. Igualmente, querría agradecer a Francisco Sandoval por su constante apoyo, esfuerzo e interés para que finalizase la presente tesis.

Como no, quiero también agradecer a todo ese torrente de personas que han estado presentes a lo largo de la realización de este trabajo aportando sus ideas, criticando, apoyando, dando ánimos, acompañando en las comidas, echando unas risas, etc. Estoy seguro de que este listado no está completo, pero aún así quiero agradecer concretamente a Jose Carlos, Jose Ramón, Pedro, Richar, Jose Manuel, Juanpe, Antonio Bandera, Rebeca, Raquel, Raúl, Ale, Paqui, Antonio Palomino, Cubi, Alejandro, Manolo, Pepe, Cisco, Javi, Débora, Yolanda, Miguel, Jesus, Blanquilla, Lydia, Carlos, Joaquin, Inma, Mariluz, Adri, Gerardo, David, Fran, Eu, Cesar, Eva, Bernardo, Eva, Zuma, Jacob, Fuen, Gloria, Juanma, Mario, Laura, Nacho, Edu, Javier, Roberta, Alessia, Ulises, Darío, Thomas, Dani, Josemi, Berti, Juan Ramón, Rafa, Carmen, *mini-Carmen*, Ana Belén, *Lola*, *al último teleko* y a *Scott*. A todos: ¡Muchas gracias!

También querría agradecer al Ministerio de Educación y Ciencia (MEC), al Ministerio de Ciencia e Innovación (MICINN) y a los Fondos FEDER por el

apoyo ofrecido a través de los proyectos “Sistema de Navegación Multiagente Cooperativo para Entornos Sanitarios” (TIN2004-07741-C02-01), “Perception and senso-motor learning system for humanoid platforms” (TIN2005-01359), “Arquitectura de ServicioS Inteligente de SoporTe a Personas con necesidades especiales (ASSIST)” (TEC2006-11689-C01), “Arquitectura de Servicios Inteligente de SoporTe a la Independencia mediante Robots (ASISTIR)” (TEC2008-06734-C02-01) y “Tecnología Asistiva para Rehabilitación Colaborativa” (TEC2011-29106-C02-01). De igual manera, querría agradecer a la Junta de Andalucía por su ayuda a través los proyectos SIAMA (TIC 249) y SIAD (TIC-3991), a la Unión Europea en el VI Programa Marco (FP6-2005-IST-5. STREP no. 045088) por el proyecto “Supported Human Autonomy for Recovery and Enhancement of cognitive and motor abilities using information technologies (SHARE-it)” (Ref.STREP no. 045088) y a Andalucía Tech.

También querría agradecer al grupo KEMLg (Knowledge Engineering and Machine Learning Group) de la Universidad Politécnica de Cataluña (UPC), por permitirnos usar su implementación del servidor CBR en la presente tesis así como en otros trabajos.

Por último, y ya que seguramente haya olvidado mencionar expresamente a alguien en los párrafos anteriores pues han sido muchos años y muchas personas, querría agradecer de forma genérica a todos los que han ayudado o participado de alguna manera en la realización de esta tesis, así como a todos aquellos que no han molestado, pues *mucho ayuda el que poco estorba*. Por lo tanto: ¡Muchas gracias a todos!

Resumen

La robótica ha cambiado enormemente desde sus orígenes hasta la actualidad. Así pues, se ha pasado de la concepción de robots, que prácticamente eran autómatas que realizaban una simple tarea repetitiva de forma automática prácticamente sin interactuar ni obtener ningún tipo de información del entorno, a los sistemas actuales con una gran versatilidad y capacidades de interacción a nivel táctil, visual y auditivo.

Uno de los grandes avances en la robótica, no porque la idea no hubiese surgido hace tiempo, sino porque prácticamente hasta hace poco no era factible técnicamente, es la aplicación de la inteligencia artificial y la interacción entre distintas entidades. Ambos elementos están altamente relacionados, pues no es posible obtener una gran interacción ni una interacción natural si las entidades que interactúan no disponen de una cierta inteligencia, capacidad de adaptación y de aprendizaje.

En el caso de un robot, las posibles interacciones que puede tener son la interacción robot-robot y la interacción robot-humano. Ambos tipos de interacción son hoy en día áreas de trabajo muy activas, pues suponen retos interesantes, complejos y con distintos puntos de vista y posibles soluciones, muchas de ellas válidas dependiendo de la situación y problema concreto.

La presente tesis se centra en la interacción robot-robot, concretamente en la coordinación entre robots, área activa desde hace años y todavía con mucho interés.

Concretamente, en este trabajo se presenta un sistema de coordinación implícita multi-agente basado en el aprendizaje de comportamientos mediante aprendizaje por demostración.

La coordinación implícita se basa en que la coordinación surge a partir de la interacción entre los distintos agentes del sistema, los cuales actúan de forma autónoma e independiente entre ellos, aunque pudiendo tener conocimiento de la existencia de los demás agentes. Este enfoque ha sido utilizado frente a la coordinación explícita porque encaja más fácilmente en una arquitectura basada en comportamientos, cuyo principio de funcionamiento, de abajo hacia arriba, consiste en el desarrollo de comportamientos simples pa-

ra que la combinación de éstos de lugar a un comportamiento emergente, de alto nivel, más complejo que los comportamientos simples desarrollados, de forma similar a lo que ocurre en las colonias de hormigas, abejas o termitas.

Se ha seleccionado aprendizaje frente al uso de algoritmos analíticos porque el aprendizaje permite una mayor libertad a la hora de crear los comportamientos y no requiere de un modelado tan estricto como, en general, los algoritmos analíticos. De entre las distintas técnicas de aprendizaje, el aprendizaje por demostración es una técnica que asocia una acción a un estado y permite entrenar un comportamiento específico a partir de ejemplos provistos por un usuario supervisor. Esta técnica de aprendizaje ha sido seleccionada porque resulta especialmente interesante en el entrenamiento de robots, pues permite absorber, implícitamente durante el aprendizaje, distintos factores difíciles de modelar, como imperfecciones en los robots (derivas en el movimiento debido a que los motores no están correctamente alineados), errores en los sensores, ruido, etc. La razón es que estos factores son tenidos en cuenta durante el entrenamiento ya que son extraídos directamente de datos reales del entorno. Así pues, esta técnica solventa uno de los problemas que supone trabajar con robots: su modelado.

Por último, dentro del aprendizaje por demostración hay una gran variedad de estrategias de aprendizaje como aprendizaje mediante razonamiento basado en casos (CBR), aprendizaje basado en árboles de decisión, aprendizaje bayesiano, etc. Cada estrategia tiene sus ventajas y desventajas, sin embargo se ha escogido el CBR porque es muy intuitivo para las personas, está estrechamente relacionado con el modo en que piensan las personas y encaja a la perfección con el concepto de comportamiento reactivo, cuya principio de funcionamiento consiste en acoplar las entradas de los sensores a los actuadores o, visto de otra forma, asociar una situación a una solución, que es lo que hace el CBR.

El sistema ha sido probado con robots AIBO ERS-7 de Sony, usando localización visual mediante marcas artificiales de colores en un entorno similar al de la competición robótica de fútbol Robocup. Para ello se entrenan comportamientos simples, como correr por la banda, ir a portería o evitar a un oponente, y se combinan mediante una capa superior que los activa y desactiva según la situación lo requiera.

Abstract

Robotics has largely evolved since its origins till nowadays. At first, a robot was just a machine that could be programmed to perform repetitive tasks in an automatic way without variation nor environment feedback. A robot, nowadays, is a complex and versatile system with high interaction capabilities at tactile, visual and auditive level.

The last advances in robotics have enabled Artificial Intelligence and interaction between different entities. Although these advances are really old ideas, they have not been really possible till nowadays due to the advances in technology. These two elements have a close relationship, as it is not possible to achieve a deep nor natural interaction without some intelligence, adaptation or learning skill between the entities that interact.

Considering a robot, possible interactions are robot-robot or human-robot interaction. Both are, nowadays, very active working areas that present really interesting and complex challenges. Indeed, there are very different possible solutions to these problems, depending the solution on the situation and specific problem to solve.

The present dissertation focuses in robot-robot interaction and, within this area, in robot coordination, an active area for the last 20 years. Specifically, this work presents a multi-agent implicit coordinated system based on behaviours learned via Learning from Demonstration (LfD).

Implicit coordination is based on the interaction between the agents' system which act on their own, whether they know or not about the existence and position of other agents. Hence, in implicit coordination each agent takes its own decisions based on whatever it knows about its environment, but not explicitly coordinating with other agents. We choose this approach instead of an explicit coordination one because implicit coordination fits easier in a behaviour-based architecture, since a behaviour-based architecture consists of the development of simple behaviours that, when combined, result in an emergent more complex behaviour than the simple developed ones. This is similar to ant, bees and termites colonies behaviour.

On the other hand, we have chosen learning instead of analytic algorithms

because learning allows a higher freedom in behavior creation and does not require a strict model as, in general, analytic algorithms do. Among the different approaches to learning, we have chosen LfD. This technique associates an action to an state and consists in training a behaviour using specific examples given from supervisor. Thus, this technique is specially interesting for robot learning, as it implicitly absorbs, during the training, different factors that are not easy to model, such as robot imperfections (drift in movement due to bad motor calibration), sensor errors, noise, etc. Moreover, this technique models these factors in a realistic way, as they are extracted directly from reality. Thus, this technique solves one of problems that means working with robots: robot modelling.

Finally, LfD can be achieved using different learning techniques, such as Case-Based Reasoning (CBR), decision-tree learning, Bayesian learning, etc. Each strategy has its advantages and disadvantages. However, CBR has been chosen because it is very intuitive to use for people, it is closely connected to the way people think and it fits perfectly in the concept of reactive behaviour. The reason is that reactive behaviours consists in coupling sensor inputs to actuators or, from another point of view, coupling a situation to a solution, that is what CBR does.

This system has been tested using AIBO ERS-7 robots from Sony. Robot's localization is based in colored artificial visual landmarks in the environment, similar to artificial landmarks used in Robocup soccer league. Thus, robots are trained in simple behaviours, such as *reach goal*, *move following the sideline* or *avoid opponent*, and then they are combined and activated, using a high level layer, depending on the specific situation they face.

Índice general

Índice de figuras

Índice de tablas

Índice de acrónimos

AIBO Robot de Inteligencia Artificial.

ANN Redes Neuronales Artificiales.

CBR Razonamiento Basado en Casos.

CSV Valores Separados por Comas.

DLA Arquitectura en Capas Distribuída.

DWA Método de Ventana Dinámica.

GA Algoritmos Genéticos.

HSI Hue, Saturation, and Intensity.

HSV Hue, Saturation, and Value.

IA Inteligencia Artificial.

LfD Aprendizaje por Demostración.

MAS Sistemas Multi-Agente.

MRS Sistemas Multi-Robot.

OpenCV Librería de Código Abierto para Visión por Ordenador.

PFA Método de Campos Potenciales.

Pyro Python para Robótica.

RA Realidad Aumentada.

RBS Sistemas Basados en Reglas.

ROS Sistema Operativo Robótico.

SPA Percibir-Planificar-Actuar.

URBI Interfaz Universal para Plataformas Robóticas.

VFH Histograma de Campo de Vectores.

Capítulo 1

Introducción

1.1. Introducción

En este capítulo se presenta una introducción a la robótica, así como su evolución desde sus orígenes hasta la actualidad, finalizando este apartado con los Sistemas Multi-Robot (MRS) (*Multi-Robot System* en inglés). Tras esto, se indicarán los objetivos de la presente tesis y la estructura del documento.

1.2. La robótica

Los últimos años han sido testigo del increíble e inexorable avance de la robótica desde sus tímidos primeros pasos hasta su situación actual. En sus comienzos, allá por los años 20, el concepto de robot, acuñado en la obra R.U.R de Karel Capek, no era más que un término perteneciente a la ciencia ficción. Sin embargo, en poco menos de 100 años este concepto ha pasado de ser algo ficticio a una realidad cotidiana, pues la robótica ha llegado a invadir incluso los hogares domésticos con robots para entretenimiento y ocio como el Robot de Inteligencia Artificial (AIBO) (*Artificial Intelligence roBOt* en inglés), Pleo o Sphero, o con robots aspiradoras como la Roomba, llegando a convertirse estos robots, en algunos casos, casi en un miembro más de la familia.

Todo esto ha sido posible gracias a los avances en muchas de las áreas de las que depende la robótica, como la visión, la Inteligencia Artificial (IA), la electrónica, etc. En general, uno de los mayores avances ha sido la evolución en la miniaturización electrónica, que ha permitido grandes capacidades de cómputo en dispositivos ligeros y de pequeño tamaño, permitiendo a la robótica autónoma mayores capacidades de cómputo, autonomía y movilidad.

1.2.1. Orígenes de la robótica

La primera referencia en la literatura a un ser parecido a un robot tiene lugar en *Las Argonáuticas* en torno al siglo IV a. C. El robot en cuestión sería Talos (Fig. 1.1), que según algunas interpretaciones sería un ser similar a un autómata construido por Hefesto, dios de la metalurgia, como un regalo a Minos, rey de Creta. Su misión era la de de proteger Creta y vigilar el correcto cumplimiento de las leyes.

Figura 1.1: Representación de Talos, la primera referencia a un robot según algunas interpretaciones.[2]

Sin embargo, y dejando aparte las posibles interpretaciones sobre la naturaleza de Talos, el primer *robot real* (entendido en su mínima expresión como una simple máquina automatizada con capacidad de movimiento) fue creado en la antigua Grecia en el siglo IV a. C. Según las crónicas, Arquitas de Tarento inventó y construyó un pájaro que movía las alas y era capaz de volar cortas distancias. En realidad, este pájaro era una pequeña máquina de vapor y se puede considerar como el primer autómata y dispositivo autopropulsado,

Dos siglos después, Ctesibio realizó varios inventos mecánicos basados en hidráulica como la clepsidra (un reloj de agua) con figuras móviles o un órgano hidráulico (que podría ser considerado como el precursor del órgano moderno). Varios siglos después, en el siglo I d.C. Herón de Alejandría creó, según los relatos, diversos autómatas como un teatro de figuras mecánicas.

Ya no hay muchas más referencias a inventos relacionados con la *robótica* o automática desde esta época hasta finales de la Edad Media, en el siglo XIII, en el que se vuelven a encontrar referencias a algunos autómatas como un supuesto hombre de hierro creado por Alberto Magno y los relojes animados de Al-Jazari.

[2]Imagen pública extraída de: `http://commons.wikimedia.org/wiki/File:Didrachm_Phaistos_obverse_CdM.jpg`. Última visita: 07-09-2015

A partir de esta época, y ya en la época del renacimiento, se encuentran más referencias a robots y autómatas, entre las que destacan el robot diseñado por Leonardo Da Vinci en el siglo XV (cuyos bocetos no fueron encontrados hasta mediados del siglo XX), los inventos de Jacques Vaucanson en el siglo XVIII, entre los que cabe destacar un músico mecánico que podía llegar a tocar distintas canciones, un pato automático (Fig. 1.2), que podía podía mover algunas partes y realizar funciones como beber o nadar, o los autómatas del suizo Pierre Jaquet-Droz, también en el siglo XVIII, conocidos como *La pianista*, *El dibujante* y *El escritor*, los cuales llegaron a ser contemplados por autoridades de Europa y Asia.

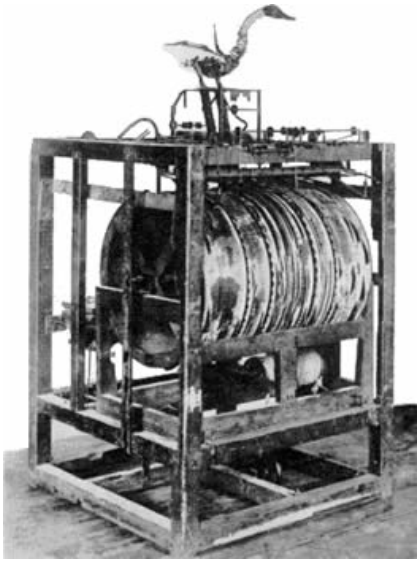

Figura 1.2: Pato de Vaucanson, destruido en 1789.[3]

Hasta el momento la mayoría de autómatas desarrollados (a excepción de los relojes) tenían como fin el ocio y el divertimento. Sin embargo, a principios del siglo XIX Joseph-Marie Jacquard inventó una máquina para la construcción de telares controlable mediante unas cartas perforadas. Estas cartas determinaban los patrones a desarrollar y facilitaban enormemente el funcionamiento de los telares. Este fue un invento realmente significativo para la producción en la industria textil pero, a su vez, fue un hito que inspiró ulteriores desarrollos de máquinas automáticas, ya que dio pie para que Charles Babbage comenzase a trabajar en su Máquina Analítica, una de las primeras maquinas computacionales y, posteriormente, George Boole inventase la lógica simbólica, que es utilizada en las computadoras actuales.

[3]Imagen pública (PD-US) extraída de: `http://commons.wikimedia.org/wiki/File:Vaucanson_duck1.jpg`. Última visita: 07-09-2015

Estos avances supusieron un empuje decisivo para el uso de los robots (aún autómatas) en la industria. Si bien es verdad, que el coste de estos sistemas en esta época era aún excesivo para muchas empresas y el exceso de mano de obra hacía más rentable contratar personas que comprar un robot que hiciese el trabajo.

1.2.2. Robótica industrial

Aunque ya se habían dado los primeros pasos para la aplicación de la *robótica* en la industria, a principios del siglo XIX no era común el uso de máquinas autómatas en este ámbito excepto por casos puntuales como el mencionado telar de Jacquard. Sin embargo, distintos sucesos en este siglo provocarían el desarrollo de la robótica industrial en un breve lapso de tiempo.

Para comenzar, el término robot fue acuñado a principios del siglo XIX por el autor checo Karel Capek (la palabra checa *robota*, significa "trabajo forzado") en su obra R.U.R. (Robots Universales Rossum). Aunque, esta obra tuvo un notable éxito, no fue hasta la llegada del bioquímico, divulgador científico y prolífico escritor Isaac Asimov que divulgaría el término robot como una palabra de uso común en la ciencia ficción, acuñando incluso el término robótica, como la ciencia dedicada al estudio de los robots, en su novela corta Runaround.

Dejando de lado el anecdótico origen de la palabra robot y su divulgación entre el público en general, hay que esperar hasta poco después del fin de la Segunda Guerra Mundial para encontrar los primeros trabajos que dieron lugar a la robótica industrial y, con ésta, la gran evolución que experimentó la robótica en este último siglo. Concretamente, a finales de los 40 comenzó en los laboratorios de Oak Ridge y Argonne National Laboratories el desarrollo de un robot para el manejo remoto de material radiactivo. Este robot era del tipo *maestro-esclavo*, estando diseñado para reproducir fielmente los movimientos de brazos y manos realizados por el operador que lo controlaba.

Poco después de la finalización de este proyecto, a mediados de los años 50 el inventor George C. Devol diseñó el primer robot programable, que era un brazo primitivo que se podía programar para realizar tareas específicas. Tras los primeros diseños, Devol se unió con Joseph Engelberger para crear la primera compañía fabricante de robots, llamada Consolidated Controls Corporation, posteriormente convertida en Unimation (de Universal Automation). Tras esto, a principios de los años 60, Devol y Engelberger consiguieron un contrato con la General Motors para instalar su brazo robótico, el Unimate, el cual con el tiempo se convertiría en el hoy conocido robot PUMA (Fig. 1.3). El robot Unimate es considerado el primer robot industrial de la

historia y su función era la de manipular y ensamblar piezas pesadas.

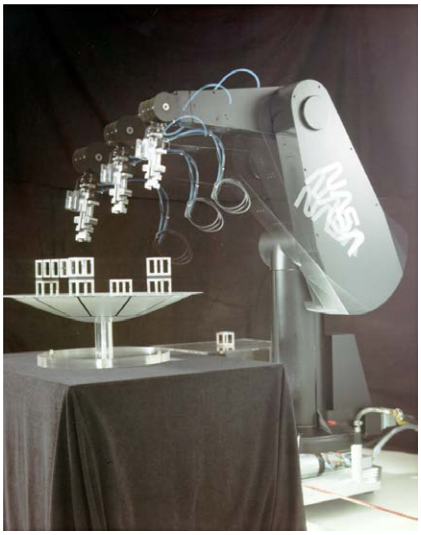

Figura 1.3: Brazo industrial PUMA, desarrollado por Unimation.[4]

Tras el éxito de estos primeros prototipos industriales comenzó, en la década de los 70, el imparable avance de la robótica industrial gracias al interés de grandes empresas como General Electric, General Motor, etc. De esta forma comenzó a mejorarse los diseños de los brazos robóticos aumentando sus grados de libertad, su precisión, facilidad de uso, etc, permitiendo su uso en tareas más generales y variadas, como soldadura o pintado, abarcando hoy en día todos los sectores industriales imaginables.

1.2.3. Robótica autónoma

La historia de los robots autónomos discurre, al menos parcialmente, paralela a la de los robots industriales, ya que si bien es verdad que algunos robots industriales pueden considerarse también robots autónomos, pero el objetivo de los robots autónomos es, en principio, más genérico que el de los robots industriales.

Aunque no existe una definición concreta de lo que es un robot autónomo, ya que esta depende en parte del entorno específico del robot y de su grado de autonomía, pero en general se considera que los primeros robots autónomos de la historia fueron creado a finales de la década de los 40 por William Grey Walter. Estos robots, llamados Elmer y Elsie, tenían forma

[4]Imagen pública extraída de: `http://commons.wikimedia.org/wiki/File:Puma_Robotic_Arm_-_GPN-2000-001817.jpg`. Última visita: 07-09-2015

de tortuga y se limitaban a moverse hacia una luz. Sin embargo, ya mostraban las características generales que se considera que debe poseer todo robot autónomo, y que son:

1. Obtener información sobre su entorno.
2. Actuar sin intervención humana.
3. Moverse o mover partes de su estructura a través de su entorno operativo sin intervención humana.
4. Evitar daños a personas, bienes o a sí mismos, a menos que sean parte de las especificaciones de diseño.

Sin embargo, los robots industriales de esta época no pueden considerarse autónomos ya que no disponían de estas capacidades, puesto que para ello sería necesario que el robot tuviese, entre otras cosas, sensores para poder captar el entorno y obtener información de él y una IA que le permitiese actuar en función de la información obtenida del entorno.

1.2.4. Robótica autónoma en la industria y el espacio

A pesar de que W. G. Walter construyese a Elmer y Elsie en la década de los 40, como ya se ha comentado no fue hasta principios de los 60 cuando se tuvo en cuenta la necesidad de utilizar sensores para interactuar con el entorno y conseguir que los robots fuesen más flexibles gracias a la capacidad de reacción ante cambios en el entorno que les rodeaba. Entre los factores que favorecieron el desarrollo de la robótica autónoma, están especialmente su uso en la industria y en ambientes peligrosos o adversos, como la exploración de otros planetas.

Así pues, en ésta época comenzó a impulsarse el uso de robots industriales, lo que llevó a un mayor estudio en dicha área para mejorar y ofrecer nuevas funcionalidades, como la capacidad de detectar y reaccionar ante cambios en el entorno, aportando más versatilidad a los robots y permitiendo su uso en entornos menos controlados que en los que se usaba inicialmente.

Concretamente, la primera mano mecánica, la MH-1, capaz de *sentir* bloques y apilarlos sin la ayuda de ningún operario fue desarrollada a principios de los 60 por H.A. Ernst. En esta época, también se presentaron otros diseños de manos sensibles, como la de Tomovic y Boni, capaz de detectar el tamaño y peso de un objeto, o el brazo articulado (VESATRAN) de la American Machina y Foundry Company. A finales de los años 60, trabajos como el de McCarthy en el Stanford Artificial Intelligence Laboratory, que

desarrolló un computador con *manos*, *ojos* y *oídos* (manipuladores, cámaras de TV y micrófonos), muestran como los robots de esta época ya presentan una considerable capacidad de detección y reacción ante cambios en el entorno. Sin embargo, el ambiente en el que trabajan estos robots industriales continúa siendo un entorno muy controlado.

Por otro lado, en esta misma época también se estaban invirtiendo grandes esfuerzos en la carrera espacial, que duró desde finales de los 50 hasta mediados de los 70.

En este ámbito la robótica estaba enfocada al uso de robots teledirigidos en tareas en las que era inviable (por coste o peligrosidad) el uso de humanos. Sin embargo, para poder controlar remotamente un robot es necesario tener información del entorno del robot (sensores), siendo lo más usual el uso de la visión. Por otro lado, en los casos en los que el control remoto constante es prácticamente inviable, como en las misiones a Marte en las que los retrasos en las comunicaciones limitan de forma considerable la capacidad de operación remota, es necesario dotar de cierta inteligencia y capacidad de decisión a los robots para que puedan actuar de forma autónoma mientras reciben órdenes del operador.

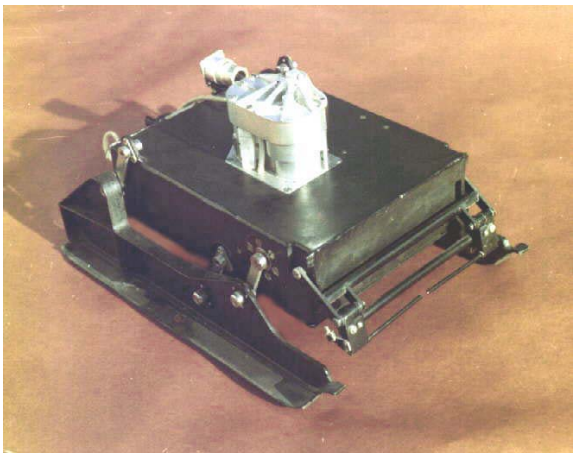

Figura 1.4: Robot autónomo Prop-M Rover de la misión Mars 2.[5]

Así pues, en estas tareas se empezaba a hacer necesario el uso de robots con capacidad de actuación autónoma. Un ejemplo de estos robots se encuentra por ejemplo en la misión Mars 2, a principios de los 70. Esta misión rusa estaba formada por un módulo orbital y uno de descenso, que contenía un robot rover de exploración (Fig. 1.4). El objetivo de esta misión era explorar

[5]Imagen pública extraída de: `http://en.wikipedia.org/wiki/File:Mars_propm_rover.jpg`. Última visita: 07-09-2015

y obtener información de Marte. Debido a que los retrasos en las comunicaciones hacían inviable un control remoto seguro, el robot rover (llamado Prop-M Rover) debía ser capaz de moverse, evitando obstáculos, y obtener muestras de análisis del suelo de forma autónoma en un entorno prácticamente desconocido. Así pues, además de capacidad de captar información del entorno este robot debía tener una mínima capacidad de IA (apenas surgida hacía 30 años) para poder reaccionar ante los posibles e imprevisibles obstáculos del entorno.

1.2.5. Inteligencia Artificial

La IA es un elemento básico en un robot autónomo, ya que determina el *grado de autonomía* de este, entendiendo por grado de autonomía la capacidad del robot para actuar en situaciones diferentes. Si bien es cierto que este elemento no es estrictamente necesario en un robot autónomo, pero si es lógico que cuanto *más desarrollada* sea la IA del robot, mayor capacidad de actuación y reacción tendrá ante cambios en el entorno.

La IA comenzó a desarrollarse a principios de la década de los 40, aproximadamente cuando comenzaron a desarrollarse y extenderse los primeros robots industriales. Los primeros trabajos en este área se deben a unas publicaciones de Alan Turing, las cuales no tuvieron gran repercusión en aquel momento. Poco después de estas publicaciones, Warren McCulloch y Walter Pitts realizaron estudios sobre el funcionamiento de una red neuronal (Fig. 1.5) y demostraron que la máquina de Turing podía ser implementada en una red finita de neuronas. Tras esto, a finales de los años 40 Donald Hebb desarrolló un algoritmo (basado en lo que se conoce como *Regla de Hebb*) que permitía el aprendizaje de estas redes, creándose de esta forma la escuela conexionista, considerada como el origen de lo que se conoce hoy en día como IA. También a mediados de esta época George Pólya presentó los primeros conceptos sobre heurística, que con el tiempo se convertiría en un elemento fundamental en la IA.

Sin embargo, no fue hasta la década de los 50 cuando la IA comenzó a cobrar importancia. Así pues, en esta época Alan Turing planteaba, en un artículo que tuvo una enorme repercusión, la pregunta de si una máquina podía pensar, y establecía una prueba para comprobar si una máquina era capaz de actuar de forma inteligente o no. Fue pocos años después, a mediados de los 50, cuando surge el término IA y se convierte en disciplina en la Conferencia de Computación de Dartmouth. En esta época, Herbert Simon, Allen Newell y J.C. Shaw desarrollaron el primer lenguaje de procesamiento de información, el IPL-11, y el primer programa de IA, el GPS de sus siglas en inglés *General Problem Solver*, capaz de demostrar teoremas matemáticos.

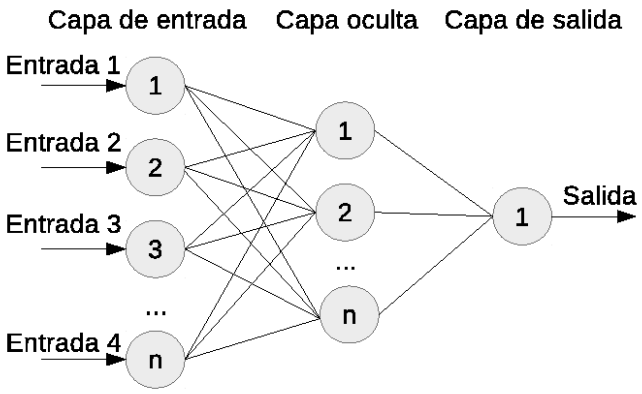

Figura 1.5: Ejemplo de red neuronal.

Si bien la IA estuvo ligada inicialmente a juegos como el ajedrez y las damas, llegando a su punto álgido con el desafío entre Deep Blue y Gary Kaspárov en 1996, pero fue aplicada a otros campos desde casi sus orígenes.

Así pues, a finales de los 50 y comienzos de los 60 Frank Rosemblatt inventó y desarrolló el perceptron, que es una extensión del modelo matemático concebido por McCullock y Pitts. Este perceptrón fue inicialmente aplicado al reconocimiento de patrones visuales. Sin embargo, debido a las limitaciones que presentaba el perceptrón, las redes neuronales cayeron en el olvido hasta el surgimiento del perceptrón multicapa.

Otra de las áreas en las que fue aplicada fue en la comprensión del lenguaje. Como ejemplos de esta aplicación, estarían el programa llamado SAD-SAM (Sentence Appraiser and Diagrammer, and Semantic Analyzing Machine), desarrollado por Robert K. Lindsay a comienzos de los 60, que era capaz de extraer conclusiones a partir de oraciones, el sistema SIR (Semantic Information Retrieval), desarrollado por Bertrand Raphael a mediados de los 60, o el programa SCHRDLU, que permitía interrogar y dar órdenes a un robot que se movía dentro de un mundo virtual de bloques, desarrollado por Terry Winograd a finales de los 60.

Otra de las áreas a las que fue aplicada la IA fue a la medicina. Así pues, el sistema Dendral, desarrollado a mediados de los 60 por Edward Feigenbaum en la Universidad de Stanford, tenía como objetivo el reconocimiento de moléculas orgánicas desconocidas. El desarrollo de este sistema llevó 10 años y se considera el primer sistema experto.

Como resultado de todos estos avances, entre finales de los 60 y principios de los 70 SRI International desarrolló el primer robot móvil de propósito general, Shakey, que era capaz de razonar sobre sus propias acciones, es decir,

que en vez de realizar secuencias de acciones establecidas ante una orden determinada, el robot era capaz de analizar el comando recibido y determinar, por sí mismo, que secuencia de acciones realizar. Este proyecto fue importante por varios factores. Por un lado aunaba distintas áreas de investigación, como la robótica, visión por ordenador, procesamiento de lenguaje natural e IA (como de hecho ocurre en casi todos los robots actuales). Por otro lado, fue el primer proyecto que unía el razonamiento lógico con la acción física.

Sin embargo, a finales de los años 60 se publicó el libro "Perceptrons: an introduction to computational geometry" de Marvin Minsky y Seymour Papert en el que mostraban las limitaciones del perceptron. Esta publicación causó un abandono (temporal) de las redes neuronales a favor de los sistemas expertos, debido a los buenos resultados obtenidos con el sistema experto Dendral. Así, a mediados de los 70, Ted Shortliffe desarrolló en la Universidad de Stanford el programa MYCIN, una continuación del programa Dendral. Este sistema tenía como objetivo el diagnóstico de enfermedades infecciosas en la sangre, consiguiendo una tasa de aciertos de aproximadamente el 65 %, lo cual mejoraba las estadísticas de la mayoría de los médicos no especializados en el diagnóstico de infecciones bacterianas (frente a los especializados cuya tasa era del 80 %). En esta época también comenzó a surgir la Inteligencia Artificial Distribuida como un campo de la IA y, concretamente, lo que hoy en día se conocen como Sistemas Multi-Agente (MAS) (*Multi-Agent System* en inglés), cuyo objetivo se centra en la interacción entre los agentes que componen el sistema. Así pues, este enfoque se basa en unir las capacidades de resolución de los distintos agentes de forma que la interacción de todos ellos permita la resolución de problemas más complejos, estando la inteligencia del sistema distribuida entre los distintos agentes y la interacción entre ellos.

Ya en la década de los 80 se empezaron a comercializar las primeras aplicaciones basadas en sistemas expertos (obviamente para el sector profesional). En esta época también se desarrollaron los primeros trabajos en el área de la conducción autónoma. Concretamente, fue el equipo de Ernst Dickmanns de la Universidad de Bundeswehr quien construyó el primer coche robot capaz de conducir por sí mismo (aunque en carreteras vacías por seguridad). En esta época comienzó también el resurgimiento de las de las redes neuronales gracias al algoritmo de retropropagación (descrito por Paul J. Werbos).

Durante la década de los 90 hay grandes avances en todas las áreas relacionadas con la IA, como en aprendizaje máquina (*machine learning* en inglés), razonamiento basado en casos, planificación, visión, etc. Un ejemplo de estos avances es el robot Polly, desarrollado por Ian Horswill, el primer robot basado en comportamientos capaz de navegar usando visión y operar

a una velocidad considerable (1 m/s). Otros hitos importantes fueron los avances en conducción autónoma, realizándose experimentos en carreteras con tráfico, el éxito de la máquina Deep Blue frente a Gary Kasparov o la primera competición de la RoboCup, con 40 equipos de robots y más de 5000 espectadores.

A partir de esta fecha, y hasta la actualidad, continúan los avances en todas las áreas de la IA, siendo uno de los puntos remarcables la liberación de los sistemas expertos al público en general, surgiendo por ejemplo los juguetes Furby y AIBO a finales de los años 90 o la aspiradora autónoma iRobot's Roomba a principios del 2000. Ejemplos de sistemas expertos para todos los públicos más recientes serían la aplicación de Google now de Google, Siri de Apple o Cortana de Microsoft, que reconocen lenguaje natural para responder a preguntas y hacer recomendaciones a los usuarios.

1.2.6. Sistemas Multi-Robot

Una vez la robótica autónoma alcanzó una madurez considerable se fueron planteando retos más y más complejos, de forma que se encontraron problemas que eran difíciles de resolver por un único robot, estudiándose la posibilidad de que varios robots colaborasen entre sí para realizar una misma tarea que un único robot no podría hacer o tardaría más que varios robots colaborando en paralelo.

Así fue como surgió, a finales de los 80, los MRS (*Multi-Robot System* en inglés), que no es más que un sistema formado por múltiples robots autónomos individuales que interactúan entre sí para realizar una tarea concreta. De esta forma, los MRS se pueden considerar como una extensión de la robótica autónoma *individual*, posibilitando la resolución de problemas más complejos aunque, a su vez, presentando nuevos retos, ya que los robots que forman el sistema deben ser capaces de coordinarse entre sí de forma correcta. También hay que tener en cuenta que los MRS presentan una serie de ventajas frente a los sistemas monolíticos, ya que son más tolerantes a fallos, más flexibles y, en general, más eficientes al realizar las tareas pues tienen la capacidad de estar en múltiples lugares al mismo tiempo, dividiendo las tareas complejas en tareas más simples.

Los trabajos en coordinación e interacción de múltiples agentes inteligentes comenzaron en la década de los 70, pero no fue hasta 10 años después cuando comenzaron a aplicarse estos trabajos en el área de la robótica, concretamente en robótica cooperativa. Así pues, el primer trabajo fue el proyecto CEBOT (Cellular Robotics), desarrollado por T. Fukuda, Y. Kawauchi y H. Asama, que consistía en la construcción de sistemas complejos basándose en pequeños robots que se comunicaban entre ellos e interactuaban para

realizar una tarea concreta.

Por esta misma época (finales de los 80 y principios de los 90) surgieron otros trabajos basados en la misma idea, como el proyecto ACTRESS (ACTor-based Robot and Equipments Synthetic System), un sistema robótico distribuido formado por múltiples robots que pueden comunicarse entre ellos para realizar las tareas encomendadas, o el proyecto GOFER, con un enfoque similar al proyecto ACTREES pero incluyendo la posibilidad de comunicación entre robot y el usuario.

Pocos años después, a mediados de los 90, surgió el proyecto ALLIANCE, cuyo objetivo era conseguir una cooperación tolerante a fallos entre robots heterogéneos, pudiendo reaccionar a cambios en el entorno, errores mecánicos o eliminación de algunos robots del equipo, por ejemplo.

A finales de los 90 surgen los proyectos M+, un sistema distribuido para la cooperación de múltiples robots que incluye planificación y reparto de tareas así como mecanismos de cooperación basado en negociación, y Murdoch, un sistema de asignación de tareas para agentes heterogéneos mediante un sistema de negociación que permite escoger el agente más adecuado para cada tarea mediante un esquema suscriptor/publicador en el que cada agente se puntúa según su adecuación a una tarea.

Figura 1.6: Equipo de robots en la Robocup 2004.[6]

En esta época también tuvo lugar la primera RoboCup (Fig. 1.6), un proyecto internacional con distintas categorías, cada una con un enfoque, pero con el objetivo común de impulsar y promover la investigación en robótica

[6]Imagen pública extraída de: `http://commons.wikimedia.org/wiki/File:Robocup.legged.leauge.2004.nk.jpg`. Última visita: 07-09-2015

autónoma, la IA y los MRS. Las distintas categorías de la RoboCup son RoboCupSoccer (una competición de fútbol con robots autónomos impulsando, entre otros, el estudio de los MRS y su coordinación), la RoboCupRescue (cuyo objetivo es crear robots que ayuden en tareas de búsqueda y rescate de personas), RoboCupJunior (cuyo objetivo es acercar la robótica a los más jóvenes) y la RoboCupHome (cuyo objetivo es la realización de robots asistenciales que ayuden en las tareas diarias). La RoboCup también sirve como un gran laboratorio para la validación de modelos robóticos (individuales y colectivos) tanto a nivel de simulación como a nivel físico. De esta forma, muchos de los trabajos de investigación desarrollados dentro del área de la robótica están contenidos en parte o totalmente dentro de las distintas áreas que componen la RoboCup.

Por último, a principios del 2000, se hayan otros trabajos como el proyecto ASyMTRe (Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration), un sistema que automatiza la solución de tareas en equipos de robots determinando como y quien realizará una tarea concreta, permitiendo afrontar las tareas con equipos de robots distintos de forma que unos robots apoyen a otros en caso de necesidad, o el proyecto Centibots, un ambicioso proyecto de la Universidad de Stanford con el apoyo de DARPA que tenía como objetivo coordinar un gran número de robots (100 robots, repartidos entre 80 Amigobots y 20 Pioneer 2 AT) para realizar la tarea de mapear una zona o área de interés.

Todos estos trabajos e iniciativas, junto con muchos otros, han dando paso, poco a poco, al desarrollo de los MRS, siendo hoy un área activa de investigación en la que todavía quedan muchos problemas que resolver. Entre estos problemas, están la complejidad que pueden alcanzar estos sistemas, debido a las interacciones entre los robots y a todos los factores a tener en cuenta para no repetir tareas, evitar obstaculizarse mutuamente y, en definitiva, coordinarse correctamente para obtener los beneficios de un sistema distribuido. Otros problemas más específicos son, por ejemplo, la dependencia con el componente central en sistemas coordinados centralizados, que generalmente provoca un cuello de botella y ralentiza la coordinación entre los robots, o las comunicaciones en sistemas con una gran cantidad de robots o en entornos donde éstas están limitadas, lo que dificulta enormemente las posibilidades de coordinación y la interacción en estos sistemas.

1.3. Objetivos de la tesis

La investigación en el área de los MAS y MRS comenzó hace ya bastantes años (en la década de los 70), sin embargo no ha sido hasta hace pocos años,

cuando la robótica y la tecnología han alcanzado una suficiente madurez, que se ha podido estudiar más a fondo y de una forma más realista y práctica en estos ámbitos.

Los MAS están englobados directamente dentro del área de la Inteligencia Artificial Distribuida, un campo de la IA, y están formados por un conjunto de entidades o agentes autónomos que ayudan a resolver un determinado problema [93]. Por lo tanto, desde el punto de vista de la Inteligencia Artificial Distribuida un agente puede ser definido como un elemento, que forma parte de un conjunto más grande, que ayuda a resolver un problema que no podría resolver de forma individual [93]. Como se puede observar, la definición de agente es bastante vaga y genérica. De hecho, no existe una definición formal y precisa de lo que es un agente autónomo [123], ya que existe una gran cantidad de agentes distintos y es difícil dar una definición exacta. Sin embargo, existen algunas definiciones de agente algo más concretas o, al menos, más precisas que la dada anteriormente. Así pues, se puede considerar que un agente es cualquier entidad en un entorno que es capaz de percibir dicho entorno y actuar de forma autónoma interactuando con él [123].

Teniendo en cuenta esta definición, un agente podría ser un programa software, un robot o incluso una persona [141]. De hecho, un MRS puede verse como un caso concreto de MAS [52,152] en el que cada robot puede ser considerado como un agente con la habilidad de solucionar unas tareas locales y coordinarse con sus compañeros [152]. Así pues, y aunque hay diferencias conceptuales entre ellos, pero sí que existe una relación entre los MAS y los MRS, pues ambos requieren una coordinación entre los agentes o robots para que el conjunto del sistema pueda realizar tareas que, individualmente, no son capaces de realizar los elementos que lo componen.

El interés en los MRS y los MAS robóticos estriba en que los sistemas formados por múltiples robots ofrecen una serie de ventajas frente a los sistemas robóticos individuales [9], como son:

- mayor robustez, ya que al tratarse de un sistema formado por distintos robots en caso de que un robot fallase los demás robots podrían seguir realizando su labor, si bien la tarea global tardaría más en realizarse.

- mayor adaptabilidad, ya que al repartirse las tareas es más fácil adaptarse a cambios en el entorno, pues cada robot sólo tendrá que adaptarse a los factores que le afecten directamente según su tarea concreta, mientras que si se tratase de un único robot tendría que adaptarse a todos los factores que afronta cada uno de los robots por separado. Además, si los distintos robots que forman el sistema están especializados en una tarea concreta, es más fácil que se adapten a un problema concreto que un robot adaptándose a todos los problemas específicos que surjan.

- mayor flexibilidad, en parte relacionado con las dos anteriores, pues en caso de fallo o cambios externos, el sistema es potencialmente más flexible pudiendo reasignar las tareas, o bien unificar todos los robots en una única tarea para actuar como un único robot.

Es por esto que los MRS se usan en aplicaciones complejas y que requieren robustez o abarcar un gran área rápidamente, como por ejemplo el mapeo de terrenos, ambientes hostiles o peligrosos [42], tareas militares [63, 95], operaciones de rescate [87], etc.

Por otro lado, no todo son ventajas, ya que el coste de estos beneficios es el correcto diseño de un sistema que, en general, es más complejo que el diseño de un robot autónomo individual, pues es necesaria una correcta coordinación y cooperación entre los distintos elementos del sistema para su correcto funcionamiento. Sin embargo, el problema de la coordinación de los distintos robots de un MRS deriva en otros problemas o decisiones más específicas como son la arquitectura, la heterogeneidad del sistema, la comunicación entre los miembros del equipo, el reparto de tareas entre los robots o el aprendizaje entre otros. Por lo tanto, el desarrollo de un MRS supone una serie de decisiones y especificaciones que afectarán al desarrollo del sistema final.

Así pues, en la presente tesis, el objetivo es el desarrollo de un sistema de coordinación multi-agente basado en el aprendizaje de comportamientos.

La coordinación entre los agentes es implícita pues en un MAS una de las características de los agentes es que son autónomos y actúan por sí mismos, sin haber un elemento que centralice las acciones de los distintos agentes. Sin embargo, eso ni impide que los agentes puedan conocer la existencia de los demás agentes ni se transmitan entre ellos de forma explícita información sobre el entorno y su propio estado.

La implementación de los comportamientos se realiza mediante aprendizaje en vez de mediante algoritmos analíticos porque el aprendizaje puede ofrecer una mayor libertad y facilidad a la hora desarrollar comportamientos que no se ajusten fácilmente a una ecuación analítica. Concretamente, dentro de las distintas técnicas de aprendizaje se usa Aprendizaje por Demostración (LfD) (*Learning from Demonstration* en inglés) [8, 130] porque es una técnica que asocia una acción a un estado y consiste en entrenar un comportamiento específico a partir de ejemplos provistos por un usuario supervisor, lo cual permite un entrenamiento sencillo, pues tan sólo es necesario controlar el robot para que realice el comportamiento que quiere que se aprenda, y además, permite absorber, implícitamente durante el aprendizaje, distintos factores difíciles de modelar, como imperfecciones en los robots (derivas en el movimiento debido a que los motores no están correctamente alineados), errores

en los sensores, ruido, etc. Así pues, esta técnica de aprendizaje solventa uno de los problemas que supone trabajar con robots: su modelado.

Dentro del aprendizaje por demostración se ha seleccionado la técnica de aprendizaje Razonamiento Basado en Casos (CBR) en vez de otras como aprendizaje basado en árboles de decisión o aprendizaje bayesiano, porque el CBR resulta muy intuitivo y fácil de comprender y, sobre todo, porque encaja a la perfección con el concepto de comportamiento reactivo, cuyo principio de funcionamiento consiste en acoplar las entradas de los sensores a los actuadores o, expresado de otra forma, asociar una situación a una solución, que es lo que hace el CBR.

Por último, este sistema ha sido probado con robots AIBO ERS-7 de Sony, conectados mediante WiFi y usando localización visual mediante marcas artificiales de colores en un entorno similar al de la competición robótica de fútbol Robocup. Para ello se han entrenado comportamientos simples, como correr por la banda, ir a portería o evitar a un oponente, y se han combinado mediante una capa superior que los activa y desactiva según la situación lo requiera.

1.4. Estructura de la tesis

La presente tesis está dividida en los siguientes capítulos:

- Capitulo 2: Arquitectura de control y sistema de navegación. En este capítulo se presenta un estado del arte de las arquitecturas de control y sistemas de navegación más usuales. Tras la presentación de cada una de las opciones disponibles, se seleccionará las más adecuada al sistema que se desea desarrollar. Concretamente, la arquitectura de control usada será una arquitectura híbrida, formada por una capa de bajo nivel basada en comportamientos aprendidos y una de alto nivel que permite la activación o desactivación de estos. Así pues, en los capítulos 3 y 5 se presentará la implementación la capa reactiva para comportamientos individuales y coordinados respectivamente, a la que se añadirá finalmente una capa de más alto nivel (capítulo 6) que los activa y desactiva para permitir obtener comportamientos emergentes más complejos.

- Capitulo 3: Aprendizaje de comportamientos básicos. En este capítulo se presenta la implementación de comportamientos aprendidos mediante CBR (*Case-Based Reasoning* en inglés). El CBR es una técnica para aprendizaje y adaptación que ayuda a resolver problemas actuales mediante la recuperación y adaptación de experiencias pasadas, y se ha

escogido porque encaja a la perfección con el enfoque de navegación basada en comportamientos. Como técnica de aprendizaje se usará LfD. Por último, en este capitulo se aplica el enfoque propuesto para la creación de comportamientos aprendidos a una prueba de concepto que consiste en el aprendizaje de unos comportamientos simples aplicados a un único robot AIBO. El objetivo en este capítulo es comprobar la idoneidad del aprendizaje basado en CBR para el aprendizaje de comportamientos que, en posteriores capítulos, permitirán la coordinación de varios robots.

- Capitulo 4: Sistema de localización. El objetivo final de esta tesis es realizar un sistema coordinado multiagente, para lo cual es necesario tener una referencia de las posiciones de los distintos agentes del sistema, bien de forma relativa entre ellos o de forma absoluta. Así pues, en este capítulo se realiza una breve presentación de las distintas técnicas de localización disponibles. Dado que el enfoque inicial del sistema ha sido un enfoque reactivo, se tratará de usar una técnica de localización que sea lo más reactiva posible, teniendo en cuenta que el hecho de usar una localización ya implica que el sistema no será totalmente reactivo. Tras realizar distintas pruebas, en este capítulo se concluye que las opciones más viables para el sistema de localización son el uso de localización basada en marcas fiduciarias (primera opción seleccionada) o localización basada en marcas visuales artificiales corrigiendo el posicionamiento mediante filtrado (opción más estable pero *menos reactiva* que la anterior).

- Capitulo 5: Coordinación reactiva basada en aprendizaje. Tras presentar el algoritmo de navegación (aprendizaje por demostración basado en CBR) en el capítulo 3 y la técnica de localización a usar (localización basada en marcas fiduciarias) en el capítulo 4, se propone la aplicación de este mismo enfoque para la creación de comportamientos coordinados mediante aprendizaje. Para ello, en este capítulo se muestran los resultados de una prueba de concepto basada en este enfoque que permite la coordinación de dos robots mediante el uso de CBR. En estas pruebas preliminares el sistema se basará únicamente en una capa reactiva y la localización se basará en marcas ARToolkit. Tras los experimentos, en este capítulo se comprueba la correcta coordinación de los robots mediante comportamientos aprendidos, aunque también se encuentran algunos problemas e inconvenientes que se solventarán o minimizarán en el capítulo 6.

- Capitulo 6: Coordinación híbrida basada en aprendizaje. En este últi-

mo capítulo se presentan los resultados finales de coordinación implícita de dos robots AIBO basada en comportamientos aprendidos mediante CBR. En estas pruebas se añade al sistema una capa de más alto nivel que permite la activación y desactivación de los comportamientos aprendidos según la situación, permitiendo una coordinación más compleja que en el capítulo anterior así como la realización de trayectorias más elaboradas. En este capítulo se solucionan algunos problemas detectados en los capítulos anteriores relacionados con la localización, el entrenamiento de los comportamientos o el entorno de trabajo usado. Por último, en los experimentos realizados en este capítulo se consiguen resultados más interesantes que en los realizados en los capítulos anteriores gracias a la división de los comportamientos, permitiendo una mejor organización y comprensión del sistema global.

Capítulo 2

Arquitectura de control y sistema de navegación

2.1. Introducción

Los MRS suponen una serie de ventajas frente a los sistemas robóticos individuales. Sin embargo, en los MRS hay que tener en cuenta, en general, los mismos problemas que en los robots individuales más una serie de problemas y decisiones propios de los sistemas con múltiples robots y que no son necesarios tener en cuenta cuando se trabaja con robots individuales. Así pues, antes de afrontar los problemas propios de los MRS, y dado que éstos también incluyen problemas asociados a los sistemas basados en robots individuales, se planteará en primer lugar el problema de la arquitectura de control y la navegación individual de cada robot, seleccionando y desarrollando en los capítulos siguientes los demás elementos clave del MRS final.

Como se ha comentado en el capítulo 1, el objetivo final de este trabajo es la realización de un sistema coordinado basado en el aprendizaje de comportamientos. Sin embargo, los comportamientos pueden ser implementados mediante aprendizaje o mediante algoritmos analíticos, que es el enfoque más clásico y usual. Igualmente, hay que determinar como se interconectan y combinan estos comportamientos para permitir la navegación del robot, lo que se correspondería con la arquitectura de control del sistema de navegación.

Así pues, en primer lugar se presenta la plataforma de desarrollo sobre la cual se realizarán todas las pruebas del presente trabajo (sección 2.2), que es el robot AIBO ERS-7 de Sony. Tras esto, se presentarán las distintas arquitecturas de control disponibles para navegación, así como las decisiones que han llevado a escoger una arquitectura híbrida basada en comportamientos reactivos (sección 2.3). Posteriormente, se presentarán las distintas opciones

para implementar el algoritmo de navegación (sección 2.4), indicando porqué se selecciona la navegación basada en aprendizaje así como la técnica de IA que se usará para realizar el aprendizaje de los comportamientos. Por último, se presentarán las conclusiones (sección 2.5) del capítulo.

2.2. Plataforma de desarrollo

La navegación se puede definir como el proceso de trasladarse desde una posición origen a una posición destino de una forma segura. Por lo tanto, la navegación robótica es un problema básico en robots autónomos y, de hecho, desde los orígenes de la robótica móvil ésta ha sido un área muy activa de investigación y desarrollo [4,32], ya que cada aplicación y entorno de trabajo supone problemas de navegación distintos.

Para que un robot sea capaz de moverse por un entorno es necesario que sea capaz de desplazarse, tenga un destino que alcanzar y pueda percibir el entorno que le rodea para poder determinar como alcanzar dicho destino.

Respecto a la capacidad de desplazamiento, los medios de desplazamiento más usuales en robots son el uso de ruedas o de extremidades. Cada uno tiene sus ventajas y desventajas, estando asociados normalmente a unas circunstancias y situaciones concretas. Así pues, las plataformas con ruedas tienden a ser muy estables y fáciles de usar y controlar y permiten una estimación de cuanto se ha desplazado el robot respecto a una posición inicial con una cierta precisión, aunque ésta sea baja. Sin embargo, en terrenos con desniveles o irregularidades, si bien son igualmente válidos, pero pueden no ser tan adecuados, como otros robots basados por ejemplo en extremidades. Por su lado, los robots basados en extremidades tienen como ventaja la posibilidad de saltar o pasar por encima de determinados obstáculos más fácilmente que los robots con ruedas. Por contra, su control resulta extremadamente complejo, en comparación con las plataformas con ruedas. De hecho, las plataformas basadas en extremidades tienden a ser bastante inestables y el cálculo de la posición basado en la odometría resulta muy poco fiable pues no es tan fácil estimar cuanto espacio se ha avanzado en este tipo de plataformas.

Por otro lado, otro factor importante a tener en cuenta es cómo se capta la información del entorno. Esto suele realizarse mediante los sensores disponibles por la plataforma de desarrollo, la cual puede ser cerrada o abierta, permitiendo en este caso añadir más sensores. Tradicionalmente, uno de los sensores más típicos y usuales era el anillo de sensores de distancia (como el anillo de sonares) ya que son simples, baratos, fáciles de procesar y aportan información espacial básica sobre el entorno. Sin embargo, este sensor sólo aporta información espacial, por lo que tareas como el reconocimiento

de objetos pueden resultar complejas en ocasiones. También hay que tener en cuenta que este tipo de sensores es habitualmente empleado en robots móviles con ruedas, en los cuales es fácil obtener la posición de los obstáculos respecto al plano de tierra del robot. Sin embargo, en robots con extremidades, como el robot AIBO, el uso de un anillo de sensores de distancia no es la mejor solución debido al constante movimiento del robot al desplazarse, lo que ocasiona que el procesamiento de la información obtenida del anillo de sensores sea más complejo que en el caso de robots con ruedas. Aparte de estos problemas, el uso del sonar es especialmente problemático cuando la precisión, el alcance o la densidad de los sensores es baja (cada sensor abarca un gran ángulo del entorno y la distancia estimada no es fiable), lo que provoca que la información obtenida del entorno sea demasiado ambigua como para resultar útil. En estos casos, es usual el uso de sensores complementarios o alternativos.

Por lo tanto, para solventar los problemas que puede presentar el uso del sonar se recurre al uso de otros sensores, como puede ser una vídeo cámara. El problema de este tipo de sensores es que, en general, no es fácil extraer la información necesaria del entorno de las imágenes obtenidas debido, entre otros factores, a la dependencia de éstas con los cambios en las condiciones del entorno, como las oclusiones, los cambios de luminosidad, el ruido, etc. [32] presenta un estudio sobre la navegación de robots basada en visión en el que se indican las ventajas y desventajas de la mayoría de estrategias usadas, así como la gran influencia que tienen las condiciones del entorno en la captura de los datos y, por tanto, en la extracción de información cualitativa del entorno. Sin embargo, en lo que respecta a la navegación basada en visión hay que distinguir entre robots con ruedas y robots con extremidades. Esto es así porque, en los robots con ruedas, la alteración y los saltos en el flujo de imágenes es menor que en los robots con extremidades, permitiendo el uso de técnicas para procesamiento basados en la información de imágenes anteriores. Sin embargo, en el caso de los robots con extremidades, debido al balanceo de la cámara durante el desplazamiento del robot, no es posible, en general, realizar un procesamiento continuo de imágenes mediante técnicas diferenciales convencionales debido al drástico cambio de información en las imágenes usadas. Por tanto, en este tipo de robots es común el uso de técnicas de procesamiento de imágenes cualitativas, las cuáles permiten la búsqueda de características fácilmente detectables en cada fotograma independientemente, sin necesidad de disponer de un flujo de imágenes consecutivas y sin discontinuidades entre ellos. Así pues, y a pesar de los problemas comentados, los sistemas basados en robots con extremidades están ligados en la mayoría de los casos al uso de la visión para su correcto funcionamien-

to [78, 79, 108, 109, 129].

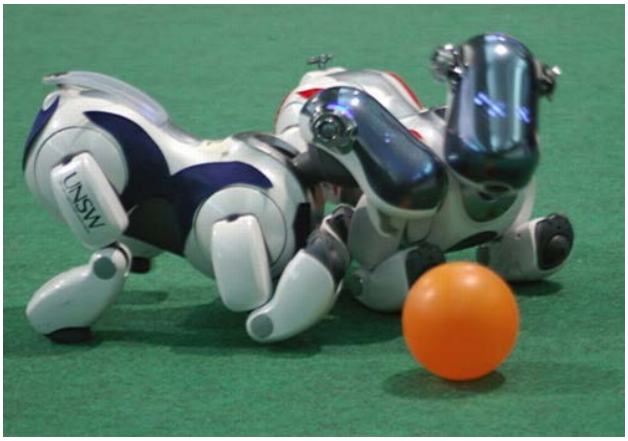

Figura 2.1: Dos AIBOs ERS-7 compitiendo en la Robocup.[1]

En el caso concreto de la presente tesis, las pruebas y experimentos serán realizados con robots AIBO ERS-7 de Sony (Fig. 2.1), que son robots con extremidades cuyo principal sensor es una cámara direccional integrada en la cabeza. Este robot también dispone de un sensor de infrarrojos en el pecho y otro en la cabeza junto a la cámara, aunque su utilidad resulta muy limitada. Así pues, la plataforma a usar, y a tener en cuenta durante el presente trabajo, será una plataforma con extremidades en la que la percepción del entorno se basará en visión.

2.3. Arquitectura de control

Como se ha comentado, para que un robot sea capaz de moverse por un entorno y alcanzar su destino es necesario, dado por hecho su capacidad de desplazamiento y percepción del entorno, determinar la secuencia de acciones que le permitirá llegar al destino y realizarlas en respuesta a la información obtenida del entorno [147]. A la forma en que estas acciones se organizan e integran para obtener un resultado concreto es a lo que se llama arquitectura de control.

A lo largo de los años han surgido una gran variedad de arquitecturas de control, cada una con sus ventajas y desventajas. Todas las arquitecturas de control buscan, entre otras cosas, la creación de un sistema robusto, flexible y fiable teniendo en cuenta un entorno de trabajo concreto. Sin embargo, es

[1]Imagen pública extraída de: `http://en.wikipedia.org/wiki/File:RUNSWift_AIBOS.jpg`. Última visita: 20-09-2015

justo este último factor, el entorno de trabajo que se tiene en cuenta, el que hace una arquitectura sea más adecuada que otra según la situación a tener en cuenta. De forma general, las arquitecturas de control se pueden clasificar en tres grandes grupos [88]:

- Navegación deliberada o centralizada. La navegación deliberada es el esquema de navegación más clásico dentro de la robótica móvil [51, 147]. La navegación deliberada se basa en el esquema tradicional de Percibir-Planificar-Actuar (SPA) (Fig. 2.2.b) (del inglés *Sense-Plan-Act*). Este paradigma está basado en la idea de planificación de las acciones y movimientos en base a un modelado del mundo con el que el robot debe interactuar. Los esquemas deliberativos buscan, mediante el modelado del entorno, la solución o camino óptimo para alcanzar el destino, aplicándose posteriormente dicha planificación de acciones. Así pues, este esquema se basa en una estructura jerárquica de arriba hacia abajo, pues las acciones se generan en las capas altas y se transmiten hacia las capas bajas de la arquitectura. El principal problema de este esquema es que el cálculo de la solución óptima puede ser costoso computacionalmente y temporalmente, por lo que este esquema está especialmente enfocado a entornos estáticos y poco dinámicos [138] donde, una vez encontrada una solución, esta será válida durante un gran período de tiempo. Además de este problema, hay que tener en cuenta que en entornos dinámicos o con incertidumbre es más complejo realizar y actualizar un modelo válido del entorno [135], modelo que es necesario para el correcto funcionamiento de un sistema SPA. De hecho, cuanto más fiable y correcto sea el modelo mejores resultados dará el sistema. Por otro lado, para disponer de un entorno válido del entorno es necesario crear dicho modelo, lo cual puede ser también un problema añadido a tener en cuenta.

  La principal ventaja de estos esquemas es que, si la hay, suelen aseguran una ruta viable y óptima para alcanzar el destino, ya que al tener un modelo del entorno, si éste es correcto y está actualizado, se puede confirmar si se alcanzará o no el destino.

- Navegación reactiva o basada en comportamientos. El esquema de navegación reactiva surgió como alternativa al paradigma SPA para solventar los problemas que presenta esta arquitectura en entornos dinámicos o desconocidos. Este esquema de navegación fue propuesto por Brooks [25] y consiste en acoplar o relacionar la información sensorial que obtiene el robot en un instante determinado con los comandos de movimiento que éste debe realizar (Fig. 2.2.a). Biológicamente este

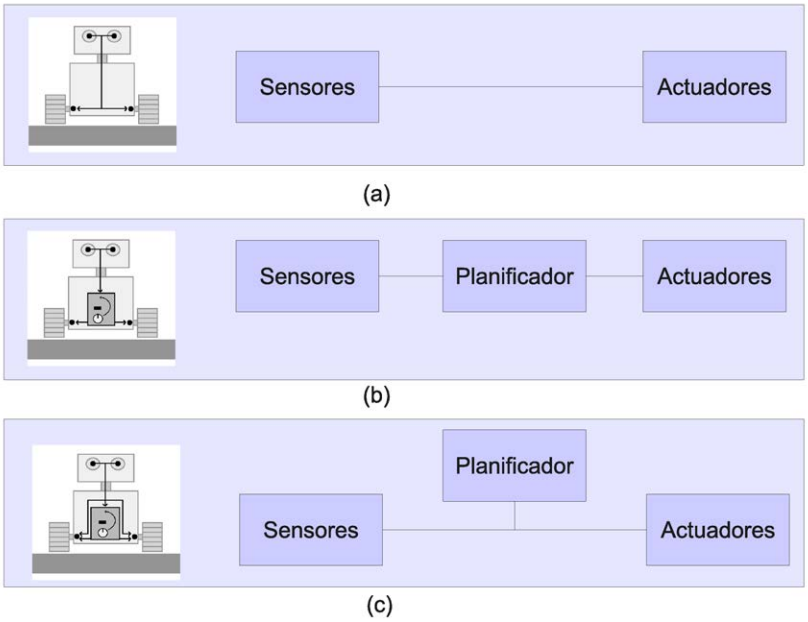

Figura 2.2: Esquemas de navegación reactivo (a), deliberado (b) e híbrido (c).

esquema está inspirado en la idea de imitar los actos reflejos o reacciones instintivas, que en esta arquitectura se suelen llamar conductas o comportamientos. Esta arquitectura ofrece una forma de determinar las distintas acciones a realizar mediante la combinación de conductas o comportamientos activados según la información recibida de los sensores, basándose en la idea de que la unión de comportamientos o conductas de bajo nivel da como resultado un comportamiento o conducta más complejo, conocido como comportamiento emergente. Así pues, en un esquema de navegación reactiva pura únicamente se tiene en cuenta la información que se obtiene en el instante actual, y no información obtenida anteriormente. Este esquema, por lo tanto, sigue un enfoque desde abajo hacia arriba, en el sentido de que las acciones surgen de las capas de bajo nivel, generando como resultado una acción más compleja y de más alto nivel que las originales.

Dado el esquema de funcionamiento de un esquema reactivo, no es necesario un modelado del entorno, ya que la información recibida directamente de los sensores se acopla a los actuadores del robot mediante una función de transferencia concreta, que es lo que se conoce como comportamiento. Por lo tanto, en estos esquemas no es posible esta-

blecer el posicionamiento del robot respecto a un modelo del entorno, ya que no hay. Sin embargo, y debido a su simplicidad, estos esquemas son rápidos y robustos frente a errores en los sensores y ruido [138]. Además, y por la propia naturaleza de este esquema, se ajusta muy bien a ambientes altamente dinámicos o cambiantes [136]. Por otro lado, el comportamiento emergente resultante puede ser impredecible, no necesariamente eficiente [135] y además no se asegura que se pueda alcanzar el objetivo a pesar de que exista alguna forma de llegar, pues este esquema puede caer en mínimos locales al hacer uso únicamente de la información actual [138]. Otro problema a tener en cuenta en este esquema es que no es fácil establecer la coordinación entre los distintos comportamientos de forma adecuada [147]. Dos de las arquitecturas más conocidas dentro de los esquemas reactivos son la arquitectura Subsunción y el esquema de motor [88], estando el primero enfocado a la competición entre comportamientos y el segundo a la cooperación entre ellos.

- Navegación híbrida o reactiva-deliberada. Esta arquitectura de control surgió posteriormente a las arquitecturas deliberativas y reactivas para solucionar sus debilidades pero aprovechando las ventajas de ambas [147]. Así pues, este esquema obtiene la ventaja de la planificación y optimización de los esquemas deliberados y la capacidad de respuesta ante entornos dinámicos o desconocidos de los esquemas reactivos. Los esquemas híbridos [88, 144] están normalmente formados por una capa de bajo nivel reactiva, una capa de alto nivel deliberativa y una de control entre ambos (Fig. 2.2.c). De esta forma, la capa de bajo nivel reactiva permite una rápida respuesta ante cambios en el entorno mientras que la capa de alto nivel deliberativa ofrece un posicionamiento respecto al entorno o a un modelo de éste y una solución óptima para alcanzar el destino [147].

  Aunque existen una gran variedad de arquitecturas híbridas, una de las más conocidas y usadas es la arquitectura en tres capas o arquitectura 3T (del inglés *3 Tier*) [135], formada por las 3 capas ya comentadas (deliberada, reactiva y de control), y de la cual derivan muchas de las arquitecturas de control actuales.

Como es lógico, cuando el entorno de trabajo es un entorno dinámico o no totalmente conocido, las arquitecturas de control más adecuadas son las arquitecturas reactivas e híbridas, pues son capaces de adaptarse a estos entornos respondiendo con suficiente velocidad. Sin embargo, y dado que las arquitecturas híbridas ofrecen unas mejores prestaciones en general que las

arquitecturas reactivas (Tabla 2.1), lo más usual es el uso de arquitecturas híbridas [88], siendo de hecho usada en una gran variedad de robots y entornos distintos [18, 54, 56, 98, 116, 135]. Por lo tanto, la arquitectura de navegación usada en el presente trabajo será una arquitectura híbrida.

Características	Deliberativo	Reactivo	Híbrido
Flexibilidad del sistema	Muy malo	Muy bueno	Muy bueno
Tiempo de respuesta	Muy malo	Muy bueno	Bueno
Solución óptima	Muy bueno	Muy malo	Bueno
Robustez del sistema	Regular	Bueno	Muy bueno
Capacidad de planificación	Muy bueno	Regular	Bueno

Tabla 2.1: Características más destacadas de las arquitecturas de control.

Concretamente, la arquitectura del sistema se comenzará por la definición e implementación de la capa reactiva, añadiendo capas de nivel superior según se vayan necesitando para la implementación del sistema. Así pues, y aunque la arquitectura del sistema será una arquitectura híbrida, en el presente capítulo y en los capítulos 3 y 5 se tratará sólo de la implementación la capa reactiva para comportamientos individuales y coordinados respectivamente.

Como ya se ha comentado, la idea principal de *los comportamientos reactivos* es la de usar comportamientos simples (que acoplan los sensores con los actuadores) de forma que de la interacción que tiene lugar entre los distintos comportamientos individuales surja un comportamiento más complejo llamado comportamiento emergente [9]. En los próximos apartados se desarrollará la implementación de estos comportamientos.

2.4. Sistema de navegación

Una vez determinada la arquitectura de control, es necesario determinar como se implementará el sistema de navegación. Como ya se ha comentado, el sistema de navegación será inicialmente reactivo, añadiéndose capas de nivel superior según sea necesario. Así pues, actualmente sólo se tratará la capa de navegación a nivel reactivo.

La navegación robótica es un problema que ha sido estudiado desde los orígenes de la robótica autónoma, existiendo una gran variedad de opciones y enfoques distintos. Sin embargo, todos ellos se pueden englobar en dos grupos generales, que son: la navegación mediante algoritmos analíticos y la navegación mediante aprendizaje.

El enfoque analítico tiene su origen en la robótica clásica y busca una solución genérica, predecible y repetible mediante una formulación matemática. Este enfoque suele ser más sencillo de implementar cuanto más conocido y controlado sea el entorno de trabajo. Por su parte, el enfoque basado en aprendizaje busca, por lo general, una solución mediante el entrenamiento a través de un experto o a través de información disponible del entorno, pudiendo llegar a desarrollar esquemas de navegación más flexibles que la navegación basada en algoritmos analíticos, aunque su generalidad y predictibilidad dependerán en gran medida de cuán exhaustivos sean los entrenamientos o la información disponible del entorno. A continuación se describen con mayor detalle cada uno de estos paradigmas y las distintas alternativas disponibles dentro de cada uno de ellos.

2.4.1. Navegación analítica

La navegación mediante algoritmos analíticos es la navegación más conocida y usual siendo, de hecho, los primeros algoritmos de navegación usados. Este enfoque busca modelar analíticamente, mediante una fórmula matemática, el comportamiento de navegación del robot. Así pues, cuanto más conocido y controlado sea el entorno de trabajo más fácil será determinar la fórmula matemática para una correcta navegación. La navegación analítica se caracteriza por buscar una solución genérica, para un entorno con unas características concretas, predecible y repetible. Su inconveniente es que es bastante complejo caracterizar el comportamiento de navegación completo si no se conocen a priori todas las situaciones que se deberán afrontar o no se conoce perfectamente el entorno de pruebas, que es lo que suele ocurrir en entornos de pruebas reales. Por otro lado, este enfoque suele ser sensible a los parámetros de configuración del algoritmo, que se deben adaptar y optimizar para situaciones específicas, siendo su optimización compleja para situaciones generales. Es por esta misma razón que este enfoque es apropiado para entornos con circunstancias particulares, no siendo, en general, flexible cuando se trata con entornos cuya configuración puede cambiar considerablemente, siendo necesario reajustar los parámetros de configuración del algoritmo para un comportamiento óptimo.

2.4.1.1. Método de Campos Potenciales

La navegación mediante el Método de Campos Potenciales (PFA) [61], *Potential Fields Approach* en inglés, es uno de los esquemas de navegación reactiva más conocidos y empleados debido a su sencillez y velocidad. Gracias a su simplicidad este esquema puede ser aplicado en entornos dinámicos no

estructurados permitiendo una velocidad de respuesta rápida ante obstáculos inesperados.

La navegación mediante PFA considera al robot como una partícula libre cargada que se haya bajo la influencia de un campo potencial, de forma que los obstáculos tienen la misma carga que el robot y el objetivo una carga de polaridad distinta a la del robot. Así pues, los obstáculos generan un campo de repulsión artificial alrededor suyo y el objetivo a alcanzar genera un campo potencial de atracción hacia él. Debido a la sencillez de este método, ha sido usado no sólo en navegación para la evitación de obstáculos sino también para tareas como selección de comportamientos [83] o coordinación [149]. Sin embargo, este esquema de navegación también tiene algunos inconvenientes, como las oscilaciones o la caída en mínimos locales. La ecuación general del PFA es:

$$U(rob) = k_{at} \cdot dist(rob, dest)^2 + \sum k_{rep} \cdot \frac{1}{dist(rob, obs)^2} \tag{2.1}$$

Determinando las constantes k_{at} y k_{rep} ($k_{at} > 0, k_{rep} < 0$) la mayor o menor atracción o repulsión al objetivo o a los obstáculos respectivamente. Así pues, el vector de movimiento del robot ($\vec{v}_R$) será igual al gradiente negativo de dicho campo ($-\nabla U(rob)$).

El PFA es un algoritmo de navegación simple y genera unas trayectorias suaves a la vez que evita las colisiones con los obstáculos. Sin embargo, tiene una serie de inconvenientes, como son [138]:

- Sensibilidad a la existencia de mínimos locales en el campo potencial. En el caso de que el robot alcance un mínimo local es posible que éste se quede atrapado indefinidamente en él. Este problema puede ser resuelto mediante el uso de un esquema de navegación híbrido que evitase los mínimos locales.

- Generación de oscilaciones en las cercanías de obstáculos.

- Dificultad o incapacidad para desplazarse por zonas con obstáculos cercanos entre sí, como pasillos estrechos.

- Dependencia de los parámetros de configuración con el entorno. Este problema tiene relación con los anteriores, ya que si en el entorno existen zonas estrechas por las que el robot debe pasar la configuración de parámetros tendrá que disminuir el rango de seguridad del robot para permitir el acceso a dichas zonas, mientras que si las zonas de paso son suficientemente amplias se puede aumentar el rango de seguridad del robot con los obstáculos. De esta forma, la configuración de los

parámetros origina los anteriores problemas de oscilaciones así como la imposibilidad o dificultad de atravesar obstáculos cercanos entre sí.

2.4.1.2. Histograma de Campo de Vectores

La navegación mediante Histograma de Campo de Vectores (VFH) [23], *Vector Field Histogram* en inglés, se basa en la representación estadística del entorno del robot. Concretamente, lo que hace es representar en un histograma la densidad de obstáculos del entorno. Esta técnica permite determinar las regiones del entorno donde hay una menor densidad de obstáculos, de forma que el robot se dirija a estas regiones preferentemente.

Este algoritmo fue diseñado para ser eficiente, robusto y presta especial atención a la incertidumbre de los sensores y al modelado de los errores. Esto es debido a la representación estadística de los obstáculos (mediante un histograma de ocupación de celdas), la cual es especialmente interesante cuando hay errores en los sensores.

El funcionamiento del VFH se basa en una representación bidimensional de los obstáculos, que es transformada en un histograma polar de una dimensión a partir de la posición actual del robot. Una vez realizado ésto, se selecciona la dirección a seguir como el ángulo en el que la densidad de obstáculos sea inferior a un determinado umbral, así como el ángulo más próximo a la dirección del destino. Finalmente, se dirige el robot en la dirección establecida.

Este algoritmo fue mejorado posteriormente [134] (llamándose VFH+ y posteriormente VFH*) para tener en cuenta en el proceso de navegación el tamaño del robot, reducir las oscilaciones del método original, aumentar la eficiencia y evitar mínimos locales.

Funciona bien y ofrece una buena respuesta temporal, sin embargo tiene varios inconvenientes, como que no es capaz de dirigirse intencionadamente hacia un obstáculo una vez detectado y que se ralentiza en la presencia de una gran cantidad de obstáculos. Por otro lado, y como ocurre en general con los esquemas analíticos, es sensible a los parámetros de configuración.

2.4.1.3. Ventana Dinámica

La navegación mediante el Método de Ventana Dinámica (DWA), *Dynamic Windows Approach* en inglés, es una técnica para evitar obstáculos [38] basada en la dinámica del robot y está especialmente diseñada para poder tratar con las limitaciones de velocidad y aceleración del robot.

Para ello se genera un espacio de búsqueda de posibles rutas y luego se busca la solución óptima dentro del espacio de búsqueda establecido impo-

niendo las restricciones de velocidad y aceleración iniciales. De esta forma se busca la trayectoria óptima que permite alcanzar la meta de una forma segura y con la mínima cantidad posible de obstáculos dentro del intervalo de tiempo actual. Esta trayectoria se va calculando para cada intervalo de tiempo, de forma que si en un intervalo de tiempo se puede aproximar demasiado a un obstáculo a la velocidad actual, esta se reduce, por ejemplo.

Visto de otro modo, este algoritmo comprueba todas las posibles velocidades que puede alcanzar el robot de una forma segura (sin colisiones) en una determinada ventana temporal, por lo que permite navegar al robot a altas velocidades por el entorno de pruebas. Estas velocidades son optimizadas teniendo en cuenta una serie de funciones relacionadas con la distancia a la meta y a los obstáculos más cercanos.

2.4.1.4. Bandas Elásticas

La navegación mediante Bandas Elásticas [105], *Elastic Bands* en inglés, consiste en deformar una trayectoria que permita alcanzar el destino deseado (trayectoria que será calculada por un planificador de caminos) en función de unas fuerzas artificiales que se crean a partir de la distribución de obstáculos presentes en el entorno. La trayectoria resultante de la deformación de las bandas elásticas es, por lo general, suave y eficiente, siendo una técnica adecuada para entornos parcialmente conocidos en los que la trayectoria calculada por el planificador previo sea válida a largo plazo y no deba ser recalculada constantemente. La principal ventaja de este algoritmo es evitar a los planificadores de alto nivel el cálculo de una nueva ruta ante pequeños cambios en el entorno, aplicando modificaciones a la ruta original en tiempo real. Sin embargo, cuando el entorno de aplicación es desconocido o muy dinámico esta técnica no es adecuada, pues una desviación excesiva de la ruta original puede causar que ésta se vuelva insegura o no fiable. Otro problema de esta técnica, es la sensibilidad con los parámetros de configuración, siendo complejo un ajuste óptimo para circunstancias no predecibles.

2.4.1.5. Diagrama de Proximidad

La navegación basada en Diagrama de Proximidad (Nearness Diagram) es, en su concepción, muy parecido al VFH, ya que consiste en dividir los 360 grados alrededor del robot en sectores, almacenando cada uno de estos sectores la proximidad de los obstáculos que contiene. De esta forma se estima la proximidad de los obstáculos al centro del robot y se calcula el camino libre de obstáculos más cercano a la dirección en la que se encuentra el destino.

Existen modificaciones de este método, en las que se indica la proximidad

de los obstáculos respecto a la parte exterior del robot, por lo que esta modificación permite su aplicación a robots con distintas formas y tamaños más fácilmente que el algoritmo original, estimando de una forma más correcta la seguridad durante el recorrido.

Su principal ventaja respecto al VFH es que se comporta mejor que éste en entornos con una gran cantidad de obstáculos. Su principal problema, es el mismo que ya se ha comentado con los demás algoritmos analíticos, y es la necesidad de ajustar adecuadamente los parámetros de configuración del algoritmo para obtener un resultado óptimo en cada entorno.

2.4.2. Navegación basada en inteligencia artificial

La navegación basada en aprendizaje tienen su origen en la IA y los algoritmos de aprendizaje surgidos de ésta. El aprendizaje puede ser definido como *cualquier cambio en un sistema que permita realizar una tarea mejor la siguiente vez que se realice o que permite realizar una tarea que antes no se podía hacer* [132], pudiendo manifestarse este aprendizaje como conocimiento obtenido por observación o aprendizaje, por ejemplo. Por lo tanto, la navegación basada en aprendizaje intenta solventar algunos de los problemas de la navegación mediante métodos analíticos permitiendo el movimiento del robot mediante el conocimiento y la experiencia adquirida por el sistema. Así pues, las técnicas basadas en aprendizaje suelen basarse en la capacidad de imitar la percepción, aprendizaje y razonamiento humanos para solventar problemas complejos [28]. Sin embargo, y como todas la técnicas, tiene sus ventajas y desventajas.

Una de las ventajas de este enfoque es la flexibilidad que ofrece frente a los enfoques analíticos, más rígidos al tratar de resolver todas las posibles situaciones que debe afrontar el robot mediante una ecuación analítica. El método basado en aprendizaje, por su parte, pretende particularizar la respuesta del sistema según el escenario en el que se encuentre, ofreciendo mayor flexibilidad al permitir aprender comportamientos concretos para cada situación específica, lo cual es difícil conseguir mediante algoritmos analíticos. Por lo tanto, el enfoque basado en aprendizaje ofrece una alternativa a la resolución de problemas complejos mediante la simplificación de dichos problemas particularizando según cada circunstancia específica.

Por su parte, la desventaja de este tipo de navegación es la falta de generalidad, pues es difícil conseguir un aprendizaje lo bastante completo como para poder afrontar cualquier situación posible. La forma de solucionar este problema es aumentar el entrenamiento y conocimiento almacenado por el robot para incluir mayor cantidad de situaciones a afrontar. Sin embargo, lo más usual es realizar un aprendizaje que permita disponer de distintos com-

portamientos para las situaciones más usuales a afrontar, lo que permitiría seleccionar el comportamiento más adecuado en cada situación, solventando parcialmente el problema.

A continuación se describen algunas de las técnicas de IA más usuales basadas o usadas en aprendizaje [28], como son el CBR (*Case-Based Reasoning* en inglés), las Redes Neuronales Artificiales (ANN) (*Artificial Neuronal Network* en inglés) o los Sistemas Basados en Reglas (RBS) (*Rule-Based System* en inglés).

2.4.2.1. Razonamiento basado en reglas

Los sistemas basados en reglas (RBS) tuvieron sus orígenes en torno a los años 70, y se basan en la solución de problemas mediante la representación explícita del conocimiento a través de un conjunto de reglas heurísticas generadas por un experto en el área [46], siendo una regla una expresión formada por una condición y una acción a realizar en caso de que se cumpla la condición.

Los sistemas RBS suelen estar formados por varios módulos que almacenan las variables y conjunto de reglas del sistema (base de conocimiento), que comprueban si se cumplen o no las condiciones de las distintas reglas (motor de inferencia), que determinan si las reglas se deben aplicar y en que orden y, por último, que ejecutan las acciones asociadas a las reglas.

Uno de los problemas de este tipo de sistemas es que las reglas deben estar correctamente definidas para solucionar el problema deseado, por lo que es necesario tener un gran conocimiento de dicho problema y que éste sea comprensible para poder establecer las reglas correctas [28]. Es por esto que los RBS suelen aplicarse fundamentalmente a problemas bien estructurados que pueden ser descritos con relativa facilidad por un conjunto de reglas deterministas bien definidas, como un sistema de control de tráfico, sistemas de seguridad o transacciones bancarias.

Otro problema de estos sistemas es que no son adecuados cuando existen un elevado número de excepciones o situaciones particulares, ya que en estos casos el número de reglas necesarias para describir dichas situaciones sería muy elevado lo que haría el sistema muy complejo y difícil de mantener, pudiendo además degradar su respuesta. Como es obvio, y por esta misma razón, estos sistemas no suelen proporcionar soluciones adecuadas cuando se pretenden resolver circunstancias excepcionales e imprevistas no contenidas en el conjunto de reglas del sistema. Así pues, es conveniente aplicarlos en entornos donde no haya una gran cantidad situaciones excepcionales o bien restringir o limitar las condiciones del entorno para evitar dichas situaciones. Otras situaciones en las que no son convenientes aplicar los RBS es en sis-

temas cuyas interacciones son complejas o los procesos que tienen lugar no se entienden completamente. Así por ejemplo, los RBS son usados en entornos relativamente cerrados con condiciones específicas (en los que se pueden aplicar reglas más generales), como por ejemplo en la identificación de enfermedades específicas en animales o cultivos [76, 154] o en la identificación de elementos como nombres propios, organizaciones o localizaciones en texto no estructurado [3].

2.4.2.2. Redes neuronales artificiales

Las redes neuronales artificiales (ANN) son unas de las técnicas de IA más clásicas, y están basadas en el funcionamiento de las neuronas biológicas. Las ANN son ampliamente utilizadas y desde su origen han sido aplicadas a una multitud de áreas como: clasificación de patrones [94], clustering [145], funciones de aproximación [29], predicción [128], optimización o control [13].

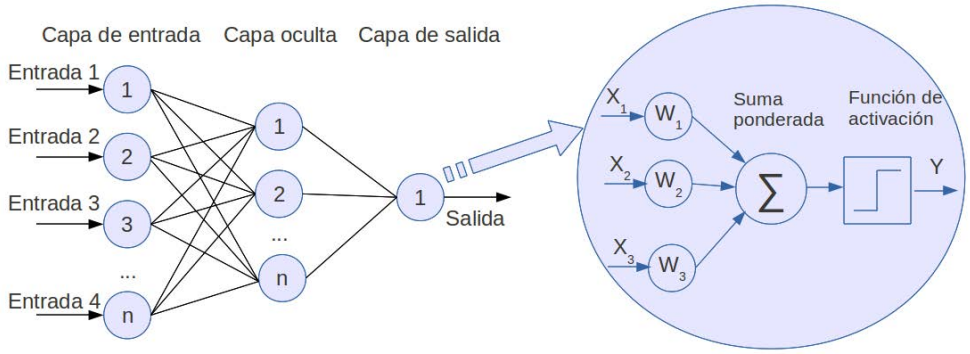

Figura 2.3: Red neuronal y estructura interna de una neurona.

Las ANN están compuestas por una gran cantidad de neuronas interconectadas entre sí (Fig. 2.3) siendo la unidad básica de procesamiento la neurona. El funcionamiento básico de una neurona consiste, simplemente, en realizar una suma ponderada (por los pesos de cada entrada) de los distintos valores de entrada de la neurona, aplicando el valor obtenido a una función de activación para determinar si la neurona está activa o no. Así pues, para que una ANN funcione, antes es necesario realizar un entrenamiento de ésta, que permitirá configurar de forma adecuada los pesos [91] de las distintas entradas de las neuronas que forman la ANN, permitiendo así obtener la respuesta esperada. De esta forma, las ANN son capaces de inducir conocimiento sobre el comportamiento de un sistema a partir de la información de dicho sistema.

Por último, una vez que una ANN ha sido entrenada, y si el entrenamiento ha sido satisfactorio, al aplicar a su entrada un patrón concreto debería obtenerse la respuesta esperada en función de la aplicación específica de la red.

Como se ha indicado, las ANN son ampliamente usadas, siendo especialmente interesantes para la solución de problemas con una gran cantidad de datos en los que el algoritmo o las reglas de resolución del problema son desconocidos o difíciles de expresar [155], para tareas en las que es necesario procesamiento paralelo o hay errores en los datos de entrada, o bien cuando hay que encontrar relaciones entre datos que aparentemente no tienen relación.

Sin embargo, y a pesar de sus grandes ventajas, las ANN también tienen sus inconvenientes. Uno de ellos [28] es que son, en esencia, una caja negra que no ofrece información sobre como se ha realizado el aprendizaje, no permitiendo la explicación o justificación de una determinada respuesta, al igual que es difícil, en caso de que el sistema no aprenda algún patrón concreto, averiguar porqué no se ha aprendido y cómo forzar su aprendizaje. Igualmente, y derivado del mismo problema, si un entrenamiento no converge, no hay forma, en principio, de averiguar porqué. Por otro lado, las ANN suelen requerir una gran cantidad de datos para realizar su entrenamiento, siendo también necesario clasificar y organizar los datos a presentar a la red para separar los patrones de entrenamiento y patrones de comprobación. Por último, esta técnica tampoco es adecuada cuando el comportamiento del sistema no se puede generalizar por un único modelo debido, por ejemplo, a que haya muchas excepciones o no haya, como tal, un patrón en el sistema.

2.4.2.3. Algoritmos genéticos

Los Algoritmos Genéticos (GA), *Genetic algorithm* en inglés, son una técnica que se basan en imitar la evolución biológica y la selección natural. Así pues, el algoritmo hace evolucionar una población de partida, que es modificada de forma aleatoria, al igual que ocurriría en el caso de mutaciones y recombinaciones genéticas, realizando tras un cierto período de evolución un filtrado o selección siguiendo unos determinados criterios. A partir de esta selección se determinan los individuos que encajan mejor en la solución esperada, y éstos son los individuos que sobreviven (siendo los demás descartados) pudiendo evolucionar nuevamente hacia la nueva generación hasta la próxima selección. Sin embargo, para que los GA funcionen de forma adecuada es necesario asegurar que la población inicial sea lo suficientemente grande y variada como para garantizar la variedad de soluciones, razón por la cual suele generarse de forma aleatoria.

La secuencia de pasos de un GA, como se ha comentado, es la combinación (reproducción) de las soluciones actuales, su mutación (añadiendo variabilidad), y la selección de la siguiente población. Todos estos pasos se repiten hasta que se de una condición de fin (alcanzar un valor óptimo o por encima de un umbral determinado), cuando no se produzcan más cambios en las generaciones (el algoritmo converge) o cuando se realicen un número determinado de iteraciones.

Los GA son computacionalmente simples y robustos y permiten, de forma implícita, un alto grado de paralelismo, siendo especialmente útiles para la optimización de parámetros, aunque su idoneidad depende del problema concreto, siendo recomendable limitar el espacio de búsqueda (evolución) del algoritmo, de la facilidad para definir una función para evaluar la adecuación de una determinada respuesta y de si las soluciones se puedan codificar de una forma relativamente sencilla. Así por ejemplo, los GA han sido aplicados para la calibración de sistemas para modelado de la calidad del agua [96], selección de características para previsión de la calidad del aire [59] o para la estimación de la densidad aparente del suelo [30].

Sin embargo, los algoritmos genéticos también tienen sus inconvenientes, ya que pueden tardar en converger o incluso no converger y, en caso de converger, si la función a optimizar tiene muchos puntos cercanos al valor óptimo, no se puede asegurar que se vaya a obtener el valor óptimo.

2.4.2.4. Razonamiento basado en casos

El CBR es una técnica de aprendizaje derivada los estudios de Roger Schank y Robert Abelson según la cual el ser humano resolvería los problemas según la memoria de experiencias pasadas y la anterior resolución de problemas similares [120]. Es decir, que cuando una persona debe afrontar un problema, lo primero que hace es comprobar si se ha enfrentado anteriormente a dicha situación, o una similar, y qué solución se aplicó en dicha situación previa.

Así pues, el CBR intenta resolver problemas reutilizando y adaptando experiencias o soluciones (llamadas casos en el CBR) aplicadas anteriormente para resolver problemas similares [75]. La esencia del CBR se basa en la asunción de que *problemas similares tienen soluciones similares* [65].

El CBR dispone de una base de conocimiento (base de casos en el CBR) formada por la experiencias pasadas, que son llamadas casos, y en las cuales, cada vez que se afronta un problema nuevo, se buscan problemas similares y, en caso de encontrarlos, se usan o adaptan las soluciones aplicadas a dicho problema a la situación actual.

Por lo tanto, para que un sistema CBR funcione tan sólo es necesario

definir la estructura de los casos (las variables que definen el problema y las que definen la solución) y disponer de información sobre el entorno o el problema a solucionar.

Esta técnica es especialmente interesante pues, mediante la actualización de la base de casos, el CBR mejora continuamente su capacidad de razonamiento y las soluciones dadas a los problemas. Además, el CBR es capaz de trabajar con grandes cantidades de datos y múltiples variables. Sin embargo, el CBR no es capaz de inferir soluciones de problemas que no sean parecidos a casos anteriores. La ventaja del CBR es que, teniendo información suficiente sobre el problema y sus soluciones, no es necesario entender el problema para que el sistema funcione correctamente. Por otro lado, una ventaja del CBR es que, si fuese necesario modificar alguna respuesta concreta o averiguar porque está dando una respuesta específica, se puede estudiar la base de casos para encontrar si hay casos contradictorios y eliminarlos o corregirlos si se dispone de información suficiente. Esta transparencia del CBR es una gran ventaja frente a otras técnicas de aprendizaje más opacas, pues ofrece un gran control potencial sobre el sistema de aprendizaje.

El CBR es una técnica de aprendizaje muy conocida y usada debido a sus ventajas, habiendo sido aplicada a tareas muy diferentes [14, 16, 17, 50, 90], como el soporte a la toma de decisiones [60], monitorización de la calidad el aire [58], predicción de ciclones tropicales [118], análisis de imágenes [99] o la estimación de necesidades para la asistencia de personas con discapacidades [31]. Dentro del área de la robótica expresamente, el CBR ha sido usado para planificación de alto nivel [44, 131], selección de comportamientos y de acciones [73, 114] o selección de parámetros de comportamiento [68].

A modo de resumen, entre las principales ventajas que presenta este paradigma se pueden destacar las siguientes [2, 62, 65, 148]:

- La base de casos del CBR representa lo que el sistema ha aprendido y puede ser analizada y accedida fácilmente, permitiendo saber en todo momento el conocimiento almacenado del sistema así como depurar el sistema y obtener información sobre porqué se aplica un resultado (o caso) en vez de otro [65].

- Si existen situaciones concretas a las que hay que aplicar una solución específica o se conoce alguna solución concreta a una situación determinada se puede introducir manualmente de forma expresa en la base de casos [65, 148] fácilmente.

- El aprendizaje en el CBR se basa en añadir nuevos casos a la base de casos, por lo que, en principio, se puede añadir nueva información

mientras se está ejecutando, permitiendo un aprendizaje incremental de una forma sencilla [65].

- Debido a su aprendizaje incremental, no es necesario un entrenamiento exhaustivo para poner en funcionamiento esta técnica, pudiendo utilizarse con una cantidad mínima de información, siendo esta aumentada posteriormente si fuese necesario.

- La aplicación de esta técnica no requiere un modelado, entendimiento o conocimiento general del problema al que se va a aplicar, sino que tan sólo se necesita acumular experiencia en situaciones específicas [2] y la conversión de dicha experiencia, de forma adecuada, en casos para formar la base de datos de casos del CBR. Esto es una gran ventaja respecto a otras técnicas, pues sólo requiere tener las soluciones a unos determinados problemas, sin necesidad de entender las soluciones ni cómo se ha llegado a ellas.

Por lo tanto, y como resultado de sus ventajas, esta técnica tiene también algunas desventajas, y es que no permite la generalización del problema a resolver, pues se basa en experiencias concretas. También relacionado con este problema, el CBR no permite inferir o deducir soluciones si no hay casos mínimamente similares al problema actual [28]. Otro problema al usar esta técnica es que es conveniente comprobar si existen soluciones contradictorias ante situaciones similares, lo cual puede requerir un mantenimiento de la base de datos para evitar incoherencias.

2.4.3. Selección del algoritmo de navegación

Como se ha comentado en la sección 2.4.1, los algoritmos analíticos han sido ampliamente usados desde los orígenes de la robótica. Sin embargo, sus comportamientos no siempre resultan intuitivos o lógicos según la forma de actuar de las personas (realizando, por ejemplo, trayectorias muy próximas a los obstáculos o navegando justo en el punto medio de éstos, en función de los parámetros configurados), a pesar de que en general sean eficientes y seguros. Sin embargo, el principal problema de los algoritmos analíticos es su dependencia con una serie de parámetros que necesitan ser optimizados para cada problema o entorno específico y para las plataformas robóticas a utilizar, ya que los parámetros, por lo general, dependen de la distribución de obstáculos del entorno y de la cinemática y dinámica del robot, así como de la calibración de sus sensores. Desafortunadamente, y como se ha comentado en la sección 2.2, los robots a usar son robots con extremidades, por lo que la cinemática no es fácil de obtener, especialmente cuando se desvían de los

cálculos teóricos debido a particularidades de los robots, como que un motor tenga más potencia que otro lo que hace que el robot tenga tendencia a girar más hacia un lado, por ejemplo. Así pues, el desarrollo de expresiones analíticas para relacionar los sensores (visión en el caso del AIBO) con los actuadores (los comandos de movimiento) puede resultar una tarea compleja.

Por otro lado, las técnicas basadas en aprendizaje permite solventar problemas complejos de caracterizar, como puede ser la navegación, sin la necesidad de realizar un modelado analítico de éste. Además, la navegación basada en aprendizaje permite una mayor flexibilidad a la hora de establecer comportamientos *ad hoc*. Por contra, este enfoque no ofrece soluciones óptimas ni ofrece siempre una solución a pesar de que pueda haberla (como ocurre con los algoritmos analíticos), requiriendo, además, de un período de entrenamiento.

Así pues, y tras estudiar los pros y los contras de cada enfoque, la opción de navegación basada en aprendizaje se plantea como el enfoque más adecuado para la implementación de la navegación sobre robot AIBO de Sony, debido sobre todo a la mayor flexibilidad y facilidad de implementación y modificación de los comportamientos frente a la navegación analítica. Por lo tanto, y una vez determinado que se usará navegación basada en aprendizaje, será necesario determinar cuaĺ de las técnicas de IA comentadas encaja mejor con la tarea a realizar.

En primer lugar, los GA son una técnica de IA que se basa, esencialmente, en la optimización de parámetros [77], permitiendo encontrar soluciones a problemas, aunque hay que determinar correctamente los parámetros a optimizar, lo cual podría resultar más complejo que el uso de otras técnicas más directas. Por otro lado, aunque han sido aplicados a la navegación robótica [57, 77, 124], pero en general han sido aplicados a la planificación de caminos sobre un mapa del entorno [77, 124], lo cual requiere, al menos, dicha información de partida. Por otro lado, los GA funcionan como una caja negra, en el sentido de que aunque encuentran una solución a un problema, pero no se sabe cómo ni porqué devuelven dicho valor, no pudiendo, a posteriori, depurarse o estudiar el problema para intentar mejorarlo fácilmente.

Por otro lado, los RBS son adecuados cuando se pueden establecer fácilmente las reglas de comportamiento del sistema, hay pocas situaciones excepcionales y el problema a resolver es comprensible y se tiene un buen conocimiento de éste. Así pues, esta técnica no es adecuada para la navegación basada en visión por varias razones. En primer lugar, es necesario conocer el problema en profundidad, y la navegación que se obtendría sería una navegación genérica, no pudiendo particularizarse fácilmente, teniendo el mismo problema que la navegación analítica. Por otro lado, al estar la

navegación basada en visión existe una alta posibilidad de que haya errores en la detección de los elementos del entorno mediante visión, lo cual dificultaría el establecimiento de reglas en el sistema para una correcta navegación. Así pues, este enfoque no parece el más adecuado para realizar la navegación mediante visión basada en aprendizaje.

Respecto a las ANN han sido ampliamente usadas desde los orígenes de la robótica en una gran cantidad de ámbitos. Sin embargo, son una caja negra en la que no se sabe como ha aprendido el sistema ni que ha aprendido y es difícil forzar el entrenamiento de situaciones específicas que puedan surgir posteriormente. Además, las ANN suelen requerir una gran cantidad de datos para realizar su entrenamiento. Así pues, y aunque sería viable su uso, pero no se considera que las ANN sean la técnica más adecuada para la navegación por aprendizaje que se pretende realizar pues, debido a la cantidad de información que suelen requerir las ANN, los entrenamientos para adquisición de información podrían ser largos y tediosos. Aparte de eso, una característica que sería deseable en el sistema de aprendizaje es la capacidad de saber y tener acceso a lo que ha aprendido el robot así como añadir conocimiento fácilmente, y estas no son características que ofrezcan las ANN.

Por último, está la alternativa del CBR. El CBR permite un aprendizaje incremental añadiendo información directamente a la base de casos así como saber qué a aprendido el sistema en cada instante. Además, el CBR es adecuado para situaciones en las que se puede obtener información fácilmente, sin necesidad de tener una gran cantidad de información, y no requiere ni un modelado ni un entendimiento del problema, sino tan sólo determinar los parámetros que describen la situación o problema a resolver y la solución. Así pues, esta técnica solventa los inconvenientes que presentaban las ANN y permite una implementación sencilla de un comportamiento de navegación reactiva, pues permite acoplar de forma inmediata los sensores (que definen el problema del caso) a la acción o comando de movimiento del robot (que define la solución del caso). Así pues, la técnica de aprendizaje que mejor se adapta al problema de navegación reactiva sería la de CBR.

2.5. Conclusiones

En este capítulo se ha presentado un estado del arte de arquitecturas de control y de sistemas de navegación, estudiando sus ventajas y desventajas y seleccionando finalmente las opciones que mejor se adaptan al objetivo del trabajo a realizar.

Como arquitecturas de control, las opciones más viables eran la arquitectura híbrida y reactiva, dado el entorno dinámico en el que se realizarán las

pruebas del sistema. De entre ambas opciones, se ha escogido la arquitectura híbrida pues en general ofrece unas mejores prestaciones que las arquitecturas reactivas. Sin embargo, dado que actualmente no es necesaria ninguna capa de alto nivel, se ha comenzado por la capa reactiva de bajo nivel, a la que se añadirán posteriormente, según se considere necesario, capas superiores para hibridizar el sistema y permitir respuestas más complejas. Así pues, actualmente el sistema se puede considerar que es un sistema reactivo, hasta que se le añada alguna capa de nivel superior.

Respecto al algoritmo de navegación, si bien la solución más clásica y usual es utilizar un algoritmo analítico como PFA o DWA, en el presente caso se ha optado por una implementación basada en aprendizaje. La razón es que el aprendizaje permite una mayor libertad a la hora de definir las posibles trayectorias a realizar o las posibles respuestas del sistema. Por contra, tienen problemas como que no ofrecen siempre respuestas seguras (por lo que es conveniente añadir un control de seguridad para detener el robot en caso de peligro) o que la obtención de la información necesaria para el aprendizaje puede resultar a su vez compleja y tediosa.

Sin embargo, se considera que las ventajas que ofrece el uso de una técnica basada en aprendizaje puede ser interesante frente al enfoque clásico. Así pues, y concretamente dentro de las distintas técnicas de aprendizaje, como pueden ser ANN, RBS o CBR, se ha escogido esta última, principalmente por su transparencia y posibilidad de depuración y acceso a la información aprendida, lo que permite, en caso de problemas, determinar o intentar determinar que está pasando, porqué e incluso tratar de buscar una solución, algo que en el caso de las otras técnicas, más opacas en ese sentido, no es fácil de conseguir.

Capítulo 3

Aprendizaje de comportamientos básicos

3.1. Introducción

El uso de los MRS puede ofrecer ventajas frente al uso de los sistemas robóticos individuales. Sin embargo, el uso de los MRS supone tener que resolver los mismos problemas que en robots individuales, más una serie de problemas añadidos derivados de la coordinación y cooperación de los robots que forman el sistema. Así pues, en primer lugar se solucionarán los problemas relativos a robots individuales, para pasar posteriormente a los problemas propios de los MRS, como la coordinación.

Como ya se ha comentado en el capítulo 2, los robots usarán una arquitectura híbrida para navegación. Sin embargo, dado que inicialmente no se realizarán tareas de alto nivel, en primer lugar se planteará únicamente el desarrollo de la capa navegación reactiva de los robots, la cual se implementará mediante comportamientos aprendidos usando CBR, añadiendo posteriormente capas de alto nivel según se consideren necesarias. Así pues, en el presente capítulo se estudia un método simple y directo, alternativo al clásico método analítico, que permite enseñar comportamientos a un robot para que se mueva siguiendo unos determinados patrones de navegación de una forma sencilla y cómoda.

La idea básica de este método consiste en relacionar la información visual que obtiene el robot con los comandos de movimiento que está realizando en un instante determinado mediante un comportamiento aprendido. Para ello se usará aprendizaje LfD controlando remotamente al robot. Concretamente, el entrenamiento consistirá en controlar remotamente el robot para que realice unas trayectorias determinadas, que determinarán el comportamiento

reactivo. De esta forma, durante el aprendizaje lo que se hace es relacionar directamente los comandos de movimiento enviados al robot con los parámetros extraídos de la imagen, los cuales están asociados a la situación específica en la que se ejecutan dichos comandos.

Una de las ventajas de este enfoque es que para poder obtener comportamientos con distintas respuestas, tan sólo es necesario entrenar de nuevo el sistema en el nuevo comportamiento deseado. Otras ventajas son que no es necesario realizar ningún modelo cinemático explícito del robot y que los errores sistemáticos y mecánicos del robot (como por ejemplo que el robot gire más a la izquierda que a la derecha) son absorbidos implícitamente por el entrenamiento [47, 102]. Esto es debido a que, cuando el operador controla el robot, implícitamente absorbe y asimila la estructura y movimientos del robot debido a la capacidad de adaptación innata de las personas.

Así pues, en primer lugar se presentará un breve estado del arte sobre la aplicación del CBR en el área de la navegación robótica (sección 3.2). Tras esto, en la sección 3.3 se comenta en más detalle el CBR, la técnica seleccionada para realizar el aprendizaje de los comportamientos y, tras esto, se describirá en profundidad el sistema implementado (sección 3.4). Posteriormente se aplicará el enfoque propuesto a dos tareas que servirán de prueba de concepto para comprobar que el sistema propuesto puede ser viable para la implementación de comportamientos reactivos aprendidos y, finalmente, se presentarán los resultados (sección 3.5) y las conclusiones (sección 3.6) de dichos experimentos, indicando tanto las ventajas como las desventajas del enfoque seleccionado.

3.2. CBR aplicado a navegación autónoma

La técnica de aprendizaje seleccionada para implementar los comportamientos aprendidos es el CBR. La razón es, por un lado, que esta técnica permite un fácil acceso a la información almacenada, lo que permite una depuración y comprensión del sistema más fácil que otras técnicas de IA. Por otro lado, el funcionamiento del CBR se basa en almacenar experiencias, descritas mediante pares solución-problema, para utilizarlas posteriormente en la resolución de situaciones concretas. Por lo tanto, el CBR encaja perfectamente con la implementación de un comportamiento reactivo, cuya base de funcionamiento es acoplar la información de los sensores (que define el problema actual) y a la acción o comando de movimiento del robot (que define la solución al problema).

Sin embargo, y a pesar del paralelismo que pueda existir entre el planteamiento del CBR y los comportamientos reactivos, el CBR no ha sido usado

generalmente para la navegación reactiva. De hecho, las tareas relacionadas con la navegación robótica en las que se ha usado el CBR han sido generalmente las siguientes:

- Selección de comportamientos, estrategias de navegación y parámetros. En esta categoría se muestran trabajos [15, 44, 68–71, 73, 106, 114, 115, 125, 131] que usan el CBR para la selección de comportamientos o estrategias de navegación, por lo que se usa para navegación de alto nivel, usando por debajo distintos tipos de algoritmos, desde secuencias de movimiento preestablecidas, hasta algoritmos analíticos como PFA.

  También se engloba en esta categoría los trabajos orientados a la selección de parámetros para mejorar el algoritmo de navegación usado, en general para que se adapte mejor a las circunstancias concretas. Estos trabajos se engloban en la misma categoría que los de selección de comportamiento pues, en muchos trabajos, estos unen la selección de un comportamiento y sus parámetros concretos mediante CBR para adaptarlos al entorno.

  Así por ejemplo, [69] propone un esquema que permite seleccionar, mediante CBR, los comportamientos a activar, así como los parámetros de dichos comportamientos, según la situación en tiempo real. De esta forma, los parámetros usados por los comportamientos son más adecuados que si se usan unos parámetros fijos. Este trabajo continúa el trabajo de [106] añadiendo una representación espacio-temporal del entorno que permite describir mejor las situaciones y obtener mejores respuestas por parte del CBR. Para ello, [69] crea una descripción espacial y temporal del entorno, de forma que primero se realiza en el CBR una búsqueda de los casos más similares respecto a las características espaciales, seleccionando entre éstos, el que tenga una descripción temporal del entorno más similar al actual. Este enfoque fue posteriormente mejorado en [71] y [70] permitiendo el aprendizaje de nuevos parámetros y la optimización de los casos ya almacenados. Otro trabajo similar, que parte también de [69], es [68], en el que se presenta un sistema para selección de parámetros (como la ganancia para dirigirse hacia la meta o evitar un obstáculo o cuanto se puede aproximar a los obstáculos) de los comportamientos que controlan la navegación de un robot.

  Por otro lado, y centrándose sólo en la selección de tareas, comportamientos y estrategias, hay otros trabajos como [73], que presenta un algoritmo de navegación en dos fases en el que el CBR es usado para determinar los comportamientos a activar en función de una serie de comandos genéricos. Concretamente, en una primera fase se determina,

mediante GA, una secuencia de comportamientos genéricos para realizar una tarea concreta en una plataforma genérica. Posteriormente, en una segunda fase, se aplican sobre una plataforma real estos comportamientos genéricos usando el CBR para determinar a que comportamientos concretos, se corresponde en esa plataforma el comportamiento genérico a ejecutar, seleccionando y activando los comportamientos en tiempo real.

Siguiendo otro enfoque distinto, [125] presenta una arquitectura híbrida aplicable a juegos en tiempo real, compuesta por una capa de alto nivel analítica, una capa intermedia basada en CBR y una capa reactiva con una serie de comportamientos y acciones en el juego, de forma que el CBR es usado para aprender que acciones son más adecuadas dependiendo de la situación concreta del juego.

En un enfoque algo más similar al presentado aquí, [15, 114] presentan un sistema basado en CBR para selección de las acciones o secuencias de acciones de un equipo de robots AIBO en la competición Robocup. En este caso el CBR decide, a partir de una base de casos creada manualmente, que acción (pasar pelota, esquivar, etc.) deben realizar los robots dependiendo de la situación concreta. Sin embargo, estos trabajos están orientados a estrategias de más alto nivel que el propuesto en este trabajo.

Por otro lado, [115] presenta en un sistema con 4 capas (selección de líder o coordinador, selección de estrategia y roles, asignación de roles y ejecución de estrategia) en el que el CBR se utiliza en la segunda capa para la selección de estrategias y roles de un equipo de robots, consistiendo una estrategia en un plan de acción a largo plazo, indicando una serie de roles para los robots y las tareas que deben realizar.

Por último, [44,131] proponen un sistema basado en CBR para estimar el interés de una persona en interactuar con un robot. Concretamente, el CBR funciona como una capa de alto nivel que estima, partiendo de entrenamientos previos, el interés potencial de la persona en interactuar, determinando así la estrategia de navegación (basada en campos potenciales) que realizará el robot para tratar de interactuar con la persona.

- Planificación global de caminos. En esta categoría se muestran varios trabajos [24, 39, 43, 49, 65] que aplican el CBR a la planificación global de caminos bajo distintas condiciones de partida.

  Así, por ejemplo [24, 39] aplican el CBR a la planificación global de caminos en entornos estáticos partiendo de un mapa métrico conocido

a priori, donde los casos absorben la estructura del entorno, usándose en [24] abstracciones que permiten simplificar los casos y mejorar la eficiencia del CBR.

Por su parte, [43] aplica el CBR a la planificación global de caminos en entornos dinámicos basándose en un mapa topológico del entorno conocido a priori [43]. En este trabajo se trataba de encontrar un camino posible desde un origen a un destino en un mapa real de Pittsburgh a partir de rutas almacenadas, las cuales tendrían en cuenta factores como los patrones de tráfico o el número de vías. El problema es que en el caso de cambios inesperados en el mapa no es posible descubrir nuevas soluciones a menos que se reorganizase el mapa topológico regularmente [65], lo cual supone un coste computacional elevado. Este problema es debido a la estrecha relación entre la descripción del entorno y la localización del robot.

Por otro lado, [65] propone un método para planificación global de caminos en entornos dinámicos usando el CBR para evitar situaciones donde haya que afrontar obstáculos, minimizando las posibilidades de colisión. Este trabajo soluciona el problema de la continua reorganización del mapa de [43] para obtener nuevos resultados mediante el uso de un mapa basado en cuadrículas, que permite obtener soluciones nuevas más fácilmente que en un mapa topológico. Sin embargo, este trabajo indica que la navegación global basada en CBR es especialmente útil cuando hay una gran densidad de obstáculos, de forma que solo haya pocas soluciones al problema que se desea solucionar. Esto es un problema inherente al CBR pues, ante problemas en los que hay muchas posibles soluciones, es difícil determinar la solución correcta si no hay suficiente información sobre el problema. En este caso, al haber pocos obstáculos hay multitud de posibles caminos y, en principio, todos o una gran cantidad de ellos, son igualmente buenos.

Por último, [49] aplica también el CBR a la planificación global de caminos con la idea de reutilizar y adaptar rutas anteriormente válidas en vez de encontrar rutas nuevas. La diferencia con los enfoques anteriores es que [49] utiliza un grafo de segmentos del camino en los casos CBR en vez del camino completo, de forma que sea más sencilla la búsqueda, el reconocimiento de la situación y su posible adaptación a nuevas situaciones. En este trabajo, y debido al enfoque utilizado, es necesario un algoritmo analítico para aquellos casos en los que el CBR no disponga de información sobre el camino actual para poder alcanzar el destino, o bien para enlazar dos tramos segmentos de camino que no tienen conexión.

- Soporte a la navegación [112]. [112] propone un sistema de ayuda a la navegación para entornos semi-estructurados que ayudaría a determinar situaciones en las que no hay posibles salidas, o que camino es mejor coger bajo determinadas circunstancias. El CBR se acopla a un sistema de navegación (que ya permite la navegación y mapeo del entorno) que detectará situaciones peligrosas, conflictivas, sin salida, etc, indicándolo al sistema de navegación para evitar alcanzar dichas situaciones.

Como se puede observar, los trabajos relacionados con navegación basados en CBR utilizan el CBR para gestionar, resolver o mejorar tareas generalmente de alto nivel, estando normalmente las tareas de bajo nivel o bien codificadas mediante secuencias preestablecidas o bien implementados mediante algoritmos analíticos, pero no mediante CBR.

De hecho, la literatura que se ha encontrado disponible sobre la aplicación del CBR a tareas de bajo nivel en navegación [136, 137, 139] está relacionada con el propio grupo de investigación. En estos trabajos, el CBR se usa con un enfoque como el presentado en el presente trabajo, realizando aprendizaje LfD mediante control remoto asociando movimientos concretos ante situaciones específicas de forma reactiva a bajo nivel. De hecho, el desarrollo actual parte de la experiencia obtenida en dichos trabajos. Sin embargo, la diferencia entre estos trabajos y el presente es que los trabajos anteriores se basan en una plataforma con ruedas usando sonar y el presente trabajo se basa en una plataforma con extremidades usando visión. Por lo tanto, en este capítulo se comprobará si este enfoque para la creación de comportamientos reactivos es aplicable igualmente sobre una plataforma AIBO.

3.3. CBR: Funcionamiento y configuración

El paradigma del CBR surgió a finales de los años 70 [121] a partir de los estudios de Roger Schank y Robert Abelson y su idea básica es que *problemas similares tienen soluciones similares* [65]. Así pues, el CBR intenta resolver problemas reutilizando y adaptando, si es necesario, experiencias o soluciones aplicadas anteriormente a problemas similares [1].

El primer sistema que puede ser llamado un razonador basado en casos fue CYRUS [2], desarrollado por Janet Kolodner a principio de los años 80. Este primer sistema era poco más que un sistema de pregunta-respuesta con conocimiento sobre varios viajes y reuniones del secretario de estado Cyrus Vance. Sin embargo, y a pesar de su simplicidad, el modelo de memoria desarrollado para esta aplicación sirvió de esqueleto para posteriores sistemas basados en CBR, como MEDIATOR, PERSUADER, CHEF o JULIA.

Dentro de las técnicas de IA, el CBR es un tipo de sistema experto, ya que su objetivo es tratar de imitar el comportamiento de un experto en alguna materia, ya sea un mecánico, un abogado o un médico. Para que un sistema experto sea capaz de imitar el comportamiento de una persona es necesario que este sistema tenga una capacidad de razonamiento o deducción similar a la de la persona (al menos en el ámbito de aplicación del sistema) o bien disponga de conocimiento suficiente sobre como actuar ante los problemas, imitando la actividad anterior del experto aunque, obviamente, con limitaciones. Así pues, el CBR, como sistema experto, necesita disponer de una base de conocimiento para poder actuar. En su caso, y dado que el CBR resuelve problemas en base a soluciones y experiencias anteriores, su base de conocimientos (o base de casos) estará formada por experiencias o casos, siendo un caso una porción de conocimiento que representa una experiencia o solución asociada al contexto en el que tuvo lugar. Por lo tanto, un caso estará formado por par contexto-experiencia o problema-solución, determinando el contexto el ámbito en el que una solución concreta es aplicable.

Por otro lado, el funcionamiento de un sistema CBR puede verse de distintas formas, siendo las más usuales expresar su funcionamiento como un modelo de proceso en forma de ciclo o como un modelo estructurado de tareas. Ambos son complementarios entre sí, aunque el primero de ellos suele ser más usado y ofrece una visión general más clara del funcionamiento del CBR.

Tanto este ciclo de funcionamiento como la base de conocimiento que permite funcionar a un sistema CBR se explican en mayor detalle en los siguientes apartados.

3.3.1. Ciclo del CBR

El CBR se puede ver como un ciclo, formado normalmente por 4 procesos, que se ejecuta cada vez que se presenta un nuevo problema al CBR, finalizando el ciclo cuando se obtiene una solución a dicho problema, dependiendo la solución del conocimiento que posea el CBR.

Las 4 fases que componen el ciclo normal de un sistema CBR (Fig. 3.1), conocido generalmente como ciclo 4R por el nombre de sus 4 fases, son: recuperar, reutilizar, revisar y retener. Sin embargo, y debido a que existen una gran variedad de sistemas CBR, habrá sistemas que incluyan estas 4 fases, mientras que habrá otros que tan sólo incluyan algunas de ellas.

En cualquier caso, y de forma general, cada vez que se presenta al CBR un nuevo problema, este problema debe ser transformado en un caso de entrada (C_{in}) siguiendo la estructura de parámetros definida para el CBR en cuestión. En general, un caso está formado por un par problema-solución, sin embargo,

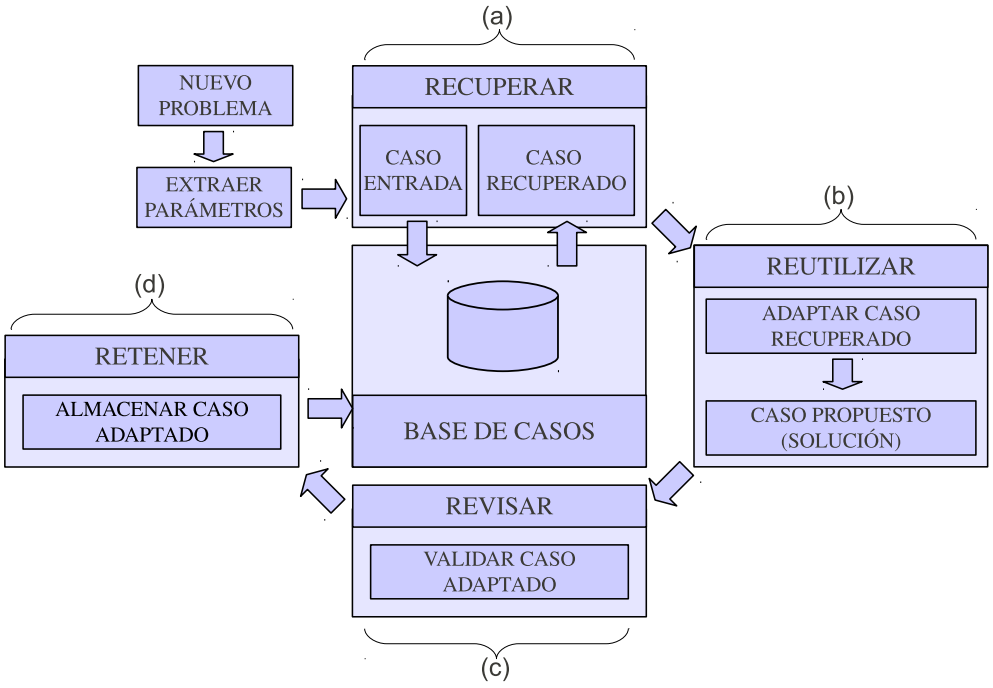

Figura 3.1: Típico ciclo del CBR

el caso de entrada (C_{in}) creado a partir del problema actual estará formado únicamente por el problema o contexto, estando el campo correspondiente a la solución o experiencia vacío, pues es la respuesta que el CBR debe dar. Así pues, y tras la conversión del problema en un caso de entrada (C_{in}), se introducirá en el ciclo CBR, realizándose las siguientes tareas en cada una de sus fases:

- *Recuperar* (o *Retrieve* en inglés). En esta fase (Fig. 3.1.a) el CBR busca, en la base de casos, el caso (C_{rec}) o casos cuyo problema del par problema-solución se parezca más al problema actual (C_{in}).

- *Reutilizar* (o *Reuse* inglés). En esta fase (Fig. 3.1.b), si el caso recuperado de la base de casos (C_{rec}) es lo suficientemente parecido al caso de entrada (C_{in}), se aplica como solución al problema actual. Si no es lo suficientemente parecido, el caso recuperado es adaptado, de forma que se pueda aplicar a la situación actual.

- *Revisar* (o *Revise* en inglés). En esta fase (Fig. 3.1.c) se comprueba, en caso de que se haya aplicado adaptación al caso recuperado, que la solución propuesta soluciona correctamente el problema.

- *Retener* (o *Retain* en inglés). Normalmente, en caso de que se haya aplicado adaptación y la solución propuesta sea correcta, en esta fase (Fig. 3.1.d) se almacena en la base de casos el caso adaptado para su uso en futuras situaciones. De forma general, esta fase contempla la adición de casos a la base de casos, tanto si son casos adaptados, como casos generados manualmente.

A continuación se explican en profundidad cada una de las fases así como los parámetros o factores que es importante tener en cuenta en cada una de ellas. Tras la explicación en detalle de las fases, se hará una recopilación de los parámetros que permiten describir completamente un sistema CBR, y que serán usados a lo largo de este trabajo para detallar las distintas implementaciones del CBR.

3.3.1.1. Recuperar

El objetivo de esta fase es encontrar, dentro de la base de casos, el o los casos más parecidos al caso que describe la situación o problemática actual. Es decir, el objetivo de esta fase es obtener, de la experiencia previamente almacenada en el sistema, la más parecida al problema actual. Así pues, esta fase comienza por la correcta conversión del problema o situación a los parámetros que describen el problema en forma de caso. Por lo tanto, para una correcta identificación de los casos similares es importante que los parámetros de descripción del problema se ajusten lo más posible a éste.

En el proceso de búsqueda dentro de la base de casos no se está buscando un caso que coincida exactamente con el caso actual, sino el más parecido. Para ello es necesario utilizar un algoritmo de búsqueda como pueden ser el de [148] búsqueda por inducción, por prototipos o por vecindad. De entre estas técnicas, la más genérica y usual es la de búsqueda por vecindad [65], que consiste en establecer una función de distancia que permite determinar que casos se parecen más al actual. Sin embargo, la velocidad de respuesta de esta técnica depende de la cantidad de casos que haya en la base de casos. Las otras técnicas mencionadas solucionan este problema, siendo especialmente útiles cuando la descripción del problema tiene características que permiten filtrar los casos a considerar (búsqueda por inducción) o cuando la base de casos es demasiado grande y es necesario acotar la búsqueda entre un conjunto inicial para acelerar la selección posterior (búsqueda por prototipos). La desventaja de estas técnicas es la posibilidad de pérdida de casos en la búsqueda y el preprocesado que es necesario aplicar a la base de casos.

Así pues, y considerando la técnica de búsqueda por vecindad, que es la más usual, para su aplicación es necesario utilizar una función de similitud

o parecido que permita determinar que caso se parece más al actual. Para obtener esta medida de similitud hay que determinar que distancia se adapta mejor al problema en cuestión. Entre las distintas distancias más usuales están, por ejemplo, la distancia Euclídea, la distancia Tanimoto o la distancia Manhattan, pudiendo cada una de ellas estar ponderadas por un determinado vector de pesos o no. Estos pesos determinarían la importancia de unas características frente a otras, siendo los pesos iguales en caso de que no haya unas características más importantes que otras.

3.3.1.2. Reutilizar

En esta fase el caso recuperado en la fase anterior es reutilizado para solucionar el problema presente. El caso recuperado es el caso más parecido dentro de la base de casos, pero eso no quiere decir que sea el caso adecuado al problema actual. Así pues, en esta fase hay que estimar como de parecido es el caso recuperado respecto al caso de entrada y si es necesario aplicar algún tipo de adaptación a la solución del caso recuperado para que se adecúe mejor al problema actual.

Para realizar esta adaptación existen distintas técnicas posibles, siendo algunas de las más usuales [148]:

- Adaptación nula. Una técnica directa y simple que consiste en aplicar directamente la solución recuperada al problema actual sin aplicar ningún tipo de adaptación. Esta solución es útil en sistemas sencillos que requieren soluciones muy simples o cuando no es fácil obtener un algoritmo de adaptación de la solución adecuado.

- Adaptación estructural. Esta técnica consiste en aplicar la adaptación del caso directamente a la solución recuperada mediante el ajuste de los parámetros que se consideren necesarios.

- Adaptación derivada. Esta técnica consiste en aplicar el algoritmo, método o reglas usados para generar la solución original de la base de casos y aplicarla para generar una nueva solución para el problema actual. El problema de esta técnica de adaptación es que sólo es aplicable en problemas que se conocen perfectamente y cuyas soluciones se han generado mediante algún método fácilmente aplicable de forma autónoma.

De entre estas técnicas de adaptación, una de las más usuales es la adaptación estructural, pues es la más factible si el problema no se conoce perfectamente. Sin embargo, en casos en los que no se desea o no sea conveniente realizar ajustes sobre la solución original, es mejor usar la técnica de

adaptación nula estudiando, posteriormente, la posibilidad de añadir nuevo conocimiento a la base de casos para solventar los problemas afrontados y no resueltos de una forma completamente satisfactoria.

3.3.1.3. Revisar

En la fase de revisión se evalúa si la solución adaptada es adecuada para resolver el nuevo problema planteado. Así pues, esta fase sólo tiene sentido en aquellos casos en los que se haya aplicado una adaptación al caso recuperado, ya que se da por hecho que los casos ya almacenados solucionan correctamente los problemas a los que están asociados.

La validación de la solución aplicada puede aplicarse sobre un modelo del sistema o sobre una simulación que permita estimar las consecuencias de la solución aplicada, aunque lo más normal es que se aplique la solución adaptada por el sistema y se estudien las consecuencias de dicha solución a posteriori mediante observación directa. Por último, otra posibilidad sería la validación manual mediante un experto que estudie la solución dada al problema concreto. En cualquier caso, en aquellas situaciones en las que la solución no se considere satisfactoria, se puede marcar como incorrecta para no repetirla en casos futuros, borrarla o bien adaptarla corrigiendo la solución.

3.3.1.4. Retener

En esta última fase se añaden, a la base de casos, nuevos casos que permiten la resolución de nuevos problemas que no están, actualmente, almacenados. En general, en esta fase se añaden todas las soluciones adaptadas que hayan sido validadas en la fase anterior, de forma que se incorporen a la base de casos, siendo posibles soluciones a tener en cuenta en futuros problemas.

Sin embargo, no sólo se almacenan dichas soluciones, pues esta fase también contempla la adición de nuevos casos que puedan ser aplicados a problemas en los que las soluciones adaptadas no han sido adecuadas. En definitiva, esta fase supone una fase importante en el CBR ya que es la que le permite evolucionar, aprender y aumentar su conocimiento, bien sea a partir de los aciertos, almacenando las soluciones adaptadas válidas, bien de los errores, generando en ese caso nuevos casos que permitan afrontar dichos problemas, bien de nuevos casos introducidos expresamente de forma manual.

De forma global, en esta fase se consideran dos tipos de aprendizaje:

- Aprendizaje por observación. Este aprendizaje es el que se realiza cuando se introduce en la base de casos experiencias obtenidas por un operador humano mediante observación directa de situaciones y soluciones

a dichas situaciones. En este tipo de aprendizaje los casos añadidos se consideran válidos pues son introducidos directamente por un observador que debe validarlos antes de introducirlos. Este aprendizaje es el que se realiza para introducir el conjunto inicial de casos [142] en cualquier sistema CBR, y se asemeja a la etapa de entrenamiento inicial de otros sistemas basados en aprendizaje, como pueden ser las ANN.

Este tipo de aprendizaje también se puede dar cuando, tras haber aplicado al sistema CBR una serie de casos de entrada a los que no ha sido capaz de dar una respuesta satisfactoria, se añaden al CBR nuevas experiencias (un segundo aprendizaje) obtenidas por un observador, que permitan al sistema afrontar esas situaciones que no ha sido capaz de afrontar por sí mismo tras el primer aprendizaje.

En definitiva, este aprendizaje es un aprendizaje manual, que exige la presencia de un observador humano para validar y generar las experiencias a aprender.

- Aprendizaje por experiencia. Este aprendizaje se realiza implícitamente cuando una nueva solución es evaluada satisfactoriamente y se incorpora en la base de casos y es una etapa importante en un sistema CBR, pues puede permitir el aprendizaje del sistema a partir de la resolución de nuevos problemas sin necesidad de un supervisor (siempre que la validación de casos pueda realizarse de forma no supervisada).

  El enfoque general del aprendizaje por experiencia consiste en añadir, a la base de casos, los casos adaptados que hayan sido validados, entrando a formar parte de la base de conocimiento y pudiendo ser usados para resolver futuros problemas. A este enfoque se le conoce como aprendizaje positivo.

  Sin embargo, existe otro enfoque, conocido como aprendizaje negativo, que consiste en que, cuando una solución adaptada no resuelva adecuadamente el problema, o bien se corrige expresamente dicha adaptación incorrecta (mediante alguna técnica especial o a través de la intervención de un operador experto) o bien se almacena dicho caso, en una base de casos fallidos, para evitar que se vuelva a producir el mismo error de adaptación en el futuro.

Todas estas técnicas de aprendizaje son válidas y, de hecho, se pueden emplear de forma conjunta. Sin embargo, no es normal aplicarlas todas, limitándose generalmente a algunas de ellas para no complicar en exceso el sistema CBR ni ralentizarlo.

Por último es importante destacar que, cada vez que se añaden casos a la base de casos, se corre el riesgo de que surjan problemas dependiendo de la estructura de la base de casos, como se comentará en los siguientes apartados. Así por ejemplo, en el caso de las estructuras planas la inserción de nuevos casos es inmediata, aunque se corre el riesgo de aumentar en exceso el número de casos así como de añadir casos muy similares, por lo que es conveniente aplicar periódicamente algún algoritmo de filtrado y de control del número de casos en la base de conocimiento. Por su parte, en el caso de bases de datos jerárquicas, cada vez que se añade un nuevo elemento es necesario procesar la base de datos para indexar de nuevo los casos o para determinar que éstos están en el grupo correcto dentro de la base de datos. Otro proceso que puede ser necesario al añadir nuevos casos es la reorganización o reestructuración de los casos para su acceso óptimo. Todo esto supone, por tanto, un trabajo extra que hay que añadir al proceso de aprendizaje, pues si la base de conocimiento no está en óptimas condiciones el sistema CBR no podrá trabajar adecuadamente.

3.3.2. Representación del conocimiento

El CBR es una técnica de IA que trata de resolver nuevos problemas a partir de la experiencia en la resolución de problemas anteriores. Por lo tanto, un elemento crucial en todo sistema CBR es su base de conocimiento o base de casos, que es donde se almacena toda la experiencia y, por tanto, donde reside la capacidad de resolución de problemas del CBR.

Las distintas experiencias en el CBR son almacenadas en la base de conocimiento en forma de casos, pudiendo definirse un caso como una *pieza de conocimiento contextualizada que representa una experiencia que enseña una lección fundamental que permite al sistema CBR alcanzar su objetivo.* Así pues, un caso almacena información, que sería la solución a un problema, de forma contextualizada, asociando así dicha solución al problema o situación a la que se puede aplicar, estableciendo una relación entre un solución y la situación en que fue aplicada. Por lo tanto, y de forma simplificada, un caso está formando un par problema-solución o situación-conocimiento. Así pues, una etapa importante en el CBR es la definición de los parámetros que caracterizarán cada par problema-solución, pues si estos no están representados adecuadamente, el CBR no devolverá soluciones correctas.

Por otro lado, también hay que tener en cuenta la forma en que los casos son almacenados en la base de casos, pues esta influirá en gran medida en la eficiencia que tendrá el sistema final y, como consecuencia, en la calidad de la solución que se obtendrá. La dependencia del CBR con la forma en que se almacenan los datos es importante pues el ciclo del CBR se basa en

la búsqueda de dichos casos, y cuanto más fácil y rápido sea buscarlos y encontrarlos, más eficiente será el sistema y mejor será la solución que se obtenga de éste.

3.3.2.1. Representación del caso

Un caso almacena la experiencia o solución asociada a un contexto o problema concreto, estableciendo así una relación entre un problema concreto y la solución aplicada a éste. Por lo tanto, un caso está formado por los pares problema-solución[1], y debe describir de forma completa tanto el problema a resolver como su solución.

Por otro lado, e independientemente de la estructura de la base de casos, el problema es usado como índice de búsqueda de la base de datos, ya sea de forma directa o indexado o categorizado de alguna manera.

Por lo tanto, la representación del par problema-solución mediante parámetros (Fig. 3.2), que formarán finalmente un caso, es una parte importante y crucial en un sistema CBR, pues determinará factores como la facilidad de búsqueda de los casos, su relevancia para el problema, su inteligibilidad o los posibles problemas en su búsqueda, en caso de que los parámetros para la descripción de los casos sea ambigua y se obtengan soluciones que no corresponden realmente al problema actual.

Aunque no existe un consenso sobre que información debe contener un caso, pero si existen dos elementos a tener en cuenta al decidir que debería estar representado en un caso: la funcionalidad y la facilidad de obtener la información representada en el caso [148].

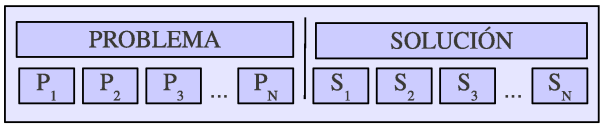

Figura 3.2: Representación de un caso CBR.

Así pues, para el correcto funcionamiento del CBR es de vital importancia la correcta definición y elección de los parámetros que describirán el problema y su solución, es decir, la correcta caracterización del problema y de su solución. Además es importante tener en cuenta que este conjunto de parámetros debe ser lo más reducido posible para minimizar el impacto computacional,

[1]Existen sistemas CBR en los que cada caso está formado por el trío problema-solución-resultado, indicando también el resultado obtenido al aplicar dicha solución, aunque no es lo más usual.

pues a mayor cantidad de parámetros, más costosa resultará la búsqueda de casos.

Poniendo un ejemplo genérico de navegación, que a fin de cuentas es la aplicación a la que se va a destinar el CBR en el presente trabajo, el problema, dependería de la posición del robot (R_x, R_y), del destino a alcanzar (D_x, D_y) y de los obstáculos en el camino (O_x, O_y), por lo que se podría caracterizar por dichos parámetros, mientras que la solución a dicho problema, es decir, el comando de movimiento que debe realizar el robot, podría caracterizarse por la velocidad frontal y la velocidad de rotación (V_f, V_{rot}). Sin embargo, aunque estos parámetros parecen lógicos, pero surgen problemas o dudas respecto a ellos. Por ejemplo, en el caso de la definición del problema, habría que determinar cuantos obstáculos se tienen en cuenta, o si las posiciones se indican de forma absoluta o de forma relativa a la posición del robot, o si las posiciones serán cartesianas o polares. Todas estas dudas son importantes, y pueden tener un gran impacto en el sistema, ya que en el caso de usar posiciones absolutas, habría que realizar un entrenamiento más completo abarcando todas las posibles posiciones absolutas a tener en cuenta, por ejemplo. De igual manera, en el caso de la solución, el usar las velocidades puede haber problemas pues, si se utiliza dicho sistema con un robot distinto al original, puede que no acepte comandos de movimiento basados en velocidades, por lo que igual sería más oportuno usar comandos de movimiento normalizados. Todos estos, son problemas que hay que tener en cuenta y estudiar sobre el terreno cuando se realiza la definición del caso que formará la base de casos del CBR, teniendo en cuenta todas las ventajas y desventajas de cada opción.

3.3.2.2. Estructura de la base de casos.

La base de casos en un sistema CBR es la que almacena todo el conocimiento y experiencia del sistema, y está formada por el conjunto de casos que lo componen. Al igual que ocurre con la definición del caso propiamente dicha, la base de casos, que no es más que una memoria para almacenar información, puede tener distintas estructuras de almacenamiento con sus ventajas y desventajas, lo cual tendrá una influencia directa en las prestaciones del sistema final, sobre todo en el tiempo de respuesta y en la recuperación satisfactoria de los casos más adecuados en cada situación.

Por lo tanto, existen diferentes estructuras para la realización, por ejemplo, de búsquedas más rápidas usando una clasificación por categorías, así como búsquedas mediante indexación de los casos por determinada similitud o búsquedas directas.

Aunque existen una multitud de posibles estructuras para organizar la

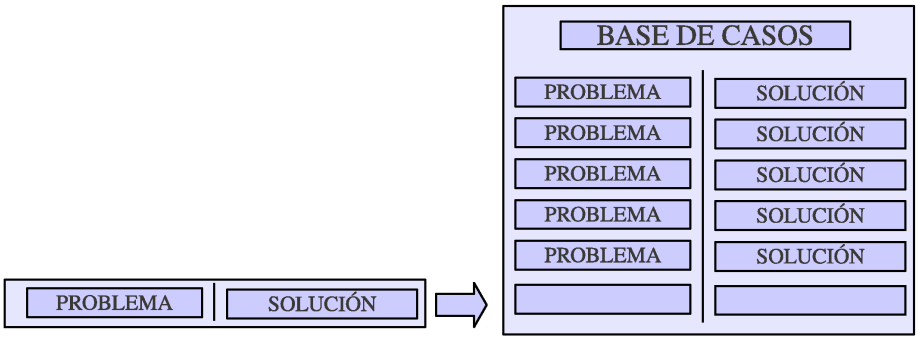

Figura 3.3: Representación de una base de casos plana.

base de casos en un sistema CBR, todas ellas se pueden dividir en dos grandes grupos:

- Estructura plana. En las estructuras planas (Fig. 3.3) los casos se almacenan secuencialmente de forma independiente, sin establecer ninguna relación ni agrupación explícita entre ellos. Esta estructura es sencilla de implementar, garantiza que en la búsqueda de casos se devuelve el más parecido que exista en la base de casos y la inserción de nuevos casos es tan simple como añadirlo al final de la base de casos. Sin embargo, también tiene algunos inconvenientes, ya que en la búsqueda se realiza la comparación del caso de entrada con todos los casos de la base, por lo que la velocidad de respuesta en las búsquedas dependerá de la cantidad de casos que almacene la base de casos. Por esta razón, y a pesar de que no requiere ningún post-procesamiento al añadir nuevos casos, es conveniente aplicar periódicamente algún algoritmo que permita comprobar que el número de casos de la base de casos no crece en exceso para no reducir el rendimiento del sistema. En definitiva, es una estructura adecuada cuando se puede controlar que el número de casos de la base de casos no crezca en exceso.

- Estructura jerárquica. En las estructuras jerárquicas los casos se organizan en la base de casos de forma jerárquica formando grupos según su parecido teniendo en cuenta determinados parámetros. Esta estructura es más compleja que la estructura plana, pero a cambio de dicha complejidad, permite realizar búsquedas mucho más rápidas permitiendo el uso de bases de casos con un elevado número de casos. Para ello, en el proceso de búsqueda en vez de realizar una búsqueda exhaustiva como en las estructuras planas, se discriminan los casos (según los grupos a los que pertenezcan) que no se consideran parecidos al de entrada,

buscando únicamente en los que se consideren similares. Por lo tanto, esta estructura solventa el problema de las estructuras planas, pero a cambio de una serie de problemas, como la mayor complejidad de implementación, la posibilidad de pérdida de casos significativos debido a la discriminación de casos en la búsqueda jerárquica, o la complejidad al añadir nuevos casos, que supone, cada vez que se añada un caso, realizar una reestructuración y reorganización de la base de casos para que ésta sea consistente. Algunos ejemplos de estructuras jerárquicas usados habitualmente son la estructura jerárquica basada en el modelo de memoria dinámica y la basada en el modelo de categorías y ejemplares.

Por lo tanto, las ventajas de las estructuras planas son que siempre devuelven el caso más similar al actual (no descartan posibles soluciones debido a agrupaciones o accesos a grupos incorrectas) y que la inserción de casos es sencilla y barata en términos computacionales. Sin embargo, la búsqueda en estas estructuras son más lentas que en las jerárquicas si hay una gran cantidad de casos debido a que las estructuras planas buscan en toda la base de casos. Sin embargo, este problema tiene una fácil solución manteniendo el número de casos de la base de casos acotado. Las estructuras jerárquicas, por su parte, son más eficientes en las búsqueda pero la implementación y la inserción de casos resulta más compleja que en las estructuras planas. Por otro lado, y al igual que las estructuras planas requieren un control del número de casos en la base, las estructuras jerárquicas requieren un mantenimiento para asegurar la coherencia de los índices y agrupaciones de los casos almacenados. Además, las estructuras jerárquicas son sensibles a la pérdida potencial de casos significativos si en la búsqueda se accede a un grupo de búsqueda no equivocado.

3.3.3. Parámetros para descripción del CBR

En este apartado se resumen, de forma breve, los elementos o parámetros que son necesarios determinar para describir un sistema CBR así como su funcionamiento. Estos elementos son:

- Estructura de la base de casos. Hay que determinar si la base de casos tiene una estructura plana o jerárquica (lo que determinará el tamaño máximo aconsejable de ésta así como su velocidad de búsqueda) y los algoritmos de indexación y/o agrupación usados en función del tipo de estructura seleccionado.

- Estructura del caso. La estructura de parámetros que define un caso es crucial para una correcta búsqueda posterior de los casos así como de

los más parecidos.

- Algoritmo de búsqueda en la fase recuperación. Los algoritmos de búsqueda más usuales son búsqueda por inducción, por prototipos o por vecindad, dependiendo el uso de uno u otro, sobre todo, de la estructura de la base de casos.

- Algoritmo de similitud en la fase recuperación. El algoritmo de similitud suele ser una distancia distancia de similitud como la distancia Euclídea, Tanimoto, Manhattan, etc. En el caso de usar una distancia ponderada, hay que indicar los pesos de cada parámetro también.

- Algoritmo de adaptación en la fase de reutilización. Los posibles algoritmos de adaptación son: adaptación nula, adaptación estructural y adaptación derivada, siendo necesario en el caso de las dos última indicar como se realizará la adaptación sobre el caso recuperado.

- Algoritmo de evaluación en la fase de revisión. Se indicará como se realizará la validación de las soluciones adaptadas a los nuevos problemas en caso de que se aplique adaptación.

- Algoritmo de aprendizaje en la fase de retención. Hay que determinar los tipos de aprendizaje que se usarán en esta fase para el sistema CBR, siendo las alternativas más usuales aprendizaje por observación y por experiencia (positiva y/o negativa).

3.4. Navegación basada en comportamientos aprendidos

En este capítulo se presenta un sistema que permite la navegación reactiva de un robot mediante aprendizaje. Concretamente, el objetivo en este capítulo es demostrar que se pueden desarrollar distintos comportamientos reactivos que permitan la navegación de un robot mediante aprendizaje, sin necesidad de modificar (o realizando modificaciones mínimas) el sistema. Así pues, para generar distintos comportamientos tan sólo sería necesario entrenar al robot para que realice el comportamiento deseado y determinar los parámetros que definen tanto la situación actual como su solución. La idea es, por lo tanto, comprobar que el enfoque de navegación basada en comportamientos aprendidos es viable y flexible, de forma que pueda ser usada posteriormente para implementar la navegación coordinada del MRS, que es el objetivo de la presente tesis.

La arquitectura del sistema es una arquitectura híbrida, aunque este capítulo se centra únicamente en la implementación de la capa de navegación reactiva basada en comportamientos aprendidos, dejando la descripción de las capas superiores para próximos capítulos. La razón de usar aprendizaje en vez de un algoritmo analítico es que el aprendizaje permite una mayor flexibilidad a la hora de definir los comportamientos así como una mayor facilidad de implementación y modificación de éstos, como se comentó en el capítulo 2.

Dentro de las distintas técnicas de IA y aprendizaje, se usará la técnica de CBR porque su aplicación es inmediata en situaciones en las que se puede obtener información fácilmente, permite tener acceso a la base de conocimiento del sistema en cada instante y no requiere un modelado ni un entendimiento profundo del problema. Además, el CBR encaja perfectamente con la idea de los comportamientos reactivos, pues permite acoplar de forma inmediata los sensores (cuya información definiría el problema del caso) a la acción o comando de movimiento del robot (que definiría la solución del caso), ya que los casos del CBR están formados precisamente por pares problema-solución.

Por otro lado, el aprendizaje se realizará mediante aprendizaje por demostración (LfD) [8,130]. Este aprendizaje consiste en realizar un entrenamiento, con el robot controlado remotamente, de forma que el operador sólo tenga que considerar como quiere mover al robot y hacia donde quiere dirigirlo en función de lo que el robot está percibiendo. A través de este entrenamiento, el CBR asocia las acciones del operador (los comandos de movimiento del joystick) a un estado (la información visual extraída de la imagen actual) de una forma sencilla. La sección 3.4.3 describe en más detalle el LfD así como sus ventajas. Tras finalizar este entrenamiento, el CBR dispone de información suficiente para que el robot puede comenzar a funcionar de forma autónoma utilizando la información obtenida durante el entrenamiento.

Para comprobar la viabilidad del enfoque, se realizarán pruebas con un robot AIBO de Sony, cuyo principal sensor de entrada es una cámara, en un entorno con una pelota rosa y un pequeño campo de fútbol.

Concretamente, para realizar los experimentos se van a entrenar dos comportamientos como prueba de concepto, que permitirán estimar la validez del sistema en esta plataforma. En ambas tareas se ha utilizado el mismo programa, cambiando únicamente el entrenamiento y la definición del caso CBR. De esta forma, se comprueba la versatilidad y flexibilidad del sistema para realizar distintos comportamientos, pues sólo es necesario redefinir los parámetros del caso y realizar un entrenamiento para implementar un comportamiento. Dado que es una prueba de concepto, estas dos tareas son usadas simplemente para comprobar la viabilidad del enfoque propuesto. Si

bien las tareas a realizar son básicas, pero cumplen su objetivo, además de permitir una primera toma de contacto con el sistema de navegación reactiva basado en CBR. Las dos tareas a realizar son, en principio, independientes y no se ejecutarán de forma simultánea, sino que, como ya se mostrará más adelante, se enlazarán de forma secuencial. También hay que tener en cuenta que en estas tareas no se tienen en cuenta la presencia de otros robots en el campo, ya que esto se tendrá en cuenta en próximos capítulos (concretamente a partir del capítulo 5).

Así pues, la primera de las tareas consiste en la búsqueda de una pelota de color conocido a priori en un campo de fútbol, restricción que es considera normal al plantearse un problema basado en visión [146]. Hay que tener en cuenta también que, aunque el seguimiento de una pelota es un problema bastante conocido y que ha sido solucionado usando distintas aproximaciones, pasando desde las puramente reactivas hasta la predicción usando filtros de Kalman [66, 122], pero el objetivo en sí no es otro que, como ya se ha comentado, el de realizar unas pruebas preliminares de navegación reactiva basada en CBR usando para ello un comportamiento sencillo del que poder sacar conclusiones fácilmente.

Por otro lado, la segunda tarea será la búsqueda de la portería en un campo de fútbol mediante detección de líneas en la imagen. La localización basada en detección de líneas ha sido usada normalmente en el entorno de la Robocup, habitualmente en combinación con balizas de colores y porterías [79,129]. Así por ejemplo, algunos métodos se basan en el color [53, 108, 109] para detectar las líneas por medio de un escaneo vertical de la imagen, procesando posteriormente toda esta información mediante técnicas de Montecarlo para estimar la posición del robot y calcular la acción a realizar a partir de la posición del robot. La principal diferencia del enfoque propuesto con respecto a los métodos descritos es que en la navegación reactiva basada en CBR no existe un modelo explícito del entorno, no obteniéndose la posición del robot respecto al campo, sino estimándose implícitamente mediante el aprendizaje, devolviendo directamente el comando de movimiento que debe realizar el robot para alcanzar la portería. En cualquier caso, y al igual que en el ejemplo de la búsqueda de la pelota, es necesario tener en cuenta que esta tarea es usada simplemente para probar la versatilidad de la propuesta de construcción de comportamientos basados en CBR.

Como se puede observar, ambas tareas utilizan una definición del entorno distinta (en la primera se tienen en cuenta la detección de zonas de color, mientras que en la segunda se usan las líneas detectadas en la imagen). Sin embargo, en ambas se usa el mismo sistema y el mismo enfoque basado en entrenamiento, mostrando la versatilidad del sistema propuesto. Por último,

tras finalizar los entrenamientos y las pruebas, ambas tareas se enlazarán secuencialmente, de forma que se ejecutará primero la tarea de buscar la pelota y, una vez encontrada y capturada, se llevará la pelota a la portería para chutar a gol. Estas tareas, así como las ventajas del aprendizaje por demostración y el esquema global del sistema se desarrollan en más detalle en los siguientes apartados.

3.4.1. Comportamiento de búsqueda de pelota

Este comportamiento tiene por objetivo que el robot sea capaz de buscar, tras un entrenamiento adecuado, una pelota de un color conocido a priori (rosa) en un entorno controlado (un campo de fútbol en el que no existen más objetos del mismo color que la pelota), se acerque a ésta y la capture entre las piernas frontales, deteniéndose el comportamiento cuando se detecte que la pelota ha sido capturada.

Para el funcionamiento de este comportamiento el robot deberá detectar en la imagen el color de la pelota y obtener la posición estimada de ésta respecto al robot. Estos parámetros formarán la descripción del problema y se detallarán más adelante.

Si el robot no detecta en el campo de visión ninguna pelota, el robot girará hacia su izquierda hasta que la pelota entre en el campo de visión.

Por último, recordar que como el comportamiento funciona a nivel reactivo y no existe ningún modelado del entorno, tampoco existirá un posicionamiento global del robot en el entorno, siendo todos los objetos referenciados al propio robot.

3.4.2. Comportamiento de búsqueda de portería

Este comportamiento tiene por objetivo que el robot sea capaz de buscar la portería, tras el correspondiente entrenamiento, en un entorno controlado (un campo de fútbol blanco marcado por líneas negras) utilizando para estimar la posición de la portería las lineas detectadas en la imagen. De esta forma, el robot deberá dirigirse a la portería y detenerse delante de ésta. Si posteriormente el robot detecta que no se haya delante de la portería, volverá a moverse hacia ésta.

Para el funcionamiento de este comportamiento el robot deberá ser capaz de detectar las lineas en la imagen y estimar, de forma implícita, la posición de la portería respecto al robot a partir de éstas. Hay que tener en cuenta que como sólo se usan las lineas del campo de fútbol, y éste es simétrico, el robot se acercará a una portería pero no sabrá a cual. Al igual que en el caso de la

búsqueda de pelota, los parámetros de las líneas formarán la descripción del problema y se detallarán más adelante.

En el caso de que el robot no detecte ninguna linea en el campo de visión, este se detendrá por seguridad, ya que podría tener la cabeza demasiado cerca de un obstáculo que no le permitiese al robot ver nada y colisionar.

Finalmente, y al igual que en el comportamiento anterior, indicar que como el comportamiento funciona a nivel reactivo y no existe ningún modelado explícito del entorno, tampoco existirá un posicionamiento global del robot en el entorno.

3.4.3. Aprendizaje por demostración

El aprendizaje por demostración o LfD [8, 130] es una técnica de aprendizaje que permite asociar una acción a un estado y que consiste en entrenar un comportamiento específico a partir de ejemplos provistos por un usuario supervisor.

Concretamente, y como ya se ha comentado, el entrenamiento se realiza mediante control remoto del robot a través de un joystick o cualquier otro interfaz que resulte cómodo al operador. Durante este entrenamiento, el operador únicamente tiene que mover al robot según el patrón que quiera que éste realice cuando se mueva de forma autónoma. Así pues, a través del entrenamiento el CBR almacena los comandos de movimiento que realiza el usuario (la acción) asociándolos a los parámetros que describen la situación actual (el estado) de una forma sencilla e inmediata.

Hay que tener en cuenta que durante los entrenamientos el operador normalmente controla al robot desde su propio punto de vista. Sin embargo, este cambio de punto de vista (pues el robot tiene que moverse desde su propio punto de vista) no es importante siempre y cuando los objetos a tener en cuenta y que conforman el universo del robot estén en su campo de visión.

La figura 3.4 muestra un ejemplo de aplicación del LfD y las distintas fases de las que se compone. Así pues, la figura 3.4.a muestra el caso en el que el robot aún no ha sido entrenado y no sabe hacer nada, de forma que al lanzarle la persona la pelota, no sabe que hacer. En la figura 3.4.b se observa como la persona lanza la pelota y controla el robot mediante un joystick para enseñarle que tiene que hacer cuando vea la pelota. Durante este entrenamiento, los comandos del operador se están almacenando también en el CBR. Finalmente, en la figura 3.4.c se observa como al lanzar el operador la pelota, el robot ya ha aprendido que debe hacer, accediendo a la información almacenada en el CBR, capturando finalmente la pelota.

El uso de este tipo de aprendizaje tiene una serie de ventajas, derivadas la mayoría de ellas de la capacidad innata que tienen las personas para

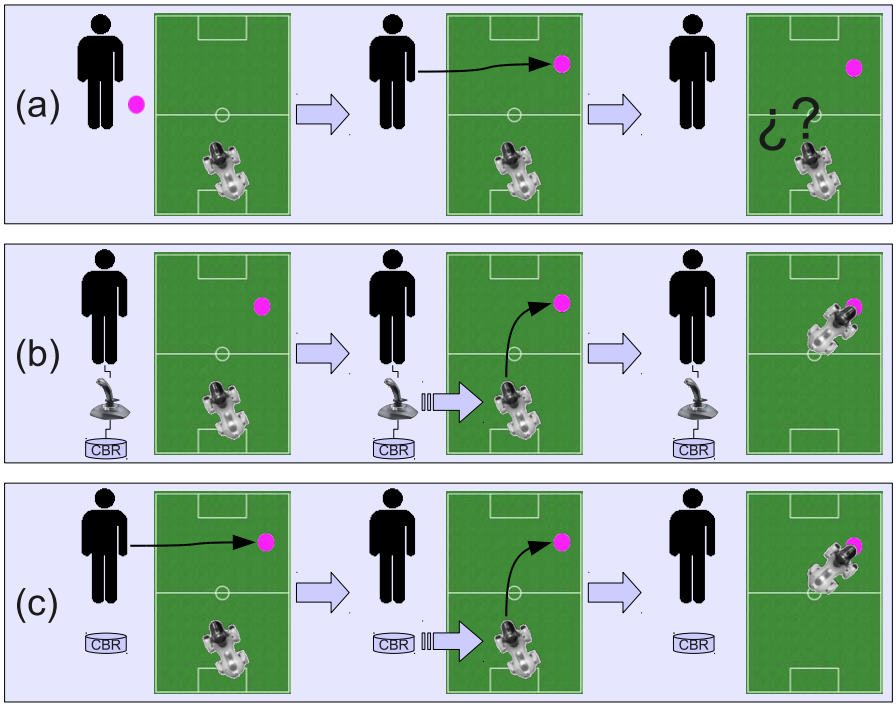

Figura 3.4: Aprendizaje por observación: a) El robot no está entrenado y no sabe que hacer; b) se entrena al robot para que busque la pelota; c) el robot recuerda como buscar la pelota.

adaptarse a distintas situaciones y entornos. Así pues, cuando el operador está entrenando al robot, este debe asimilar la estructura y movimiento del robot, de forma que el robot aprenda como se movería una persona *si tuviese la misma estructura corporal que el robot*. Esta es una idea clave en el enfoque presentado, pues sin esta capacidad de adaptación de los humanos se perderían la mayoría de ventajas del este enfoque. Así pues, a partir de esta simple idea, se consiguen las siguientes ventajas [47, 102]:

- La absorción de la dinámica y cinemática del robot. Esto sucede debido a que cuando el operador maneja el robot está asimilando de forma implícita las características propias del robot, como su forma de moverse y su velocidad. De esta forma, se evita tener que realizar un modelo cinemático del robot.

- La adaptación del esquema de navegación fácilmente a otros robots. Esta ventaja deriva del punto anterior ya que, si los parámetros del caso no son dependientes de la cinemática del robot, y dado que ésta se absorbe durante el entrenamiento por el operador es sencillo adaptar

un entrenamiento a otro robot con características distintas. En cualquier caso, si no fuese factible dicha adaptación, tan sólo sería necesario repetir el entrenamiento con el nuevo robot.

- La absorción de los errores en los parámetros de entrada y de salida. En el caso del robot usado, los parámetros son extraídos de imágenes reales capturadas por el robot, por lo que mediante el entrenamiento los errores sistemáticos de los sensores, como la distorsión de barril de la cámara, u otros errores que se puedan producir debido ruidos en la imagen o movimientos bruscos del robot son absorbidos por el operador implícitamente ya que las imágenes reales con las que se realiza el entrenamiento ya incluyen esos errores y distorsiones. Respecto a los parámetros de salida, en el caso de los robots con extremidades es fácil que los motores de las extremidades no sean exactamente iguales, lo que hace que el robot pueda tener una deriva hacia un lado, por ejemplo, o que gire más en un sentido que en otro. Este tipo de imperfecciones y errores mecánicos son absorbidos también de forma implícita al manejar el robot.

- La generación comportamientos ad hoc y la mejora de los existentes fácilmente. Para que el robot realice un comportamiento concreto sólo es necesario entrenarlo en dicho patrón de conducta, y en el caso de ampliar uno anterior habría que realizar un nuevo entrenamiento y añadirlo al ya existente, de una forma sencilla. En este último caso habría que tener especial cuidado si ante una misma situación se realizasen distintos comandos de movimiento, lo que podría provocar comportamientos indeseados.

- Evitar el uso de expresiones analíticas para la implementación del comportamiento, pues el propio entrenamiento remoto genera el algoritmo para la navegación del robot. Esto es especialmente interesante cuando se quieren realizar comportamientos que no son fáciles de implementar analíticamente pero son fáciles de entrenar.

- Una fácil adaptación del comportamiento de navegación a distintas situaciones. Gracias a la capacidad de adaptación del CBR, el sistema podría, potencialmente, adaptarse de forma sencilla a otras situaciones diferentes a las entrenadas, siempre que éstas no fuesen muy diferentes y los parámetros de entrada usados sean los suficientemente genéricos. En éste último caso, siempre se puede realizar un nuevo entrenamiento en la nueva situación y añadirlo al entrenamiento anterior.

3.4.4. Funcionamiento del sistema de navegación

El sistema de navegación mediante LfD basado en CBR supone un método simple y directo para acoplar la información visual y los comandos de movimiento a enviar al robot, implementando así un comportamiento reactivo aprendido fácilmente. Sin embargo, para que este sistema funcione de forma autónoma es necesaria una información o experiencia previa. Así pues, el sistema de navegación CBR estará formado por las siguientes fases:

- Aprendizaje supervisado. Durante la fase de aprendizaje supervisado o aprendizaje por demostración se realiza el aprendizaje de los distintos patrones de navegación que tendrá que realizar el robot en función de cada situación concreta. Para ello lo que se hace es controlar de forma remota al robot mientras se almacena en la base de casos del CBR la información con el comportamiento que se desee entrenar. Como resultado de esta fase se obtendrá la experiencia inicial o base de casos inicial que utilizará el CBR cuando trabaje de forma autónoma para obtener soluciones a partir de la experiencia previa (adquirida en esta fase).

- Navegación autónoma. Durante la fase de navegación autónoma el robot funcionará de forma autónoma sin supervisión, enviado al CBR información sobre el entorno y el estado del robot y devolviendo el CBR los comandos de movimiento previamente aprendidos que más se adecúen a la situación actual.

3.4.5. Implementación del sistema de navegación

En este apartado se describe la implementación del sistema de navegación basado en comportamientos aprendidos. Como se ha comentado, este sistema está formado por dos fases, entrenamiento supervisado y navegación autónoma, por lo que se describirán los bloques que componen el sistema en cada una de ellas.

La figura 3.5.a muestra el diagrama de bloques del sistema durante el aprendizaje supervisado. Como se puede observar, el sistema está dividido en la aplicación del robot y la aplicación del ordenador, conectadas ambas via WiFi. Así pues, la aplicación del robot envía a la del ordenador el estado del robot, así como la imagen capturada por el robot, de forma que la aplicación del ordenador extrae los parámetros de las imágenes recibidas y los une con los parámetros de estado del robot así como con los comandos de movimiento del operador para formar la estructura del caso que se almacenará en la base de casos. Simultáneamente a la creación y almacenamiento de los casos, los

comandos de movimientos del operador son enviados al robot para que se mueva por el entorno según se le indique durante el entrenamiento.

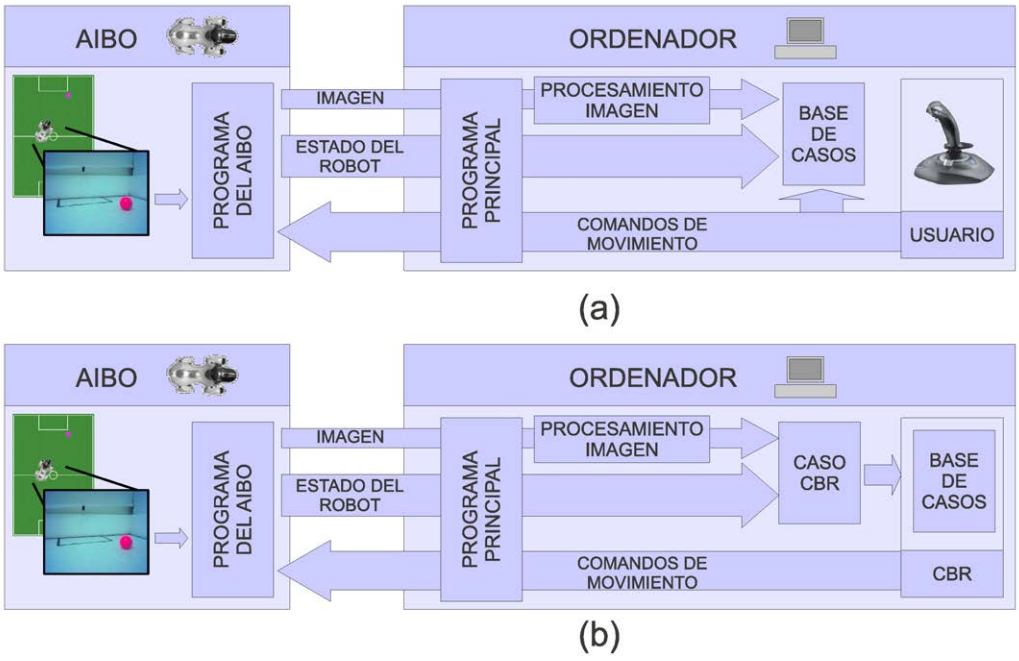

Figura 3.5: Diagrama de bloques del sistema: a) en modo entrenamiento por observación b) en modo navegación autónoma.

Por su parte, la figura 3.5.b muestra el diagrama de bloques correspondiente a la fase de navegación autónoma. Como se puede observar, y a diferencia de la fase de entrenamiento, cuando se reciben en el programa principal del ordenador el estado del robot y los parámetros de la imagen, se forma un caso CBR de entrada que se introducirá en el módulo CBR que devolverá el comando de movimiento que ejecutó el operador en la situación más similar a la descrita por los parámetros de entrada, solución que será enviada de vuelta al robot para que navegue de forma autónoma.

Como se puede observar en la figura 3.5, los bloques de los que se componen la fase de entrenamiento remoto y navegación autónoma son prácticamente los mismos. De hecho, el funcionamiento es prácticamente el mismo, con la excepción de que el envío de comandos al AIBO conmuta entre el usuario y el CBR en función del modo de funcionamiento, por lo que los módulos de los que se componen son iguales. Concretamente, estos módulos son:

- Programa del AIBO. El programa que se ejecuta en el AIBO está desarrollado en C++ y utiliza el entorno Tekkotsu (sección 3.4.5.1), el cual

permite el desarrollo del programa a alto nivel (avanzar, retroceder, girar, mover cabeza) sin necesidad de acceder a comandos de bajo nivel (mover motor 1 de la extremidad frontal izquierda).

Este programa se encarga de ejecutar en el robot los comandos de movimiento recibidos desde el programa del ordenador, así como enviar a éste las imágenes capturadas por el robot y el estado del robot. Los parámetros del estado del robot incluyen si éste está levantado o tumbado, la posición de la cabeza, si se ha detectado la pelota rosa y, si se ha detectado, su posición en la imagen y su tamaño (la detección de la pelota rosa se realiza en el AIBO gracias a la funcionalidad ofrecida por Tekkotsu). La comunicación entre el robot y el ordenador se realiza via WiFi.

- Programa principal del ordenador. Este programa se ejecuta en el ordenador y está desarrollado en Java (excepto el módulo de procesamiento de imagen). Esta aplicación esta desarrollada usando el entorno de desarrollo Tekkotsu, que facilita la comunicación con el AIBO y su control mediante un joystick, así como la realización de los interfaces gráficos para mostrar la imagen de la cámara y el estado del robot al usuario.

  El programa principal del ordenador se encarga de recibir la imagen capturada por el robot y los parámetros de estado de éste, procesando la información y dirigiéndola hacia los módulos adecuados. Así pues, en el caso del comportamiento de búsqueda de portería, la imagen es enviada al módulo de procesamiento de imagen, el cual devuelve los parámetros de las rectas detectadas en la imagen. Por su parte, en el caso del comportamiento de búsqueda de pelota, el programa del AIBO ya envía los parámetros de ésta al programa del ordenador, por lo que no es necesario aplicar ningún procesamiento a la imagen en el ordenador.

  Cuando el sistema está en modo aprendizaje, esta aplicación une los parámetros extraídos de la imagen, el estado del robot y los comandos de movimiento del usuario y los almacena en una base de casos que posteriormente usará el CBR. Por otro lado, cuando está en modo navegación autónoma esta aplicación envía los parámetros de la imagen y del estado del robot al módulo CBR, para que éste busque en la base de casos el caso más parecido y devuelva el comando de movimiento que será enviado al robot.

- Módulo de procesamiento de imagen. Este módulo es el encargado de procesar las imágenes recibidas en el ordenador y extraer los parámetros

necesarios de éstas cuando se ejecuta el comportamiento de búsqueda de portería. Este módulo está implementado en C++ y hace uso de la Librería de Código Abierto para Visión por Ordenador (OpenCV) (sección 3.4.5.2) para detectar las líneas en la imagen mediante la Transformada de Hough. Dado que la aplicación principal está desarrollada en Java, el módulo de procesamiento de imagen se ejecuta a través de Java Native Interface (JNI).

Por lo tanto, el objetivo de este módulo es aplicar la transformada de Hough para detectar todas las líneas en la imagen. Tras detectar las líneas, las filtra (pues es posible que una línea en la imagen genere múltiples líneas detectadas en la imagen) y envía finalmente a la aplicación principal las 4 líneas principales detectadas.

- Módulo CBR. El módulo CBR es un servidor CBR que permite la conexión y desconexión de clientes (vía socket) para el acceso a la base de casos. Este módulo está desarrollado en Java y es una modificación del servidor CBR desarrollado por el grupo KEMLg (Knowledge Engineering and Machine Learning Group) de la Universidad Politécnica de Cataluña (UPC).

3.4.5.1. Entorno Tekkotsu

Tekkotsu[2] es un entorno de desarrollo de código abierto creado por la universidad Carnegie Mellon que facilita la creación de aplicaciones para robots como el AIBO, usado en el presente trabajo. Para ello, este entorno de desarrollo aporta al programador una serie de herramientas y funcionalidades que permiten el desarrollo de aplicaciones de una forma relativamente sencilla. Entre las distintas funcionalidades que ofrece están, por ejemplo:

- Interfaz gráfico para ordenador para el control remoto del AIBO y la activación de comportamientos.
- Algoritmos para procesamiento de visión en el AIBO, como algoritmos segmentación de color.
- Comportamientos básicos para el AIBO que permiten al robot andar y mover la cabeza de una forma sencilla.
- Y muchas otras funcionalidades como Herramientas de depuración de errores, creación de máquinas de estado, control de eventos, temporizadores, etc.

[2] `http://tekkotsu.org/`. Última visita: 24-09-2015

Por otro lado, respecto a la estructura de este entorno, Tekkotsu está formado por dos módulos principales: el módulo que se ejecuta en el AIBO y el que se ejecuta en el ordenador.

El módulo del AIBO se programa en C++ y hace de servidor para el módulo del ordenador, permitiendo lanzar comportamientos o aplicaciones para el movimiento, procesamiento de visión, control de leds, etc.

Por su parte, el módulo del ordenador está programado en Java y permite controlar y monitorizar al AIBO mediante un interfaz gráfico, conectándose al robot como cliente. Además del interfaz gráfico, este módulo también ofrece herramientas para compilar y depurar los comportamientos a ejecutar en el AIBO.

3.4.5.2. Librería OpenCV

OpenCV[3], *Open source Computer Vision library* en inglés, es una librería de código abierto desarrollada originalmente por Intel Corporation e implementada en C y C++. OpenCV está orientada especialmente para el procesamiento de visión y tratamiento digital de imágenes en tiempo real, facilitando la tarea de programación de aplicaciones de visión al ofrecer más de 300 funciones y algoritmos para procesamiento digital de imagen y visión.

OpenCV incluye, por ejemplo, algoritmos para segmentación, detección de bordes, identificación de objetos, reconocimiento de caras y gestos, seguimiento de movimiento, etc. Entre los distintos algoritmos de los que dispone está la Transformada de Hough, que permite la detección de curvas fácilmente parametrizables (como por ejemplo rectas) en imágenes. Para ello, este algoritmo transforma cada punto de la imagen del *espacio de la imagen* a un *espacio paramétrico* que vendrá determinado por los parámetros de la curva a detectar.

Así pues, y como se ha comentado, se ha hecho uso de OpenCV para procesar las imágenes recibidas del robot y detectar las posibles rectas, asociadas a las líneas del campo, mediante la transformada de Hough.

3.4.6. Descripción del CBR

En esta sección se describe la estructura de la memoria de casos del CBR, los parámetros que forman los casos y los distintos algoritmos utilizados en cada una de las fases del CBR.

Como se ha comentado anteriormente, el CBR va a ser utilizado para realizar dos tareas distintas: la búsqueda de una pelota rosa y la búsqueda de

[3] `http://opencv.org/`. Última visita: 24-09-2015

portería a partir de las líneas en la imagen. Así pues, si no se hace referencia expresa a los parámetros que usa una u otra tarea, se debe suponer que los parámetros indicados en cada apartado serán usados en ambas tareas.

3.4.6.1. Estructura de la base de casos

La estructura de la base de casos de un sistema CBR puede ser plana o jerárquica. Como ya se ha comentado la jerárquica es más rápida en las búsquedas, aunque la inserción de nuevos casos es más compleja que en en las estructuras planas. Por su parte, la búsqueda en las estructuras planas puede ser más lenta que en las jerárquicas, especialmente si el número de casos de la base es elevado, por lo que conviene limitar el número de casos máximos en la base de casos mediante algún algoritmo de agrupación o *clustering* en inglés.

Teniendo en cuenta que las tareas a entrenar son tareas relativamente sencillas, que no es previsible que requieran una gran cantidad de casos para funcionar correctamente, y con la idea de simplificar en la medida de lo posible el sistema CBR, se ha optado por una estructura de la base de casos plana a la que se aplicará un algoritmo de agrupación en el caso de que haya que reducir el número de casos en la base para que el sistema CBR no reduzca sus prestaciones.

Para reducir el número de casos, si es necesario, se aplica un algoritmo que recorre la base de casos entera y comprueba los casos cuya entrada esté por debajo de un umbral D_{umbral}. Este umbral está determinado de forma heurística para reducir la cantidad mínima de casos de forma que su número se mantenga acotado. Respecto a la función de similitud usada para determinar si los casos están por debajo o no del umbral, había distintas posibilidades, como usar una distancia Euclídea, Tanimoto o Manhattan. Sin embargo, se ha seleccionado la distancia Manhattan en el algoritmo de agrupación porque es una distancia adecuada cuando los campos que forman el caso CBR son muy diferentes y no están relacionados entre sí, ya que esta distancia evalúa las diferencias entre la forma global de los vectores más que de los elementos de forma independiente. Además, en la práctica se comprobó que daba buenos resultados. Así pues, al aplicar este algoritmo, lo que se hace es agrupar todos los casos similares en una única clase, que será el nuevo caso usado en la base de casos en lugar de los casos originales.

3.4.6.2. Definición del caso

La representación más simple para los casos en un sistema CBR es un vector de características-valores, donde un caso puede ser considerado como

un espacio de N dimensiones. Como ya se ha comentado, un caso está formado por los pares problema-solución, que también se pueden ver como los pares entrada-salida. Por lo tanto, para definir la estructura del caso es necesario determinar los parámetros de su entrada y su salida. En la definición del caso no se contempla la posibilidad de añadir una componente que describa la eficiencia o medida de la evaluación del caso pues ésta será determinada por observación por el operador durante la navegación autónoma, considerando que los casos son correctos si el sistema es capaz de realizar la tarea para la que ha sido entrenado.

Así pues, se comenzará por la parametrización de la salida del caso, más sencilla que la de la entrada, como se verá a continuación. Dado que las dos tareas que se van a aprender mediante CBR consisten en un sistema de navegación reactiva, los parámetros que formen la salida del caso consistirán directamente en los comandos de movimiento que envíe el operador mediante el joystick al robot durante los entrenamientos. Para hacer más genéricos estos comandos, se enviarán normalizados de -1 a 1, siendo 1 la velocidad máxima permitida por el robot hacia delante/izquierda, y -1 hacia atrás/derecha. El 0 indica parada.

Respecto a la parametrización de la entrada del caso, hay que tener en cuenta que es conveniente introducir en el caso únicamente la información necesaria para describir la situación del entorno, ya que un sistema CBR trabajará mejor si se elimina toda la redundancia posible en el caso de entrada, pues las búsquedas serán más rápidas al haber menos parámetros con los que comparar. Además, si existe información redundante existe una mayor posibilidad de almacenar casos en principio diferentes, pero que realmente describirían situaciones similares, lo que podría causar comportamientos indeseados en el sistema. Así pues, es importante añadir única y exclusivamente los parámetros que resulten representativos para describir la situación, y no cualquier parámetro que describa parcialmente el entorno.

Por lo tanto, en la parametrización de la entrada hay que tener en cuenta qué parámetros son los mínimos a tener en cuenta para su definición. Dado que la tarea a aprender es la navegación, esta se puede caracterizar por un origen, un destino y los obstáculos que pueda haber en el camino. Como el sistema es reactivo y no se usa ningún modelado del entorno que permita una localización global, la posición de origen sería la del propio robot en coordenadas relativas, por lo que no hay que considerar este factor. Por otro lado, en los experimentos no se consideran obstáculos, por lo que tampoco habrá que tenerlos en cuenta para el caso, quedando únicamente los factores que describan donde se haya el destino a alcanzar de forma relativa al robot.

Por lo tanto, hay que determinar los parámetros que definen el destino

que debe alcanzar el robot en cada una de las tareas a aprender. Así pues, y dado que ambos comportamientos tienen objetivos distintos, se estudiará por separado la parametrización de entrada cada uno de los comportamientos a aprender:

- Comportamiento de búsqueda de pelota. En el caso de la tarea de búsqueda de la pelota de color conocido (rosa), el destino a alcanzar es la pelota de color rosa. Dado que un sistema reactivo puro no tiene memoria explicita, toda la información de entrada a usar por el CBR debe estar contenida dentro de la imagen actual. A partir de una imagen se pueden extraer tanto parámetros cuantitativos como cualitativos [6]. En el caso de los parámetros cuantitativos, como componentes principales, histogramas, etcétera, los parámetros cuantifican la imagen en un conjunto de números, mientras que los parámetros cualitativos tratan de encontrar información significativa en la imagen mediante el análisis de su contenido dependiendo de la aplicación a desarrollar. Este último enfoque suele dar como resultado un conjunto de características de interés dependientes de la aplicación. Dado que lo que se busca es usar la información mínima que permita describir el problema, y el CBR es dependiente del problema, las técnicas cualitativas resultan más interesante para describir el problema presente.

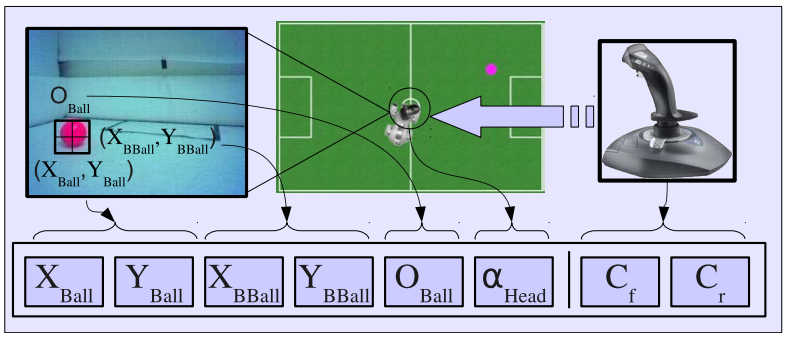

Figura 3.6: Parámetros de definición del caso para búsqueda de pelota.

Sin embargo, es importante que el conjunto de características de la imagen sean representativas, fáciles de detectar y lo más inmune posible a oclusiones y cambios en la iluminación. Dado que la pelota en la imagen se presenta como un área de color homogéneo que crece cuando más cerca esté de la cámara, se puede usar la posición del centroide de dicha área en la imagen para determinar la posición de la pelota y su tamaño para determinar su distancia a la cámara (dado que se conoce su tamaño real). Para poder conseguir cierta inmunidad contra los cambios

de iluminación, se puede realizar la detección del color en espacios como HSI o HSV, en los que el canal que indica el color es independiente del que codifica la luminosidad. Es importante también tener en cuenta las posibles oclusiones de la pelota, por lo que es conveniente añadir un indicador de oclusión como parámetro de entrada. Así pues, la figura 3.6 muestra los parámetros visuales que se tienen en cuenta para la formación del caso de entrada.

Sin embargo, hay que tener en cuenta un factor más, ya que la cabeza del robot puede no estar alineada con el cuerpo. En ese caso, se puede obtener una imagen en la que, por ejemplo, se vea la pelota enfrente del robot, cuando realmente está a la derecha o a la izquierda. Así pues, otro parámetro importante que define la situación del entorno es la orientación de la cabeza, pues define desde que punto de vista se está mirando el entorno.

Teniendo en cuenta todas estas reflexiones, se llega a la definición final del caso para la tarea de detección de pelota, en la que el caso estará determinada por la siguiente pareja $\{P, S\}$:

$$Case = \{\{X_{ball}, Y_{ball}, X_{BBball}, Y_{BBball}, O_{ball}, \alpha_{head}\}, \{C_f, C_r\}\} \quad (3.1)$$

Siendo (X_{ball}, Y_{ball}) la posición de la pelota en la imagen en píxeles, (X_{BBball}, Y_{BBball}) el tamaño de la pelota en ancho y alto en la imagen en píxeles, O_{ball} un indicador de si la pelota está ocluida o cortada, α_{head} la orientación de la cabeza respecto al cuerpo del robot en grados, y C_f y C_r los comandos de movimiento frontal y de rotación a enviar al robot normalizados entre -1 y 1.

- Comportamiento de búsqueda de portería. En el caso de la tarea de búsqueda de la portería a partir de las líneas detectadas en la imagen, el destino a alcanzar es la portería. Este comportamiento, al igual que el anterior, es reactivo, por lo que no dispone de ningún tipo de mapa del entorno y la información a usar para determinar la posición de la portería debe estar contenida en la imagen capturada.

  Para la detección de las líneas en la imagen, como ya se ha comentado, el método más usual es la aplicación de la Transformada de Hough, la cual devuelve directamente los parámetros que definen las líneas detectadas en la imagen. Esta transformada puede aplicarse a cada imagen recibida y, a partir de las líneas que ésta detecta, realizar de forma instantánea una estimación de la posición del robot y de la portería en función de

las líneas que se estén visualizando actualmente. Así por ejemplo, si se detectan las líneas de la figura 3.7.a, se puede estimar que la posición del robot es aproximadamente la indicada, al igual que si se detectan las líneas de la figura 3.7.b la posición sería aproximadamente la indicada en la figura. Hay que tener en cuenta que esta estimación de la posición no se realiza de forma explicita, sino que el operador enseña al AIBO a estimar su posición en el campo de forma implícita a partir de las líneas detectadas en la imagen.

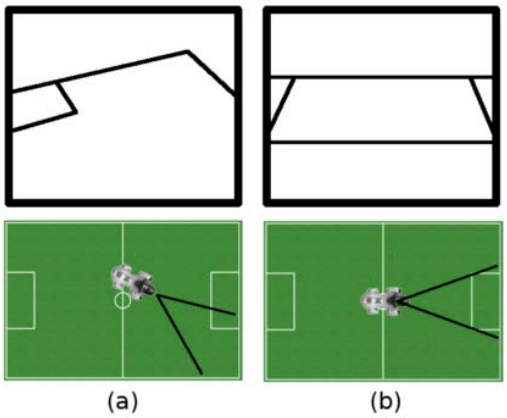

Figura 3.7: Estimación de la posición del robot a partir de las líneas.

Dado que hay que intentar usar la mínima cantidad de parámetros para la definición del caso CBR, para minimizar los posibles errores debido a la detección de las líneas, errores que ocasionan que se detecten múltiples líneas derivadas de una única línea que por el movimiento del robot se ha detectado mal, y puesto que en un campo de fútbol es difícil ver más de 4 líneas simultáneamente, se ha decidido finalmente usar sólo las 4 líneas más relevantes detectadas en la imagen. Así pues, la figura 3.8 muestra los parámetros que forman el caso de entrada para este entrenamiento.

Por último, y al igual que en el caso anterior, hay que tener en cuenta el ángulo de la cabeza, pues de éste depende el punto de vista desde el que se ve el entorno. Así pues, la definición final del caso para la tarea de detección de portería estará formada por la siguiente pareja $\{P, S\}$:

$$Case = \{\{\rho_{l1}, \theta_{l1}, \rho_{l2}, \theta_{l2}, \rho_{l3}, \theta_{l3}, \rho_{l4}, \theta_{l4}, \alpha_{head}\}, \{C_f, C_r\}\} \qquad (3.2)$$

Siendo (ρ_{ln}, θ_{ln}) los parámetros correspondientes a la recta n, α_{head} la orientación de la cabeza respecto al cuerpo del robot en grados, y C_f y

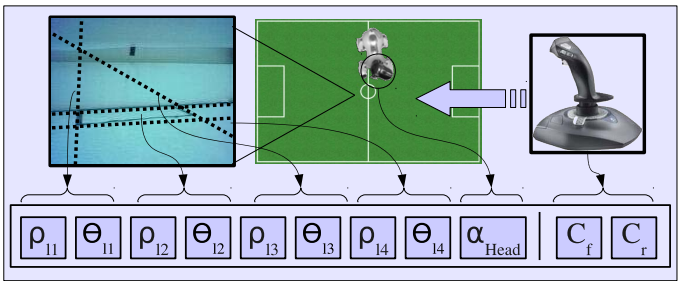

Figura 3.8: Parámetros de definición del caso para búsqueda de portería.

C_r los comandos de movimiento frontal y de rotación a enviar al robot normalizados entre -1 y 1.

Hay que tener en cuenta que en el caso de que se detecten menos de cuatro líneas, los campos correspondientes a la líneas no detectadas se rellenan a un valor nulo.

Por último, en el caso de que no se haya detectado ninguna línea en absoluto, el robot se detiene por seguridad, ya que podría tener la cabeza demasiado cerca de un obstáculo que no le permitiese al robot ver nada y colisionar.

3.4.6.3. Descripción de la fase de recuperación

Para describir la fase de recuperación usada en el CBR es necesario especificar tanto el algoritmo de búsqueda que usará esta fase como la distancia de similitud que se usará.

Respecto al algoritmo de búsqueda, se usará el algoritmo de búsqueda por vecindad, pues es un algoritmo muy utilizado cuyo único inconveniente es el tiempo de respuesta si la base de casos crece demasiado, pero como en el caso presente el número de casos se mantendrá acotado, esto no resultará un problema. Por otro lado, la función de distancia para estimar la similitud de los casos que se usará es la distancia Manhattan, pues esta distancia permite evaluar las diferencias entre la forma global de los vectores más que de los elementos de forma independiente, por lo cual resulta adecuada cuando los elementos del vector a calcular son muy diferentes y no están relacionados entre sí, como es la situación actual. Además, en la práctica se ha comprobado que ha dado buenos resultados.

3.4.6.4. Descripción de la fase de reutilización

Para describir la fase de reutilización usada en el CBR es necesario especificar el algoritmo de adaptación usado.

Dado que se han aprendido dos tareas distintas, y en cada una se utiliza una técnica de adaptación distinta, se comentará cada una por separado:

- Comportamiento de búsqueda de pelota. En el caso de esta tarea, dado que la tarea a desarrollar resulta extremadamente sencilla y en los experimentos, como se verá en el apartado 3.5, el aprendizaje por observación ha sido más que suficiente, se ha optado por usar una adaptación nula para este comportamiento.

- Comportamiento de búsqueda de portería. En esta tarea, dado que el comportamiento a desarrollar es más complejo que el anterior y es difícil poder tener en cuenta todas las posibles situaciones que el robot pueda enfrentar (ver sección 3.5 para más detalles), fue necesario añadir un algoritmo que permitiese la adaptación de los casos aprendidos durante el aprendizaje por observación, ya que la otra alternativa era la realización de un entrenamiento exhaustivo con todas las posibles situaciones que el robot puede afrontar, lo cual resultaría más complejo y tedioso.

  Así pues, para poder adaptar un caso es necesario relacionar el caso recuperado a los datos capturados del entorno. En este comportamiento se ha observado que la orientación relativa del robot en el campo (determinada por la componente θ de las líneas) tiene una mayor influencia desde el punto de vista del comportamiento que la distancia a la que se encuentra de la meta (determinada principalmente por la componente ρ de las líneas). Por tanto, la adaptación debe tener en cuenta el valor de la componente θ de las líneas detectadas en la imagen para modificar los comandos de movimiento.

  La figura 3.9 muestra un ejemplo de como afecta un giro en el robot a la posición de una línea en el dominio de Hough. La figura 3.9.a muestra un caso donde el AIBO se dirige hacia la línea formando un ángulo de 90 grados con su orientación. En la figura 3.9.b se ve el mismo caso pero con un ángulo menor. La figura 3.9.c muestra la diferencia de ángulos entre las figuras 3.9.a y b representada en el dominio de Hough. Si el AIBO está afrontando la situación en la figura 3.9.b y el caso más similar almacenado es el de la figura Fig. 3.9.a, donde el AIBO tenía que seguir hacia delante, para poder alcanzar el mismo resultado el AIBO debería girar hacia la derecha, dependiendo la magnitud de este giro

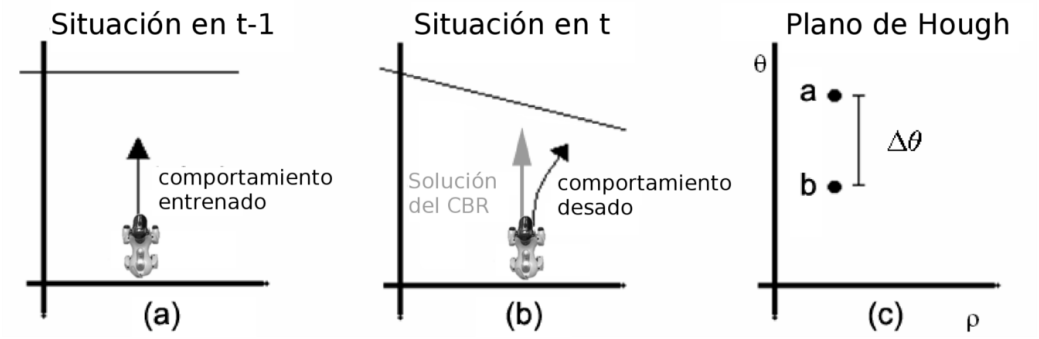

Figura 3.9: Efecto de cambios en la inclinación de una línea en el dominio de Hough.

de la diferencia en θ entre ambas líneas (figura 3.9.c). Como el comportamiento puede verse afectado por varias líneas simultáneamente, los casos son adaptados dependiendo de la media de los cambios en la componente θ de las líneas detectadas:

$$\theta_{Av} = \Sigma_i(\theta_{IN}(i) - \theta_{CBR}(i)) \tag{3.3}$$

Siendo $\theta_{IN}(i)$ el parámetro θ para la línea i en la situación actual y $\theta_{CBR}(i)$ el parámetro θ para la línea i en la salida del caso recuperado de la base de casos. Si θ_{Av} es mayor que un determinado umbral U_{adapt} (determinado heurísticamente), la salida del caso se adapta añadiéndole el factor F_{adapt} (ecuación 3.4) a la rotación propuesta por el CBR. Sino, se aplicará la rotación propuesta por el CBR directamente.

$$F_{adapt} = \frac{k \cdot \theta_{Av}}{U_{adapt}} \tag{3.4}$$

Siendo k una constante para controlar cuanto puede cambiar un caso, de forma que un bajo k produce una baja adaptación, pero movimientos más suaves, mientras que una alta k resulta en una adaptación más rápida pero movimientos más bruscos y erráticos. El valor de k se fijó heurísticamente según la respuesta del comportamiento observada en las pruebas.

La figura 3.10 muestra un ejemplo real de adaptación de un caso, en el que el robot había aprendido como moverse cuando está frente a la portería (Fig. 3.10.a) mientras que en la situación enfrentada durante la navegación autónoma (Fig. 3.10.b) está girado hacia la izquierda

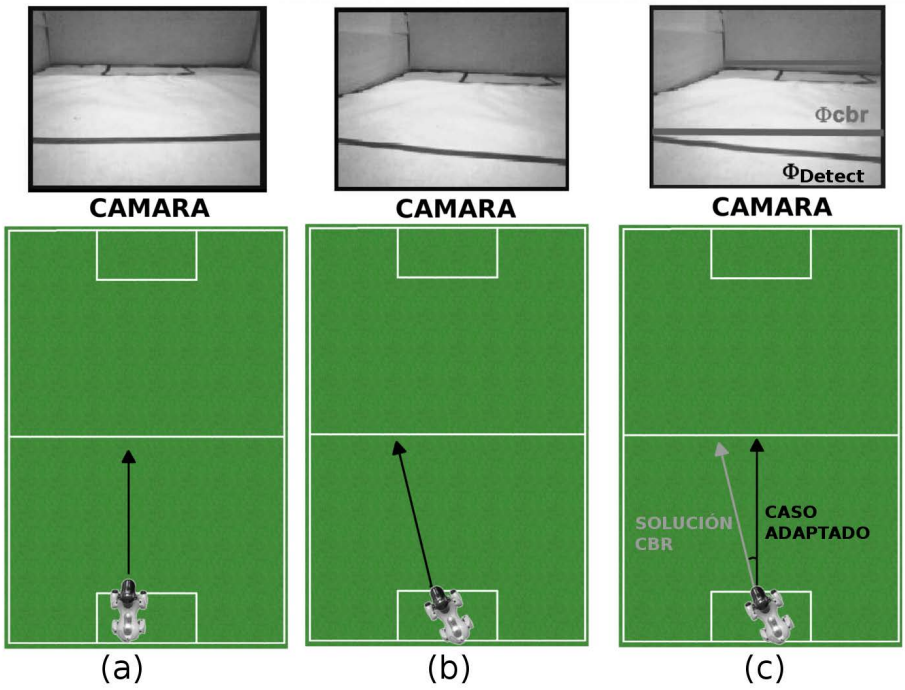

Figura 3.10: Ejemplo real de adaptación de un caso. (a) Caso almacenado en el CBR. (b) Caso enfrentado durante la navegación autónoma (c) Adaptación del caso obtenido del CBR a la situación actual.

respecto a la situación almacenada por el CBR. Si el robot se mueve hacia delante, como propone el CBR, el robot acabaría en la banda del campo, pero aplicando adaptación al caso recuperado se consigue modificar el comando de movimiento y que acabe alcanzando la portería (Fig. 3.10.c).

3.4.6.5. Descripción de la fase de revisión

Para describir la fase de revisión usada en el CBR es necesario especificar como se realiza la validación de las soluciones adaptadas y, en general, de las soluciones aportadas por el CBR.

Así pues, la evaluación o validación de los casos aportados por el CBR se hará de forma manual mediante observación, por parte de un experto, de la ejecución del sistema en modo autónomo, estudiando los ficheros de información del sistema cuando se vean situaciones anómalas o extrañas para buscar las razones y corregirlas cuando sea posible.

En cualquier caso, y de forma general, se considerará que los casos devueltos por el CBR y la adaptación de los casos son correctas siempre y cuando el robot consiga alcanzar su destino sin necesidad de intervención externa.

3.4.6.6. Descripción de la fase de retención

Para describir la fase de revisión usada en el CBR es necesario especificar los tipos de aprendizajes que se usarán en el CBR, siendo las alternativas usuales aprendizaje por observación y aprendizaje por experiencia.

Obviamente, en ambos comportamientos se usará aprendizaje por observación, pues en este aprendizaje [142] es en el que se genera la base de conocimiento que usará el CBR cuando se ejecute de forma autónoma, es decir, la base de casos inicial del CBR. Concretamente, y como ya se ha comentado, el aprendizaje por observación se realiza mediante control remoto del robot, lo cual permite asociar de forma simple y directa la información que llega al robot y que describe su entorno con los comandos de movimiento, que son la solución ante dicho entorno para alcanzar su destino.

Respecto al aprendizaje por experiencia, éste se realiza cada vez que una nueva solución es evaluada satisfactoriamente, incorporándose así a la base de casos y suponiendo un aprendizaje por experiencia del propio CBR. Es decir, este aprendizaje se realiza cuando el robot está navegando de forma autónoma, no cuando está controlado por el operador a través del joystick, y aprende como resolver nuevas situaciones mediante la adaptación de la experiencia inicial proveniente del operador. Así pues, este aprendizaje está generado por la incorporación de nuevos casos adaptados a la base de casos por lo que, como se comentó en la descripción de la fase de reutilización, su utilización dependerá del comportamiento en cuestión y del uso o no de adaptación:

- Comportamiento de búsqueda de pelota. En este caso se aplicaba adaptación nula, por lo que no se realizará ningún aprendizaje por experiencia.

- Comportamiento de búsqueda de portería. En este comportamiento, y como ya se comentó en la sección 3.4.6.4, se realiza adaptación de los casos porque sólo con el aprendizaje por observación no se llega a obtener un comportamiento adecuado del robot. Así pues, en este comportamiento sí se realiza aprendizaje por experiencia.

  Sin embargo, conviene destacar que a pesar de la adaptación y el aprendizaje por experiencia, y como se comentará en la sección 3.5.2, ha sido necesario realizar nuevos entrenamientos específicos en situaciones

concretas (aprendizaje por observación) en las que el sistema no estaba entrenado y la adaptación no era suficiente para solventar la falta de conocimiento. Así pues, este comportamiento además de aprendizaje por experiencia ha tenido varias etapas de aprendizaje por observación.

3.5. Experimentos y resultados

En esta sección se presentan los experimentos realizados para probar el correcto funcionamiento de los comportamientos comentados anteriormente. La plataforma sobre la que se han realizado las pruebas es un robot AIBO ERS7 de Sony y el entorno utilizado para los experimentos es un campo de fútbol blanco delimitado por líneas negras. Obviamente, el entorno utilizado no se adecúa a los estándares del campo en el entorno de la competición Robocup, pero el objetivo de estos experimentos no es sino probar que el aprendizaje de comportamientos mediante CBR propuesto es flexible y funcional en un entorno *similar* al de la Robocup.

Durante las pruebas el AIBO no dispone de ningún modelo del entorno, es decir, no tiene información a priori sobre el tamaño de la pelota, las dimensiones del campo ni la posición de las líneas. El entrenamiento del robot se ha realizado, como se ha comentado, mediante control remoto usando un joystick.

En este apartado se describen los experimentos realizados correspondientes al comportamiento de búsqueda de pelota, al comportamiento de búsqueda de portería y una combinación de ambos comportamientos, en la que el robot primero debe capturar la pelota para, posteriormente, dirigirse hacia la portería y marcar un gol.

Durante los experimentos se recibían las imágenes a una tasa de unos 15 fotogramas por segundo a una resolución de 216x160 píxeles. Sin embargo, debido que el ordenador usado (Pentium III a 1GHz con 512 MB de RAM) solo era capaz de procesar una de cada 4 imágenes recibidas, realmente durante los entrenamientos se almacena aproximadamente 4 casos por segundo en la base de casos del CBR. Aunque esta tasa de procesamiento pueda parecer baja, hay que tener en cuenta que la velocidad de movimiento del robot es relativamente baja, con lo que esa tasa de almacenamiento es más que suficiente.

Antes de pasar a los experimentos, conviene recordar que éstos no son más que una prueba de concepto para comprobar el enfoque propuesto basado en comportamientos aprendidos. Así pues, las tareas a realizar no son especialmente complejas y no contemplan, todavía, a otros robots ni obstáculos en el entorno, lo cual complicaría tanto la definición del caso como el entrena-

miento. Esto se tendrá en consideración en próximos capítulos.

3.5.1. Comportamiento de búsqueda de pelota

Este comportamiento tiene como objetivo que el robot sea capaz de buscar y capturar una pelota situada en cualquier punto del campo de juego.

Aunque se trate de un comportamiento muy simple, pero es el comportamiento básico con el que se han comenzado a realizar aprendizajes con CBR y es mejor partir de un ejemplo sencillo y fácilmente comprensible. En los siguientes apartados se detallan las secuencias de entrenamiento realizadas para el aprendizaje por observación de este comportamiento y posteriormente los resultados obtenidos.

3.5.1.1. Entrenamiento del comportamiento

Para el aprendizaje por observación se han realizado tres secuencias de entrenamiento distintas, que son suficientes para que este comportamiento se pueda realizar satisfactoriamente.

La figura 3.11 muestra las secuencias de entrenamiento realizadas para este comportamiento. En la figura 3.11.a el AIBO está orientado hacia la pelota a una determinada distancia y aprende como acercarse a ésta hasta alcanzarla. En la figura 3.11.b y c la pelota está inicialmente situada a la izquierda y derecha del robot respectivamente, aprendiendo el robot en cada caso como orientarse hacia la pelota, momento en el cual entraría en funcionamiento el primer entrenamiento realizado para acercarse a capturar la pelota. Así pues, la primera secuencia de entrenamiento es usada para aprender como acercarse a la pelota cuando ya está orientado hacia ésta, mientras que en las dos últimas el robot aprende a girar hasta estar orientado hacia la pelota.

El número de casos final de este entrenamiento es de 60 casos, que a una tasa aproximada de 4 casos por segundo supone un entrenamiento de unos 15 segundos para obtener este comportamiento.

Respecto al entrenamiento realizado, debe tenerse en cuenta que la base de casos del CBR sólo almacena casos si el AIBO detecta la pelota en la imagen y el robot está moviéndose, descartándose los patrones en los que está parado o no se detecte la pelota.

Además, hay que destacar que en la base de casos de este entrenamiento se añade un patrón CBR especial, definido a priori, que hace girar al robot hacia la izquierda cuando no se detecta ninguna pelota en la imagen (patrón de entrada nulo). De esta forma, si el robot no encuentra la pelota en el campo de visión, comenzará a dar vueltas hasta que la encuentre.

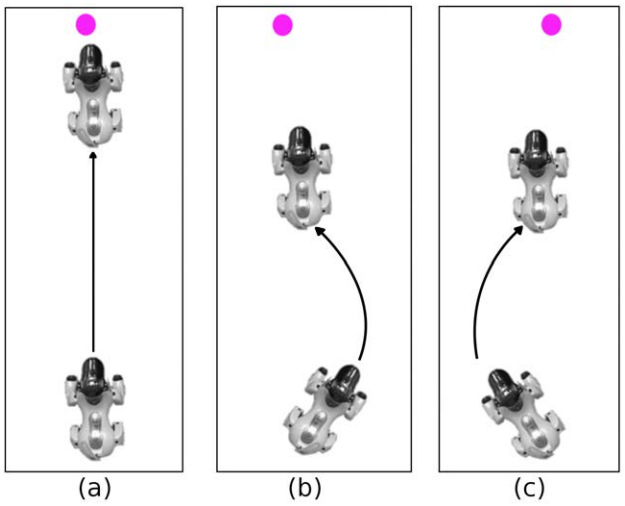

Figura 3.11: Secuencias de entrenamiento para el módulo de seguimiento de pelota mediante CBR

3.5.1.2. Navegación autónoma

En este apartado se muestran las pruebas realizadas con el comportamiento de búsqueda de pelota usando la información obtenida en el entrenamiento del apartado anterior, en los cuales se comprobó experimentalmente que el AIBO era capaz de seguir correctamente la pelota incluso en situaciones en las que no había sido específicamente entrenado (Fig. 3.12).

En la figura 3.12 también se puede ver como, cuando el AIBO no tiene la pelota dentro de su campo de visión, gira hacia la izquierda hasta que la encuentra, haciendo uso de ese caso CBR añadido específicamente para lidiar con este tipo de situaciones.

Durante estos experimentos también se comprobó que errores puntuales en el sistema como, por ejemplo, el cambio de iluminación producido por el encendido de las luces o la apertura de una persiana son fácilmente filtrados por la propia naturaleza del sistema reactivo, ya que la siguiente imagen no errónea que se obtenga será usada para conseguir una respuesta del CBR correcta y la inercia del sistema y la baja velocidad de movimiento del robot evitará que se tenga demasiado en cuenta las salidas erróneas, a no ser que éstas sean sistemáticas o duraderas.

Dada la naturaleza reactiva del sistema de navegación, esto es válido tanto si la pelota está estática como si está en movimiento. Así pues, la figura 3.13.a muestra como el AIBO pierde de vista la pelota cuando se acerca a ella debido a que esta se ha movido hacia otra posición, y aún así el robot es capaz de encontrarla de nuevo y seguirla hasta alcanzarla. En el ejemplo de la figura

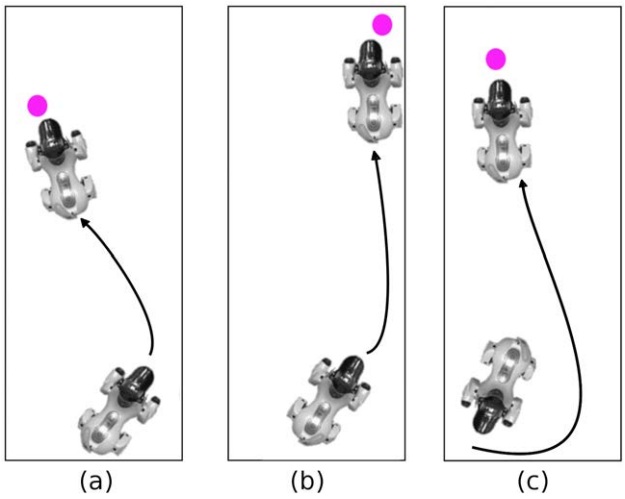

Figura 3.12: Pruebas de navegación reactiva basada en CBR con una pelota estática.

3.13.b se muestra como el AIBO es capaz de seguir la pelota sin perderla ya que en esta ocasión la pelota se ha movido más lentamente, de forma que el AIBO ha conseguido mantenerla dentro de su área de visión.

3.5.2. Comportamiento de búsqueda de portería

Tras los resultados obtenidos en el comportamiento de búsqueda de pelota, en este apartado se detallan las pruebas realizadas con el comportamiento aprendido de búsqueda de portería a partir de las líneas detectadas en la imagen.

En primer lugar, y al igual que en el caso anterior, se indicará como se desarrolló la fase de entrenamiento inicial (aprendizaje por observación) para crear la base de conocimiento básica para que el robot pueda trabajar de forma autónoma, pasando posteriormente a las pruebas realizadas.

3.5.2.1. Entrenamiento del comportamiento

Para el entrenamiento de este comportamiento se realizó una primera fase de entrenamientos formada por 3 secuencias, al igual que en el caso del comportamiento de búsqueda de pelota, que incluían las situaciones más probables con las que se podía enfrentar el robot (Fig. 3.14). Como se puede observar en la figura 3.14.a, el robot es entrenado, estando frente a la portería, para acercarse a ésta, mientras en las figuras 3.14.b y c se le enseña cómo orientarse hacia la portería cuando no lo está, momento en el que, al igual

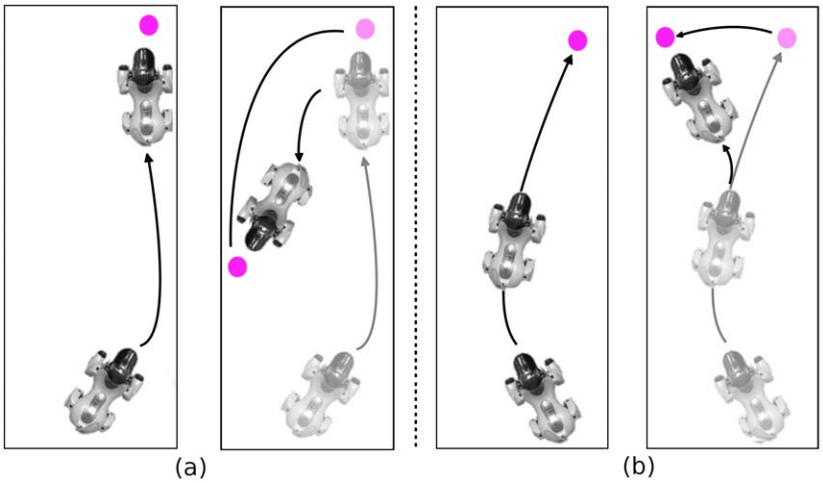

Figura 3.13: Pruebas de navegación reactiva basada en CBR con una pelota móvil.

que en el comportamiento de búsqueda de pelota, entrará en funcionamiento el primer entrenamiento realizado.

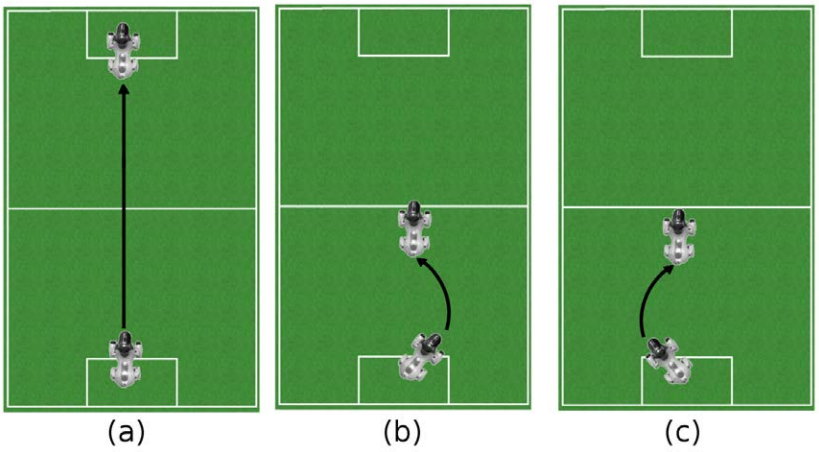

Figura 3.14: Secuencias de entrenamiento para el comportamiento de búsqueda de portería usando CBR

Estas secuencias de entrenamiento se han realizado con la idea de que el robot sea capaz de recolectar suficiente información como para aprender a moverse hacia la portería intentando estar lo más centrado posible en el campo de fútbol desde la mayoría de situaciones posibles.

Tras finalizar las secuencias de entrenamiento indicadas, la base de casos resultante estaba formada por 120 casos, que a un ritmo de 4 casos por se-

gundo, corresponde aproximadamente a unos 30 segundos de entrenamiento, que no es un entrenamiento muy largo a pesar de ser un comportamiento más complejo que el anterior.

En este comportamiento, al igual que en el de búsqueda de pelota, la base de casos del CBR sólo almacena casos si el AIBO detecta alguna linea en la imagen y el robot está moviéndose. Igualmente, en este comportamiento también se añade a la base de casos un caso especial que corresponde al caso en el que no se detecte ninguna línea y que detiene el robot por seguridad en caso de que esté demasiado próximo a un obstáculo, que podría ser la razón por la que no detecta ninguna línea.

Sin embargo, como se comenta en la siguiente sección, al realizar pruebas se comprobó que este entrenamiento básico no fue suficiente para cubrir todas las situaciones que el robot debía afrontar, y se tuvo que añadir un nuevo entrenamiento específico para esquinas a este entrenamiento inicial. Este nuevo entrenamiento sumó unos 60 casos (aproximadamente 15 segundos de entrenamiento) a la base de casos anterior.

3.5.2.2. Navegación autónoma

Una vez finalizado el entrenamiento, se realizaron distintas pruebas para comprobar el correcto funcionamiento del comportamiento. La figura 3.15 muestra el funcionamiento del sistema CBR sin adaptación (para comprobar el funcionamiento del sistema antes de usar adaptación), y se puede ver como el AIBO es capaz de alcanzar la portería incluso desde posiciones en las que no ha sido entrenado.

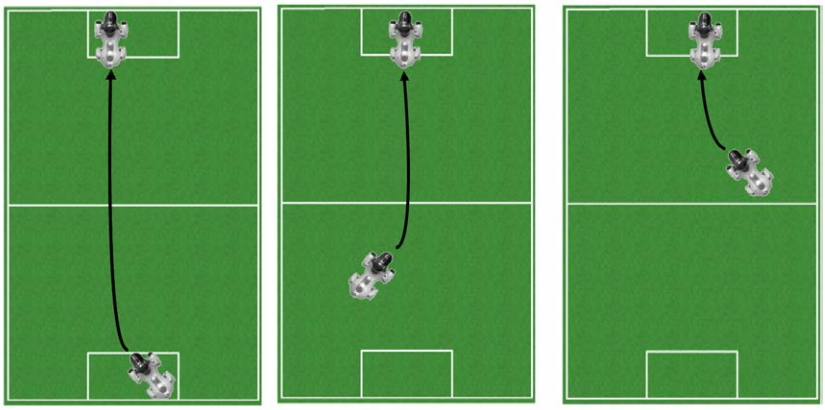

Figura 3.15: Búsqueda de portería reactiva usando CBR sin adaptación

Sin embargo, en otras pruebas de este comportamiento se comprobó que había situaciones desde las que no era capaz de alcanzar la portería (Fig.

3.16). Así pues, en la figura 3.16.a se observa como el AIBO detecta las líneas de banda de campo y de medio campo, y comienza a avanzar suponiendo que está centrado en el campo (como en la primera secuencia del entrenamiento realizado), cuando en verdad está en un lateral, no consiguiendo alcanzar la portería. En la figura 3.16.b, el AIBO inicialmente comienza a girar hacia la portería, rotando hacia la izquierda para alcanzarla. Sin embargo en un momento dado detecta las líneas de la esquina de la portería y vuelve a girar a la derecha nuevamente, finalizando en la esquina del campo. Por último, la figura 3.16.c muestra un ejemplo similar al de la figura 3.16.a con la salvedad de que al principio el AIBO gira hacia la derecha cuando detecta la línea de medio campo, orientándose hacia la portería, aunque no de forma correcta.

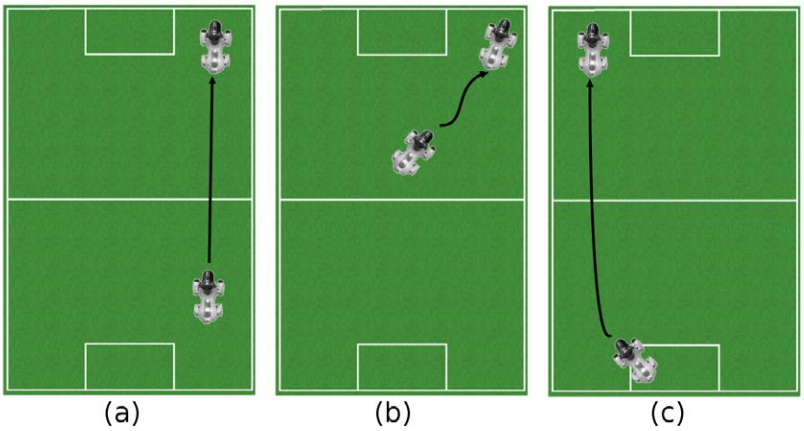

Figura 3.16: Errores buscando búsqueda de portería sin adaptación

Para solucionar estos problemas y mejorar el comportamiento entrenado, y dado que en las primeras pruebas no se habilitó la opción de adaptación para ver el funcionamiento del sistema sin ésta, se habilitó la adaptación en el CBR (permitiendo el aprendizaje por experiencia) para ver si la adaptación conseguía solucionar los problemas obtenidos y afrontar nuevas situaciones no entrenadas previamente.

Como se puede observar en la figura 3.17, tras habilita la adaptación en el CBR se han conseguidos solucionar 2 de las situaciones en las que el sistema fallaba. Sin embargo, sigue quedando una situación en la que el sistema basado en CBR no es capaz de alcanzar la portería a pesar de la adaptación de los casos (Fig. 3.17.b). Sin embargo, una de las ventajas del CBR es su habilidad para solucionar problemas puntuales, como éste, mediante nuevo aprendizaje por observación, específico para esa situación, que se añadirá al aprendizaje ya realizado.

La figura 3.18 muestra las secuencias de entrenamiento realizadas específi-

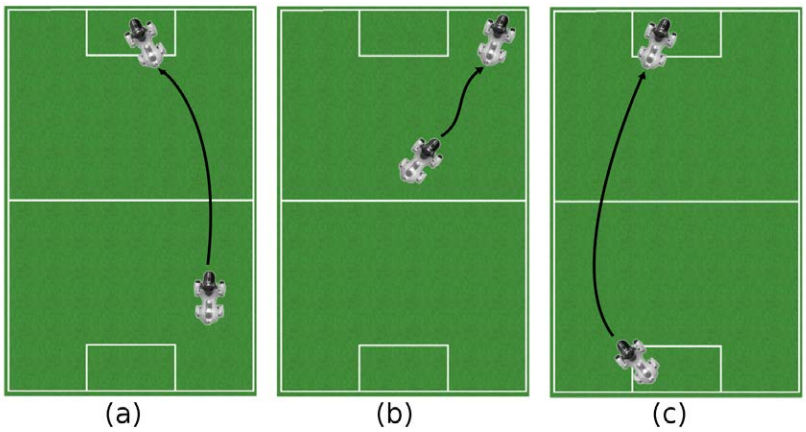

Figura 3.17: Búsqueda de portería reactiva usando CBR con adaptación

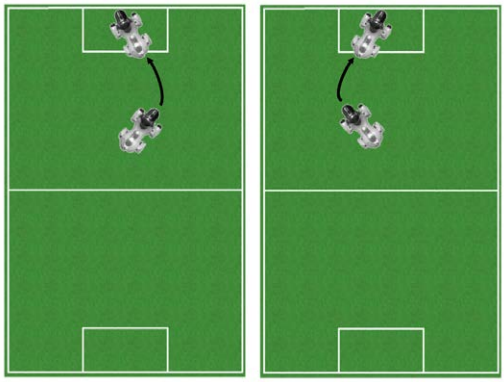

Figura 3.18: Nuevas secuencias de entrenamiento específicas para esquinas

camente para que el robot fuese capaz de alcanzar la portería desde esa posición. Como se puede ver, son dos secuencias en las que se enseña al AIBO a dirigirse a la portería estando orientado hacia las esquinas del campo. Los casos resultantes de este entrenamiento son añadidos a la base de casos ya existente, obteniéndose el resultado de la figura 3.19.

Como se puede observar, al añadir el entrenamiento específico de esquinas, y habilitar la adaptación de casos en el CBR, el comportamiento de búsqueda de portería basado en líneas es capaz de funcionar desde las posiciones del campo en las que antes no era capaz de alcanzar la portería sin mayores problemas.

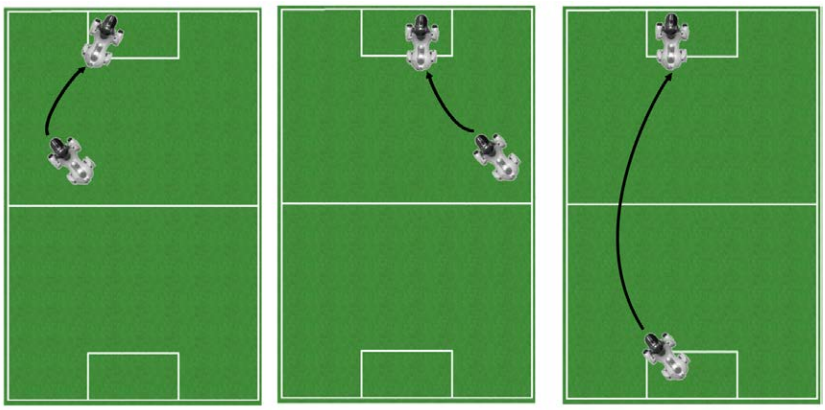

Figura 3.19: Prueba final de búsqueda de portería reactiva usando CBR con adaptación

3.5.3. Comportamiento global: búsqueda de pelota y portería

Finalmente, tras la comprobación del correcto funcionamiento de los comportamientos de búsqueda de pelota y búsqueda de portería se han enlazado secuencialmente ambos comportamientos para conseguir un comportamiento global que permita capturar la pelota y, posteriormente, llevarla hasta la portería. De hecho, la conmutación de un comportamiento a otro depende de si el robot ha capturado o no la pelota, de forma que si el robot ha capturado la pelota activa el comportamiento de búsqueda de portería, mientras que si no la ha capturado activa el comportamiento de búsqueda de pelota. Así pues, esta tarea permite mostrar como dos comportamientos simples pueden ser unidos para desarrollar una tarea más compleja de un modo sencillo.

Para ello, en primer lugar el AIBO activa el comportamiento de seguir pelota en el campo hasta que la captura, detectando la pelota capturada gracias al sensor pectoral del robot. En ese instante, se activa el comportamiento de buscar portería, con lo que el robot lleva la pelota capturada hacia la portería, donde finalmente marca gol. Si tras capturar la pelota y activar el comportamiento de búsqueda de portería el robot detecta que ha perdido la pelota (la cual puede escaparse del AIBO por los propios movimientos del robot o bien por las imperfecciones del campo montado), se volvería a activar el comportamiento de búsqueda de pelota, comenzando de nuevo la secuencia.

La figura 3.20 muestra una secuencia de imágenes del comportamiento global en funcionamiento. En primer lugar, el AIBO ejecuta el comportamiento de seguimiento de pelota. Como no detecta la pelota, gira hacia la

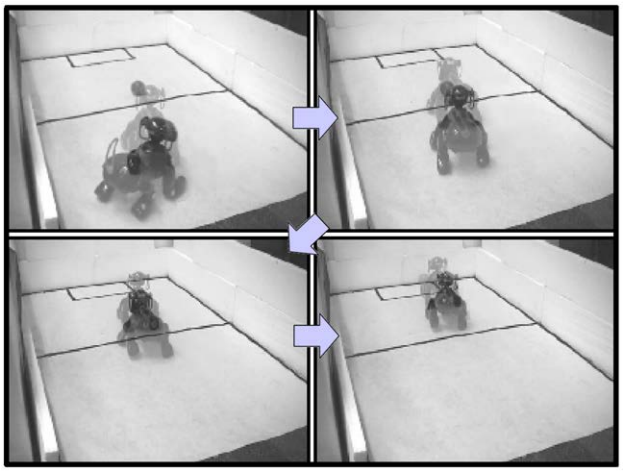

Figura 3.20: Imágenes reales del comportamiento global (prueba 1).

izquierda hasta que ésta entra dentro del campo de visión y entonces lanza el comportamiento de seguimiento de pelota para capturarla. En ese instante entra en acción el comportamiento de búsqueda de portería, que dirige al AIBO hacia la portería con la pelota capturada. Finalmente, cuando el AIBO detecta que está suficientemente cerca de la portería, chuta la pelota y marca un gol.

En la figura 3.21 se muestra otra prueba con el mismo comportamiento global, en este caso mostrando una visión cenital. En esta prueba, al igual que en el caso anterior, el robot consigue capturar la pelota, alcanzar la portería y chutar a gol sin mayores problemas.

3.6. Conclusiones

En este capítulo se ha presentado una técnica que permite la construcción de comportamientos para navegación reactiva basándose en aprendizaje LfD usando CBR. Concretamente, el sistema de aprendizaje consiste en que una persona, que hace de supervisor, realice el entrenamiento de un robot, controlado remotamente mediante un joystick, enseñando así al robot el comportamiento que se quiere realizar. De esta forma se puede relacionar de un modo sencillo y directo lo que percibe el robot (en el caso del robot usado mediante visión a través de la cámara integrada) y los comandos de movimiento que el operador está enviado al robot.

Una de las ventajas de este enfoque es que la cinemática y dinámica del robot así como los errores de calibración y los errores sistemáticos son

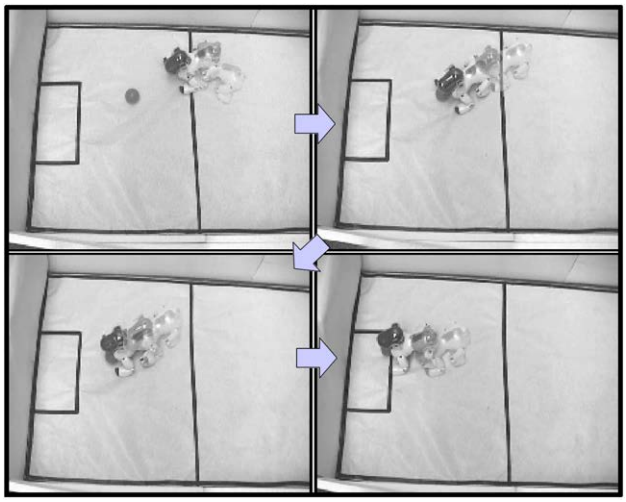

Figura 3.21: Imágenes reales del comportamiento global (prueba 2).

implícitamente tomados en cuenta por el supervisor, el cual los compensa naturalmente durante el entrenamiento. Por otro lado, este enfoque permite el desarrollo de comportamientos *ad hoc* de una forma sencilla sin tener que realizar ningún modelado del entorno. Además, y debido a la naturaleza reactiva del comportamiento, los errores puntuales no son, en principio, un problema ya que generalmente son absorbidos por la propia inercia del sistema.

Sin embargo, hay que tener en cuenta también que este enfoque tiene una considerable dependencia con la definición de los casos del CBR, por lo que es conveniente prestar especial atención a esta fase y escoger una definición de caso eficiente, ya que la información que contengan los parámetros que forman el caso no debería estar correlada entre ellos ni repetida. También hay que tener en cuenta que es recomendable que los parámetros usados en el caso sean fiables y fáciles de extraer de los sensores de entrada.

Es importante también tener presente que los entrenamientos realizados deben ser lo más genéricos posible y cubrir la máxima cantidad de situaciones pues, como se ha podido comprobar en este capítulo, la falta de entrenamiento básico en situaciones importantes puede ser suplido parcialmente por la adaptación y el aprendizaje por experiencia, pero no todas las situaciones pueden resolverse mediante adaptación, por lo que hay que hacer un estudio de las posibles situaciones que deberán enfrentarse y tratar de entrenarlas todas para conseguir una mejor respuesta del sistema.

Para comprobar la viabilidad de este enfoque se ha propuesto una prueba de concepto consistente en el aprendizaje de dos comportamientos sim-

ples que se enlazan secuencialmente para realizar una tarea más compleja. Los comportamientos implementados son sencillos, sin embargo, cumplen su objetivo, que no es otro que demostrar que se pueden desarrollar distintos comportamientos reactivos que permitan la navegación de un robot mediante aprendizaje, sin necesidad de modificar el sistema. De hecho, para generar distintos comportamientos tan sólo sería necesario determinar los parámetros del caso que definen tanto la situación actual como su solución y entrenar al robot para que realice el comportamiento deseado.

Los experimentos se han realizado con un robot AIBO ERS-7 con una cámara integrada en un entorno en el que únicamente hay una pelota rosa y un campo de fútbol de reducidas dimensiones delimitado por líneas. Destacar que en estas pruebas no se han tenido en cuenta más robots de momento, pues la coordinación con otros robots será introducida en próximos capítulos. Así pues, en los experimentos se ha mostrado como se ha conseguido un comportamiento complejo a partir de la secuencia de dos comportamientos aprendidos más simples. Concretamente, el comportamiento consistía en conseguir que un robot AIBO capturase una pelota y la llevase hasta la portería. Para ello se ha dividido este comportamiento en dos más sencillos: búsqueda de pelota y búsqueda de portería. La activación y desactivación de estos módulos depende de si el robot ha capturado o no la pelota.

Ambos comportamientos han sido aprendidos mediante LfD usando CBR, siendo las componentes del caso CBR en el primer comportamiento las coordenadas de la pelota en la imagen, su tamaño y la posición de la cabeza respecto al cuerpo, y en el segundo comportamiento las cuatro líneas dominantes detectadas en el campo de visión, definidas mediante sus parámetros en el dominio de Hough, y el ángulo de la cabeza del robot respecto al cuerpo. La salida del caso CBR en ambos comportamientos es el comando de movimiento frontal y de rotación normalizado que debe realizar el robot para alcanzar su objetivo, sea el que sea según el comportamiento. Durante el entrenamiento, se generará la base de casos CBR que almacena los comandos de movimiento realizados por el operador asociados a la situación concreta en la que los aplicó. Como se ha comentado antes, el entrenamiento de los comportamientos se ha realizado controlando remotamente el robot mediante un joystick, lo que permite definir fácilmente las trayectorias que se desee que el robot aprenda. Tras finalizar el entrenamiento, el robot dispone, a través de la base de casos CBR, de la información necesaria para poder funcionar de forma autónoma, obteniendo para ello, de la base de casos, los comandos de movimiento a realizar en cada situación.

Hay que indicar que la adaptación en el CBR es una técnica que permite el aprendizaje automático a partir de nuevas situaciones. Sin embargo, tam-

bién hay que ser conscientes de que la base de conocimientos inicial es muy importante y que si esta no cubre las situaciones más importantes que debe enfrentar el robot, la adaptación puede no ser suficiente para que el sistema afronte nuevas situaciones satisfactoriamente, como se ha comprobado en los experimentos.

Así pues, en las pruebas realizadas se ha comprobado que el planteamiento propuesto resulta adecuado para entrenar comportamientos simples con un único robot. Sin embargo, y dado que el objetivo final de este trabajo es la realización de un sistema de coordinación multi-agente basado en comportamientos aprendidos, este enfoque no se considera apropiado para el sistema que se pretende realizar. La razón es que en los comportamientos mostrados en este capítulo no se ha usado ningún tipo de localización explícita, sino que la localización se realiza de forma implícita en el mismo aprendizaje del comportamiento. Es decir, en ningún momento se conoce directamente la posición del robot respecto a ningún elemento o punto referencia concreto, pues el objetivo de los comportamientos es obtener el movimiento a realizar para alcanzar el destino en función de la situación actual, descrita por lo que el robot visualiza en ese mismo instante.

El problema es que, si se usa este enfoque basado en localización implícita para el sistema coordinado a desarrollar, y no se permite la comunicación entre los robots, las posibilidades de coordinación se limitarían a cuando éstos se viesen directamente, pudiendo evitarse como obstáculos, o bien a realizar tareas conjuntas para empujar o alcanzar una meta común en la que tengan que colaborar, pero sería difícil la realización de otro tipo de comportamientos coordinados más sofisticados. Sin embargo, si se permitiese la comunicación entre los robots, el problema sería que la definición del caso resultaría demasiado compleja al tener que añadir en la descripción de la situación actual todos los parámetros (líneas de campo o referencias visuales) de los demás robots para tener en cuenta la posición propia y la de los otros robots implícitamente en el entrenamiento. Sin embargo, esto complicaría enormemente la definición del caso, el entrenamiento y la búsqueda dentro de la base de casos, por lo que no sería una solución viable.

Así pues, se considera que la mejor solución a este problema sería disponer de un sistema de localización explícita que permita obtener la posición de los demás robots respecto a algún punto de referencia de forma expresa para permitir comportamientos coordinados más complejos, lo cual se tratará en los siguientes capítulos.

Capítulo 4

Sistema de localización

4.1. Introducción

Los MRS han sido aplicados en la última década en una gran variedad de áreas así como a multitud de tareas. De hecho, los MRS han sido aplicados en una gran variedad de escenarios [156], como los robots auto-organizados [11, 92], la coordinación de robots mediante modelos biológicos [143], tales como los enjambres de robots [80,85], vigilancia y exploración [34], tareas militares [63, 95], operaciones de rescate [87], entornos industriales [127], ambientes hostiles o peligrosos [42], etc.

La principal ventaja de los MRS es que permiten la realización de una determinada tarea más rápido y más eficientemente [9, 52, 152, 156] que un sólo robot, ya que los robots del MRS pueden estar en distintos sitios al mismo tiempo, pueden desarrollar tareas concurrentes y cooperativas y, en general, pueden descomponer una tarea compleja en otras más simples.

Sin embargo, no todo son ventajas en los MRS, ya que en estos sistemas la trayectoria de cada uno de los agentes que los componen no está definida únicamente por su posición con respecto al objetivo, sino también por su posición relativa con respecto a los demás agentes del sistema, ya que los demás agentes actúan como obstáculos, a pesar de que puedan estar cooperando en la resolución de la misma tarea. Así, por ejemplo, en entornos dinámicos con elementos en movimiento, tareas como la navegación, localización y actualización de los mapas del entorno resulta más compleja e importante en los MRS que en sistemas robóticos individuales. Por lo tanto, para la correcta navegación de los distintos agentes de un sistema MRS y para evitar colisiones y el mejor desempeño posible, es crucial una correcta y rápida localización de los miembros del sistema.

Sin embargo, el uso de una localización que permita conocer la posición de

otros robots respecto a la de uno mismo implica la necesidad de un punto de referencia en el entorno que permita relacionar el sistema de referencia local del robot con uno externo común que sirva de nexo con el resto de robots, lo que requiere un modelado del entorno. Por lo tanto, el uso de un sistema de localización implica que el sistema no sería puramente reactivo, pues un sistema reactivo supone no usar un modelado del entorno [147]. Sin embargo, dando por hecho que al usar localización el sistema no va a ser reactivo puro, hay sistemas de localización que son más *reactivos* que otros, en el sentido de que requerirán un modelado del entorno más o menos estricto, así como la posibilidad de requerir o no el uso de componentes temporales como una memoria. Así pues, se tratará de encontrar un sistema de localización lo más reactivo posible para el sistema de coordinación a desarrollar.

Como se ha comentado, en las pruebas del sistema coordinado a desarrollar se utilizarán robots AIBO ERS-7 de Sony en un entorno de trabajo similar al entorno de la liga de fútbol robótica de la Robocup. La localización en este entorno suele realizarse mediante visión, el principal sensor de estos robots, usando marcas artificiales de colores situadas en posiciones concretas del campo de la Robocup [37, 108, 110, 140]. Además, en este entorno es normal que se haga uso de filtros de Kalman [41, 89] o filtros de partículas [108, 110, 151] para mejorar la precisión y la estabilidad de la posición obtenida.

Sin embargo, este tipo de localización, a pesar de ser ampliamente usada, no encaja con el enfoque reactivo propuesto en el capítulo anterior debido al estricto modelado del entorno y el uso de una memoria (historial de observaciones de los filtros). Una alternativa más reactiva sería evitar el uso de una componente temporal (usando una localización instantánea o inmediata) y reducir las restricciones de modelado del entorno al mínimo. Así pues, una opción que encajaría mejor con un sistema reactivo sería que los robots obtuviesen la posición de los compañeros cuando se detecten directamente. Sin embargo, esto limitaría las posibilidades de coordinación únicamente a momentos puntuales o a comportamientos similares a los de los enjambres de robots en los que todos tienen una meta en común, como empujar un objeto. Otra alternativa sería el uso de un sistema de localización basado en objetos comunes a partir del cual puedan localizarse. La opción más reactiva de este enfoque sería que los robots se localizasen respecto a cualquier objeto en común, minimizando el modelado del entorno. Sin embargo, esto sería demasiado complejo, por lo que generalmente suelen usarse objetos fáciles de detectar, como objetos de colores específicos. Otra alternativa sería el uso de marcas fiduciarias, que si bien suponen un modelado más estricto del entorno (al usar marcas específicas) pero pueden permitir el posiciona-

miento instantáneo de los robots a partir de la imagen actual (sin el uso de una componente temporal), resultando un enfoque más reactivo que el usado comúnmente en la Robocup. Así pues, en este capítulo se estudiarán distintas alternativas de localización, determinando cuáles de ellas son factibles para su uso en el sistema coordinado a desarrollar manteniendo, en la medida de lo posible, el enfoque reactivo de éste.

El capítulo se estructura de la siguiente forma. En primer lugar, en la sección 4.2 se hace una breve introducción a los distintos tipos de localización, comentando sus ventajas y desventajas. Tras esta introducción, se indicarán las técnicas de localización que se tendrán en cuenta para el sistema de coordinación propuesto (sección 4.3). Posteriormente, se hará una prueba preliminar de las técnicas de localización consideradas para comprobar su adecuación y viabilidad (sección 4.4). Concretamente para probar cada uno de los sistemas de localización se realizará una prueba de navegación segura entre dos robots AIBO ERS7 usando navegación basada en PFA. Por último en la sección 4.5 se presentan las conclusiones del capítulo indicando el sistema de localización que se estima más adecuado.

4.2. Tipos de localización

Conocer la posición y orientación en un entorno es una información fundamental que todo robot debe ser capaz de calcular para moverse e interactuar con su entorno, permitiendo que éste realice las tareas que tenga asignadas [20].

Dado que es una tarea fundamental para la robótica móvil, ha sido un área muy activa [20, 32, 140] desde el comienzo de la robótica, existiendo una gran variedad de técnicas y métodos para la estima y cálculo de la posición.

Así por ejemplo, las técnicas de localización variarán en función de la información disponible del entorno (utilización de marcas naturales [150] o artificiales [117]), de los sensores disponibles (láser [133], sonar [89], visión [151] o una combinación de distintos sensores [45]) o de la plataforma robótica (con ruedas [138] o con extremidades [48, 100]).

En cualquier caso, la localización se puede dividir en dos categorías:

- Localización según el tipo de posicionamiento. En esta categoría el posicionamiento puede ser relativo o absoluto, siendo en la localización relativa (también referenciada en ocasiones como seguimiento) el posicionamiento del robot en todo momento relativo a la posición inicial de éste, mientras que en el segundo caso la posición es absoluta respecto a algún mapa o modelado del entorno.

- Localización según el modelado del entorno. En este caso se dividen las técnicas de localización según si se dispone de un modelado o mapa a priori del entorno, si el mapa se construye antes de realizar la localización o si no existe ningún tipo de mapa del entorno.

A pesar de esta clasificación en dos grandes grupos, estos no son excluyentes, pues de hecho es normal usar localización relativa junto con localización absoluta con mapa a priori (por ejemplo) para corregir el primer tipo de localización. A continuación se cuentan con más detalle cada una de estas técnicas.

4.2.1. Localización según el tipo de posicionamiento

Una de las clasificaciones más usuales cuando se trata de localización en robótica móvil es la relativa al tipo de posicionamiento [22], el cual puede ser relativo o absoluto.

En el primer caso el posicionamiento del robot se hace de forma relativa a la posición de inicio, mientras que en el segundo se hace de forma absoluta respecto al entorno, teniendo cada uno sus ventajas y sus desventajas como se comentará a continuación.

4.2.1.1. Localización relativa

En la localización relativa o local la localización del robot se realiza de forma relativa a la posición de inicio de este, que si bien no tiene porque ser conocida, en general suele ser conocida o se tiene una estimación aproximada de ésta [20, 32].

La localización relativa es una de las más sencillas y básicas, siendo usada desde los comienzos de la robótica. La forma más común de realizar la localización relativa es mediante odometría o mediante navegación inercial [22]:

- Odometría. La odometría es un método básico y simple para estimación de la posición a partir de un punto de origen. Se basa en calcular la posición actual en función de cuanto ha avanzado el robot o a que velocidad ha avanzado y durante cuanto tiempo. Para obtener esta información, normalmente se utilizan unos dispositivos acoplados a las ruedas del robot.

  La principal ventaja de este método es su sencillez, facilidad de uso, y que es autocontenido y no necesita de ninguna información sobre el entorno, permitiendo siempre ofrecer una estimación de la posición respecto al origen de movimiento. Además, este método es muy barato pues los sensores son bastante simples.

Sin embargo, esta técnica tiene una gran cantidad de desventajas. Por un lado, la estimación de la posición tiene errores provenientes de los sensores (errores de calibración y resolución limitada), del tipo de locomoción (desalineación de las ruedas, deslizamientos y diferencias de fricción según el suelo, incertidumbres en el radio exacto de la rueda o en el avance de las extremidades), etc. Estos errores son acumulativos, lo que hace que esta técnica sólo sea válida en cortos períodos de tiempo, pues el error en la estimación de la posición es acumulable (creciendo sin límite), de forma que a partir de un determinado momento la posición no sería fiable. Así pues, la odometría necesita utilizar alguna técnica que permita, de forma periódica, corregir parcialmente el error acumulado y reducir la incertidumbre, obteniendo así una mejor estima de la posición real.

Otro problema inherente a esta técnica es que es más adecuada para robots con ruedas que para robots con extremidades, pues en robots con ruedas es más sencillo estimar el avance real de éste (a pesar de los errores ya comentados). Sin embargo, en el caso de robots con extremidades los errores cometidos en la estimación del avance son mucho mayores que en el caso de los robots con ruedas, por lo que la localización basada únicamente en odometría es poco viable.

- Navegación inercial. Esta técnica, similar a la anterior, se basa en el uso de giróscopos y/o acelerómetros para estimar la rotación y aceleración del robot de forma que, mediante esta información, se pueda estimar su avance. La principal ventaja de esta técnica es la ya comentada en el caso de la odometría, y es que es un método autocontenido, que se puede aplicar sin necesidad de información externa.

  Sin embargo, el problema de esta técnica es el mismo que el de la odometría, y es que el error crece sin límite con el tiempo por lo que esta técnica sólo es válida en breves períodos de tiempo, siendo necesaria su corrección periódicamente. Además, los sensores necesarios para este tipo de localización son mucho más caros que en el caso de la odometría.

Así pues, y como resumen, si bien la localización relativa resulta sencilla y aplicable en la mayoría de entornos, pero tiene el problema de que requiere la corrección periódica de la posición estimada mediante técnicas como los filtros de Kalman [41, 89] o filtros de partículas [151] junto con marcas del entorno.

4.2.2. Localización global

En la localización global o absoluta no se dispone de ninguna información sobre la posición inicial del robot [20,32], siendo necesario estimar la posición de éste en función de la información obtenida del entorno.

Así pues, para poder realizar este tipo de localización es necesario disponer de algún tipo de información sobre el entorno, a diferencia de la localización relativa. Además, este tipo de localización puede presentar problemas si es posible que en el entorno existan ambigüedades debido a zonas de localización similares o difíciles de diferenciar.

Dependiendo de la información disponible del entorno, la localización puede clasificarse en los siguientes grupos [22]:

- Marcas activas. Este método se basa en la estimación de la posición del robot a partir de la medida de balizas activas [7, 19, 72, 104], como infrarrojos, ultrasonidos, WiFi o Bluetooth, mediante la triangulación de la señal proveniente de éstas. La posición de las balizas suele ser conocida a priori, lo que permite el posicionamiento absoluto del robot. En caso de no conocerse la posición de las balizas a priori, el posicionamiento sería relativo a éstas. Como ejemplo se podrían considerar los enjambres de robots (*swarm robotics* en inglés), los cuales suelen posicionamiento unos respecto a otros usándose como balizas mutuamente, pero sin establecer un posicionamiento absoluto. En este método, la precisión obtenida dependerá de los obstáculos e interferencias del entorno. Además, este método requiere la modificación del entorno para añadir las balizas activas.

- Detección de marcas naturales. Este método consiste en la detección de marcas naturales del entorno [41, 86, 150] que sean distinguibles y fácilmente identificables. La ventaja de este método es que no es necesaria la modificación del entorno, aunque sí es necesario determinar que tipo de marcas se usarán y, por lo tanto, es necesario un conocimiento a priori del entorno. El principal problema de este método es que las marcas deben ser fácilmente identificables y no ambiguas, pudiendo existir en ese caso incertidumbre en el posicionamiento del robot, lo que dificultaría el posicionamiento. Además, el coste computacional suele ser elevado en comparación con otras técnicas.

- Detección de marcas artificiales. En este método la localización se realiza mediante la detección de marcas artificiales pasivas [74, 110, 117] colocadas en el entorno en posiciones conocidas. La ventaja de este método es que las marcas artificiales pueden diseñarse expresamente

para ser fácilmente detectables. La desventaja, al igual que en el caso de las balizas activas, son las posibles oclusiones y la necesidad de modificar el entorno, así como la necesidad de conocer la posición de las marcas. Además, una desventaja respecto a las balizas activas es que la detección y estimación de la posición de las marcas artificiales, como por ejemplo en el caso de las marcas visuales, puede ser más costosa computacionalmente que en el caso de balizas activas. La ventaja de esta técnica respecto a la de marcas naturales es que el coste computacional de las marcas artificiales es menor y el problema de la incertidumbre debido a las marcas es muy reducida, ya que las marcas artificiales escogidas deben ser seleccionadas para evitar o reducir la incertidumbre en el posicionamiento.

Por lo tanto, como se puede observar, dentro de la localización absoluta hay una gran variedad de alternativas, siendo un elemento en común la necesidad de obtener información del entorno y, en la mayoría de los casos, modificar éste para obtener información con mayor exactitud y precisión.

4.2.3. Localización según el modelado del entorno

Independientemente de si la localización es relativa o absoluta, en cualquiera de ambos casos es normal la necesidad de disponer de alguna información o modelado del entorno, ya sea para corregir el posicionamiento relativo o para permitir el posicionamiento absoluto.

Así pues, a continuación se mostrarán las distintas variantes de localización en función del tipo de modelado del entorno disponible, que puede ser:

- Localización basada en mapas. Esta localización se basa en mapas o modelos creados a priori del entorno en el que el robot debe localizarse.
- Localización basada en construcción de mapas. Esta localización se basa en la construcción del mapa del entorno en función de la información que obtienen los sensores del robot, usando dicho modelo posteriormente para permitir la localización.
- Localización sin mapas. En este grupo se engloban localizaciones que no encajan en los grupos anteriores y que, como característica común, tienen el no necesitar una representación explícita del entorno en el que se desenvuelve el robot. En lugar de un mapa del entorno, este tipo de localización suele usar características u objetos del entorno como referencias para realizar un seguimiento o estimación del movimiento

realizado por el robot. En este tipo de localización suelen incluirse localizaciones relativas como la odometría, comentada anteriormente, así como la localización mediante flujo óptico, ninguna de las cuales necesita, en principio, un mapa del entorno.

A continuación se describen en más detalle cada uno de estos enfoques.

4.2.3.1. Localización basada en mapas

La localización basada en mapas consiste en la localización del robot en base a un modelo o mapa del entorno disponible a priori [20, 32].

El modelo del entorno puede tener un grado de detalle mayor o menor, dependiendo de la información disponible sobre éste y de la precisión necesaria para la localización del robot así como la precisión que puedan alcanzar los sensores de éste.

Normalmente, los modelos o mapas del entorno muestran todos (o los más representativos) elementos estáticos y marcas (geométrica, visuales, etc.) que se encontrarán en el entorno y que sean fáciles de identificar, permitiendo así al robot estimar su posición cuando detecte dichos objetos. Estos mapas pueden ser métricos, topológicos o metrico-topológicos, como se comentará en la sección 4.2.4.

El principal problema con este enfoque es precisamente la generación de un mapa o modelo del entorno, lo cual no es siempre sencillo de hacer, además de suponer problemas en el caso de que el entorno sea dinámico y el mapa deba modificarse, con lo que dejaría de ser fiable.

4.2.3.2. Localización basada en construcción de mapas

Debido a la dificultad que puede suponer en ocasiones la generación de un mapa del entorno que sea adecuado para la localización y navegación del robot surgió el enfoque de localización basada en la construcción del mapa. Este enfoque consiste en que el propio robot sea capaz de explorar el entorno de navegación y localización previamente generando, durante esta etapa previa, un modelo del entorno más realista que en el caso anterior mediante la extracción de características y puntos identificativos del entorno directamente. Tras esta etapa previa de exploración y generación del mapa, el robot sería capaz de localizarse y navegar por dicho entorno.

El principal problema de este enfoque radica en esa etapa previa de exploración y generación del modelo del entorno, la cual puede resultar costosa en tiempo y en cómputo. Además, los errores de localización acumulados a lo largo de la construcción del mapa pueden hacer este inexacto y, por tanto, invalido para navegación y localización a partir de un determinado momento.

Otro punto de vista de esta técnica es el de localización y mapeo simultáneo o SLAM (del inglés *Simultaneous Localization And Mapping*), que consiste en un mapeo y localización que se realizan de forma simultánea mientras el robot navega por el entorno. Esta técnica supone un área propia de investigación aún activa y con mucha actividad debido a su versatilidad, a pesar del coste computacional y complejidad que supone.

4.2.3.3. Localización sin mapas

Por último, está ese tercer enfoque que se basa en la localización y navegación sin la realización ni utilización de ningún mapa, modelado ni conocimiento previo del entorno.

Este enfoque suele aplicarse en sistemas en los que la localización y navegación dependen exclusivamente de lo que se percibe en el instante actual, obteniendo de los sensores disponibles información suficiente como para localizarse y navegar de una forma segura. Sin embargo, hay que tener en cuenta que a pesar de no usar ningún tipo de mapas, este tipo de localización suele requerir algún historial de posiciones anteriores o de estimaciones respecto al punto inicial de origen.

En este grupo podría incluirse, por ejemplo, la localización relativa mediante odometría, pues no requiere de un mapa del entorno, realizando el posicionamiento a partir de las estimaciones de movimiento realizado. Otra localización que se englobaría en este grupo sería la localización basada en flujo óptico, la cual estima el movimiento del robot, y con éste la posición respecto a la posición de inicio, a partir del flujo de imágenes recibido por el robot.

El principal problema de esta técnica es que el entorno tiene que tener suficiente cantidad de elementos reconocibles en el entorno como para permitir la localización del robot pues, usando el ejemplo de la localización por flujo óptico, si el entorno fuese simétrico o de un color uniforme sería difícil extraer información para estimar el movimiento realizado. Otro problema de este tipo de localización es que el cómputo necesario para su aplicación en tiempo real puede ser bastante elevado.

4.2.4. Tipos de mapas

Aunque no siempre es necesaria la utilización de un mapa del entorno para poder realizar la localización del robot, pero si es usual disponer de un mapa de éste, siendo en general un elemento fundamental para la localización.

Este modelado o mapeado del entorno no es más que una abstracción del entorno que contiene las características más importantes, representativas o

útiles de éste para la localización y navegación del robot, como pueden ser marcas naturales (esquinas, puertas o elementos específicos y fácilmente reconocibles) o marcas artificiales. Un elemento importante a tener en cuenta es que en estos modelados se suelen representan aquellos elementos estáticos o con una alta probabilidad de encontrarse en el entorno, siendo fiables para poder realizar una localización, no representando aquellos que puedan no encontrarse por ser dinámicos, ya que el robot correlará, en general, la información que obtenga de sus sensores con la del mapa del entorno comparando ambas informaciones para poder determinar su posición.

Los modelos de representación de mapas se pueden clasifican en los siguientes grupos [20, 32].

- Mapas métricos. Los mapas métricos describen el entorno mediante las características métricas de éste, como distancias entre elementos (esquinas o bordes) representando los elementos en el mapa de forma absoluta respecto a un punto de referencia. Este tipo de mapas permite que, conociendo la posición en el mapa, se pueda determinar la distancia al destino y a todos los obstáculos representados en el mapa. Hay distintos tipos de mapas métricos, como las rejillas de ocupación [20], que dividen el mapa en una serie de celdas con una determinada probabilidad de ocupación, o los basados en características o marcas, representando elementos característicos como paredes, esquinas, o la posición de marcas visuales.

- Mapas topológicos. Los mapas topológicos [21], a diferencia de los mapas métricos, no almacenan información exacta sobre la distancia de los elementos del mapa, sino que almacenan información sobre la relación entre los elementos del mapa, representando cada posición destacada de un mapa, como una habitación o una marca concreta, mediante un nodo, los cuales están conectados entre sí mediante arcos dependiendo de si hay algún camino seguro que permita ir de una habitación a otra. Los arcos que unen los nodos del mapa pueden contener distinta información, como información métrica indicando la distancia entre un nodo y otro. Así pues, estos mapas resultan más compactos que los mapas métricos, pues sólo almacenan un grafo de los elementos importantes del mapa y su conexión, aunque a costa de perder cierta información específica sobre las posiciones métricas de los elementos. Sin embargo, estos mapas permiten un cálculo de caminos más rápido que los métricos debido a su sencillez.

- Mapas métrico-topológicos. Estos mapas [126] tratan unir las virtudes de ambos tipos de mapas, usando mapas topológicos para un rápido

cálculo de caminos, y métricos para la obtención de información detallada de zonas concretas.

4.3. Localización visual basada en marcas

Una tarea crucial a tener en cuenta tanto en robots individuales como en MRS es la capacidad de estimar la propia posición y, en el caso de los MRS, la de cada robot del sistema, ya que éstos pueden actuar como obstáculos para los demás miembros del equipo, además de entorpecerse mutuamente o repetir tareas si se desconoce la posición de los demás robots, perdiéndose la ventaja de los MRS frente a los robots individuales.

Como se ha comentado en la sección 4.2, existen una gran variedad de técnicas de localización. Dado que no existe una técnica de localización que se adecúe a todas las circunstancias y entornos, es necesario seleccionar la técnica de localización a utilizar dependiendo de distintos factores como la precisión necesaria en la localización, la información disponible del entorno, el tipo de plataforma robótica que se disponga o los sensores de ésta. Así pues, es necesario determinar la más adecuada en cada situación concreta según las necesidades.

En el presente trabajo, la plataforma de desarrollo a usar es el robot AIBO ERS-7 de Sony, cuyo principal sensor de captura de información es una cámara unidireccional, y el entorno de trabajo es un entorno basado en el de la competición de fútbol Robocup. El campo de fútbol de la Robocup, si bien suele modificarse ligeramente en cada temporada, pero en general suele ser un campo de color predefinido con las líneas delimitadoras, como las de un campo de fútbol normal, con una serie de marcas artificiales añadidas (balizas de colores y porterías [108] [110]) que sirven para el posicionamiento de los robots ya que su posición, tamaño y color son conocidos a priori. En un entorno como éste, en el que se conoce toda esta información necesaria para localizarse a priori, lo normal es usar una localización absoluta basada en un mapa que indique la posición de las distintas marcas.

Sin embargo, y debido a problemas como las posibles oclusiones debidas a otros robots o la incertidumbre debida a la simetría de las marcas, es habitual complementar esta localización mediante el uso de filtros de Kalman [41, 89] o filtros de partículas [110, 151] para mejorar el posicionamiento. De hecho, este enfoque es el más usado en este entorno, pues es robusto y da buenos resultados [108, 110].

Lamentablemente, y como se comentó anteriormente, este tipo de localización no encaja con el enfoque reactivo presentado en el capítulo anterior, pues implica un modelado bastante estricto del entorno, así como el uso de

memoria (historial de observaciones previas). Una alternativa más acorde y que también permitiría un posicionamiento con bastante precisión sería el uso de marcas fiduciarias (*fiducial marker* en inglés). La ventaja de este enfoque es que se podría obtener la posición de los robots respecto a la marca a partir de la imagen actual y, por lo tanto, de una forma más reactiva al no usar más que la información actual de los sensores (eliminando la componente temporal). Sin embargo, este enfoque todavía supone un modelado del entorno bastante específico. Así pues, otro enfoque más acorde con la filosofía de los comportamientos reactivos sería la localización basada únicamente en los objetos en común que puedan encontrarse en el entorno (como objetos de un color específico) y que estén visualizando los robots en el momento actual. Este enfoque sería el más reactivo, al requerir un menor modelado del entorno, pero también sería el menos preciso y más limitado en las situaciones en las que podría usarse. Así pues, a continuación se comentará el funcionamiento de la localización basada en objetos comunes, marcas fiduciarias y balizas de colores usando filtrado, presentando posteriormente algunos experimentos realizados para comprobar la viabilidad de su uso para el sistema de coordinación que se pretende desarrollar.

4.3.1. Localización basada en objetos comunes

Un requisito general para establecer la localización entre varios robots es un sistema de referencia común, es decir, establecer de alguna manera una referencia entre el sistema de referencia local del robot y el sistema de referencia global del entorno. Dado el enfoque reactivo del sistema propuesto, la localización debería ser lo más básica posible, suponiendo el mínimo modelado del entorno y no necesitar una componente temporal o memoria. Así pues, lo ideal sería usar una localización inmediata o instantánea respecto a cualquier tipo de objeto o patrón que se pudiese identificar en el entorno. Sin embargo, esto es demasiado complejo, por lo que hay que delimitar los objetos que pueden servir como referencia a objetos concretos y fáciles de identificar, aunque lo más genéricos posible, como podrían ser objetos de un color específico (dado que la localización a usar se basa en visión).

Una forma usual de obtener la posición usando referencias visuales o balizas es mediante triangulación, la cual permite el posicionamiento mediante la aplicación de la trigonometría a partir de la información obtenida de dos o más balizas. Sin embargo, dado que se trata de evitar el uso de memoria en el sistema, para el uso de la triangulación es necesario que las marcas u objetos de referencia a usar estén todos dentro del campo de visión al mismo tiempo, lo cual no es fácil ni viable en muchas ocasiones.

Una forma de solucionar este problema sería hacer uso de la idea de la

visión estéreo, de forma que dos robots estimen su posición a partir de la disparidad que presenta un objeto desde su punto de vista. En este caso, sería necesario sólo un elemento común en el campo de visión para establecer el posicionamiento, así como permitir la comunicación entre los robots.

Estos dos enfoques serán los comentados en primer lugar, pues son dos alternativas que, en principio, encajarían con un enfoque reactivo sin memoria y con un modelado mínimo del entorno al usar como referencia objetos *generales*, cuya única limitación es su color.

4.3.1.1. Localización basada en triangulación

La triangulación consiste en el uso de la trigonometría para determinar la posición de puntos o distancias y es una de las técnicas más clásicas para determinar la posición de un robot en función a una serie de balizas o marcas repartidas en el entorno [20, 32] independientemente de si las balizas son visuales [27], sonoras [119] o de radiofrecuencia [111]. De hecho, la triangulación por radiofrecuencia es un tema de estudio muy activo últimamente, pues resulta muy práctico al suponer un modelado mínimo del entorno, pero suele tener errores muy grandes y sólo permite estimar la posición, no la orientación.

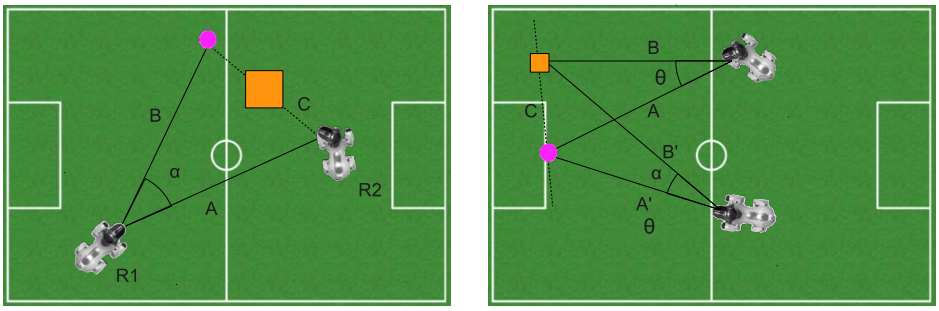

Figura 4.1: (a) Estimación de posición de objetos ocultos usando triangulación. (b) Estimación de la posición mediante triangulación usando dos objetos en común.

Así pues, dejando de lado la triangulación por radiofrecuencia queda la alternativa de la triangulación por visión, mostrándose en la figura 4.1 un par de ejemplos en los que se podría aplicar. En el ejemplo de la figura 4.1.a, el robot R1 podría estimar mediante triangulación la distancia a la que está el compañero de la pelota y enviarle su posición aproximada. Para ello, sería necesario aplicar la ley del seno (ec. 4.1) y del coseno (ec. 4.2) usando como

información de partida la distancia a la que está su compañero, la pelota y el ángulo entre ambas.

$$\frac{A}{sen(\alpha)} = \frac{B}{sen(\beta)} \tag{4.1}$$

$$C^2 = A^2 + B^2 - 2ABcos(\alpha) \tag{4.2}$$

Sin embargo, la aplicación más usual de la triangulación, aunque partiendo del mismo enfoque, es el mostrado en la figura 4.1.b, en el que los robots puede estimar su posición a partir de la visualización de dos marcas u objetos reconocibles e identificables, sirviendo estos objetos como eje de referencia para que los robots puedan compartir su posición.

El problema de este enfoque, sin el uso de memoria, es que requiere que las marcas a usar estén visibles simultáneamente y sean identificables, lo cual es difícil de conseguir en la realidad. Además, aunque en teoría baste con dos marcas para establecer el posicionamiento mediante triangulación, en la práctica, debido a los errores, es recomendable el uso de más elementos para corregir el posicionamiento obtenido.

4.3.1.2. Localización basada en visión estéreo

Para evitar la necesidad de que haya varios objetos simultáneamente en el campo de visión para posicionarse, se puede utilizar otro enfoque basado en la idea de la visión estereoscópica.

La visión estereoscópica o visión estéreo es una técnica que permite obtener información tridimensional a partir de imágenes en 2 dimensiones. Esta técnica está basada en la visión humana, y de forma más genérica en la visión de muchos animales, los cuales usan las imágenes capturadas por los ojos para estimar la profundidad o distancia a la que se encuentran objetos u otros animales.

Concretamente, la estereoscopía artificial permite obtener la distancia de los objetos visualizados respecto a las cámaras que forman un par estéreo. El cálculo de la profundidad en la visión estéreo parte de la suposición de que las dos cámaras que capturan las imágenes tienen la misma distancia focal, sus ejes ópticos son paralelos y las imágenes generadas por ambas cámaras están en el mismo plano (Fig. 4.2.a). Teniendo en cuenta estas consideraciones, la profundidad o distancia del objeto observado respecto a las cámaras está determinada por la siguiente ecuación:

$$D_{focal} * D_{camaras} = D_{objeto} * Disp \tag{4.3}$$

Siendo D_{focal} la distancia focal de las cámaras, $D_{camaras}$ la distancia entre las cámaras, D_{objeto} la distancia al objeto a localizar o profundidad y $Disp$ la disparidad entre las dos imágenes, calculada como $Disp = X_1 - X_2$, representando la diferencia en píxeles de la posición de un objeto entre las dos imágenes de las cámaras.

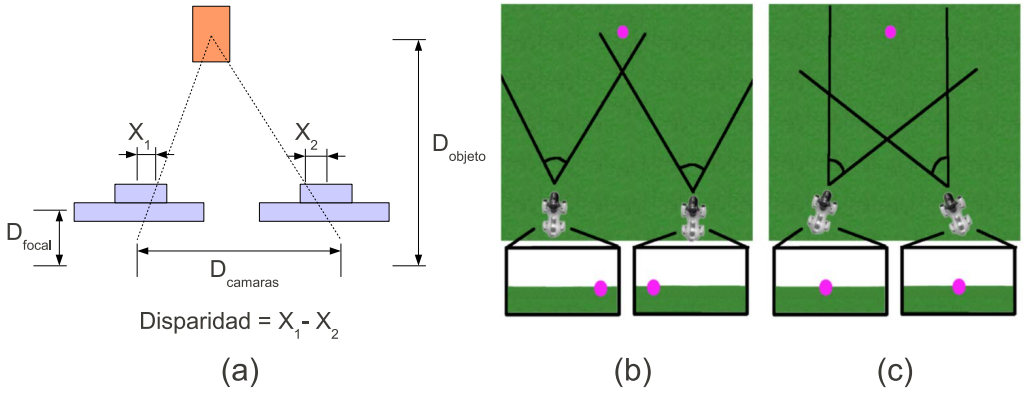

Figura 4.2: (a) Visión estéreo y parámetros para el cálculo de la disparidad. (b) y (c) Importancia de la orientación de los robots para la localización mediante visión estereoscópica.

Sin embargo, esta ecuación se puede ver de otra forma, pues si se supone conocida la distancia de un objeto respecto a dos cámaras que cumplan las condiciones anteriores, lo que se puede estimar es la distancia entre estas dos cámaras. Es decir, partiendo de dos robots (Fig. 4.2.b), cada uno con una cámara direccional, que están observando un objeto en común de tamaño conocido (lo que permite obtener la distancia de éste), se podría calcular la distancia entre ambos robots, permitiendo establecer una estimación de la posición relativa entre ellos. El uso de esta técnica requiere, por tanto, que ambos robots compartan su información visual para determinar la disparidad de las imágenes de cada uno de los robots, por lo que se trataría de una técnica de localización cooperativa.

Sin embargo, para poder usar esta técnica de localización es importante tener en cuenta las suposiciones de partida de la ecuación 4.3 para el cálculo de la profundidad. Por lo tanto, para que la estimación de la distancia entre los robots sea fiable estos deben estar alineados y en el mismo plano (como las cámaras del par estéreo), pues cuanto más desalineados estén los robots más errores se obtendrá en la estimación de su posición. Así, por ejemplo, la figura 4.2.b muestra un ejemplo de situación ideal en el que esta localización podría ser aplicable obteniéndose una buena estimación de la distancia entre los robots. Sin embargo, la figura 4.2.c muestra una situación en la que esta

técnica no sería aplicable pues los valores obtenidos no serían correctos debido a la variación de ángulo de las cámaras de ambos robots. Así pues, para usar esta técnica habría que estimar si es aplicable o no, lo cual sería realmente complejo en un ambiente real. En el caso del entorno de un campo de fútbol como el de la Robocup, sin embargo, una opción para calcular si ambos robots están en paralelo, o aproximadamente en paralelo, sería estimar su orientación cuando visualicen alguna línea del campo, lo que les daría una idea de su orientación. En cualquier caso, para poder determinar si esta localización es aplicable o no prácticamente habría que tener una buena estimación de la posición posición de los robots, dado el requisito de la alineación y la posición en el mismo plano de éstos para su aplicación. Por último, otro problema de este enfoque es que es necesario saber qué robot está a la derecha y cuál a la izquierda para poder aplicar la técnica adecuadamente.

4.3.2. Localización basada en marcas fiduciarias

La localización basada en marcas fiduciarias requiere, como es lógico, el uso de marcas específicas fácilmente reconocibles en el entorno. Las marcas fiduciarias son marcas de referencia concretas, como pueden ser los códigos QR, las marcas ARToolkit o las marcas reacTIVision, que permite obtener la posición de la cámara respecto a la marca y viceversa. Este tipo de localización, por tanto, implica un modelado del entorno más concreto y exigente que en el caso anterior, que se basaba únicamente en la detección de objetos de un color determinado, que si bien implica igualmente un modelado del entorno, pero era más genérico. Por lo tanto, este enfoque resulta menos *reactivo* que el planteamiento anterior, aunque tiene la ventaja de que permite obtener el posicionamiento únicamente a partir la imagen actual, por lo que no requiere ningún uso de memoria.

Una alternativa más reactiva al uso de marcas fiduciarias sería el reconocimiento de marcas naturales. El problema de este enfoque es que suele requerir una mayor capacidad de cómputo, además de que es necesario que en el entorno haya elementos característicos que puedan funcionar como marcas naturales, y dado que el entorno de pruebas usado en los experimentos es un entorno artificial similar al de la Robocup no resultaría un enfoque viable.

Así pues, de entre las distintas alternativas de marcas fiduciarias disponibles, se ha decidido usar concretamente marcas ARToolkit [64,82], ya que permite el posicionamiento 3D de la cámara respecto a la marca y se ha usado anteriormente para el desarrollo de aplicaciones de Realidad Aumentada (RA) con buenos resultados. Así pues, ARToolkit es una librería realizada en C distribuida bajo licencia GPL que está especialmente diseñada para la implementación de aplicaciones de RA. Para ello, esta librería proporciona una

serie de funciones para la captura de vídeo y para la búsqueda de patrones ARToolkit en tiempo real.

Concretamente, los patrones ARToolkit (Fig. 4.3.a) son unas marcas visuales planas de tamaño conocido a priori y formadas por un marco de color negro que contiene en su interior un diseño previamente configurado que permite la identificación de cada marca concreta. De esta forma, cuando se detecta una marca la librería puede calcular la distancia y la orientación de ésta a partir de su tamaño y de la distorsión provocada por el ángulo desde el que se percibe. Por lo tanto, la librería permite obtener la posición de la marca respecto a la cámara que la visualiza y viceversa.

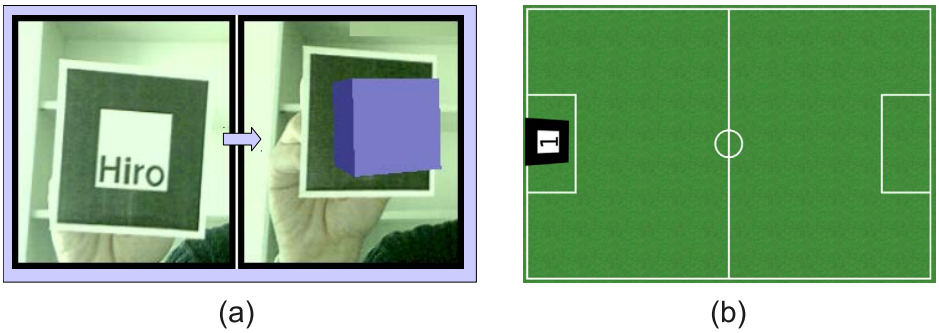

Figura 4.3: (a) Marca de ARToolkit y uso en RA. (b) Marca ARToolkit para localización.

La ventaja de esta librería es que soluciona una de las principales dificultades en el desarrollo de la RA, que es el posicionamiento preciso de objetos reales (las marcas) respecto a la cámara, lo que proporciona el punto de vista del observador para poder insertar los objetos virtuales en el mundo real de forma realista (Fig. 4.3.a).

En el presente trabajo, el posicionamiento que ofrece ARToolkit permite obtener la posición de la cámara (el robot) respecto a la marca. Así pues, usando una marca ARToolkit como referencia (Fig. 4.3.b), sería posible establecer la posición de varios robots respecto a ella fácilmente a partir de la imagen actual. De esta forma, esta librería permitiría obtener un posicionamiento preciso, manteniendo, dentro de lo posible, la filosofía de los comportamientos reactivos al obtener el posicionamiento a partir de la imagen actual.

4.3.3. Localización basada en marcas y filtrado

Esta localización consiste en el uso de balizas específicas repartidas en el entorno para ofrecer un posicionamiento global de una forma sencilla. Así pues, para este tipo de localización se suelen usar balizas de colores y tamaños conocidos a priori situadas en el entorno en posiciones fijas, facilitando de esta forma la localización del observador (en este caso robots).

Sin embargo, debido a que este posicionamiento es sensible a oclusiones, incertidumbre en las marcas, etc., es normal que el posicionamiento obtenido se corrija mediante el uso de algún tipo de filtrado que permitan mejorar la estimación y estabilidad de la posición, siendo los más normal el uso de filtros de partículas [151] o de filtros de Kalman [41,89]. Ambos tipos de filtros son ampliamente conocidos y usados y ofrecen buenos resultados, aunque los filtros de partículas ofrecen una mayor flexibilidad y fácil implementación que los filtros de Kalman, razón por la que se ha seleccionado la utilización de los filtros de partículas. De hecho, la localización basada en balizas de colores y filtros de partículas es una técnica de localización muy conocida y usada en el entorno de la Robocup [67, 108, 110, 151].

Lamentablemente, y como se comentó anteriormente, esta opción es la que menos encaja con un enfoque reactivo, pues implica un modelado muy específico y estricto del entorno, al forzar unas marcas concretas en posiciones fijas, a la vez que implica el uso de memoria (histórico de observaciones), lo cual va contra el enfoque general de los comportamientos reactivos. Sin embargo, se considerará el uso de esta localización, aunque sea como última opción debido a su enfoque *menos reactivo*, dado que se trata de una técnica madura y ampliamente probada y usada que ofrece un posicionamiento estable y, en general, más fiable que las técnicas anteriores.

Así pues, y como es lógico, para aplicar esta técnica es necesario un módulo que permita la detección de las balizas y otro que implemente un filtro de partículas. A continuación se explican en más detalle cada uno de estos módulos.

4.3.3.1. Detección de balizas de colores

El entorno de la Robocup, el entorno de pruebas usado en los experimentos, suele incluir marcas artificiales de colores en posiciones concretas que el robot puede usar para una localización visual global [37, 108, 110, 140]. Estas marcas cambian de año en año, modificando bien la distribución de colores, la posición en el campo, etc.

Sin embargo, una característica en común es que son marcas de colores fáciles de identificar en el entorno, de tamaño y posición conocida, lo que

permite que al detectarlas se pueda estimar la posición de forma absoluta en el campo.

En cualquier caso, y aunque existen distintas técnicas y enfoques [26,108, 110], la mayoría de detectores de balizas suponen realizar, de una forma u otra, los siguientes pasos:

- Conversión de la imagen a un espacio de color en el que sea fácil detectar los colores, como HSI o HSV, y segmentar la imagen para detectar los colores que forman parte de las balizas. Para la segmentación es especialmente importante definir correctamente los colores a detectar así como sus umbrales.

- Detección de regiones. Una vez segmentada la imagen, se detectan las zonas, formadas por píxeles sin ninguna relación entre sí, que se corresponden con un mismo color y zona delimitada. A esta detección de regiones se le puede aplicar un filtrado para eliminar regiones que se dan por hecho que no son viables en la realidad, como regiones demasiado pequeñas, grandes o que estén en posiciones que no sean posibles o poco probables.

- Detección de balizas. Tras detectar cada una de las regiones por separado, es el momento de determinar que regiones de las detectadas cumplen las condiciones para formar parte de una baliza. Estas condiciones, para el caso de una baliza como las de la Robocup, serían por ejemplo que los colores se correspondan con los de una baliza válida, que las regiones estén una encima de otra o que ambas regiones tengan un tamaño similar.

- Cálculo de la posición respecto al robot. Tras detectar una baliza con una cierta seguridad, se obtendrá su tamaño y posición en la imagen. Así pues, y dado que se conocen tanto el tamaño predefinido de la baliza así como los ángulos de visión de la cámara del AIBO (56.9º en horizontal y 45.2º en vertical), es posible estimar la distancia de la marca así como su ángulo respecto a la cámara del robot.

Por último, la figura 4.4 muestra un ejemplo de detección de balizas siguiendo los pasos indicados anteriormente (conversión de espacio de color y segmentación, detección de áreas, detección de balizas y cálculo de posición).

4.3.3.2. Filtros de partículas

Los filtros de partículas (también conocido como filtro de bootstrap, filtro de Monte Carlo o filtro de condensación) se engloban dentro de los conocidos

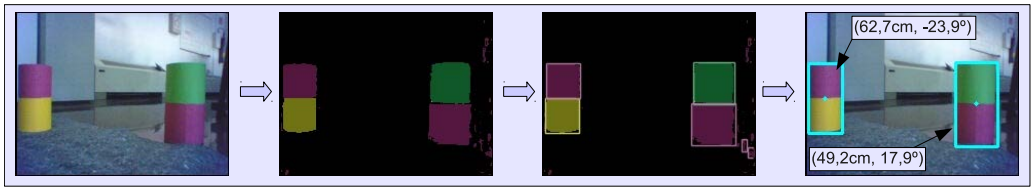

Figura 4.4: Proceso para detección de balizas.

como métodos de Monte Carlo, que son un conjunto de técnicas de muestreo estadístico que permiten representar cualquier distribución con un número reducido de muestras sin perder representatividad.

Concretamente, los filtros de partículas se basan en el Filtro Bayesiano, un algoritmo recursivo que permite estimar la distribución de probabilidad del estado de un sistema en un cierto instante t en función de los datos disponibles de momentos anteriores. A esa densidad de probabilidad se le suele llamar creencia (*Belief* en inglés). Aplicando esto al caso práctico de la localización, el estado del sistema sería la localización del robot, mientras que los datos en instantes anteriores serían los movimientos y las observaciones realizadas por el robot.

Por lo tanto, los filtros de partículas permiten estimar el estado de un sistema (la localización de un robot) en un cierto instante t a partir de valores actuales y pasados (movimientos del robot y sus observaciones) aproximando la función de distribución a estimar mediante una serie de muestras aleatorias conocidas normalmente como partículas. Para ello, cada partícula está formada por un posible estado del sistema (una localización concreta) y un peso que indica la probabilidad de que el estado que representa esa partícula sea el correcto. Así pues, el cálculo del peso de la partícula dependerá del parecido que haya entre el sistema real y el que describe la partícula. De esta forma, el estado actual del sistema es modelado como un conjunto de partículas cuyos pesos dependen de los valores (movimientos y observaciones) anteriores del sistema.

Una vez descrito en que consiste un filtro de partículas, se comentará su funcionamiento que, como todo filtro predictor-corrector, está formado por dos fases que se repiten constantemente. Estas fases son:

- Fase de predicción. En esta primera fase se predice el estado del sistema en el instante t a partir de la función de distribución (el conjunto de partículas) en el estado $t-1$. Para ello hace falta lo que se conoce como modelo del sistema, que en el caso de la localización también se conoce como modelo de movimiento, y que se es un modelo que permite estimar cuanto ha avanzado el robot entre el instante anterior y el actual. En

general, el avance del robot suele calcularse a partir de la odometría o de la velocidad teórica de éste. Sin embargo, dado que este parámetro no es exacto, hay que añadirle un ruido que dependerá de la estimación del error del movimiento del robot. Así pues, a través de este modelo del sistema se puede determinar cuanto avanza el robot en función de los comandos de movimiento ejecutados.

- Fase de actualización. Esta fase puede ser dividida a su vez en dos fases:
  - Corrección. En esta fase se corrige la predicción realizada en la fase anterior en base a las observaciones obtenidas por el robot. Para ello, se ajustan los pesos de cada partícula en función de si el estado que describe se parece o no al determinado por las observaciones. Para realizar esta corrección es necesario lo que se conoce como un modelo de observación, que en el caso de la localización sería un modelo que determina la probabilidad de estar en una posición concreta en función de las observaciones obtenidas, probabilidad que dependerá de los posibles errores en la estimación de la observación o de los sensores de entrada. Así pues, a partir del modelo de observación se puede calcular la probabilidad de que el estado representado por las partículas se corresponda a la observación actual. Existen distintas formas de ajustar los pesos de las partículas, aunque una alternativa usual es el uso de una función de distribución, como puede ser una distribución normal, con una desviación que dependerá del error del sensor de entrada o de las observaciones.
  - Normalización y remuestreo. En este último paso se normalizan los pesos de las partículas (la suma de todos los pesos debe valer 1) y se obtiene una nueva población de partículas en función de sus pesos, de forma que las partículas con un mayor peso (mayor probabilidad) generen más partículas, aumentando la probabilidad global asociada a la siguiente población.

Así pues, realizando estos pasos de forma repetitiva los filtros de partículas consiguen aproximar la posición del robot en función de sus movimientos y de observaciones (las marcas que visualiza).

4.4. Experimentos y resultados

En este apartado se mostrarán los experimentos con los distintos sistemas de localización comentados. Para probar cada uno de los sistemas de

localización se ha implementado un sistema de navegación segura con dos robots AIBO ERS-7 de Sony, realizando la navegación mediante PFA (sección 2.4.1.1). El entorno de pruebas en todos los casos es un espacio abierto sin obstáculos excepto en los experimentos en los que se usa localización basada en objetos comunes, en los que habrá una pelota y un cubo naranja que permiten la localización de los robots.

Concretamente, la prueba consistirá en realizar una navegación segura entre los dos robots mientras tratan de alcanzar su meta. Es decir, el objetivo es que ambos robos naveguen evitando colisionar entre sí e interferirse mutuamente, para lo cual será necesario que los robots sean capaces de estimar adecuadamente sus posiciones relativas. Para ello, se permitirá la comunicación entre los robots de forma que puedan intercambiarse la estimación de sus posiciones respecto al objeto de referencia usado en cada una de las localizaciones.

4.4.1. Localización basada en objetos comunes

En primer lugar se probará el sistema de localización basado en objetos comunes en el campo de visión. Como se ha comentado, esta localización podría hacerse mediante visión estéreo o mediante triangulación. Sin embargo, dado que la triangulación es una técnica ampliamente usada y probada, y que, como se ha comentado, requeriría el uso de memoria pues necesita la posición de dos objetos de referencia para establecer el posicionamiento, no se realizarán pruebas con este sistema de localización, probándose únicamente la localización basada en visión estéreo para comprobar su funcionamiento al no tratarse un enfoque habitual.

Así pues, la localización basada en visión estéreo requiere que haya, al menos, un objeto identificable en el campo de visión de ambos robots para que puedan determinar sus posiciones a partir de este objeto. Por lo tanto, para las pruebas en el entorno habrá un cubo naranja y una pelota rosa, que serán los objetos comunes que permitirán la localización.

Concretamente, la prueba del sistema consistirá en que los dos robots naveguen de forma segura evitando colisionar e interferirse mientras se aproximan a sus destinos, que serán los objetos usados como referencia. En ambas pruebas se busca que ambos robots naveguen usando únicamente localización basada en visión estéreo para no colisionar.

Para el uso de la localización basada en visión estéreo es necesario conocer la distancia focal de las cámaras de los robots AIBO, que se ha estimado en 190 píxeles. Así pues, y teniendo en cuenta la ecuación 4.3, conociendo la distancia de los objetos a las cámaras y la disparidad entre los objetos se podría estimar la separación entre los robots. Para poder estimar la dispa-

ridad, los robots se comunicarán entre ellos para compartir la posición del objeto detectado.

4.4.1.1. Implementación del sistema de pruebas

La implementación del sistema se describe gráficamente en la figura 4.5 por su diagrama de bloques. Este sistema se ha implementado haciendo uso del entorno Tekkotsu.

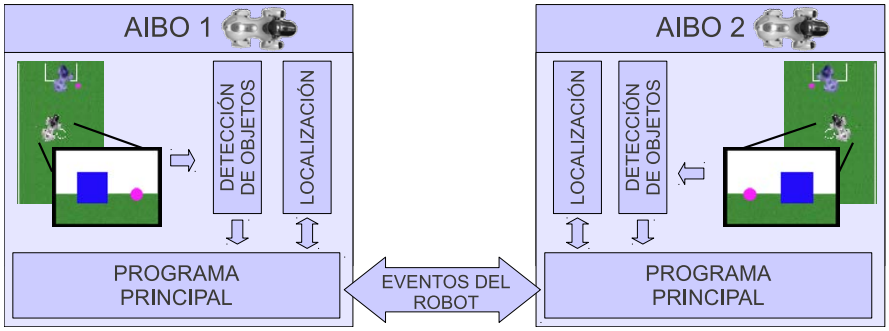

Figura 4.5: Diagrama de bloques del sistema.

Como se puede observar, tanto el sistema de navegación segura como el de localización se ejecutan de forma autónoma en cada AIBO. Concretamente, el sistema está formado por un módulo, el programa principal, que recibe información de otros dos módulos, el de detección de objetos y el de localización propiamente dicho. Las funciones concretas de cada uno de estos módulos son:

- Detección de objetos. Este módulo realiza la detección del color que tiene asociado cada objeto (rosa para la pelota y naranja para el cubo). La detección de los colores se realiza mediante una clase de Tekkotsu que cada vez que detecta en la imagen una zona con el color configurado genera un evento local con su centro, dimensiones y distancia estimada. La información del objeto es almacenada en una lista de objetos locales, actualizando la información si ya se había detectado. Cada objeto detectado es almacenado un máximo de dos segundos, eliminándose posteriormente.

- Programa principal. Este módulo se encarga de distintas tareas, entre las que se encuentran:

  - Comunicación de eventos. Esta tarea se encarga de enviar los eventos generados por el módulo de detección de objetos al otro robot

así como de recibir y procesar los eventos recibidos. Así pues, cuando se recibe un evento remoto se añade a una lista de objetos remotos, actualizando la información si ya se había detectado. Al igual que con los locales, cada objeto se almacena sólo durante 2 segundos. La comunicación entre los robots se realiza via WiFi mediante una clase de Tekkotsu que facilita el envío y procesamiento de eventos entre los robots.

- Navegación. Como se ha comentado, la navegación de los robots se basa en PFA, siendo el elemento atractor el objetivo a conseguir (pelota rosa o cubo naranja) y los repulsores el otro robot y el objeto que no sea el destino a conseguir.

- Módulo de localización. Este modulo se encarga de establecer la posición relativa del robot respecto a un objeto en común usando localización por visión estéreo.

4.4.1.2. Experimentos realizados

En esta sección se presentan los experimentos realizados con el sistema de localización basado en objetos comunes en el campo de visión. Concretamente, se usará la localización basada en visión estéreo, como ya se ha comentado.

Para las pruebas se han usado dos robots AIBO ERS-7. Dado que el objetivo es probar el sistema de localización, el comportamiento de los robots es sencillo así como el entorno de pruebas, formado únicamente por una pelota rosa y un cubo naranja. En los experimentos los robots tendrán que navegar de forma segura sin colisionar mientras tratan de alcanzar su objetivo. Concretamente, el objetivo será el cubo naranja para el robot que esté más cercano a éste, siendo la pelota rosa el objetivo del otro robot. En el caso de que sólo esté la pelota, ésta será el objetivo de los dos robots.

Es importante destacar que al comienzo de los experimentos los robots están colocados de forma paralela y en el mismo plano para que los resultados obtenidos por la localización estéreo sean aceptables. Hay que tener en cuenta también que en un entorno tan simple como el usado no hay referencias que permitan a los robots estimar su orientación y, por tanto, determinar si la utilización de esta localización es válida o no, por lo que se considerará válida durante toda la duración de las pruebas, a pesar de que según los robots se mueven y se desalinean su aplicación no sea correcta.

Así pues, la figura 4.6.b muestra la evolución de un primer experimento en el que se sitúan los robots frente a la pelota rosa, consiguiendo ambos alcanzarla sin colisionar. Esto muestra que la técnica de localización mediante

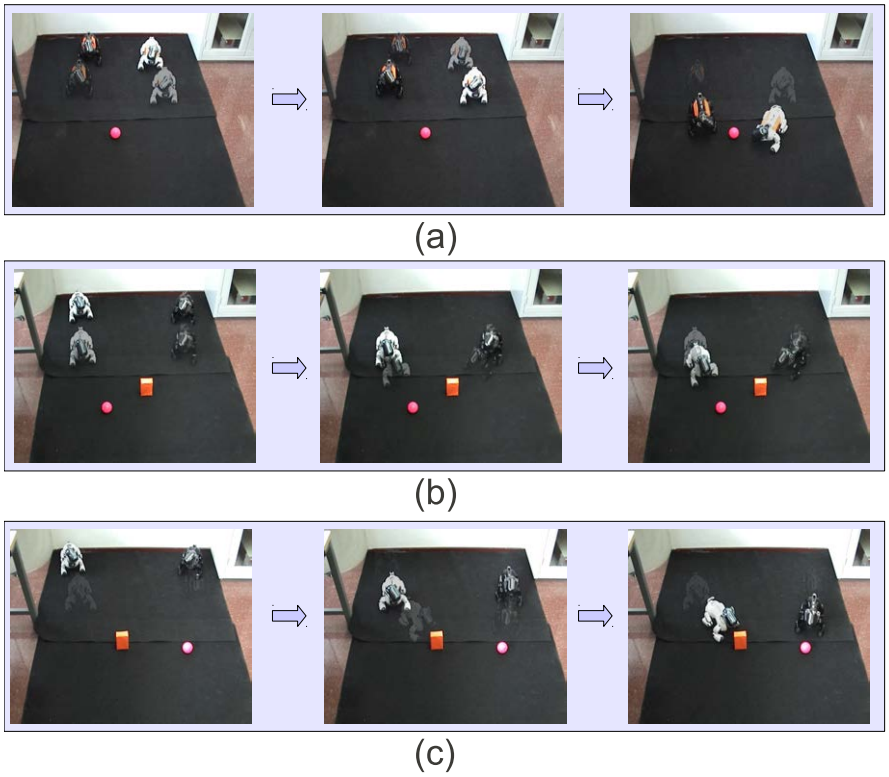

Figura 4.6: Navegación segura entre AIBOSs usando localización por objetos comunes. (a) Un único objeto. (b) Dos objetos (el AIBO negro es el más cercano al cubo) (c) Dos objetos (el AIBO blanco es el más cercano al cubo).

visión estéreo permite realizar una estimación de la posición del otro robot de forma que evitan colisionar, a pesar de que al final del experimento los resultados de la localización no sean válidos.

Por otro lado, la figura 4.6.b y c muestran las pruebas realizadas en un escenario con dos objetos simultáneos. Al igual que en el caso anterior, la posición inicial de los AIBOs permite una adecuada estimación de la posición usando visión estéreo respecto a los objetos en el campo de visión. Como se puede observar, en ambas pruebas los robots consiguen alcanzar su objetivo correctamente, siendo en la figura 4.6.b el AIBO negro es el más cercano al cubo naranja mientras que en la figura 4.6.c la situación es la contraria. Sin embargo, y como ya se comentó en el apartado 4.3.1.2, su aplicación está restringida a situaciones específicas en las que prácticamente ya se tiene una estimación de las posiciones de los robots, pues deben estar alineados y en el mismo plano, por lo que su uso difícilmente sería aplicable en un entorno real.

4.4.2. Localización basada en marcas fiduciarias

En este apartado se probará el sistema de localización basado en marcas fiduciarias, y concretamente en marcas ARToolkit. Este sistema de localización permite la localización de los robots a partir de la imagen actual, pero requiere que éstos tengan en su campo de visión una marca para poder localizarse mutuamente. Así pues, en esta prueba el objetivo será que los dos robots avancen hacia la marca en paralelo evitando chocar. De esta forma se asegura que los robots tengan la marca en todo momento en el campo de visión siempre y cuando no hagan giros demasiado bruscos.

En estos experimentos los robots se pueden comunicar entre ellos para enviar su posición al compañero, de forma que puedan determinar sus posiciones relativas. Respecto al intercambio de información, este se hará mediante Arquitectura en Capas Distribuída (DLA) (*Distributed Layered Architecture* en inglés), un sistema que permite una comunicación sencilla y transparente entre programas.

4.4.2.1. Arquitectura DLA

La arquitectura DLA [97, 98] es una arquitectura en capas, fácilmente aplicable a cualquier arquitectura de control, que permite una comunicación transparente y sencilla de forma no bloqueante entre los clientes DLA, usando para ello memoria compartida o sockets. Así pues, esta arquitectura facilita las comunicaciones tanto síncronas como asíncronas entre distintos sistemas.

El DLA, desarrollado dentro del propio grupo de investigación, está basado en un esquema cliente-servidor, de forma que el servidor tiene que ejecutarse en GNU/Linux, mientras que los clientes pueden ejecutarse tanto en GNU/Linux como en Microsoft Windows. Fue desarrollada en 2006 y se distribuye bajo Licencia Pública General de GNU (GPL).

Se ha usado este sistema porque es fácil y sencillo, además de permitir una fácil ampliación del sistema con más robots en caso de necesidad. De hecho, para el envío y recepción de información tan sólo es necesario crear una conexión con el DLA indicando el nombre asociado a la conexión y enviar los datos en el formato que se considere oportuno (texto plano o binario). De igual manera, para leer una conexión sólo es necesario saber su nombre identificativo y la codificación de los datos enviados a través de ella.

4.4.2.2. Implementación del sistema

En este apartado se describe la implementación del sistema usado para probar la localización basada en marcas ARToolkit. Esta implementación, al igual la anterior, hace uso del entorno Tekkotsu. Sin embargo, en este caso

la implementación está dividida entre el AIBO y el ordenador, pues no es posible introducir la librería ARToolkit en el AIBO.

Así pues, la figura 4.7 muestra el diagrama de bloques del sistema, siendo la función de los bloques más relevantes la siguiente:

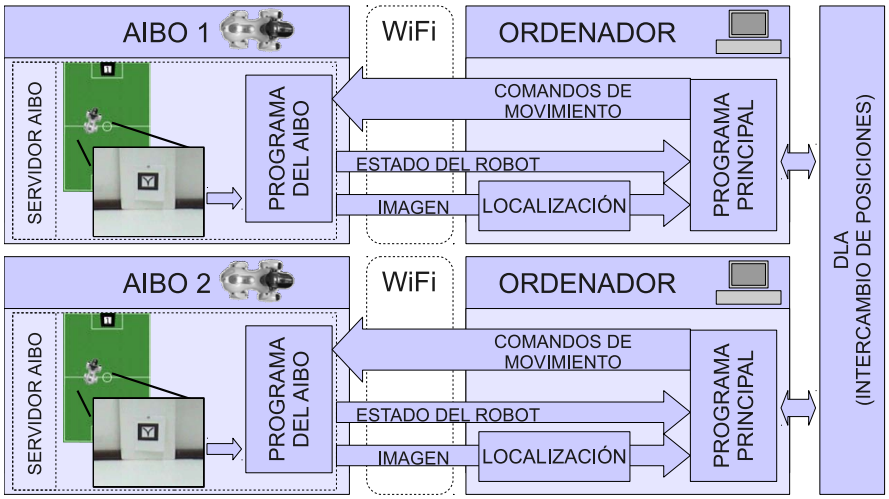

Figura 4.7: Diagrama de bloques del sistema.

- Programa del AIBO. El programa que se ejecuta en el AIBO se encarga de ejecutar en el robot los comandos de movimiento recibidos desde el programa del ordenador, así como enviar a éste las imágenes capturadas por el robot y su estado via WiFi. A partir de este momento, este módulo no cambiará, siendo su funcionalidad esperar la conexión del cliente para enviarle información y ejecutar los comandos de movimiento. Por lo tanto, a partir de ahora cuando se mencione el módulo Servidor AIBO se hará referencia a este módulo.

- Programa del ordenador. Este módulo, implementado en Java, muestra el interfaz gráfico de usuario y se encarga de la distribución de la información recibida entre los distintos módulos así como de la navegación, basada en PFA (sección 2.4.1.1), siendo el elemento atractor el objetivo a conseguir (la marca ARToolkit) y el repulsor el otro robot.

- Módulo de localización. Este modulo se encarga de establecer la posición relativa del robot respecto de la marca ARToolkit, devolviendo este parámetro al programa principal. Este módulo está implementado en C++ y se enlaza con el programa principal a través de JNI.

- DLA. Este módulo permite el intercambio de la posición de los robots respecto a la marca de forma fácil y sencilla.

Tras describir la implementación del sistema de navegación, a continuación se comentan las pruebas realizadas al sistema de localización basado en marcas ARToolkit.

4.4.2.3. Experimentos realizados

En esta sección se presentan los experimentos realizados con el sistema de localización basado en marcas ARToolkit. Para las pruebas se han usado dos robots AIBO ERS-7 y el entorno de las pruebas es una zona despejada de obstáculos de aproximadamente 2x2m^2. Dado que el objetivo es probar el sistema de localización, el comportamiento de los robots en sí es muy simple, siendo el objetivo que los dos robots se muevan hacia la marca sin colisionar ni interferirse, realizando una navegación segura entre ellos, para lo cual es necesario que estimen sus posiciones relativas de forma adecuada.

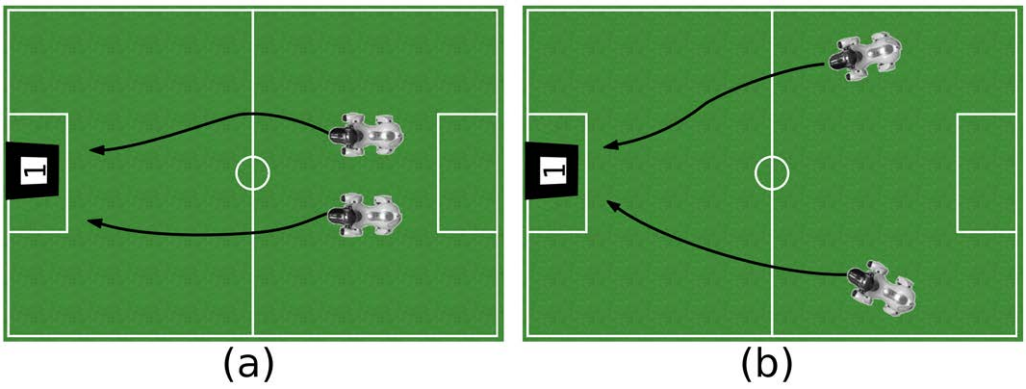

Figura 4.8: Navegación segura entre AIBOSs usando localización basada en marcas ARToolkit.

La figura 4.8.a muestra el resultado de una de las pruebas, en la que los robots comienzan muy cerca, lo que hace que el PFA los fuerce a alejarse, alcanzando finalmente la marca sin problemas. La figura 4.8.b muestra otro ejemplo en el que los robots están separados finalizando igualmente en las proximidades de la marca. Estas pruebas muestran que el uso de la localización basada en marcas de ARToolkit sería viable para la realización de comportamientos coordinados. Además, en este caso, a diferencia del anterior, el posicionamiento sólo requiere que los robots visualicen la marca, dependiendo la precisión de la localización de la distancia a la que esté el robot de la marca. Sin embargo, hay que destacar que había ocasiones en

las que se perdía la marca (debido al movimiento del robot o a una incorrecta identificación) lo que provocaba posiciones erróneas y que el robot se detuviese, momento en el cual volvía a detectar la marca (pues en general se perdía debido al movimiento del robot). En cualquier caso, en general el sistema de localización basado en marcas ARToolkit ha funcionado de forma adecuada en estas pruebas.

4.4.3. Localización basada en marcas y filtrado

En este apartado se prueba el sistema de localización basado en balizas de colores como las del campo de la Robocup con corrección de la posición mediante filtrado, concretamente mediante filtros de partículas.

Dado que este sistema de localización posee memoria, a diferencia de los anteriores, tan sólo requiere que los robots visualicen periódicamente alguna de las balizas de colores del campo para corregir y estimar su posición correctamente. De esta forma, se evita la necesidad de visualizar constantemente alguna de las balizas del entorno como en los otros dos ejemplos comentados.

En esta prueba, al igual que en la anterior, los robots estiman su posición usando el sistema de localización implementado, y la enviarán a su compañero mediante DLA, facilitando así la navegación segura entre ambos.

4.4.3.1. Implementación del sistema

En este caso, la implementación del sistema es idéntica a la mostrada en la figura 4.7, con la salvedad de que el sistema de localización en vez de usar la librería ARToolkit usa una librería desarrollada durante la realización de la tesis que permite la detección de las balizas de colores así como la corrección de esta posición usando un filtro de partículas.

Respecto a los parámetros del filtro de partículas, indicar que estaba compuesto por 500 partículas, la varianza del error de la velocidad usada en el modelo del sistema se ha estimado heurísticamente en 25 mm/s y la del error de posicionamiento del modelo de observación se ha estimado en 25 cm y 5º (usando coordenadas polares).

El resto de la implementación es idéntico, usándose el entorno Tekkotsu para el desarrollo general del sistema, JNI para enlazar el módulo de localización (en C++) con el programa principal el Java, y DLA para el intercambio de la posición de los robots, por lo que no se repetirá el esquema de bloques del sistema ni su descripción.

4.4.3.2. Experimentos realizados

Respecto a los experimentos realizados, dado que la localización basada en balizas de colores y filtro de partículas devuelve un posicionamiento concreto respecto a las balizas, al igual que ARToolkit respecto a sus marcas, y dado que el sistema usado era el mismo, el resultado de estas pruebas ha sido muy similar al de las pruebas anteriores realizadas con ARToolkit.

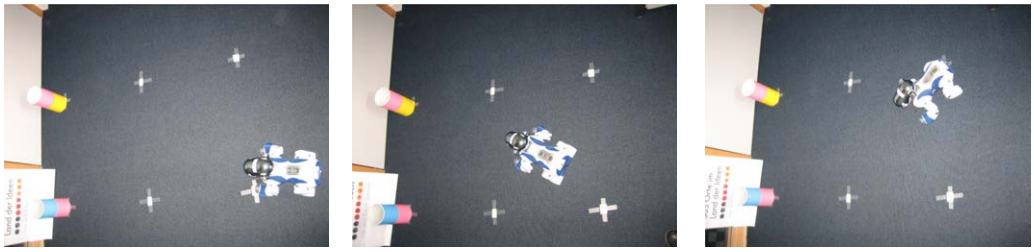

Figura 4.9: Pruebas de localización estática.

La principal diferencia observada entre estos dos sistemas de localización es que el posicionamiento obtenido con la localización basada en filtros de partículas es más estable que en el caso de ARToolkit.

Por último, la figura 4.9 muestra algunas pruebas de posicionamiento estático realizadas al sistema de localización basado en balizas de colores con corrección mediante filtro de partículas.

4.5. Conclusiones

El objetivo del presente documento es la realización de un sistema de coordinación multi-agente, siendo un elemento básico para poder realizar dicha coordinación la localización de cada agente.

Por lo tanto, en este capítulo se han presentado tres técnicas distintas para poder estimar la posición de los robots de un MRS. Cada una de estas técnicas tiene sus ventajas y desventajas, pues unas permiten un posicionamiento preciso y estable, pero no encajan en el esquema reactivo presentado en el capítulo anterior, mientras que otras encajan mejor en un enfoque más reactivo pero poseen otras desventajas.

Concretamente, los sistemas de localización probados han sido un sistema de localización basado en objetos comunes en el campo de visión usando visión estéreo y comunicación entre los robots, un sistema de localización basado en marcas fiduciarias (concretamente marcas ARToolkit) con comunicación para determinar las posiciones relativas de los robots, y un sistema de localización basado en balizas de colores como las de la Robocup y corrección

del posicionamiento mediante filtros de partículas, habilitando igualmente la comunicación entre los robots.

Para probar cada uno de estos sistemas se han realizado pruebas con dos robots AIBO ERS-7 en un entorno despejado de obstáculos. Las pruebas han consistido en conseguir que los robots naveguen de forma segura entre ellos mientras tratan de alcanzar su propio destino (el cual depende de las pruebas realizadas).

En todas las pruebas realizadas se ha conseguido que los robots navegasen evitando colisionar e interferirse. Sin embargo, eso no quiere decir que todos los sistemas de localización ofreciesen las mismas prestaciones. Así pues, en el sistema de localización basado en visión estéreo usando como referencia los objetos comunes en el campo de visión, a pesar de ser la localización con un modelado del entorno menos restrictivo y no usar memoria, el posicionamiento era muy impreciso y de hecho la aplicación de esta localización prácticamente requería tener una estimación de la posición de los robots, pues éstos deben estar alineados y en el mismo plano. Así pues, este sistema no se considera adecuado para ser usado en el sistema de coordinación que se pretende realizar, a pesar de ser la localización que mejor encaja con un enfoque reactivo.

El segundo sistema de localización propuesto se trata de una localización basada en marcas ARToolkit. Esta localización ha ofrecido un posicionamiento adecuado aunque la localización obtenida no era del todo estable y en ocasiones era difícil obtener un valor debido al movimiento de la cámara. Sin embargo, en general se considera que el resultado ha sido adecuado permitiendo, como se ha mostrado, que los robots navegasen de una forma reactiva evitando colisionar, por lo que este sistema es una alternativa a tener en cuenta para la localización del sistema de coordinación.

Por último, el sistema de localización basado en balizas de colores y filtros de partículas ha sido el que ha devuelto un posicionamiento más estable. Sin embargo, este sistema de localización no encaja con el enfoque reactivo propuesto en el capítulo 3 debido a que requiere un modelado del entorno más estricto así como al uso de memoria (histórico de observaciones).

Por lo tanto, y como conclusión, para el sistema de coordinación multiagente se optará por el uso del sistema de localización basado en marcas ARToolkit porque se considera que ofrece un posicionamiento adecuado como para ser usado en el sistema de coordinación a realizar. Como alternativa, si este sistema de localización mostrase alguna deficiencia o problema al probarse más exhaustivamente, se optaría por el enfoque de localización basada en balizas de colores y corrección con filtros de partículas, aunque el sistema perdiese su enfoque reactivo inicial.

Capítulo 5

Coordinación reactiva basada en aprendizaje

5.1. Introducción

Los MRS tienen una serie de ventajas frente a los sistemas robóticos individuales [52, 152, 156], ya que los MRS mejoran la resistencia a fallos, el área de cobertura y el tiempo necesario para realizar una determinada tarea. Por contra, estos sistemas son más complejos de diseñar que un sistema robótico individual, pues requieren una correcta coordinación entre los distintos robots para funcionar de forma eficiente. Por lo tanto, la coordinación es un elemento clave en los MRS, pues sin una correcta coordinación los distintos robots se entorpecerían en vez de ayudarse, perdiendo todas sus ventajas frente a un robot individual.

La coordinación, de forma general, puede dividirse en coordinación explícita [12,35] y coordinación implícita [81,107]. La coordinación explícita consiste en que los robots del sistema se coordinen de forma explícita entre ellos, es decir, que las acciones o tareas de cada robot se deciden de forma conjunta. Generalmente, este tipo de coordinación requiere que los robots lleguen a un acuerdo en común o que un robot o agente centralizado decida que deben hacer los demás robots. Por su parte, la coordinación implícita consiste en la coordinación surgida de la interacción entre los distintos robots a partir de las decisiones individuales de cada uno de ellos. Es decir, en este tipo de coordinación cada robot toma sus propias decisiones, pudiendo tener en cuenta o no la información de los demás robots, pero las decisiones son tomadas individualmente. Así pues, la coordinación surge, generalmente, en la forma de un comportamiento emergente, similar al que se observa en las colonias de hormigas, termitas o abejas.

De entre estos dos tipos de coordinación, la coordinación implícita es más adaptable a cambios dinámicos en el entorno, debido a que los robots actúan por sí mismos sin necesidad de llegar a ningún acuerdo entre ellos, por lo que el tiempo de respuesta ante cambios en el entorno suele ser menor que en sistemas coordinados explícitamente. Además, la coordinación implícita resulta más sencilla de encajar en una arquitectura basada en comportamientos, como la mostrada en el capítulo 3, que la coordinación explícita. La razón es que para conseguir un comportamiento coordinado implícito, partiendo de comportamientos reactivos, tan sólo es necesario que éstos comportamientos tengan en cuenta a los otros robots para actuar de acuerdo a la distribución e información del sistema, evitando colisionar e interferirse. Así pues, este tipo de coordinación es casi inmediata de conseguir a partir de comportamientos reactivos de una manera sencilla.

Por lo tanto, y partiendo del trabajo presentado en el capítulo 3, en este capítulo se presenta un sistema coordinado basado en el uso de comportamientos reactivos aprendidos mediante LfD usando CBR. Así pues, y al igual que en dicho capítulo, el entrenamiento se realiza controlando al robot de forma remota, obteniendo de esta forma las ventajas ya comentadas en la sección 3.4.3, como que no es necesario ningún modelo cinemático explícito del robot o que los errores sistemáticos y mecánicos son absorbidos implícitamente por el operador durante el entrenamiento [47, 102]. Además, para implementar comportamientos con distintas respuestas, tan sólo es necesario definir los parámetros que definen el caso del nuevo comportamiento y entrenar de nuevo el sistema para que realice el comportamiento deseado. Por otro lado, si durante el entrenamiento el operador tiene en cuenta la posición de los otros robots en el entorno, el comportamiento aprendido será un comportamiento coordinado implícitamente, ya que las respuestas del comportamiento dependerán, al menos en parte, del otro robot. De esta forma se consigue una coordinación implícita de un modo inmediato y sencillo, pues cada robot actúa y se mueve de acuerdo a su propio entrenamiento y no por un acuerdo con los otros robots, aunque esté teniendo en cuenta a los demás robots para realizar sus movimientos.

Así pues, para que los comportamientos puedan tener en cuenta a los demás robots es necesario que los robots conozcan tanto su posición como la de los compañeros, por lo que se hace necesario un sistema de localización. Como se comentó en el capítulo 4, de los distintos sistemas de localización considerados, el que se usará para las pruebas será la localización basada en marcas ARToolkit, pues ofrecía un posicionamiento de una forma sencilla y permitía mantener el enfoque reactivo propuesto anteriormente. Por otro lado, para que los robots puedan conocer la posición de sus compañeros y fa-

cilitar la coordinación del sistema se permitirá la comunicación e intercambio de información (como la posición) entre ellos.

Para comprobar la viabilidad del enfoque propuesto, se ha realizado una prueba de concepto que consiste en que dos robots AIBO ERS-7 realicen un movimiento coordinado evitando colisionar entre ellos mediante aprendizaje. Esta prueba de concepto es similar a la del capítulo 3, aunque en este caso sí se tiene en cuenta otro robot en movimiento en el entrenamiento. Para determinar la posición de los robots y que puedan coordinarse adecuadamente sin colisionar se usará, como se ha comentado, localización basada en marcas ARToolkit. Por último, para el intercambio de las posiciones de los robots se usará DLA [97,98], ya que permite una comunicación sencilla y transparente entre programas.

Por lo tanto, el capítulo se estructura de la siguiente manera. En primer lugar, se realiza una introducción a la coordinación y a las estrategias de coordinación más usuales (sección 5.2). Dado que la implementación se realiza mediante comportamientos basados en aprendizaje usando CBR, se presentará también un breve estado del arte de CBR aplicado a coordinación (sección 5.3). Tras esto, se describirá el sistema de coordinación propuesto (sección 5.4). Para comprobar la viabilidad del sistema se presentará una prueba de concepto consistente en el entrenamiento de dos robots AIBO ERS-7 para que se muevan de forma coordinada en paralelo sin colisionar entre ellos. Finalmente, se presentan los resultados de esta prueba de concepto (sección 5.5) y las conclusiones del capítulo (sección 5.6).

5.2. Coordinación

Aunque existen distintas definiciones de coordinación, pero se podría entender como la administración de las interdependencias entre distintas actividades o acciones. Teniendo en cuenta esta definición, la coordinación en los MRS resulta fundamental, pues es la que permite administrar las dependencias entre las acciones que realizan los robots evitando así el caos y la anarquía entre los miembros del sistema y permitiendo la eficiencia del MRS frente a los sistemas individuales.

Como es lógico, la coordinación entre distintos robots puede conseguirse de muchas formas distintas, sin embargo, en general la coordinación se puede dividir en dos grandes grupos [55,153]:

- Coordinación explícita. En la coordinación explícita, los robots se comunican explícitamente entre ellos y las acciones de cada agente en el grupo son calculadas de forma expresa. En este tipo de coordinación,

las acciones del equipo pueden ser calculadas para determinar la mejor acción individual, evitando la duplicación de acciones y minimizando el esfuerzo. Sin embargo, este enfoque tiende a ser poco flexible y poco tolerante a fallos, debido a que en general suele haber un elemento que centraliza la coordinación. Por otro lado, si el número de componentes del sistema crece suficientemente el coste computacional y la cantidad de mensajes para establecer la comunicación entre los robots pueden resultar excesivos en este tipo de coordinación. Así pues, la coordinación explícita es habitualmente usada en entornos pocos dinámicos, pues no responde adecuadamente a cambios rápidos en el entorno, con equipos de pequeño tamaño.

- Coordinación implícita. La coordinación implícita se basa en la dinámica de la interacción entre los robots y el entorno para lograr el objetivo colectivo, a menudo en la forma de un comportamiento emergente. Este comportamiento emergente surge, pues, de la combinación de otros comportamientos más sencillos [33, 84, 101] que realizan cada uno de los robots por separado. Así pues, en este enfoque, los robots no trabajan unidos explícitamente, sino que cada uno actúa por su cuenta, tomando en cuenta la existencia de los otros robots, para modificar sus acciones o simplemente para evitar estar en su camino. Las acciones de los agentes en este enfoque generan una acción combinada emergente [153] que se acepta como cooperativa, tales como las acciones de las colonias de insectos [36, 40], por ejemplo. Este mecanismo es típico en animales y es bastante eficiente cuando todos los robots son similares y cuando hay gran cantidad de miembros en el equipo. Por otro lado, la independencia de los robots del sistema en la coordinación implícita hace que este enfoque sea más adecuado para entornos dinámicos pues las respuestas del sistema resultan más rápidas y flexibles. Por contra, y respecto a la coordinación explícita, este tipo de coordinación puede ser menos eficiente (duplicación de tareas por falta de comunicación o interferencias entre los robots).

  Es importante destacar que, a pesar de que cada robot actúe por sí mismo, los distintos miembros del sistema pueden compartir expresamente información como su posición o los objetos que están captando, siempre y cuando la toma de decisiones se realice de forma autónoma, aunque influenciada por la información de los otros miembros. Sin embargo, y a diferencia de la coordinación explícita, la coordinación implícita suele requerir poca comunicación o ninguna entre los distintos miembros.

Como se puede observar, ambos enfoques son muy diferentes y cada uno

de ellos tiene un ámbito de aplicación distinto, que dependerá de las circunstancias y el problema concreto a resolver. En el caso presente, se ha optado por usar coordinación implícita por varias razones. Por un lado, porque se adapta más fácilmente a entornos dinámicos y a cambios en el entorno que la coordinación explicita. Por otro lado, la coordinación implícita es más sencilla de implementar a partir de una estructura basada en comportamientos como la propuesta, pues tan sólo es necesario que los comportamientos tengan en cuenta a los otros robots para actuar de forma coordinada.

5.3. CBR aplicado a Coordinación

Entre las distintas técnicas de IA aplicadas a la coordinación, se encuentran por ejemplo las Redes Bayesianas, los GA o el CBR [5]. Concretamente este último es el que se va a utilizar en el presente trabajo, como ya se ha comentado.

El CBR ha sido aplicado anteriormente para coordinación, por lo general a alto nivel. Así por ejemplo, [15] presenta un sistema que permite la cooperación de dos robots para un comportamiento de pase de pared basado en CBR. Concretamente, el CBR es usado para determinar si la situación es adecuada para realizar un pase de pared y, en ese caso, activar la secuencia de acciones que permite el comportamiento coordinado de pase de pared. Un enfoque similar muestra [113], donde el CBR es usado para la toma de decisiones. Concretamente, un robot es el encargado de obtener, en base a la situación actual, el caso CBR más adecuado y se lo envía al resto de robots para que todos ejecuten la secuencia de acciones (determinada por la salida del CBR) de forma coordinada. Otro ejemplo de aplicación del CBR para coordinación es [115], donde el CBR se utiliza para la selección de los roles de los robots y planificación a largo plazo de la coordinación de un equipo de robots. Otro uso distinto del CBR para coordinación sería el propuesto en [10], donde el CBR permite configurar los parámetros que determinan la coordinación de un sistema.

En el presente trabajo, y a diferencia de los trabajos mencionados, el CBR será aplicado, directamente, para implementar los propios comportamientos coordinados, ya que el mismo comportamiento, al aprenderse, tiene en cuenta a los demás robots.

5.4. Coordinación basada en comportamientos reactivos aprendidos

El objetivo en este capítulo implementar un sistema que permita la coordinación de varios robots a partir de comportamientos reactivos aprendidos mediante LfD usando CBR. Concretamente, lo que se pretende es comprobar que el sistema de aprendizaje presentado en el capítulo 3, que demostró en los experimentos que era flexible y viable para el aprendizaje de un robot individual, permite también su uso para la implementación de comportamientos coordinados entre varios robots.

De esta forma, se conseguiría un sistema flexible que permitiría la creación de diferentes comportamientos coordinados usando el mismo programa, siendo necesario tan sólo definir los parámetros del caso a usar y entrenar el comportamiento. Además, al usar el sistema presentado en el capítulo 3, se consiguen las ventajas ya comentadas, como que no es necesario crear ningún modelo cinemático del robot o que los errores sistemáticos y mecánicos son absorbidos implícitamente por el operador durante el entrenamiento.

Hay que destacar que en este capítulo, al igual que en el capítulo 3, se trabajará únicamente a nivel reactivo, por lo que los comportamientos obtenidos no serán demasiado complejos. Para obtener comportamientos más complejos sería necesario añadir una capa de nivel superior [103] que permita activar y desactivar los comportamientos en función de la situación.

La coordinación entre los robots será una coordinación implícita, como ya se ha comentado, pues es más adecuada para entornos dinámicos y resulta más sencilla de encajar en un sistema basado en comportamientos. De hecho, si durante el entrenamiento mediante control remoto el operador tiene en cuenta los distintos robots en el entorno, el propio aprendizaje absorbe de forma implícita el comportamiento coordinado con los demás robots. De esta forma se consigue, de un modo sencillo, una coordinación implícita, pues cada robot actúa de forma independiente, pero teniendo en cuenta a los demás robots.

Por otro lado, para que los robots se puedan tener en cuenta mutuamente se usará un sistema de localización común basado en marcas visuales artificiales, como ya se comentó en el capítulo 4, y se permitirá la comunicación entre los robots para intercambiarse mutuamente su posición. Concretamente, se usarán marcas ARToolkit pues permiten un fácil posicionamiento a partir de la imagen actual, por lo que no requiere una componente temporal, aunque sí un mínimo modelado del entorno. Así pues, en el presente capítulo en vez de relacionar mediante el CBR características visuales de bajo nivel a los comandos de movimiento a realizar por el robot, como se hacía en el capítulo

3, lo que se asociará serán las posiciones de los robots con los comandos de movimiento a realizar.

Respecto al intercambio de información entre los robots, esta se realizará mediante DLA (apartado 4.4.2.1). Se ha escogido este sistema de intercambio porque su uso resulta sencillo y permite fácilmente extender el sistema independientemente del número de robots usados.

Para estudiar la viabilidad del enfoque propuesto y las posibilidades de coordinación que ofrece el sistema presentado, se realizarán pruebas con dos robots AIBO ERS-7 en un entorno sin obstáculos usando localización basada en marcas ARToolkit, como se ha comentado. Concretamente para los experimentos se realizará una prueba de concepto consistente en la implementación de un comportamiento coordinado entre los dos robots, los cuales deben moverse en paralelo a una distancia determinada (según el entrenamiento realizado) mientras se aproximan a la portería contraria.

El comportamiento a realizar resulta sencillo, pero hay que recordar que tan sólo es una prueba de concepto que permitirá determinar si el enfoque propuesto es viable y permite, además de la navegación individual de un robot, la navegación coordinada de varios robots. También hay que tener en cuenta que este comportamiento es más complejo que los mostrados anteriormente, pues ambos robots están en movimiento y la cantidad de situaciones potenciales crece respecto a las mostradas para un único robot. Así pues, esta prueba permitirá comprobar si el entrenamiento resulta mucho más complejo que en el caso de un único robot y que factores hay que tener en cuenta para realizar el entrenamiento de la forma más sencilla y rápida posible. Además, esta prueba de concepto permitirá estimar los posibles problemas del enfoque actual para un sistema coordinado y la necesidad de usar una capa superior o no.

En los siguientes apartados se describe con más detalle el comportamiento coordinado, los elementos más importantes del sistema y su funcionamiento e implementación.

5.4.1. Comportamiento coordinado en paralelo

El comportamiento coordinado en paralelo tiene por objetivo, como se ha comentado, conseguir que dos robots AIBO se muevan en paralelo de forma coordinada mientras se aproximan hacia su destino (la portería contraria). La distancia a la que se moverán dependerá del entrenamiento realizado. Así pues, si durante el entrenamiento se mueven cerca, tenderán a replicar este comportamiento en modo autónomo y viceversa.

El entorno de las pruebas es una zona libre de obstáculos con varias marcas ARToolkit donde están únicamente los dos robots AIBO. La posición

de la portería respecto a las marcas ARToolkit se conoce a priori.

Para el funcionamiento de este comportamiento los robots deberán detectar en la imagen una de las marcas ARToolkit que habrá en el entorno, pudiendo determinar la posición de la portería contraria a partir de las marcas. Si alguno de los robots dejase de captar alguna de las marcas del entorno, el robot se pararía y empezaría a girar hacia la izquierda hasta que visualice alguna de las marcas y pueda posicionarse.

Para el correcto funcionamiento del comportamiento, dado que tiene en cuenta la posición del otro robot, también es necesario tener esta información. Para ello los robots se intercambian sus posiciones a través de DLA. De esta forma, ambos robots puedan conocer sus posiciones relativas y actuar en consecuencia. De no permitirse la comunicación entre los robots, como se ha comentado en los capítulos anteriores, las posibilidades de coordinación entre éstos se verían limitadas a los momentos en los que se detectasen directamente o a tareas en las que tuviesen un objetivo común, como empujar un objeto todos a la vez, por ejemplo.

Hay que tener en cuenta que, a pesar de ser un comportamiento sencillo, no consiste únicamente en moverse recto en paralelo junto al compañero para alcanzar la portería, pues los robots se acercarán y se alejarán y tendrán que estar corrigiendo la distancia a la que se encuentran el uno del otro, por lo que este comportamiento tiene una dependencia considerable del compañero, a pesar de su simplicidad.

Otro problema potencial es la posibilidad de colisión entre los dos robots. Para evitar posibles colisiones, se ha implementado de forma analítica un comportamiento que, en caso de colisión inminente, detiene al robot, permitiéndole únicamente girar.

5.4.2. Localización basada en marcas ARToolkit

Para poder desarrollar el sistema de localización se ha usado la librería ARToolkit [64,82], ya comentada en el capítulo anterior. La principal ventaja de esta librería es que permite obtener la posición del robot respecto a la marca visualizada a partir de una única imagen, no siendo necesario recordar las posiciones anteriores y manteniendo, en ese sentido, el enfoque de los comportamientos reactivos.

Por lo tanto, para poder obtener la posición de la portería respecto al robot bastaría, en principio, con tener una marca en una posición conocida respecto a la portería. Sin embargo, y para facilitar que los robots puedan tener en el campo de visión siempre una marca que les permita posicionarse respecto a la portería contraria (dado que la posición de las marcas respecto a la portería es conocida), se usará la configuración de marcas mostrada

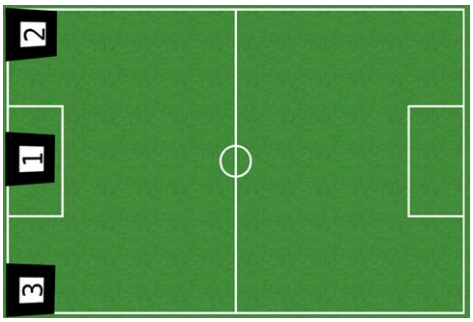

Figura 5.1: Campo con marcas ARToolkit para localización.

en la figura 5.1. De esta forma se reducirán posibles problemas derivados del posicionamiento, pudiéndose centrar la atención en el comportamiento coordinado propiamente dicho.

5.4.3. Aprendizaje con varios robots: entrenamiento secuencial

El aprendizaje de los comportamientos de los robots, al igual que se comentó en el capitulo 3, se hará mediante LfD [8, 130], es decir, enseñando a los robots el comportamiento coordinado que deben realizar mientras son controlados remotamente. De esta forma se consigue que los robots aprendan a asociar su objetivo (la portería contraria) y la posición relativa del otro robot a los comandos de movimiento a realizar en cada situación concreta.

Sin embargo, en este caso, a diferencia del entrenamiento comentado en el capitulo 3, para poder entrenar los dos robots serían necesarios dos operadores, cada uno de los cuales enseñaría a navegar al robot acorde a su propia forma de moverse. Esto quiere decir que la forma de moverse del robot, así como la seguridad de la trayectoria, depende de quien maneja al robot durante los entrenamientos. Así pues, si no se puede disponer de dos operadores para los entrenamientos, o se prefiere controlar todo el proceso de aprendizaje de los robots, una alternativa al entrenamiento simultáneo de los robots es la realización de un entrenamiento secuencial.

El entrenamiento secuencial consiste en dos fases. En una primera fase, se entrena a un robot inicial (R_i) con un comportamiento concreto. Tras esto, el robot entrenado (R_i) se pone a funcionar en modo autónomo ejecutando el entrenamiento realizado (sin tener en cuenta a ningún otro robot) mientras se entrena en paralelo al segundo robot (R_{coord}) para que tenga en cuenta al primero (R_i). En principio, con estas dos fases sería suficiente para realizar un entrenamiento coordinado (el del robot R_{coord}), el cual podría aplicarse a

ambos robots para que navegasen de forma coordinada. Sin embargo, si se desea realizar un entrenamiento más realista, se podría aplicar una tercera fase en la que el entrenamiento coordinado del robot R_{coord} se aplicase a uno de los robots mientras se vuelve a realizar un entrenamiento al otro robot, teniendo en este caso ambos robots en cuenta a su compañero.

5.4.4. Funcionamiento del sistema

El sistema coordinado basado en comportamientos aprendidos parte, como se ha comentado, del sistema de navegación desarrollado en el capítulo 3. Así pues, el funcionamiento básico del sistema será el mismo que ya se comentó anteriormente, estando formado por dos fases:

- Aprendizaje supervisado. Durante esta fase se crea la base de casos que será utilizada posteriormente durante la fase de navegación autónoma. Esta fase es necesaria si no hay ningún aprendizaje anterior que se pueda usar para los comportamientos o si se desea realizar un nuevo comportamiento no aprendido anteriormente. Concretamente, durante el aprendizaje supervisado se realiza el entrenamiento mediante control remoto de los robots (como ya se ha comentado, o los dos en paralelo o usando un entrenamiento secuencial) enseñando a los robots a realizar el comportamiento buscado. Así pues, el resultado de esta fase será la experiencia o base de casos que utilizará el CBR cuando trabaje de forma autónoma (siguiente fase) para obtener soluciones a la situación actual a partir de la experiencia adquirida.

- Navegación autónoma. Durante la fase de navegación autónoma el robot funcionará de forma autónoma sin supervisión, enviando al CBR información sobre el entorno y el estado de los robots, de forma que el CBR devolverá los comandos de movimiento previamente aprendidos que más se adecúen a la situación actual.

5.4.5. Implementación del sistema

En este apartado se describe la implementación del sistema de coordinación basado en comportamientos aprendidos. Como se ha comentado, este sistema está formado por dos fases que son entrenamiento supervisado y navegación autónoma, por lo que el sistema debe ser capaz de alternar entre estas dos fases.

Así pues, la figura 5.2 muestra el diagrama de bloques general, en el que se puede apreciar como el sistema puede conmutar entre los comandos provenientes del joystick (el usuario en modo entrenamiento) o el CBR (robot

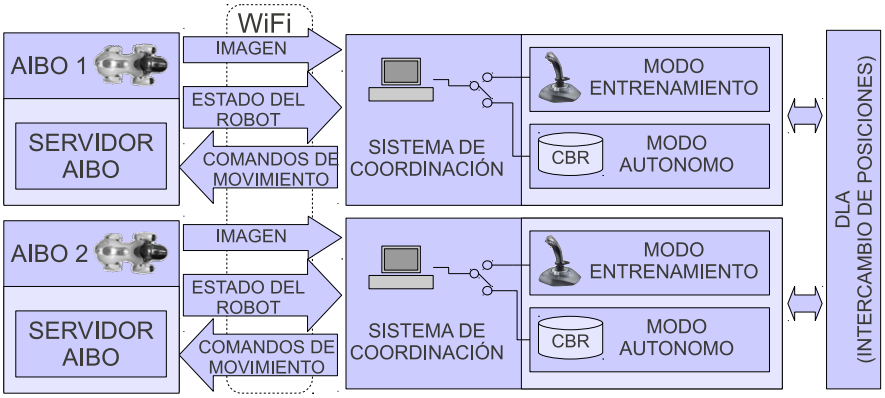

Figura 5.2: Diagrama de bloques general del sistema. El envío de comandos al robot puede conmutar entre el joystick (modo entrenamiento) y el CBR (modo de navegación autónoma).

entrenado y funcionando en modo autónomo). Por lo tanto, para el entrenamiento supervisado, que como se ha explicado se hace de forma secuencial, en primer lugar se entrenaría uno de los robots, conmutando a modo entrenamiento y desconectando la comunicación con el DLA y, tras esto, el robot entrenado conmutaría a modo de navegación autónoma mientras se entrena al segundo robot en el comportamiento coordinado (con la comunicación por DLA habilitada). Respecto al funcionamiento en modo autónomo, tan sólo habría que habilitar el envío de comandos desde el CBR en ambos robots.

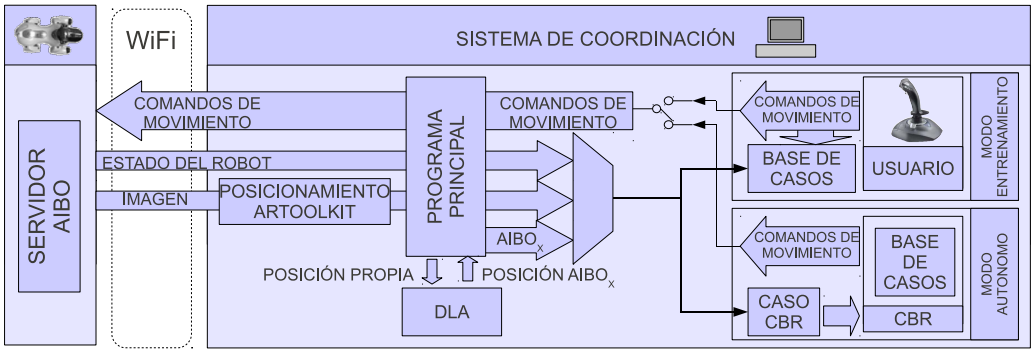

Figura 5.3: Diagrama de bloques del sistema de coordinación.

De los módulos mostrados en la figura 5.2, el módulo Servidor AIBO es igual al presentado en la sección 4.4.2.2, mientras que el diagrama de bloques del sistema de coordinación se detalla en la figura 5.3. Como se puede observar, el diagrama es muy similar al del capítulo 3 aunque cambiando el módulo de procesamiento de imagen por uno que permite el posicionamiento

usando marcas ARToolkit y añadiendo comunicación entre los dos sistemas de coordinación de los robots mediante DLA, lo que permite intercambiarse sus posiciones respecto a la marca y añadir información sobre el posicionamiento del compañero al CBR. De hecho, y como se puede observar en la figura 5.3, la entrada del CBR está formada por el destino relativo del propio robot (obtenido a partir del posicionamiento mediante ARToolkit), el estado del robot (obtenido directamente del robot) y el posicionamiento del compañero $AIBO_x$ (obtenido a través de DLA).

A continuación se comentan en más detalle las funciones y la implementación de los distintos módulos mostrados en la figura 5.3, que son:

- Servidor AIBO. Este programa es el mismo que ya se mostró en el capítulo 4 (sección 4.4.2.2), por lo que está implementado usando Tekkotsu y se encarga de enviar al ordenador el estado y las imágenes capturadas por el robot y ejecutar los comandos recibidos de éste. La comunicación entre el robot y el ordenador se realiza via WiFi.

- Sistema de coordinación. Este es el programa que se ejecuta en el ordenador y permite el entrenamiento y la coordinación (en modo autónomo) de los robots. Está formado principalmente por los siguientes módulos:

  - Programa principal. Este módulo está desarrollado en Java (excepto el módulo de posicionamiento por ARToolkit) usando el entorno Tekkotsu para la comunicación y control remoto del AIBO. La funcionalidad de este módulo es similar a la comentada en el apartado 3.4.5, aunque no exactamente igual. Así pues, este módulo es el encargado de controlar, gestionar y distribuir toda la información que llega del robot y del DLA a los distintos módulos que componen el sistema de coordinación. Este módulo, además, se encarga de mostrar los interfaces gráficos para mostrar la imagen de la cámara y el estado del robot al usuario, permitiendo un sencillo control del robot mediante teclado o joystick para el entrenamiento.

    Por lo tanto, el programa principal se encarga de recibir la imagen del robot, los parámetros de estado de éste y la posición estimada del robot respecto a la portería (obtenida del módulo de posicionamiento) así como de enviar la posición propia al compañero y de leer la posición del robot compañero a través de DLA.

    Concretamente, en modo entrenamiento, este módulo se encarga de enviar al robot los comandos de movimiento realizados por

el operador mediante el joystick, generando en paralelo el caso a almacenar en la base de casos CBR a partir del destino a alcanzar, el estado del robot, la posición del compañero y los comandos de movimiento del usuario. Por otro lado, en modo navegación autónoma, este módulo envía el destino del robot, su estado y la posición del compañero al módulo CBR para recuperar, de la base de casos, el caso más parecido, obteniéndose así el comando de movimiento entrenado previamente, el cual será enviado al robot para que lo ejecute.

- Módulo de posicionamiento mediante ARToolkit. Este módulo, implementado en C++, es el encargado de obtener la posición del robot respecto a la portería mediante la librería ARToolkit, llamada desde la aplicación principal mediante JNI. Así pues, cada vez que recibe una imagen la procesa y devuelve la posición desde la que se está visualizando la marca en el campo de visión, lo cual permite obtener la posición del robot respecto a la portería (dado que se conoce la posición de las marcas respecto a la portería).
- Módulo CBR. Este módulo es idéntico al presentado en el apartado 3.4.5, siendo su objetivo permitir el acceso a la base de casos CBR.

Para finalizar de detallar la implementación del sistema realizado es necesario describir los distintos parámetros de configuración del CBR, lo cual se hará en los siguiente apartados.

5.4.6. Descripción del CBR

En esta sección se detallan los parámetros que describen el CBR, como la estructura de la memoria de casos del CBR, los parámetros que forman los casos o los distintos algoritmos utilizados en cada una de las fases.

5.4.6.1. Estructura de la base de casos

La estructura de la base de casos, al igual que cuando se entrenó un único robot, será una estructura plana. A pesar de que el comportamiento coordinado a desarrollar no es especialmente complejo, pero es más complejo que en el caso de un único robot del capítulo 3, por lo que la base de casos estará formada por una mayor cantidad de casos. Para mantener acotado su número, se aplicará un algoritmo de agrupación sobre la base de casos para que el sistema CBR no reduzca sus prestaciones.

El algoritmo de agrupación utilizado en primer lugar selecciona una semilla y comprueba si hay casos similares en la base de casos. Dos casos se

consideran similares si su distancia es menor a un umbral D_{umbral} (establecido heurísticamente). Todos los casos que pertenezcan a un mismo grupo son promediados en un prototipo que se convierte en el nuevo caso de la base de casos, sustituyendo la semilla inicial. Tras esto, se selecciona una nueva semilla y se realiza el mismo proceso hasta que todos los casos de la base de casos hayan sido procesados por el algoritmo. Este algoritmo permite mantener acotado el número de casos de la base de casos, así como reducir la redundancia y fluctuaciones o movimientos espasmódicos debido a casos con entradas similares y salidas contradictorias o parcialmente distintas.

El algoritmo de agrupación es especialmente interesante cuando diferentes bases de casos (provenientes de distintos entrenamientos) son combinadas para formar una única base de casos que aune los entrenamientos en uno sólo. En este caso, es posible que la base de casos combinada posea casos que se correspondan a situaciones similares pero tengan soluciones distintas o incluso contradictorias, las cuales serían unificadas mediante el algoritmo de agrupación.

Respecto a la función de similitud se ha usado la distancia Manhattan porque evalúa la distancia en función de la envolvente del vector, y no de las componentes individuales, lo cual resulta más conveniente en este caso debido a que las componentes del caso (como se indica en el siguiente apartado) no están relacionadas entre así.

5.4.6.2. Definición del caso

Esta sección presenta la definición del caso para un robot aislado (usado durante el entrenamiento secuencial) y para un comportamiento que permita la coordinación de dos robots. Sin embargo, antes de comenzar con la definición del caso es importante recordar que un caso es un vector formado por dos componentes entrada-salida, que describen un problema y su solución ($\{P, S\}$) respectivamente. Al igual que en el capítulo 3 no se añade una componente para la eficiencia del caso pues ésta será determinada por el operador mediante observación directa durante la navegación autónoma, considerando que los casos son correctos si el sistema es capaz de realizar la tarea para la que ha sido entrenado.

También hay que recordar que respecto a la parametrización de la entrada del caso, es conveniente introducir únicamente la información necesaria para describir la situación del entorno, lo que favorecerá que el sistema CBR trabaje mejor al eliminar toda la redundancia posible en el caso de entrada.

Por lo tanto, y teniendo esto en cuenta, la mínima información necesaria para describir el caso de entrada, dado que el robot debe alcanzar un destino y tener en cuenta a su compañero para coordinarse con él, serán la posición del

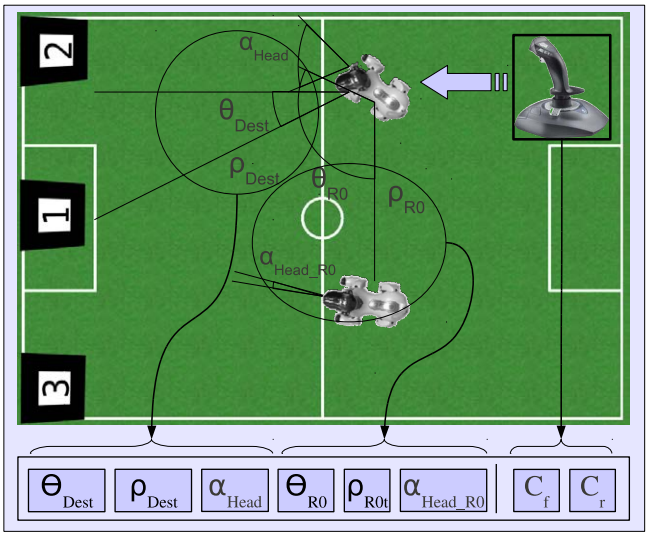

Figura 5.4: Parámetros usados en la definición del caso CBR y definición del caso CBR para el entrenamiento coordinado.

destino y la posición del compañero. En el caso de entrenamiento secuencial, el robot que se entrena en primer lugar de forma aislada sólo tendrá información de su destino, rellenándose los parámetros de la posición del compañero a un valor inválido. En la figura 5.4 se muestran los parámetros que componen el caso de entrada tanto para 1 como para 2 robots. Respecto a la salida, al igual que en el capítulo 3 estará formada por los comandos de movimiento que envíe el operador mediante el joystick al robot, normalizados de -1 a 1, siendo 1 la velocidad máxima permitida por el robot hacia delante/izquierda, y -1 hacia atrás/derecha. El 0 indica parada.

Así pues, la ecuación 5.1 muestra la definición del par $\{P, S\}$ usada para el sistema de coordinación. Como se puede observar, en el caso de realizar el entrenamiento con más robots, tan sólo sería necesario añadir a la definición del caso otro conjunto de parámetros para indicar el posicionamiento del nuevo robot del equipo. Por lo tanto, la definición del caso es:

$$Case = \{\{\rho_{Dest}, \theta_{Dest}, \alpha_{Head}, \rho_{R0}, \theta_{R0}, \alpha_{Head\_R0}\}, \{C_f, C_r\}\} \tag{5.1}$$

Siendo $(\rho_{Dest}, \theta_{Dest})$ la posición del destino (la portería) relativa al robot en polares, α_{Head} la orientación de la cabeza respecto al cuerpo del robot en grados, (ρ_{R0}, θ_{R0}) la posición relativa del compañero en polares, α_{Head_R0} la orientación de la cabeza respecto al cuerpo del compañero en grados, y C_f y C_r los comandos de movimiento frontal y de rotación a enviar al robot

normalizados entre -1 y 1.

En el caso de que el robot no pueda obtener su posición respecto a las marcas, enviará un conjunto de parámetros inválidos $(-1, -1, -1)$ [1] para indicar que no se puede establecer su posición.

5.4.6.3. Descripción de la fase de recuperación

Para la descripción de esta fase es necesario especificar tanto el algoritmo de búsqueda que se usará como la distancia de similitud.

Respecto al algoritmo de búsqueda, se utilizará el algoritmo de búsqueda por vecindad, pues es un algoritmo muy utilizado [65] cuyo único inconveniente es el tiempo de respuesta si la base de casos crece demasiado, pero como ya se ha comentado se aplicará un algoritmo de agrupación para evitar esto.

En lo que respecta a la función de distancia para estimar la similitud de los casos, se usará la distancia Manhattan, pues es adecuada cuando las componentes del vector son diferentes y no están relacionados entre sí, como es el caso. Además, en la práctica se ha comprobado que devolvía casos adecuados a las situaciones presentadas.

5.4.6.4. Descripción de la fase de reutilización

Para describir la fase de reutilización usada en el CBR es necesario especificar el algoritmo de adaptación usado.

A pesar de que el comportamiento a entrenar en este caso es más complejo que los mostrados anteriormente, pero se ha decidido no aplicar adaptación para comprobar hasta que punto es posible realizar un entrenamiento relativamente completo que cubra con las situaciones que deban afrontar los robots. Además, de esta forma se puede estudiar el resultado del entrenamiento realizado sin las interferencias provocadas por la adaptación, pudiendo determinar así los posibles errores o situaciones no contempladas durante el entrenamiento.

5.4.6.5. Descripción de la fase de revisión

Esta fase se describe indicando como se realiza la validación de las soluciones adaptadas y, en general, de las soluciones aportadas por el CBR.

En este caso, al igual que se hizo anteriormente, la validación de los casos aplicados se hará de forma manual mediante observación directa por parte de un experto. Así pues, si durante la ejecución en modo autónomo el sistema

[1] Esos valores se consideran inválidos al recibirlos, pues ρ no puede ser negativo

se coordina adecuadamente y alcanza la meta, se considerará que los casos aplicados han sido correctos. En caso de observarse alguna conducta extraña o no alcanzar su destino, se estudiarían los ficheros de depuración del sistema para determinar el problema y solucionarlo si es posible.

5.4.6.6. Descripción de la fase de retención

Por último, la fase de retención se describe especificando los tipos de aprendizajes que se usarán en el CBR. Así pues, y dado que no se aplica adaptación, se usará únicamente aprendizaje por observación, pues es el que permite generar inicialmente la base de casos para que el sistema pueda trabajar de forma autónoma posteriormente.

5.5. Experimentos y resultados

Esta sección resume los resultados de la prueba de concepto propuesta para comprobar la viabilidad del sistema para su aplicación en la coordinación de varios robots. Es importante recordar que, aunque las pruebas realizadas son simples, pero el objetivo es comprobar que los AIBOs pueden desarrollar trayectorias coordinadas de forma reactiva mediante aprendizaje. De hecho, este capítulo se centra en la coordinación mediante comportamientos aprendidos a los cuales se aplicará, en el próximo capítulo, una capa de nivel superior que permitirá la realización de comportamientos coordinados más complejos.

Las pruebas se han realizado usando dos robots AIBO ERS-7 de Sony y el entorno de las pruebas es un área sin obstáculos de unos de 2x2 m^2 con tres marcas de ARToolkit de 16x16 cm^2 con posiciones conocidas respecto a la portería, lo que permite la localización visual de los robots. Las dimensiones del campo de pruebas están limitadas por la distancia a la cual el robot AIBO es capaz de ver las marcas ARToolkit y localizarse respecto a ellas de una forma fiable y sin excesivos errores (menos de 2 m). Hay que recordar que si un robot no detectase ninguna marca se pararía (por seguridad) y comenzaría a girar hacia la izquierda hasta que pudiese localizarse.

El entrenamiento de los robots, como ya se ha comentado, se realiza de forma secuencial. Así pues, en primer lugar (fase 1) se realiza el entrenamiento mediante control remoto de un robot para que realice una trayectoria de aproximación a la portería. Tras esto, se procede con la fase 2, en la que el robot entrenado se ejecuta en modo navegación autónoma con el entrenamiento realizado mientras se entrena el segundo robot en un comportamiento coordinado con el primero, tratando de ponerse en paralelo con él mientras

se acerca a la portería. De esta forma, el segundo robot aprende a asociar la posición relativa del destino así como la del compañero con los comandos de movimiento que debe realizar para alcanzar el destino, evitando chocar e interferir con el compañero aunque manteniendo una posición respecto a éste, permitiéndose así una navegación coordinada basada en comportamientos aprendidos.

A continuación se detallan las distintas fases de entrenamiento así como los resultados obtenidos en las pruebas.

5.5.1. Entrenamiento secuencial: fase 1

En esta fase, como ya se ha comentado, se entrena un único robot para que sea capaz de realizar trayectorias que le permitan alcanzar la meta (la portería). En este entrenamiento el robot tendrá únicamente en cuenta el destino a alcanzar, ya que no hay más robots en las pruebas.

Posteriormente, en la fase 2, este entrenamiento se usará para que un robot navegue de forma autónoma, mientras el otro aprende comportamientos coordinados teniendo en cuenta a su compañero, aunque éste no lo tenga en cuenta a él.

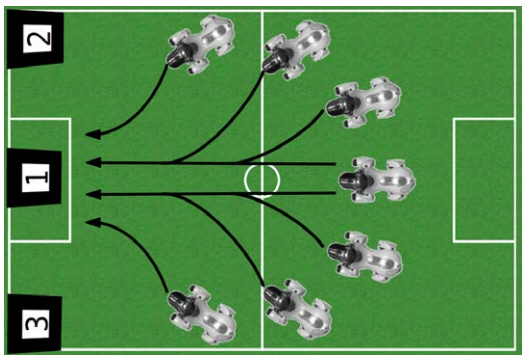

Figura 5.5: Fase 1 del entrenamiento secuencial: entrenamiento de un único robot para alcanzar la meta (la portería)

La figura 5.5 muestra diversas trayectorias, que permiten al robot alcanzar la portería desde la mayoría de puntos del campo. Las trayectorias, como se puede observar, incluyen la aproximación del robot a la portería directamente y desde los laterales.

5.5.2. Entrenamiento secuencial: fase 2

En esta segunda fase, se pone en funcionamiento un robot (R_1) con el entrenamiento individual de la fase 1, mientras el otro robot (R_2) es entrenado mediante supervisión remota teniendo en cuenta la posición del robot R_1 para realizar un comportamiento coordinado. El objetivo, como ya se ha indicado, es que los robots alcancen la portería navegando de forma coordinada pero sin colisionar ni interferirse entre ellos.

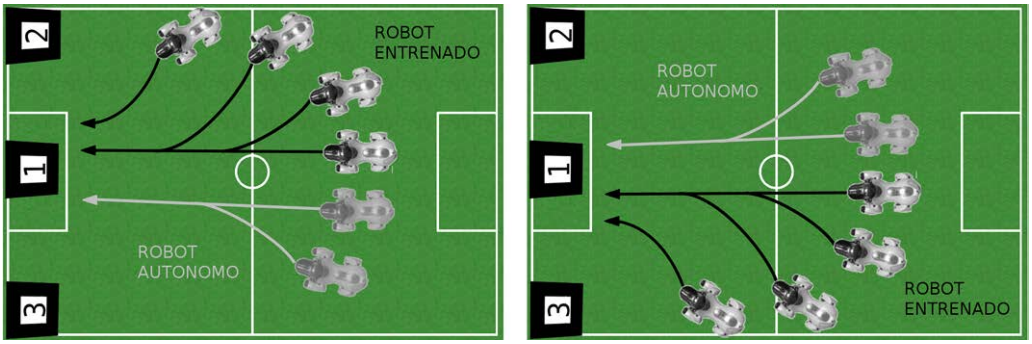

Figura 5.6: Fase 2 del entrenamiento secuencial: entrenamiento de un robot para alcanzar la meta (la portería) evitando colisionar con el otro robot (en modo autónomo).

La figura 5.6 muestra algunas de las trayectorias de entrenamiento realizadas para conseguir el comportamiento coordinado comentado. Como se puede observar, las trayectorias entrenadas son similares a las de la fase 1, pero de forma que el robot entrenado (sin sombreado) tenga en cuenta al robot autónomo (con sombreado gris). Las trayectorias han tenido que ser realizadas en varias rondas, en las que se lanzaba la ejecución del robot autónomo desde distintos puntos, como se puede observar en la figura 5.6.

5.5.3. Navegación coordinada autónoma

Tras finalizar el entrenamiento, se puede aplicar este a ambos robots para que realicen una navegación coordinada. Las pruebas realizadas al sistema propuesto (Fig. 5.7) muestran que los robots son capaces de alcanzar la meta (la portería) mientras avanzan en paralelo sin colisionar ni interferirse entre ellos.

Así pues, la figura 5.7.a muestra como los robots se aproximan al destino comenzando desde posiciones separadas en modo autónomo. Se puede observar que tienden a acercarse el uno al otro para alcanzar la portería en paralelo. En la figura 5.7.b los robots han sido colocados directamente en

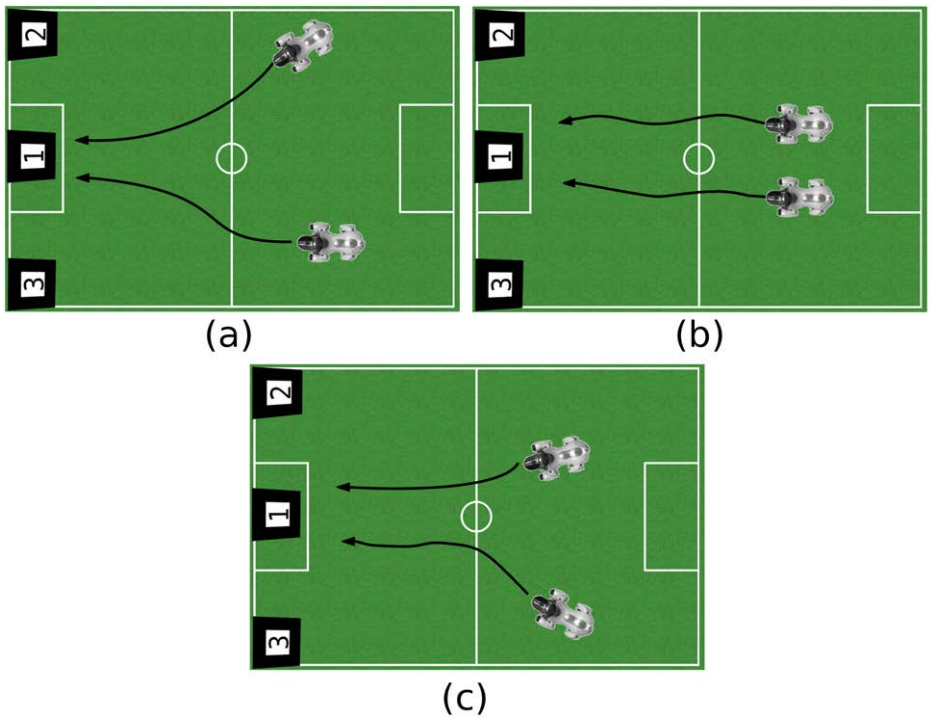

Figura 5.7: Navegación coordinada reactiva basada en comportamientos aprendidos.

paralelo, comprobándose que se mantienen así mientras se aproximan a la portería sin separarse prácticamente.

Por último, la figura 5.7.c muestra una situación similar a la de la figura 5.7.a, aunque en este caso uno de los robots realiza una trayectoria un poco más abrupta de lo esperado, aunque al igual que antes, se aproximan a la portería en paralelo.

Como se puede observar en todas las pruebas mostradas, las trayectorias realizadas por los robots no son simétricas a pesar de que comparten el mismo entrenamiento. La razón es que, en primer lugar, los entrenamientos no son exactamente iguales. Es decir, cuando se realiza el entrenamiento por la derecha del robot, el operador pudo haberse acercado más al robot o menos que en el entrenamiento por la izquierda, lo que pudo provocar que tuviese que hacer más correcciones durante la trayectoria. Esto hace que, si el robot accede en algunos momentos a los comandos que corregían una trayectoria, a pesar de no ser necesario, se generen ciertas oscilaciones, que no tienen porque ser simétricas. De hecho, se ha comprobado que esto sucede en todos los experimentos realizados con CBR mediante LfD, pues es imposible que un operador realice exactamente los mismos movimientos de forma simétrica

y en los mismos puntos y situaciones exactas, lo que provoca variaciones y diferencias en la navegación autónoma.

En cualquier caso, y como se ha comentado, en las pruebas mostradas, ambos robots consiguen alcanzar su destino de forma segura sin colisionar, por lo que estos experimentos muestran que es posible que los robots se tengan en cuenta el uno al otro a la hora de navegar, realizando así una navegación coordinada utilizando comportamientos reactivos aprendidos.

5.5.4. Comportamientos coordinados complejos

Tras estas pruebas, se intentaron realizar pruebas más complejas añadiendo, por ejemplo, un obstáculo que debía evitarse en la trayectoria. En este caso se comprobó que la cantidad de potenciales situaciones a tener en cuenta en el entrenamiento era excesivamente elevado, requiriendo incluso un estudio en profundidad de las situaciones más genéricas a tener en cuenta para no repetir los entrenamientos. La figura 5.8 muestra un ejemplo de algunas de las situaciones que habría que tener en cuenta para realizar este entrenamiento usando como obstáculo otro robot.

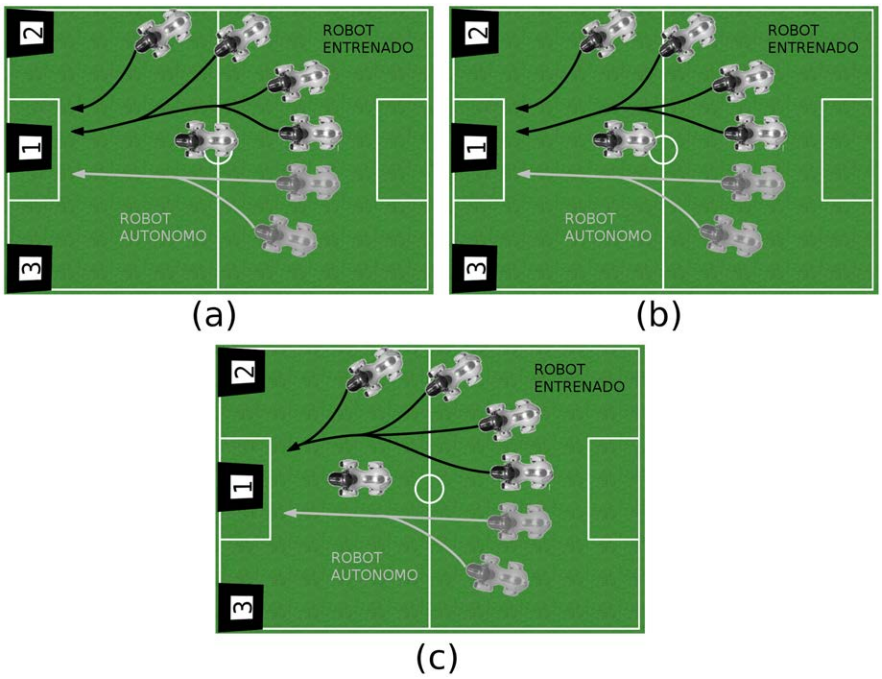

Figura 5.8: Secuencias de entrenamientos que serían necesarias para el aprendizaje de comportamientos complejos.

Como se puede observar, este entrenamiento supondría una gran cantidad

de tiempo y esfuerzo, pues no se trata sólo de las secuencias mostradas, sino que además habría que realizar también las trayectorias simétricas por la banda izquierda del campo así como las secuencias estando el robot-obstáculo por el exterior de la trayectoria, lo cual podría ser terriblemente largo y tedioso. Por otro lado, todo este trabajo sería para obtener una serie de trayectorias concretas para la aproximación a un objetivo, no siendo reutilizables para otros objetivos distintos.

Tras estudiar esta situación, se descubrió que el problema subyacente era que este comportamiento, al igual que el descrito anteriormente, está formado realmente por varios comportamientos más elementales. Así pues, en el entrenamiento que se ha presentado en las pruebas realmente se estaba entrenando el comportamiento emergente que se esperaría obtener de la suma de dos comportamientos más elementales: aproximarse a un objetivo y moverse en paralelo a un compañero, suponiendo cada uno de estos comportamientos elementales un movimiento hacia su propio destino. Sin embargo, en el entrenamiento realizado se ha establecido de forma implícita un movimiento hacia un destino de compromiso entre las respuestas de ambos comportamientos. Por lo tanto, si los comportamientos a entrenar son comportamientos complejos que se pueden descomponer en otros más simples con destinos no relacionados, podría implicar que los entrenamientos resultasen más largos y complejos, ya que seguramente habría que tener en cuenta más situaciones a entrenar, que serían el resultado de las distintas combinaciones de los entrenamientos necesarios para cada comportamiento simple por separado.

Por lo tanto, y a raíz de esta situación, se ha concluido que, aunque es posible realizar comportamientos coordinados mediante aprendizaje, pero es conveniente que éstos sean lo más simples posible, pues de lo contrario el entrenamiento supondría un esfuerzo excesivo para obtener una trayectoria específica que podría no ser fácilmente reutilizable en otras situaciones. Así pues, en el siguiente capítulo se presentará un nuevo enfoque que permitirá la creación de comportamientos más complejos basándose en comportamientos aprendidos simples, resultando de esta forma los entrenamientos más prácticos y los comportamientos más genéricos y reutilizables.

5.5.5. Limitaciones de la localización basada en ARToolkit

Aparte de los problemas comentados, también se encontró un problema, relacionado con el sistema de localización basado en marcas ARToolkit, mientras se trataba de realizar los comportamientos coordinados complejos comentados en el apartado anterior.

Concretamente, el problema detectado está relacionado con una serie de limitaciones que posee ARToolkit que no se consideró que pudiesen llegar a convertirse en un problema. Estas limitaciones son:

- Distancia máxima de funcionamiento. La distancia máxima a la que se puede obtener un posicionamiento fiable depende de la resolución de la imagen y del tamaño de la marca. Como es lógico, el tamaño de la marca se puede controlar y se pueden hacer marcas más grandes, pero la resolución máxima de la imagen del robot es de 416x360 píxeles como máximo, lo que limita la distancia máxima de trabajo a poco más de 1'5 metros, siendo a 2 metros los errores considerables.

- Oclusiones parciales. Si la marca no se visualiza completamente o parte de la marca no se detecta correctamente por efecto de brillos u oclusiones, no es posible detectar la marca y, por lo tanto, estimar la posición del robot respecto a la marca.

- Distancia mínima de funcionamiento. Derivado de los problemas anteriores, si la marca se hace muy grande para que pueda ser detectada desde una distancia mayor, al estar próximo a la marca puede darse el caso de que ésta no se vea completamente en la imagen, por lo que habría también una distancia mínima a partir de la cual el robot no podría detectar la marca ni estimar su posición.

- Errores en la detección. Debido al movimiento del robot y a la calidad de la cámara, las imágenes pueden estar movidas o borrosas, lo que dificulta el reconocimiento de las marcas provocando, en el mejor de los casos, detecciones incorrectas y posicionamientos erróneos. Sin embargo, en general si la imagen está borrosa lo más normal es que no se pueda detectar la marca y no se obtenga ningún posicionamiento.

- Ángulo de visualización. Debido a que la marca es plana, si se ve desde un ángulo demasiado pronunciado la detección se dificulta (especialmente con imágenes de baja resolución) lo que aumenta los errores obtenidos e impide el correcto posicionamiento del robot.

Como se ha comentado, estas limitaciones eran conocidas, pues la librería ARToolkit se había usado anteriormente para su aplicación usual, que es el desarrollo de aplicaciones de RA. En este ámbito, sin embargo, estas limitaciones no suelen suponer un mayor problema debido a que las cámaras usadas tienen buena calidad, una adecuada resolución y se evitan expresamente las oclusiones de las marcas.

De hecho, en las primeras pruebas realizadas con esta librería aplicada al posicionamiento de los robots, en el capítulo 4, se comprobó que estas limitaciones afectaban más de lo esperado debido a la baja resolución de la cámara del robot y a la baja estabilidad de las imágenes debido al movimiento del robot. Sin embargo, los resultados obtenidos fueron, en principio, adecuados y la posibilidad de usar esta librería para obtener un fácil y sencillo posicionamiento mediante la imagen actual era un factor importante para tener en cuenta esta opción.

Así pues, en los experimentos realizados en este capítulo se decidió usar varias marcas para minimizar los posibles problemas de posicionamiento y facilitar que los robots pudiesen tener siempre una marca en el campo de visión (ya que al no usar memoria es necesario posicionarse constantemente), funcionando el sistema de forma adecuada en la mayoría de los casos. Sin embargo, también hay que tener en cuenta que en las primeras pruebas realizadas el entorno estaba relativamente controlado y el comportamiento permitía que las marcas estuviesen siempre en el campo de visión.

Sin embargo, en las últimas pruebas realizadas, en las que se pretendía conseguir comportamientos más complejos, se comprobó que, para que los robots pudiesen posicionarse de forma continua con el enfoque actual, sería necesario situar marcas alrededor de todo el campo, lo que no se consideraba una opción factible. Por otro lado, si en vez de situar marcas alrededor de todo el campo se ponían marcas sólo en posiciones concretas, las limitaciones del tamaño mínimo y máximo así como el ángulo en que deben detectarse las marcas haría difícil la localización del robot desde determinadas posiciones del campo, por estar demasiado cerca de una marca o verla desde un ángulo muy pronunciado.

A todos estos problemas hay que añadir el hecho de que cuando las marcas se ocluyen, aunque sea mínimamente, no es posible estimar el posicionamiento respecto a ésta. Esto es importante porque en las pruebas anteriores el entorno estaba relativamente controlado, pero al intentar realizar comportamientos más complejos con más robots el entorno se volverá menos controlable y este tipo de situaciones serán difíciles de evitar.

Por todo esto se considera que, a pesar de que la localización basada en marcas ARToolkit permite un fácil posicionamiento y ha permitido continuar, dentro de lo posible, con el enfoque reactivo que se había planteado inicialmente, pero para la realización de comportamientos más complejos será necesario el uso de otro sistema de localización u otras balizas visuales, lo cual se planteará en el próximo capítulo.

5.6. Conclusiones

En este capítulo se ha presentado un sistema que permite la coordinación de varios robots a partir de comportamientos reactivos aprendidos mediante LfD usando CBR. Este sistema permite la creación de comportamientos coordinados usando el mismo programa, siendo necesario tan sólo definir los parámetros del caso a usar y entrenar el comportamiento buscado.

Por otro lado, y como se ha partido del sistema propuesto en el capítulo 3, se consiguen también las ventajas de dicho planteamiento, como que no es necesario realizar un modelo cinemático del robot o que los errores de los sensores y errores sistemáticos y mecánicos son absorbidos implícitamente por el operador durante el entrenamiento.

La principal novedad del sistema propuesto es que la coordinación se consigue a nivel reactivo usando comportamientos basados en aprendizaje LfD en vez de usar un enfoque analítico. Este planteamiento es interesante porque, a través de los entrenamientos realizados mediante control remoto, se puede enseñar al robot a realizar una trayectoria concreta, aprendiendo así trayectorias sencillas y comportamientos que no necesariamente tienen que encajar con una expresión única o sencilla de implementar usando un enfoque analítico. Sin embargo, el sistema también tiene sus limitaciones e inconvenientes, como se ha comentado, pues si el comportamiento no resulta lo suficientemente sencillo, el entrenamiento para cubrir las potenciales situaciones a enfrentar supondría un esfuerzo excesivo. Para solucionar este problema se ha decidido hibridar el sistema, como se comentará más adelante.

Para comprobar la viabilidad de conseguir comportamientos coordinados entre varios robots mediante comportamientos aprendidos, se ha propuesto una prueba de concepto consistente en el aprendizaje de un comportamiento coordinado entre dos robots de forma que tengan que navegar en paralelo mientras se aproximan a su destino. Si bien este comportamiento resulta sencillo, pero su objetivo es determinar si el enfoque propuesto es factible y permite la navegación coordinada, así como detectar posibles problemas en el enfoque propuesto.

Los experimentos de esta prueba de concepto se han realizado con dos robots AIBO ERS-7 de Sony, usando visión para localizarse mediante marcas ARToolkit e intercambiando información sobre su posición para permitir la coordinación entre ellos. La coordinación se consigue de forma implícita pues, durante el entrenamiento del robot, el operador tendrá en cuenta la posición del compañero al controlar el robot, por lo que el robot aprenderá a coordinarse con el compañero de forma implícita. El entrenamiento de los robots, como se ha comentado, se ha realizado de forma secuencial. Así pues, en primer lugar se ha entrenado un robot para que se moviese de forma

autónoma, usándose posteriormente para poder realizar el entrenamiento de un segundo robot mientras el primero funciona en modo autónomo, por lo que el entrenamiento de este segundo robot sería un entrenamiento coordinado.

En las pruebas realizadas se ha comprobado que es posible conseguir la coordinación de dos robots mediante comportamientos aprendidos. Sin embargo, también se ha comprobado un inconveniente de este enfoque, y es que cuanto más complejo sea el comportamiento a realizar, y por ende, suponga más situaciones potenciales a tener en cuenta, el entrenamiento de este comportamiento resultará más largo y complejo. Así pues, se ha considerado que para solucionar esta limitación y poder conseguir comportamientos coordinados más complejos, la opción más sencilla y práctica sería añadir una capa de alto nivel que permitiese la activación y desactivación de los comportamientos según la circunstancia. De esta forma, los comportamientos a aprender podrían ser más simples y estarían más definidos y acotados, por lo que podrían ser entrenados fácilmente, obteniéndose los comportamientos de alto nivel a partir de la unión de éstos más sencillos. Este enfoque se presentará en el siguiente capítulo.

Además, como ya se ha comentado, durante estas últimas pruebas también se ha detectado un problema con el sistema de localización basado en marcas ARToolkit. El hecho es que, aunque la librería ARToolkit permite una localización fácil y sencilla a partir de la imagen actual, pero las limitaciones que presenta, como la distancia máxima y mínima de trabajo, el ángulo mínimo para que las marcas puedan ser detectadas correctamente o la imposibilidad de detección de marcas ante cualquier oclusión de éstas, pueden suponer un gran problema para su uso en entornos dinámicos y no controlados donde no se pueda asegurar la visualización constante de marcas en el campo de visión, lo cual es probable que ocurra al realizar comportamientos más complejos con más robots. Así pues, este problema también tratará de solucionarse en el siguiente capítulo.

Capítulo 6

Coordinación híbrida basada en aprendizaje

6.1. Introducción

El estudio de los MRS incluye áreas muy distintas, como la inteligencia artificial, la biología o la psicología social. Sin embargo, una de las áreas con las que más estrechamente está relacionada es con los MAS. De hecho, un MRS puede verse como un caso concreto de MAS [52, 152] en el que cada robot puede ser considerado como un agente con la habilidad de solucionar unas tareas locales y coordinarse con sus compañeros [152].

A lo largo del presente documento se ha mostrado la aplicación de los comportamientos aprendidos mediante LfD usando CBR para navegación individual y coordinada. Concretamente, en el capítulo 3 se ha usado este enfoque para la navegación de un robot de forma individual partiendo de datos visuales en crudo (posición de una pelota en la imagen o posición de las líneas detectadas en el plano de Hough) obteniéndose directamente, sin ningún tipo de posicionamiento explícito, el movimiento a realizar para alcanzar el destino. Si bien los comportamientos aprendidos en este capítulo eran sencillos, pero mostraban la capacidad del sistema para evitar la necesidad de un modelado y para obtener trayectorias a partir de distintas fuentes de información. Sin embargo, en este capítulo también se pusieron de relevancia algunos inconvenientes, como la importancia de realizar un completo entrenamiento inicial para un correcto funcionamiento en modo autónomo, la conveniencia del uso de este enfoque en comportamientos no excesivamente complejos o la necesidad de usar algún tipo de localización explícita para permitir una coordinación compleja entre los robots.

Por su parte, el capítulo 4 estaba dedicado a buscar un sistema de localiza-

ción que permitiese una coordinación compleja entre los robots manteniendo el enfoque reactivo presentado en el capítulo 3. El problema es que, el uso de localización ya implica que el sistema en sí no es reactivo, pues el modelado que suele requerir la localización va contra la filosofía de las arquitecturas reactivas [147]. Sin embargo, dentro de las distintas técnicas de localización hay técnicas que se pueden considerar más reactivas (modelados más simples o el no uso de elementos temporales como la memoria) y localizaciones menos reactivas. Así pues, en el capítulo 4 se mostraron varias técnicas de localización tratando de mantener el espíritu reactivo del sistema. Sin embargo, se comprobó que las técnicas más reactivas no eran viables ni ofrecían un correcto posicionamiento, optándose por el uso de marcas fiduciarias, concretamente marcas ARToolkit para la localización, las cuales ofrecían una solución de compromiso, al requerir un modelado no demasiado estricto del entorno y no requerir el uso de memoria.

Por último, en el capítulo 5 se ha mostrado la aplicación de los comportamientos aprendidos para conseguir la coordinación de varios robots a partir de comportamientos reactivos aprendidos mediante LfD usando CBR. En este capítulo se mostró que era posible conseguir la coordinación de dos robots usando localización mediante marcas ARToolkit teniendo como datos de entrada la posición del destino y la del compañero relativas al robot. Concretamente, el comportamiento consistía en alcanzar el destino (la portería) moviéndose en paralelo de forma coordinada con el compañero, evitando colisionar e interferirse. Sin embargo, en estas pruebas se pusieron de relevancia algunos problemas. Por un lado, que el uso de las marcas ARToolkit para el posicionamiento del robot tenía una serie de limitaciones importantes, por lo que en este capítulo se optará por la alternativa mostrada en el capítulo 4: localización mediante balizas de colores y filtrado. Por otro lado, el entrenamiento secuencial usado, si bien era un entrenamiento *realista*, pero mostró ser un entrenamiento tedioso y no adecuado, por lo que en este capítulo se planteará otra alternativa más cómoda para el entrenamiento de varios robots. Sin embargo, el mayor problema se encontró cuando se intentó implementar un comportamiento algo más complejo, tratando que este comportamiento tuviese en cuenta también un posible obstáculo en el camino. Entonces se pusieron de relieve distintos inconvenientes, como lo importante que resulta una correcta definición de los comportamientos a aprender y su adecuada descomposición en otros más simples. Por esto, en el presente capítulo se añadirá a la arquitectura presentada una capa superior que permita combinar comportamientos aprendidos simples de forma que se puedan obtener comportamientos complejos mediante la correcta combinación de éstos. De lo contrario, el entrenamiento del sistema implicaría

realizar innumerables secuencias de entrenamiento para la obtención de un único comportamiento individual, lo cual no se considera práctico ni viable excepto en situaciones muy puntuales y específicas.

Por lo tanto, este capítulo presenta un sistema híbrido coordinado basado en comportamientos aprendidos mediante LfD usando CBR que solvente los problemas y limitaciones comentados y encontrados en los capítulos anteriores. Para ello, se usará un sistema de localización basado en balizas de colores y filtrado, en vez de la localización basada en marcas ARToolkit. Para evitar el entrenamiento de comportamientos complejos, éstos se descompondrán en comportamientos más simples, que serán entrenados fácilmente. Así pues, los comportamientos complejos se obtendrán a partir de la emergencia surgida de la adecuada combinación de los comportamientos simples, lo cual se hará gracias a la capa de nivel superior añadida en este capítulo. Respecto a la coordinación, al igual que en el capítulo anterior, se realizará de forma implícita pues el operador, al controlar remotamente los robots en el entrenamiento, tendrá en cuenta las posiciones de los robots, asociando los movimientos realizados a la situación concreta de éstos.

Así pues, el presente capítulo seguirá la siguiente estructura. En primer lugar, se detallan todos los problemas e inconvenientes observados a lo largo de los capítulos anteriores al enfoque propuesto, proponiéndose soluciones a éstos (sección 6.2). Tras comentar las soluciones planteadas, se detalla la implementación del sistema de coordinación así como de las soluciones comentadas anteriormente (sección 6.3). Tras esto, se mostrarán los resultados de los experimentos realizados con el sistema híbrido propuesto (sección 6.4), presentando finalmente las conclusiones (sección 6.5).

6.2. Coordinación reactiva: problemas y limitaciones

En este apartado se comentan en detalle los distintos problemas y limitaciones encontrados en los capítulos anteriores durante el desarrollo del sistema de coordinación a partir de comportamientos reactivos aprendidos mediante LfD usando CBR.

De entre los distintos problemas y limitaciones encontrados los más destacados han sido la descomposición de los comportamientos a desarrollar (pues si éstos son demasiado complejos o genéricos el entrenamiento puede ser demasiado tedioso de realizar), las limitaciones del sistema de localización basado en marcas ARToolkit (usado en el capítulo anterior), los problemas con el entorno de desarrollo Tekkotsu y las limitaciones con la librería CBR

usada.

6.2.1. Aprendizaje de comportamientos complejos o ambiguos

En los capítulos precedentes se han mostrado distintos comportamientos, la mayoría de los cuales han funcionado de forma adecuada, aunque obviamente los comportamientos mostrados no eran especialmente complejos y el entorno y las situaciones a afrontar estaban controladas.

A pesar de esto, se ha podido comprobar que el aprendizaje LfD mediante CBR tiene una serie de limitaciones como, por ejemplo, la dificultad para realizar el entrenamiento de comportamientos complejos debido a la cantidad de situaciones a tener en cuenta.

Otro problema de este enfoque es el aprendizaje de comportamientos ambiguos, es decir, que pueden tener distintas soluciones ante situaciones similares, ya que este tipo de comportamientos no encaja bien en un enfoque basado en CBR a no ser que haya parámetros en la entrada que permita discernir o seleccionar una solución respecto de otra, lo que implica desambiguar las posibles situaciones similares de alguna manera. Expresado desde el punto de vista del CBR, si ante una situación puede haber distintas salidas, esta situación debe poder diferenciarse de alguna forma o mediante algún parámetro para que el CBR pueda devolver distintas salidas o se pueda determinar que salida es la adecuada.

Otro problema relacionado con los comportamientos ambiguos, es que en ocasiones esta ambigüedad puede ser debida a que un comportamiento esté formado por la combinación de varios comportamientos más simples. Esto no es necesariamente un problema, si es lo que se quiere hacer y el comportamiento está claramente definido. Sin embargo, si un comportamiento está formado por distintos comportamientos con objetivos y destinos distintos, durante el entrenamiento se tendrá que enseñar al robot alguna solución de compromiso entre ambos. Por lo tanto, el problema puede venir, por un lado de que quien realice el entrenamiento no tenga clara esa solución de compromiso entre ambos comportamientos, lo que puede causar situaciones conflictivas en el entrenamiento. Por otro lado, una vez determinada una solución de compromiso, ésta sólo puede modificarse mediante un nuevo entrenamiento, reduciéndose las posibilidades de reutilización con respecto a una solución basada en el entrenamiento de comportamientos simples.

Respecto al funcionamiento autónomo del sistema tras un entrenamiento, también se ha comprobado que no siempre es fácil estimar el resultado que se obtendrá respecto al entrenamiento realizado. En general, la respuesta suele

ser bastante parecida a la entrenada, con ciertas variaciones, pero se han encontrado casos en los que, aunque el robot alcanzaba el objetivo propuesto, pero la trayectoria realizada distaba mucho de las entrenadas. Esto habitualmente ocurre porque el sistema debe afrontar situaciones no entrenadas, debido a un entrenamiento deficiente (por haber muchas situaciones posibles y descartar las menos probables en el entrenamiento) o por no disponer de una adaptación adecuada para determinadas situaciones.

Así pues, y tras meditar sobre estos problemas, se ha considerado que lo más adecuado, para conseguir una mayor versatilidad en el sistema, una mayor reutilización de los comportamientos, disminuir la complejidad de los entrenamientos y las posibles situaciones ambiguas (que ocasionan problemas con el CBR), es que los comportamientos entrenados y aprendidos deben ser lo más simples posible, teniendo especialmente en cuenta que sólo tengan un destino, no como en el ejemplo mostrado en el capítulo anterior, donde el posicionamiento del robot dependía de dos destinos que no mantenían una relación entre sí, como ya se comentó. Así pues, en este capítulo se optará por realizar comportamientos lo más simple posible.

Esto, por otro lado, supone que si los comportamientos van a ser muy simples, será necesario el uso de una capa superior que permita activarlos y desactivarlos adecuadamente. Esta capa será una capa analítica que seleccionará los distintos comportamientos, dependiendo de la situación, mediante reglas fijadas a priori, como se comentará en la sección 6.3.1.3.

6.2.2. Localización

La localización utilizada en el capítulo anterior es una localización basada en marcas ARToolkit porque este tipo de marcas permiten una localización 3D precisa a partir de una única marca y permiten obtener el posicionamiento del robot a partir de la imagen actual. Así pues, a pesar de que el uso de esta localización supone un modelado del entorno, pero permite mantener parte del enfoque reactivo del sistema ya que no requiere ningún tipo de componente temporal o memoria.

Sin embargo, como se comentó en capítulo anterior, esta localización adolecía de varios problemas e inconvenientes que dificultaban su uso, como la distancia máxima y mínima a la que se podían usar, el ángulo mínimo con el que había que visualizar las marcas para poder detectarlas correctamente (debido a que son planas), los errores obtenidos dependiendo de la distancia a la que se está de la marca, causados por la mala calidad de la cámara web, así como los problemas para la detección de las marcas debido al movimiento del robot, lo que dificulta la identificación y posicionamiento en ocasiones, siendo necesario que el robot se detenga y se estabilice para determinar su

posición.

Además, este sistema de localización requiere, para su correcto funcionamiento, que haya una marca constantemente en el campo de visión, debido a que se ha evitado el uso de una memoria para mantener el enfoque reactivo del sistema. Por otro lado, para poder detectar la marca ésta debe visualizarse completamente, pues el sistema es sensible a oclusiones. Sin embargo, esta condición no es fácil de asegurar cuando entran en juego varios robots. Todo esto complica la posibilidad de usar esta localización para un comportamiento coordinado complejo, en el que puede haber oclusiones, y no se puede garantizar la visualización de las marcas en todo instante.

Por lo tanto, para mejorar la localización y para poder realizar comportamientos coordinados más complejos, se hace necesario el uso de una componente temporal, o lo que es lo mismo, una memoria que permita recordar la última posición obtenida. Sin embargo, y ya que se va a incluir una memoria en el sistema, se puede hacer uso de algún filtrado, como los filtros de Kalman [41,89] o filtros de partículas [108,110,151] para mejorar la precisión y la estabilidad de la posición obtenida. Este filtrado mejoraría el posicionamiento y la estabilidad, pero no solucionaría los problemas y limitaciones mencionadas propias de las marcas ARToolkit. Así pues, y dado que la localización basada en marcas ARToolkit no consigue suplir las necesidades del sistema, se seleccionará la localización planteada en el capítulo 4 como última alternativa para la localización del sistema: la localización basada en marcas visuales de colores y filtrado del posicionamiento usando filtros de partículas.

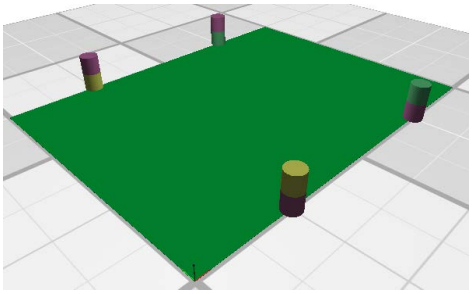

Figura 6.1: Balizas de colores y disposición en el campo.

La razón de usar marcas visuales de colores (Fig. 6.1) en vez de marcas ARToolkit es que estas marcas se pueden ver desde cualquier posición, no existiendo limitación de ángulo para detectarlas y, en caso de oclusión parcial de la marca, se puede seguir estimando su distancia a partir del ancho o del alto (según la oclusión), por lo que resultan más robustas que las marcas de ARToolkit a pesar de no permitir un posicionamiento con sólo visualizar una

única marca. Por otro lado, el uso de filtros de partículas en vez de filtros de Kalman, como ya se comentó en el capítulo 4, se debe a que ofrecen una mayor flexibilidad y fácil implementación.

6.2.3. Entorno de desarrollo

Durante el desarrollo de este trabajo se han encontrado repetidos problemas con el entorno de desarrollo Tekkotsu. El entorno, como tal, resulta práctico y rápido para desarrollar aplicaciones. Sin embargo, debido a que es un entorno en constante desarrollo y evolución, y al propio enfoque del entorno en sí, tiene una serie de limitaciones e inconvenientes, como son:

- Dificultad de depuración en el AIBO. Tekkotsu resulta bastante práctico para el desarrollo de aplicaciones a alto nivel aislando de los problemas de bajo nivel. El problema surge cuando la única forma de depurar un programa es mediante punteros y referencias a bajo nivel (OPEN-R), no siendo obvia ni sencilla la relación entre éstos y la programación realizada mediante Tekkotsu si no se conoce a fondo el entorno. Por lo tanto, en caso de tener problemas a la hora de programar, es complejo depurar los códigos dependiendo de los errores que se presenten.

- Bloqueos del sistema y reinicios inesperados en el AIBO. Si bien estos problemas no ocurren siempre, pero no es extraño que ocurran bloqueos o reinicios en el sistema incluso sin estar ejecutando ningún comportamiento propio en el AIBO.

- Fallos y limitaciones. Debido a que el entorno esta en constante desarrollo, había características que eran deseables pero, se supone que debido a la falta de tiempo y personal, éstas tardaban en añadirse al entorno. Por lo tanto, surgía la duda de si sería mejor intentar añadir esa funcionalidad al entorno por cuenta propia o esperar a que la añadiese el grupo de desarrolladores del entorno en la siguiente versión, no sabiendo con seguridad cuando saldría y si cumpliría las necesidades propias. Por otro lado, puntualmente surgían errores en algunas versiones que se podían solucionar con una nueva versión, así como problemas en versiones nuevas que no existían en una versión anterior, lo cual puede resultar frustrante cuando no se consigue averiguar cuál es el origen de un error concreto. Esto, por lo tanto, complica el desarrollo del sistema ya que hay que añadir a los problemas generados por cuenta propia los problemas generados por el entorno de desarrollo.

- Limitaciones de segmentación en el AIBO. Una característica interesante de Tekkotsu era la segmentación de objetos por color en el AIBO. Sin embargo, y gracias al compañero Ignacio Herrero Reder, se descubrió que estaba limitada a un máximo de dos objetos, siendo necesaria su implementación de forma externa para obtener la segmentación de más objetos, como de hecho se ha tenido que hacer. Por lo tanto, esta funcionalidad de Tekkotsu no ha podido ser utilizada, a pesar de la facilidad que habría supuesto para el desarrollo del sistema.

- Consumo de memoria y CPU. Como se ha comentado, Tekkotsu está formado por un programa ejecutado en el AIBO y otro en el ordenador, programado en Java. Por lo tanto, y debido a la baja eficiencia de Java y el gran consumo de memoria y CPU de este lenguaje, unido al hecho de que es en el ordenador donde se realiza el procesamiento y representación de las imágenes de los AIBO al usuario, daba como resultado que el consumo de recursos de este interfaz de usuario era excesivo para la funcionalidad realizada. De hecho, para dos robots simultáneos el interfaz gráfico ya consumía casi la totalidad de recursos del ordenador, por lo que para añadir más robots sería necesario usar otro ordenador, lo cual no se considera una solución adecuada.

- Problemas de depuración con JNI. Este problema, al igual que el anterior, está heredado del uso de Java en Tekkotsu. Así pues, si bien el interfaz JNI permite usar códigos en C y C++ desde Java, pero no permite una fácil depuración para comprobar si el programa tiene pérdidas de memoria. De hecho, y dada a la gestión y el alto consumo de memoria de Java, es bastante complejo averiguar si un incremento de memoria en el programa se debe a Java o a pérdidas de memoria debidas a una mala implementación del programa en C/C++, dificultando la detección de errores.

Como conclusión, y debido a todos estos problemas y limitaciones, se decidió buscar alternativas a Tekkotsu para la programación de la plataforma AIBO. Entre las alternativas encontradas, aparte del SDK oficial de Sony (OPEN-R) bastante árido y complejo de usar, estaban las siguientes:

- Interfaz Universal para Plataformas Robóticas (URBI) (*Universal Robotic Body Interface* en inglés). URBI es una plataforma de programación para el desarrollo de aplicaciones para distintas plataformas robóticas, entre las que se incluye el AIBO. Está desarrollado en C++ y permite un fácil manejo y control de los robots mediante un lenguaje de script (urbiscript). Además ofrece una librería (liburbi) para C++,

Java y Matlab que permite conectarse con los robots y controlarlos. Su principal ventaja es que resulta muy intuitiva y sencilla.

- Python para Robótica (Pyro) (*Python Robotics* en inglés). Pyro es un entorno, basado en *Python*, para el desarrollo de aplicaciones para distintos robots, incluido el AIBO. Su principal objetivo es el de abstraer del hardware concreto permitiendo un fácil uso y desarrollo independientemente de la plataforma. Además, permite su uso con varios simuladores, como Player/Stage, Gazebo o el simulador de Khepera.

- Sistema Operativo Robótico (ROS) (*Robot Operating System* en inglés). ROS es un entorno de desarrollo basado en Ubuntu que ofrece una gran cantidad de funciones para facilitar el desarrollo de aplicaciones robóticas. La gran ventaja de ROS es que es una plataforma con un gran soporte y funcionalidad, siendo hoy en día un entorno muy extendido.

De entre estas alternativas, Pyro se descartó porque depende de la versión 2.4 de Tekkotsu, que es una versión demasiado antigua, no siendo adecuada para el desarrollo del presente trabajo. ROS, por su parte, no tiene soporte para la plataforma AIBO[1], por lo que no era viable aunque habría sido una alternativa interesante. Por último, quedaba URBI. Este entorno sí soportaba la plataforma AIBO y podía haber sido una alternativa perfecta a Tekkotsu. Sin embargo, el problema que se encontró es que no se consiguió que URBI enviase las imágenes al ordenador con una tasa de refresco adecuada y más o menos constante (entre 5 y 10 imágenes/segundo), produciéndose en ocasiones retrasos de un segundo o más.

Por lo tanto, al no poder solucionar el problema de la recepción de imágenes en URBI, hubo que volver a la plataforma original, Tekkotsu. Sin embargo, se intentó aislar el desarrollo del programa principal de Tekkotsu en la medida de lo posible. Para ello se decidió partir del enfoque usado en [97] (Fig 6.2), donde el programa principal (Cliente Robot) se conecta con el robot (Servidor Robot) a través de un programa que hace de interfaz entre ambos (Interfaz Robot-PC), abstrayendo el hardware y realizando las conversiones adecuadas entre ambos programas.

De esta forma, Tekkotsu se utiliza únicamente para realizar el Servidor Robot en el AIBO y el Interfaz Robot-PC en el ordenador. Así pues, el Servidor Robot se encargaría de enviar al ordenador las imágenes capturadas y el estado del robot así como de ejecutar los comandos de movimiento recibidos. Por su parte, el Interfaz Robot-PC haría únicamente de interfaz entre el robot y el programa principal, permitiendo aislar el programa principal

[1] `http://wiki.ros.org/Robots`. Última visita: 24-09-2015

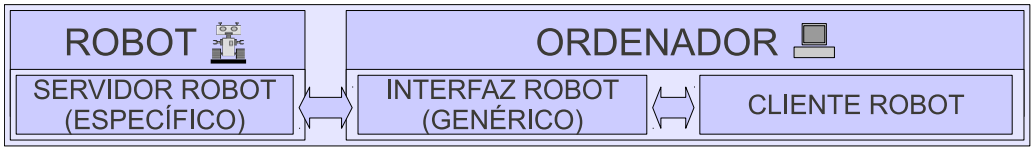

Figura 6.2: Estructura básica DLA.

de la plataforma Tekkotsu y de elementos específicos del robot. Finalmente, la comunicación entre el Interfaz Robot-PC y el programa principal (Cliente Robot) se haría mediante un protocolo de comandos (vía socket) que permitiría el envío de comandos, imágenes e información sobre el estado del robot. De esta forma, el Cliente Robot se puede realizar en C++ usando como interfaz gráfico alguna librería más ligera que la usada por Java (como FLTK).

Otra ventaja de este enfoque es que permite usar el programa desarrollado más fácilmente con robots con visión siempre que se implemente el programa interfaz entre el programa principal y el robot.

6.2.3.1. Simulador

Otro elemento importante a tener en cuenta en el desarrollo del presente trabajo fue el simulador. Al usar un robot era viable el uso del robot real a diario, pues no suponía una gran pérdida de tiempo su configuración y puesta en marcha. Sin embargo, al empezar a usar varios robots la preparación de las pruebas, las configuraciones, la carga de las baterías y los entrenamientos comenzaron a hacerse más tediosos y complejos. Por lo tanto, se hizo necesario el uso de un simulador para facilitar y agilizar las pruebas.

El primer simulador que se utilizó y que permitía realizar algunas pruebas, aunque no fuesen realistas, fue el simulador Gazebo (Fig. 6.3.a). Este simulador permite el uso de visión y tiene un modelado 3D más o menos realista. Sin embargo, no dispone de ningún modelo de robot AIBO, por lo que se usó un modelo de un robot Pioneer, a pesar de no ser adecuado, pues permitía realizar las pruebas de una forma más sencilla y fácil que con el robot real.

Sin embargo, este simulador no era adecuado, por lo que posteriormente se cambió al simulador Mirage (Fig. 6.3.b). Mirage es un entorno de simulación 3D con soporte para física, múltiples robots y múltiples cámaras. Este simulador fue añadido en un principio a Tekkotsu, pero su funcionalidad era muy reducida, por lo que ni se tuvo en cuenta en su momento. Sin embargo, tiempo después, y gracias al compañero Ignacio Herrero, se descubrió que el simulador era casi totalmente operativo (a excepción de algunos detalles,

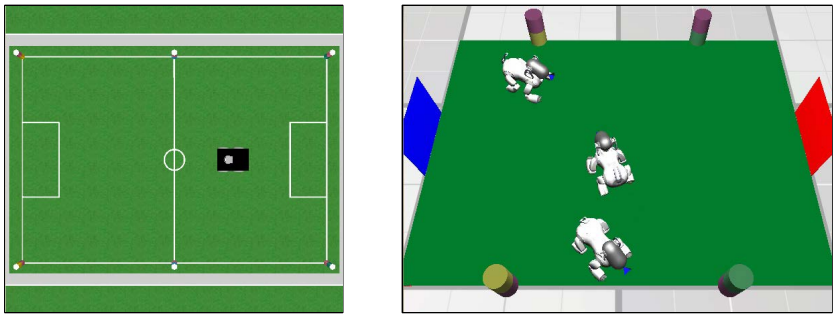

Figura 6.3: Simulador 3D Gazebo (a) y Mirage (b).

como por ejemplo la detección de colisiones) y compatible con Tekkotsu. Así pues, se comenzó a usar este simulador resultando, de hecho, crucial para el desarrollo de la tesis, pues ha facilitado enormemente su desarrollo, especialmente las pruebas preliminares con múltiples robots.

6.2.4. Implementación del CBR

El servidor CBR usado anteriormente era, como a se ha comentado, una modificación de la implementación del servidor CBR desarrollado por el grupo KEMLg de la Universidad Politécnica de Cataluña (UPC). Este servidor CBR está desarrollado en Java y permite la conexión de clientes (vía socket) para el acceso a la base de casos.

Sin embargo, un detalle que es importante tener en cuenta es que este servidor CBR sólo permite la conexión de un cliente simultáneamente, debiendo cargarse la base de casos al inicio del servidor. Hasta ahora esto no había sido un problema pues realmente sólo era necesario usar un servidor por comportamiento, no habiéndose usado más de dos comportamientos simultáneos en las pruebas mostradas anteriormente. Sin embargo, en este capítulo se usarán múltiples comportamientos en cada uno de los robots usados, lo que implica que la cantidad de servidores CBR a lanzar es excesiva, suponiendo un desperdicio de recursos innecesario, sin tener en cuenta otros problemas como que la configuración del sistema sería también más compleja.

A esto hay que sumar el relativamente poco control que se tiene sobre el CBR usado, pues su implementación interna es lo suficiente compleja como para que la realización de modificaciones internas suponga un gran esfuerzo, requiriendo el conocimiento de la implementación de este CBR.

Por otro lado, y debido a problemas tenidos con Tekkotsu ya comentados, se ha optado por una estructura en la que Tekkotsu sólo hace de interfaz para

comunicarse con el AIBO, mientras que el programa principal está implementado en C++, abandonando el desarrollo Java realizado hasta el momento. Así pues, la implementación del CBR usado hasta ahora, basado en Java, tampoco encaja con el nuevo planteamiento del sistema.

Para solucionar este problema se buscaron alternativas a este servidor CBR, como por ejemplo AIAI CBR [2], myCBR [3], COLIBRI Studio [4], eXiTCBR [5], Weka [6] o Case Based Reasoning DSS [7]. Sin embargo, todas estas herramientas están desarrolladas en Java (menos una desarrollada en PHP) y, en general, resultaban demasiado complejas y disponían de funcionalidades que no eran necesarias para el desarrollo del trabajo.

Así pues, finalmente se ha optado por implementar una versión propia de CBR en C++ con la funcionalidad necesaria para el sistema que se esta desarrollando. Además, esta implementación propia permite la adición de casos en tiempo real de forma sencilla, implementa distintas distancias de similitud, permite la carga de ficheros en tiempo real, ofrece una rápida respuesta a las búsquedas (dependiendo del tamaño de la base de casos) y, en definitiva, cumple perfectamente con las necesidades del sistema a desarrollar.

Además, y ya que se disponía de un control total del sistema, se implementaron algunas herramientas para depurar las bases de casos CBR de una forma sencilla y visual, permitiendo encontrar casos incongruentes (entradas similares y salidas distintas), así como la representación visual de los casos, o parte de éstos, permitiendo su modificación, agrupación, depuración y filtrado visualmente.

A continuación se comenta en más detalle tanto la implementación del CBR como la herramienta de depuración visual desarrollada.

6.2.4.1. LibCBR

La implementación del CBR realizada durante el desarrollo de la presente tesis se llama LibCBR. Esta librería ha sido implementada en C++ y está especialmente pensada para bases de casos formadas por pocos casos, no siendo adecuada para manejar una gran cantidad de casos.

El formato del fichero de entrada es Valores Separados por Comas (CSV) (*Comma-Separated Values* en inglés), un formato muy conocido y fácil de modificar, pues se trata de un fichero en texto plano que puede ser generado y

[2] `http://www.aiai.ed.ac.uk/project/cbr/`. Última visita: 24-09-2015
[3] `http://mycbr-project.net/index.html`. Última visita: 24-09-2015
[4] `http://gaia.fdi.ucm.es/research/colibri`. Última visita: 24-09-2015
[5] `http://exitcbr.udg.edu/Index.html`. Última visita: 24-09-2015
[6] `http://www.cs.waikato.ac.nz/ml/weka/`. Última visita: 24-09-2015
[7] `http://sourceforge.net/projects/cbrdss/`. Última visita: 24-09-2015

modificado fácilmente con cualquier editor de texto. Por otro lado, el formato utilizado es una pequeña modificación que permite la inclusión de unas cabeceras en las primeras filas del fichero para facilitar la descripción de las columnas del CBR, pudiendo indicarse, por ejemplo, el contenido de una columna, sus unidades o su rango.

```
#C#     R0POSH;     NOBJS;  O00DIST;    O00ANG;     CMD_X;     CMD_R;  CMD_PAN;  AutoGen
#C#     degree;      enum;        cm;    degree;   +-1000;   +-1000;   +-1000;  AutoGen
#H#      In_00;     In_01;     In_03;     In_04;    Out_00;    Out_01;    Out_02;OUT_NCASE
#W#       0.00;      1.00;      1.00;      2.00;         ;          ;         ;
          0.00;      1.00;    141.78;     90.34;      0.00;   1000.00;      0.00;      0.00
          0.00;      1.00;    141.78;     82.42;      0.00;   1000.00;      0.00;      4.00
```

Figura 6.4: Formato CSV del fichero CBR.

La figura 6.4 muestra un ejemplo del formato CSV indicado, donde se pueden apreciar las cabeceras añadidas al formato, cuyo significado es el siguiente:

- #C#: Las líneas que comienzan por esta cabecera son líneas de comentarios, y deben estar al principio del fichero. Estas líneas son ignoradas por el CBR, siendo utilizadas únicamente como información para que el usuario recuerde, por ejemplo, el contenido de cada columna, sus unidades o su rango. Las líneas de comentarios son opcionales.

- #W#: La línea de peso (*Weight* en inglés) contiene los pesos de las entradas, en caso de que no sean todos iguales. En el caso de que esta línea no exista se considera que todos los pesos son iguales a 1.

- #H#: La línea de cabecera (*Header* en inglés) contiene las indicaciones de qué columnas son de entrada (In_XX) y cuales de salida (Out_XX) para el CBR. Esta línea si es obligatoria para indicar al CBR de cuantos parámetros se compone la entrada y de cuantos la salida, pues puede haber parámetros en el fichero que no utilice el CBR.

Independientemente del formato de entrada usado por la librería LibCBR, esta tiene una serie de características como son:

- Estructura plana de la base de casos. Esta implementación está desarrollada especialmente para bases de casos formada por una cantidad acotada de casos.

- El algoritmo de búsqueda en la fase de recuperación es búsqueda por vecindad, pues es usual y adecuado cuando el número de casos de la base de casos está acotada.

- Implementa distintas distancias, como la distancia Manhattan, Euclídea, Chebyshev o Tanimoto.
- Permite la aplicación de un algoritmo por agrupación para eliminación de redundancias y repeticiones de casos usando distintas distancias.
- Permite la búsqueda de incongruencias en la base de datos (casos con entradas similares y salidas distintas) mostrándolas para que el usuario decida que hacer.
- Permite la inclusión, eliminación y modificación de nuevos casos en tiempo real.
- Dispone de una herramienta de depuración visual para los casos. Esta herramienta será comentada en más detalle en el siguiente apartado.

De todas estas funcionalidades, las cuatro últimas no estaban implementadas de forma integrada en la librería usada anteriormente, sino que se implementaron de forma externa. Así pues, LibCBR permite realizar estas taras de una forma más sencilla al integrarlo dentro de la propia librería. Tampoco hay que olvidar otra ventaja de la librería LibCBR, y es que puede integrarse en un programa en C++, evitando la necesidad de crear un servidor CBR por cada comportamiento como el servidor CBR anterior y, por tanto, reduciendo el consumo de recursos.

6.2.4.2. Depurador de LibCBR

El depurador visual desarrollado para la librería LibCBR permite la visualización gráfica de los casos así como su depuración de una forma fácil y sencilla.

Este depurador visual ha podido ser realizado porque, al añadir un sistema de localización, los elementos a usar en el CBR pueden ser referenciados de una forma relativa al robot, lo que facilita la posibilidad de mostrar fácilmente los casos de la base de casos. Sin embargo, este depurador también tiene sus limitaciones, pues sólo puede mostrar adecuadamente casos que dependan de un objeto, ya que la visualización de más objetos simultáneamente se vuelve demasiado compleja y confusa como para resultar práctica en la depuración.

Por otro lado, el desarrollo de este depurador muestra una de las ventajas comentadas del CBR frente a otras técnicas de IA, y es la transparencia que ofrece para el acceso a la información aprendida.

Así pues, la figura 6.5 muestra la representación visual de un caso asociado a una situación específica. Concretamente, se puede observar la situación, en la que está el AIBO con la pelota y el movimiento que realiza. Si se observa la

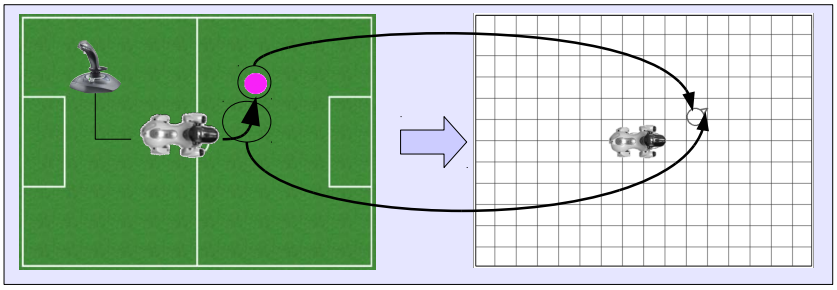

Figura 6.5: Representación de los casos en el depurador LibCBR. (a) Situación que genera el caso y salida aplicada (b) Representación del caso asociado a la situación (a).

figura, la posición de la pelota relativa al robot indica la posición del círculo respecto al robot en la representación del caso, mientras que el movimiento realizado se representa mediante una flecha-triángulo que indica la dirección del movimiento a realizar cuando el objeto está en esa posición respecto al robot. Así pues, se puede realizar una representación que permite determinar problemas, incongruencias y posiciones sin entrenamiento, entre otras cosas, de una forma relativamente sencilla.

Por otro lado, el depurador permite, además de visualizar los casos almacenados, realizar ciertas operaciones sobre ellos. Así pues, la figura 6.6 muestra alguna de la funcionalidad gráfica que permite este depurador. En primer lugar, la figura 6.6.a muestra la funcionalidad de representación de los casos de una base de casos, la cual resulta muy práctica para descubrir visualmente situaciones conflictivas (casos muy cercanos o superpuestos con salidas o movimientos distintos). La figura 6.6.b muestra la funcionalidad de eliminación de casos específicos, mediante la selección con el ratón. Por su parte, la figura 6.6.c muestra la opción de mover casos, mientras que la figura 6.6.d muestra la creación artificial de casos promediando los casos ya existentes, dando por hecho que situaciones intermedias tendrán valores intermedios para el entrenamiento usado. Mediante esta interpolación, por ejemplo, se pueden generar casos nuevos para suplir situaciones o huecos que, en la depuración, se considere que no han sido entrenados. Además, este depurador también permite aplicar un algoritmo de agrupación a la base de casos, mostrándose los casos que puedan ser conflictivos (entradas iguales y salidas distintas).

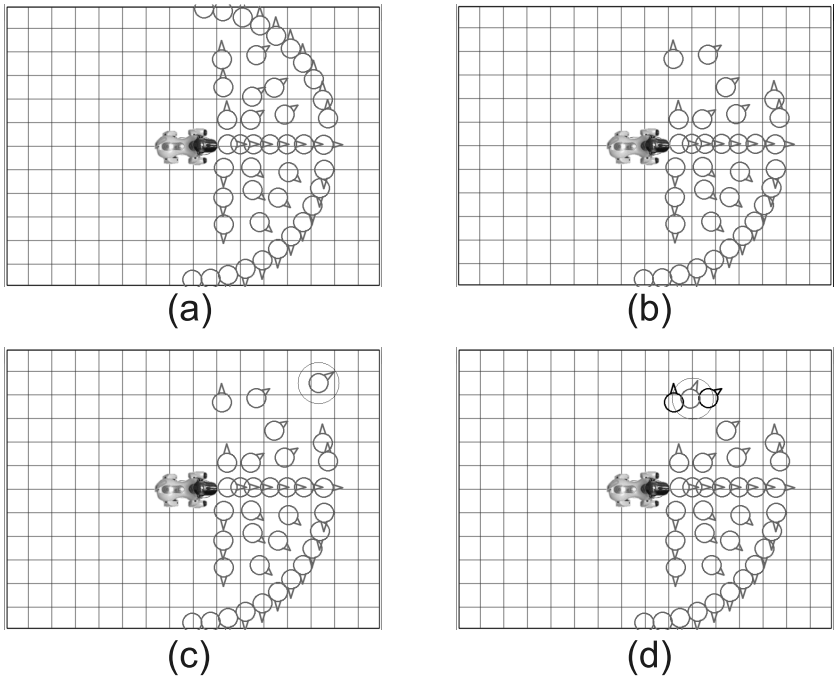

Figura 6.6: Operaciones visuales del depurador LibCBR: (a) Representación de la base de casos completa. (b) Borrado de casos. (c) Mover casos. (d) Interpolar casos.

6.3. Coordinación mediante aprendizaje

A lo largo de los capítulos anteriores se ha mostrado la evolución del sistema de coordinación propuesto descubriéndose, como se ha comentado, distintos problemas e inconvenientes, como el problema de la localización basada en ARToolkit o la dificultad en los entrenamientos de comportamientos complejos.

Así pues, el principal objetivo de este capítulo es completar el desarrollo de un sistema de coordinación multi-agente basado en comportamientos aprendidos, solucionando los problemas encontrados en los capítulos precedentes. Para ello, se optará por el aprendizaje de comportamientos simples mediante LfD usando CBR. Estos comportamientos aprendidos serán combinados mediante una capa superior que permitirá la generación de comportamientos más complejos por emergencia. Esta capa (que se detallará en los próximos apartados) se implementará de forma analítica como una serie de condiciones que permitirán activar o desactivar los comportamientos en función de la situación concreta, permitiendo así su combinación. El uso de comportamientos simples y bien definidos, como se ha comentado en la

sección 6.2.1, solventará o minimizará los problemas derivados de los comportamientos complejos, como un difícil entrenamiento debido a la cantidad de situaciones a tener en cuenta o la existencia posibles situaciones ambiguas (misma situación y distintas soluciones). Además, los comportamientos simples son más genéricos y fáciles de probar y reutilizar en distintas situaciones y entornos [136]. El entrenamiento de los comportamientos, al igual que en los capítulos anteriores, se realiza mediante control remoto.

La coordinación entre los robots se hará de forma implícita pues, como se comentó en el capítulo anterior, esta coordinación es más fácil de implementar de forma directa con un enfoque basado en comportamientos aprendidos (ya que sólo hay que tener en cuenta la posición de los demás robots a la hora de moverse y actuar) y es más adecuada para entornos dinámicos. Al igual que en el capítulo anterior, para facilitar la coordinación entre los robots y que conozcan sus posiciones de una forma sencilla se ha permitido la comunicación entre ellos, realizándose el intercambio de información entre los robots mediante DLA (apartado 4.4.2.1).

Por otro lado, y dado que la localización basada en marcas ARToolkit no cumplía con las necesidades del sistema de coordinación a desarrollar, se optará por el uso de una localización basada en balizas de colores (similares a las de la Robocup) corrigiendo el posicionamiento mediante filtros de partículas.

Hay que destacar que en este capítulo, a diferencia de los anteriores, el sistema implementado es un sistema híbrido, y no reactivo como en los casos anteriores. Esto es así por un lado por el uso de la localización comentada, pues supone tanto un modelado como una componente temporal (histórico de observaciones) que se corresponde más con un enfoque híbrido que con uno reactivo. Además, en este capítulo se ha añadido al sistema una capa de nivel superior que permite la activación de los comportamientos en función de la información global del entorno, lo que no supone un enfoque reactivo como el que se pretendía conseguir en los capítulos precedentes. Sin embargo, y como se ha comentado, ha sido necesario realizar estos cambios para obtener comportamientos más complejos que los mostrados anteriormente.

Otro factor a tener en cuenta, es que sólo se dispone de tres robots AIBO para las pruebas, robots que dejaron de fabricarse en el 2006 y que, en caso de avería o accidente, no se podrían reparar. De hecho, originalmente se disponía de 4 robots, pero debido a que los motores del cuello de uno de ellos se estropeó, actualmente no es viable su uso. Por lo tanto, y para evitar dañar a los robots en la medida de lo posible, se ha decidido evitar las colisiones y los movimientos de tiro frontal y con la cabeza, que producen un mayor estrés a las articulaciones del cuello. Sin embargo, para las pruebas es necesario el

uso de algún tipo de elemento que sirva como objetivo, ya sea una pelota o algún elemento similar. Para permitir que existan elementos en el campo, pero que no se dañe a los robots reales, se ha decidido el uso de objetos virtuales. Esto quiere decir que habrá objetos que serán visibles en la cámara los robots pero no serán visibles físicamente. La idea es la misma de la RA, pero aplicada a los robots, en vez de a las personas. Para ello será necesario el uso de un módulo que permite controlar los objetos que hay en el campo, enviando el posicionamiento de éstos a los robots, los cuales los mostrarán en la cámara realizando una estimación de su posición y tamaño para que el supervisor pueda determinar donde están durante los entrenamientos.

Con estas modificaciones, se espera poder solventar los problemas detallados en los apartados anteriores. Sin embargo, con este nuevo enfoque surge otro problema a tener en cuenta, ya que para conseguir un comportamiento complejo coordinado hay que determinar como dividirlo de forma adecuada en otros comportamientos más sencillos, lo cual se estudiará en los próximos apartados.

Por último, y para comprobar la viabilidad del enfoque híbrido propuesto y las posibilidades de realizar comportamientos coordinados más complejos, se realizarán pruebas usando tres robots AIBO ERS-7 en un campo similar al de la liga de fútbol Robocup, escalado a 2 x 1'5 metros, con balizas de colores dispuestas en la banda y sin obstáculos en el campo. Estos robots estarán divididos en dos equipos, dos atacantes y un defensor, siendo el objetivo de los atacantes marcar un gol y el del defensor evitarlo. Para marcar un gol se usará, como se ha comentado, una pelota virtual, que será visible a través de las cámaras de los robots y también a través de un módulo central que muestra la situación de todos los elementos (reales y virtuales) en el campo.

La razón de usar dos atacantes contra un defensor es, como ya se ha comentado, que se disponía únicamente de tres robots para las pruebas. Sin embargo, y suponiendo que se dispusiese de robots para formar dos equipos de dos robots cada uno, dado que los robots tienen las mismas características (misma rapidez y velocidad) y si se supone que ambos equipos tienen algoritmos similares en cuanto a estrategia, la situación de dos contra dos se puede convertir fácilmente en uno contra uno (si cada defensor cubre a un atacante), no pudiéndose estudiar o destacar la importancia de la coordinación. En el caso concreto de dos robots contra uno, y a pesar de la ventaja numérica de los atacantes, si éstos no se coordinan adecuadamente durante el ataque al defensor, la ventaja de dos contra uno se convertirá en un enfrentamiento uno contra uno o incluso uno contra uno contra uno (si los dos miembros del mismo equipo llegasen a competir entre ellos), dependiendo la victoria en este caso del azar y no de la coordinación. Así pues, esta distribución de

robots, a pesar de la ventaja numérica, requiere de la coordinación de los robots atacantes para poder superar al único defensor y marcar un gol.

Los próximos apartados detallan el comportamiento coordinado que se quiere implementar, su descomposición en comportamientos más elementales, el funcionamiento de los elementos virtuales, la implementación del sistema, etc.

6.3.1. Descripción del comportamiento coordinado

Como se ha comentado, la tarea a realizar consiste en que un equipo, formado por dos robots AIBO, debe coordinarse para superar al equipo contrario, formado por un robot AIBO, para conseguir marcar un gol en la portería contraria.

Por lo tanto, los robots atacantes, situados en su propio campo, deberán capturar la pelota, que estará en su campo, y dirigirse al campo enemigo de forma coordinada y evitando interferirse. El defensor, por su parte, para evitar alejarse mucho de la portería esperará a que los robots atacantes entren en su campo, dirigiéndose entonces a bloquear al robot que tenga la posesión de la pelota. La razón de que el defensor realice este comportamiento en vez de quedarse en la portería bloqueando el mayor espacio posible es que, si hiciese esto, haría que la coordinación de los robots atacantes perdiese sentido, ya que realmente se trataría de un problema de puntería a la hora de lanzar la pelota y no tanto de coordinación entre los robots. Así pues, al bloquear el defensor a uno de los atacantes, el atacante bloqueado deberá pasar la pelota al compañero, que intentará continuar la jugada y sortear al defensor, cogiendo antes la pelota (lo cual puede hacer que el defensor tenga tiempo de volver a bloquear al segundo atacante). Así pues, este es, a grandes rasgos, el comportamiento coordinado de alto nivel que se va a implementar. Gracias a la implementación basada en aprendizaje, no es necesario definir con más detalle la estrategia a seguir, pues los comportamientos se especificarán durante el entrenamiento y la combinación de éstos se indicará mediante la configuración de la capa de selección de comportamientos, lo cual será realizado manualmente por un experto.

La pelota que usada será una pelota virtual para evitar el factor azar en el juego y destacar la coordinación de los robots, así como para evitar que los robots se deterioren debido a los movimientos que tendrían que realizar con una pelota real. Esta pelota se puede observar mediante la cámara del robot y mediante un módulo central que genera la posición de la pelota y muestra una perspectiva de todo el entorno (robots, pelota y porterías).

Como se ha comentado, al ser un equipo de dos atacantes contra un defensor, el equipo atacante tiene, obviamente, la ventaja numérica. Sin embargo,

para que esta ventaja se refleje en un resultado, es necesario que los dos atacantes se coordinen correctamente, siendo éste el problema a resolver.

Así pues, la coordinación, como tal, dependerá de la capa de selección de comportamientos aprendidos y de la propia emergencia surgida de la combinación de los comportamientos elementales aprendidos. Esta capa está implementada de forma analítica mediante reglas de decisión preestablecidas del tipo *si-entonces*. Así pues, y según las reglas definidas a priori, esta capa determinará que comportamientos se activarán y cuales se deshabilitarán.

A continuación se detalla la descomposición en comportamientos elementales, el enfoque planteado para realizar los entrenamientos de estos comportamientos de una forma sencilla y práctica y la capa de selección de comportamientos.

6.3.1.1. Comportamientos elementales

Para obtener un correcto comportamiento coordinado emergente hay dos elementos importantes a tener en cuenta: la descomposición en comportamientos elementales y la combinación de éstos. En este apartado se detallará la descomposición que se ha considerado adecuada para el comportamiento global a realizar.

Un detalle que es importante tener en cuenta a la hora de descomponer un comportamiento en otros más elementales es que hay que intentar que éstos sean lo más genéricos posible para que puedan ser reutilizados en distintas situaciones. Por lo tanto, a la hora de descomponer los comportamientos, en vez de mencionar una pelota, una portería o un compañero de equipo, estos comportamientos elementales consideran un objeto, pudiendo ser este objeto una portería o un obstáculo, dependiendo de la situación.

Así pues, y teniendo en cuenta el comportamiento global coordinado de los dos robots atacantes, así como el comportamiento del defensor para evitar que los atacantes alcancen la portería, se ha estimado que será necesario implementar, por lo menos, los siguientes comportamientos elementales:

- Capturar objeto: el objetivo de este comportamiento es alcanzar un objeto (pelota, portería enemiga, etc.) desde la posición actual.

- Bordear objeto: el objetivo de este comportamiento es evitar un objeto cercano (obstáculo, compañero de equipo u oponente) en la dirección de movimiento.

- Evitar colisión: el objetivo de este comportamiento es evitar una colisión inminente con un obstáculo (objeto, compañero de equipo u opo-

nente). Este es un comportamiento de emergencia que se superpone sobre el resto de comportamientos para evitar una colisión.

- Moverse por la banda: el objetivo de este comportamiento es que el robot se mueva por la banda del campo para aproximarse a un objetivo (la portería enemiga, por ejemplo).

- Moverse en paralelo: el objetivo de este comportamiento es que el robot se mueva en paralelo (y manteniendo las distancias) a un objeto (por ejemplo, un compañero de equipo).

- Pasar pelota: el objetivo de este comportamiento es pasar o chutar la pelota, pudiendo lanzar la pelota hacia delante o hacia los lados.

- Defender objeto: el objetivo de este comportamiento es proteger un objeto de otro objeto (por ejemplo, defender la portería propia de un oponente).

Se ha considerado que una correcta combinación de estos comportamientos debería devolver un comportamiento de alto nivel adecuado a las situaciones más usuales en el caso de los atacantes , así como una respuesta adecuada en el caso del defensor.

Respecto a la implementación de los comportamientos, los más simples, como el comportamiento de *evitar colisión* o *pasar pelota* serán implementados analíticamente, mientras que los demás serán aprendidos, describiéndose a continuación la forma en que se ha realizado el entrenamiento.

6.3.1.2. Entrenamiento de los comportamientos

En el capítulo anterior se propuso el uso de un entrenamiento secuencial para el entrenamiento coordinado de los robots por resultar más realista. Sin embargo, este tipo de entrenamiento no es práctico y resulta bastante complejo y tedioso de realizar.

Así pues, en este capítulo se propone el entrenamiento coordinado de los robots usando un robot R_{ref} parado, que enviará su posición al robot a entrenar R_{ent}. De esta forma, mientras el robot R_{ref} está parado, se podrá controlar al robot R_{ent} para enseñarle patrones de movimiento teniendo en cuenta la posición de R_{ref}. Este entrenamiento resulta más *aséptico* y menos realista que el entrenamiento secuencial, pero resulta mucho más práctico y sencillo de realizar, además de evitar un entrenamiento extra (el del robot que se mueve de forma autónoma mientras se realiza el entrenamiento secuencial). Por lo tanto, este nuevo enfoque tiene la desventaja de perder realismo en

el entrenamiento, además de que hay que estimar qué movimiento se quiere que realice el robot respecto al robot parado, lo cual puede implicar un cierto estudio de las situaciones que se quieran afrontar cuando el robot R_{ref} esté realmente en movimiento. Sin embargo, teniendo en cuenta que los comportamientos actúan a nivel reactivo, el hecho de que el robot esté parado no es un problema pues de forma instantánea, todos los robots y objetos se consideran que están parados.

Por otro lado, como se ha comentado anteriormente, ahora se dispone tanto de un simulador como de objetos virtuales (sección 6.3.2), por lo que existen distintas posibilidades para realizar los entrenamientos, teniendo cada una de ellas unas ventajas y desventajas. Así pues, se pueden realizar entrenamientos, como hasta ahora, con el robot real y usando otro robot parado, como se ha comentado, con lo que se tendría en cuenta el tamaño de éste y se mantendrían las ventajas del aprendizaje basado en demostración, indicadas en los capítulos precedentes. Otra alternativa sería el uso de un robot real para entrenar y objetos virtuales para interactuar, por lo que habría que usar la visión del robot o la información global del entorno para poder controlar al robot durante el entrenamiento, perdiéndose parte de las ventajas anteriores (concretamente las relacionadas con el tamaño del objeto a evitar o alcanzar así como la absorción de los errores en la visión), pero se podría tener una mayor facilidad para realizar los entrenamientos. En cualquier caso, hay que tener en cuenta que como en este capítulo se trabaja con las posiciones de los robots y los objetos, el entrenamiento ya no absorbe los errores visuales, pues la información visual no se usa directamente en el entrenamiento, sino que se usa la información procesada (posición de los objetos). Por último, se podría usar para entrenar el simulador junto con objetos virtuales. En este caso se perderían la mayoría de ventajas de absorción de errores y cinemática real del robot, a costa de obtener un entrenamiento mucho más sencillo de realizar.

Así pues, y de forma general los entrenamientos se realizarán inicialmente en simulador (con objetos virtuales o con otro robot) para comprobar si la respuesta obtenida coincide con la esperada. Tras esto, se probara el entrenamiento en un robot real para ver si responde correctamente y, de no ser así, se repetirá el entrenamiento con el robot real, teniendo ya en cuenta los problemas obtenidos en el simulador, facilitándose así el proceso de entrenamiento.

6.3.1.3. Combinación de los comportamientos

Tras comentar la descomposición del comportamiento global en comportamientos elementales y como se realiza el entrenamiento de estos comporta-

mientos, falta indicar cómo se combinarán para dar lugar al comportamiento esperado.

Como ya se ha indicado, la combinación de los comportamientos se realizará mediante una capa de alto nivel que determinará, a partir de una serie de reglas preestablecidas, que comportamiento activar y qué parámetros de entrada recibirá (pues como se ha comentado, los parámetros de entrada de los comportamientos son, en principio, genéricos, pudiendo ser un objeto la pelota o un robot). Concretamente, la configuración de esta capa de selección de comportamientos la realizará un experto que determinará el comportamiento emergente que se espera obtener a partir de la combinación de comportamientos simples configurada.

Esta capa, además, calcula el comando final de movimiento enviado al robot mediante la suma ponderada de las salidas de los distintos comportamientos, dependiendo el peso de cada comportamiento de la situación concreta.

Las posibles combinaciones de comportamientos que permite esta capa son: comportamientos en secuencia, en paralelo o una combinación de ambos. Así, por ejemplo, si se quiere que el robot busque la pelota y, tras capturarla, marque un gol, tan sólo es necesario configurar esta capa para que ejecute de forma secuencial el comportamiento de *capturar objeto* dos veces, teniendo como entrada la pelota en el primero de los casos y la portería en el segundo de ellos. Por lo tanto, esta capa de alto nivel determina que entrada se enviará al comportamiento de bajo nivel. Por ejemplo, si se quiere que el comportamiento indicado anteriormente sea combinado con un evitador de obstáculos, tan sólo sería necesario ejecutar en paralelo el comportamiento *evitar objeto* con los otros dos comportamientos, estableciendo como entrada al comportamiento el obstáculo a evitar (por ejemplo, el robot más cercano) y que el objeto esté en la dirección de movimiento como condición de disparo.

6.3.2. Información y objetos virtuales

El uso de una localización global permite la posibilidad de crear un modelado completo del sistema, pudiendo conocer tanto la posición de los distintos robots como crear destinos u objetos asociados a una posición concreta. Así pues, la figura 6.7.a muestra el modelado del entorno realizado en base al posicionamiento absoluto en el campo. Como se puede ver, en esta figura están representadas las balizas, un robot, una pelota y las porterías, pudiéndose tener una perspectiva global de la situación.

De los elementos mostrados en este modelado del entorno, las balizas y el robot pueden ser reales o simulados en el simulador Mirage. Sin embargo, las porterías y la pelota son virtuales, es decir, son objetos virtuales que no

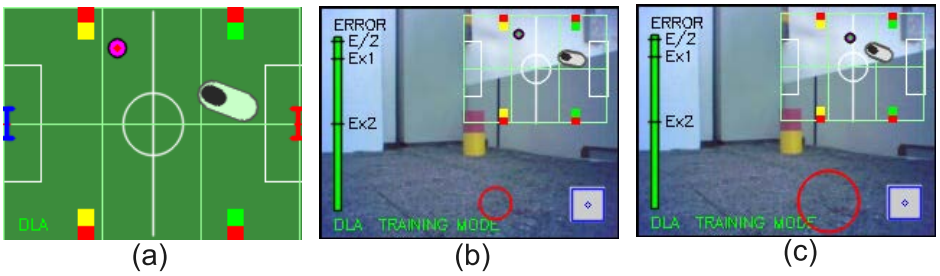

Figura 6.7: Modelo del sistema (a) y objetos virtuales (b) y (c).

existen ni en realidad ni en el simulador.

Como se ha comentado anteriormente, hay varias razones para usar elementos virtuales. Por un lado, los robots usados dejaron de producirse hace tiempo y no es viable su reparación, por lo que se ha intentado evitar en la medida de lo posible causar cualquier tipo de estrés o movimientos bruscos que pudiesen acelerar el deterioro de los robots, especialmente de los motores del cuello, que son los que más sufren durante el movimiento y las secuencias de disparo. Otra ventaja del uso de los elementos virtuales es que facilita la movilidad y la realización de entrenamientos sin necesidad de realizar grandes despliegues. Por otro lado, y a pesar de que el uso de elementos virtuales supone una pérdida de realismo, pero también supone una ventaja, pues permite eliminar el factor azar en el juego (ya que el movimiento de la pelota cuando dos robots colisionan es prácticamente indeterminado) lo que permite centrarse y dar más importancia a la propia coordinación, que es lo que se está buscando.

También hay que tener en cuenta que el hecho de usar elementos virtuales, similar al uso de la RA sólo que en vez de aplicarla a las personas se aplica a los robots, resulta interesante porque sólo a través de lo que ven los robots es posible descubrir elementos que no son visibles de otra manera. Esto hace, por un lado, que al realizar los entrenamientos de los comportamientos el operador tenga que estar atento a lo que realmente están captando los robots, no pudiendo hacer uso de otra información más que la que éstos observan, resultando en un entrenamiento más *justo* para el robot, pues el operador hará uso casi exclusivo de la información de la que dispone el robot, no haciendo *trampas* usando información adicional.

Así, por ejemplo, las figuras 6.7.b y c muestran como vería el robot la pelota virtual (el círculo rojo) a distintas distancias. Si bien la representación de la pelota es muy tosca y poco realista, pero permite estimar tanto su posición como distancia respecto al robot. Además, el operador dispone, superpuesto a la imagen de la cámara, de un mapa del entorno y la posición

de los elementos tanto virtuales como reales, lo que le permite tener una panorámica de la situación, al igual que la tiene el robot.

Por lo tanto, durante el entrenamiento se mezclarán objetos reales (los robots y las balizas) con objetos virtuales (la pelota, las porterías y otros elementos que puedan crearse), perdiendo cierto realismo al aplicar este enfoque pero favoreciendo por otro lado la posibilidad de una mayor interacción.

6.3.3. Funcionamiento del sistema

El sistema a desarrollar se trata de un sistema coordinado basado en comportamientos aprendidos, utilizando, por lo tanto, el enfoque planteado en los capítulos anteriores. Así pues, el funcionamiento general del sistema será prácticamente igual al ya presentado, aunque en este caso estará formado por tres fases en lugar de dos, que son:

- Aprendizaje supervisado. Durante esta fase, y al igual que en los capítulos anteriores, se entrena a los robots mediante control remoto, creándose así las bases de casos que se usarán en la fase de navegación autónoma. En este caso, a diferencia de los capítulos anteriores, se aplicará esta fase por cada comportamiento básico o elemental a implementar que, como se ha indicado, son varios comportamientos.

- Configuración de la capa de selección de comportamientos. Antes de pasar a la etapa de navegación autónoma, es necesario determinar las entradas de cada uno de los CBRs entrenados durante la etapa anterior (ya que, como se ha indicado, la entrada del CBR es genérica) así como la forma en que se activarán y desactivarán los distintos comportamientos en base a cada situación. Para ello, será necesario configurar la capa de selección de comportamientos, que se encarga de esta tarea.

- Navegación autónoma. Durante la fase de navegación autónoma el robot funcionará de forma autónoma sin supervisión, enviando a los distintos CBR información sobre el entorno y el estado de los robots, de forma que el CBR devolverá los comandos de movimiento previamente aprendidos que más se adecúen a la situación actual. Durante esta fase se lanzarán tantos CBRs como comportamientos elementales hayan sido entrenados, siendo cada CBR activado en función de la situación por la capa de selección de comportamientos. El comando de movimiento enviado finalmente al robot será la suma ponderada de las distintas salidas devueltas por los comportamientos activos en un momento dado.

Es importante tener en cuenta que, como se ha comentado, no todos los comportamientos elementales son aprendidos, pues alguno más simple, como el de *evitar colisión* o *pasar pelota*, se implementan de forma analítica.

6.3.4. Implementación del sistema

En este apartado se describe la implementación del sistema de coordinación híbrido basado en comportamientos aprendidos. En la figura 6.8 se muestra el diagrama de bloques general que, como se puede observar, sigue el esquema ya indicado en la figura 6.2, estando formado por tres bloques principales:

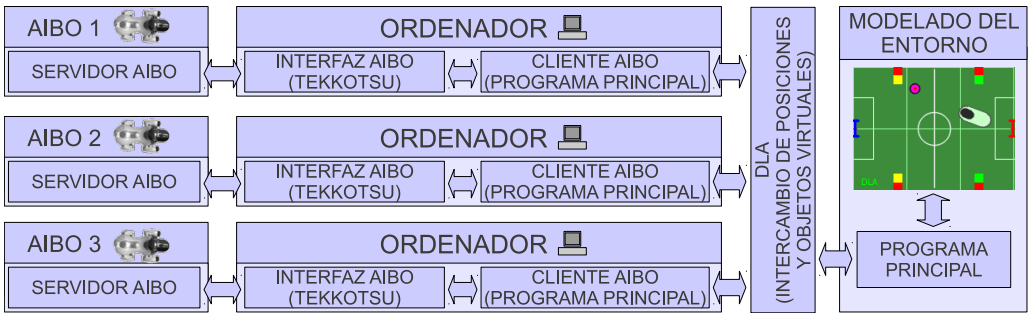

Figura 6.8: Diagrama de bloques general del sistema.

- Servidor AIBO. Este programa, es el mismo que ya se mostró en el capítulo 4 (sección 4.4.2.2), por lo que está implementado usando Tekkotsu y se encarga de enviar al ordenador el estado y las imágenes capturadas por el robot y ejecutar los comandos recibidos de éste. La comunicación entre el robot y el ordenador se realiza via WiFi.

- Interfaz AIBO. Este módulo, desarrollado usando Tekkotsu, hace de interfaz entre el AIBO y el programa principal del ordenador (Cliente AIBO), permitiendo aislar el uso de Tekkotsu a una simple capa de interfaz que se conecta con el programa principal mediante el uso de unos comandos *genéricos*. De esta forma se consigue tener más control sobre los posibles errores del programa principal y su consumo en CPU y memoria (como se indicó en el apartado 6.2.3).

- Cliente AIBO. Este módulo es el programa principal del ordenador. Desarrollado en C++ y usando la librería gráfica FLTK[8] (una librería

[8] `http://www.fltk.org/index.php`. Última visita: 10-10-2015

gráfica rápida y eficiente), este módulo es el encargado de implementar el sistema de coordinación propiamente dicho y mostrar al usuario el interfaz gráfico para el control del robot. También permite las funcionalidades de depuración del CBR así como realizar los entrenamientos de los robots. A continuación se describirán en más detalle los distintos módulos que lo componen.

- DLA. El DLA, al igual que en los capítulos anteriores, es el módulo que permite el intercambio de información (posiciones de los robots y posición de los objetos virtuales) entre los distintos robots. Además, el DLA también permite una fácil escalabilidad del sistema ya que la adición de nuevos robots supone, tan sólo, añadir una nueva conexión, lo cual resulta muy sencillo usando DLA.

- Modelado del entorno. Este modulo dispone de un interfaz gráfico para mostrar al usuario una panorámica completa del entorno, mostrando la posición de los distintos robots conectados y los objetos virtuales en el campo. Por otro lado, este módulo también aplica efectos físicos elementales (rozamiento) a los objetos virtuales en el caso de que se muevan (como la pelota), dándoles un movimiento realista. El uso de este módulo, que centraliza el posicionamiento de los objetos con los que se puede interactuar, puede verse como una variante del uso de un único robot que coordine las acciones de los demás mediante la determinación de qué información usar, eliminando la ambigüedad de información entre los distintos robots [113], permitiendo, así, centrarse en el problema de la coordinación.

De estos módulos, el más complejo es el *Cliente AIBO*, por lo que se detallará a continuación su estructura y composición de módulos. La figura 6.9 muestra el diagrama de bloques del programa del ordenador, detallando especialmente el módulo Cliente AIBO. Como se puede observar, la estructura de este módulo es similar a la del sistema de coordinación del capítulo anterior. Al igual que en el anterior diseño, el módulo está formado por dos partes que se pueden conmutar, permitiendo el entrenamiento del robot (modo entrenamiento) o la navegación autónoma (modo autónomo). Respecto a las diferencias con el diseño del capítulo anterior, los principales cambios son el uso de Tekkotsu únicamente para interfaz con el Cliente AIBO, implementado en C++ con FLTK como librería gráfica, el cambio del módulo de localización o la adición de la capa de selección de comportamientos. A continuación se detallan los distintos módulos y los cambios realizados:

- Programa principal. Este programa se encarga de controlar y gestionar

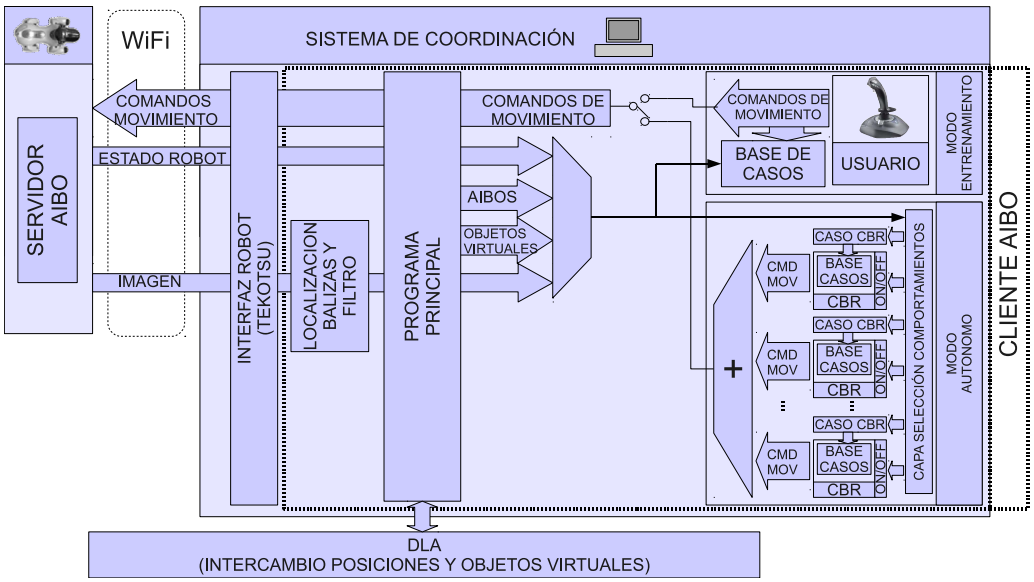

Figura 6.9: Diagrama de bloques del módulo Cliente AIBO.

toda la información que llega del robot y del DLA y distribuirla adecuadamente a los distintos módulos que componen el Cliente AIBO. También es el encargado de mostrar el interfaz de usuario, como ya se ha comentado implementado mediante FLTK, mostrando la imagen de la cámara del robot y su estado al usuario. Este interfaz permite, además, un sencillo control del robot mediante teclado o joystick.

Debido a que este módulo gestiona la información que llega, su funcionamiento será distinto en modo entrenamiento y en modo autónomo. Así pues, en modo entrenamiento, este módulo envía al robot los comandos de movimiento realizados por el operador mediante el joystick (pues como se observa en la figura 6.9 puede conmutar entre los comandos del usuario y del CBR) generando, en paralelo, el caso a almacenar en la base de casos CBR. En modo navegación autónoma, este módulo envía la información necesaria (posición del robot y de otros robots, de objetos virtuales, etc.) a la capa de selección de comportamientos para que ésta pueda activar el comportamiento CBR entrenado adecuado y pasarle los parámetros de entrada.

- Módulo de posicionamiento mediante balizas de colores y filtro de partículas. Este módulo, implementado en C++, es el encargado de procesar las imágenes recibidas (mediante OpenCV) para detectar las balizas de colores, estimar su posición respecto al robot y determinar la

posición más probable del robot actualmente, usando para ello el filtro de partículas, las observaciones de balizas ya indicadas y los movimientos del robot. La configuración del filtro de partículas es la misma que ya se indicó en el apartado 4.4.3.1.

Esta módulo devuelve al programa principal la posición estimada, la cual se enviará al DLA para que el resto de robots puedan saber la posición del robot, actuando en consecuencia.

- Capa de selección de comportamientos. La capa de selección de comportamientos es una capa de nivel superior que permite activar y desactivar los distintos comportamientos CBR entrenados en función de la situación concreta a afrontar. Esta capa es la encargada de determinar los parámetros de entrada de cada comportamiento (pues en principio los comportamientos son genéricos, pudiendo aceptar como entrada tanto la posición de la pelota como la posición de un robot) así como de activarlos.

  Este módulo se ha implementado de forma analítica mediante reglas estáticamente configuradas del tipo *si-entonces*. Así pues, esta capa comprueba el conjunto de condiciones establecidas y, en función del resultado, lanza los correspondientes comportamientos de bajo nivel. Esta capa también calcula el comando final de movimiento enviado al robot (obtenido como la media ponderada de todas las salidas de los comportamientos de bajo nivel) y establece los correspondientes pesos de cada comportamiento.

  Por último, indicar que este módulo permite la combinación de los comportamientos de tres formas distintas: comportamientos en secuencia, en paralelo o una combinación de ambos. La combinación de comportamientos se configura mediante un fichero de texto, creado por un experto, que se lee en tiempo de ejecución.

- Módulo CBR. El módulo CBR es, como ya se ha indicado, una implementación propia del CBR desarrollada para suplir algunos problemas encontrados en el servidor CBR desarrollado por el grupo KEMLg de la Universidad Politécnica de Cataluña (UPC). Concretamente, el CBR desarrollado, llamado LibCBR, se trata de una versión básica con la funcionalidad necesaria para los objetivos del presente trabajo, permitiendo su integración en el programa principal (sin tener que usar un servidor por cada comportamiento), ofreciendo una rápida respuesta, fácil depuración de las bases de datos (mediante un depurador gráfico desarrollado a tal efecto) y la inclusión y modificación en tiempo real de casos si fuese necesario.

Para finalizar de detallar la implementación del sistema realizado se indicarán los distintos parámetros de configuración del CBR, lo cual se hará en los siguiente apartados.

6.3.5. Descripción del CBR

En esta sección se detallan los parámetros que describen el CBR, como la estructura de la memoria de casos del CBR, los parámetros que forman los casos o los distintos algoritmos utilizados en cada una de las fases.

6.3.5.1. Estructura de la base de casos

La estructura de la base de casos, como ya se ha comentado, es una estructura plana, pues el CBR desarrollado está especialmente enfocado al uso de este tipo de estructura. Dado que esta estructura no es adecuada cuando hay una gran cantidad de casos, es conveniente mantener acotado el número de casos de la base de casos, por lo que se aplicará un algoritmo de agrupación para que el sistema CBR no reduzca sus prestaciones.

El algoritmo de agrupación utilizado, al igual que en los capítulos anteriores, selecciona en primer lugar una semilla y comprueba si hay casos similares en la base de casos, considerándose dos casos similares si su distancia es menor a un umbral D_{umbral} (establecido heurísticamente). Todos los casos que pertenezcan a un mismo grupo son promediados en un prototipo que se convierte en el nuevo caso de la base de casos, sustituyendo la semilla inicial. Finalmente, se selecciona una nueva semilla y se repite el proceso hasta que todos los casos de la base de casos hayan sido procesados por el algoritmo. De esta forma se puede mantener acotado el número de casos, reduciendo a la vez la redundancia y fluctuaciones o movimientos espasmódicos debido a casos con entradas similares y salidas contradictorias o parcialmente distintas. En el caso de situaciones contradictorias (entrada similares y salidas distintas) el algoritmo marcará estos casos y avisará al usuario para su supervisión.

Respecto a la función de similitud, se ha usado la distancia Chebyshev en vez de la distancia Manhattan porque se hicieron pruebas con ambas distancias y los valores obtenidos fueron mejores con la primera distancia. Además, con esta distancia se puede controlar la diferencia máxima permitida en las componentes a la hora de buscar casos similares.

6.3.5.2. Definición del caso CBR

Esta sección presenta la definición del caso para los distintos comportamientos entrenados. Un caso CBR, como ya se ha indicado, es un vector

formado por dos componentes entrada-salida, que describen un problema y su solución respectivamente. La definición de un caso puede, además, incluir una componente para la eficiencia, aunque en este trabajo no se añade porque ésta será determinada por el operador mediante observación directa durante la navegación autónoma.

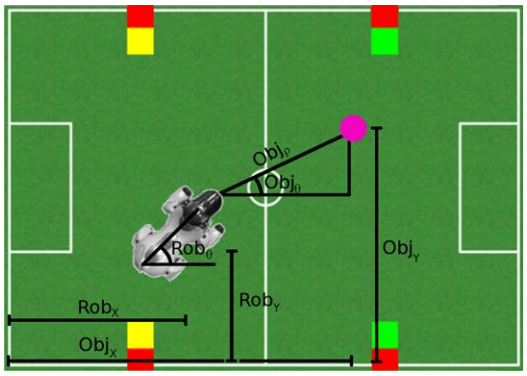

Figura 6.10: Representación de las variables usadas en los casos CBR.

Es importante recordar que, respecto a la parametrización de la entrada del caso, es conveniente que éste contenga únicamente la información necesaria para describir la situación del entorno, lo que favorecerá que el sistema CBR trabaje mejor al eliminar toda la redundancia posible en el caso de entrada.

Así pues, y teniendo esto en cuenta, en el presente trabajo la salida de todos los comportamientos es la misma (el comando de movimiento que el robot tiene que ejecutar). Sin embargo, la entrada dependerá del comportamiento en cuestión. La tabla 6.1 resume las entradas usadas para cada comportamiento propuesto para implementar mediante aprendizaje. Por su parte, la figura 6.10 muestra una representación gráfica de las variables usadas en la definición del caso de entrada.

	Entradas de los casos CBR			
	Ent_1	Ent_2	Ent_3	Ent_4
Capturar objeto	Obj_ρ	Obj_θ		
Bordear objeto	Obj_ρ	Obj_θ		
Moverse por la banda	$\theta_{ref} - Rob_\theta$	Rob_Y		
Moverse en paralelo	$\theta_{ref} - Rob_\theta$	$Obj_X^1 - Rob_X$	$Obj_Y^1 - Rob_Y$	
Defender objeto	$Obj_X^1 - Rob_X$	$Obj_Y^1 - Rob_Y$	$Obj_X^2 - Rob_X$	$Obj_Y^2 - Rob_Y$

Tabla 6.1: Entrada de los casos CBR para cada comportamiento.

Siendo $(Rob_X, Rob_Y, Rob_\theta)$ las coordenadas cartesianas absolutas del robot y de su orientación, (Obj_ρ, Obj_θ) las coordenadas polares relativas del objeto respecto a la posición del robot, (Obj_X^N, Obj_Y^N) las coordenadas cartesianas absolutas del objeto N y θ_{ref} el ángulo de referencia para que el robot se mueva en dicha dirección (0° para moverse a la derecha y 180° para la izquierda).

Todas las distancias son medidas en centímetros y los ángulos en grados. Se puede observar, por ejemplo, que el comportamiento *capturar objeto* sólo depende de las posiciones polares relativas (respecto al robot) del objeto a capturar (pelota, portería, etc.), mientras que el comportamiento de *defender objeto* depende de la posición cartesiana relativa (respecto al robot), del objeto a defender (objeto 1) y del objeto que *ataca* (objeto 2).

6.3.5.3. Descripción de la fase de recuperación

Para la descripción de esta fase es necesario especificar tanto el algoritmo de búsqueda que usará como la distancia de similitud.

El algoritmo de búsqueda, como ya se ha indicado, será el algoritmo de búsqueda por vecindad, pues es un algoritmo muy utilizado y el único que se ha implementado en la librería LibCBR. El problema de este algoritmo es el tiempo de respuesta si la base de casos crece demasiado pero, dado que se aplicará un algoritmo de agrupación para evitarlo, esto no afectará.

En lo que respecta a la función de distancia para estimar la similitud de los casos, se usará la distancia Chebyshev para poder determinar la máxima diferencia en las componentes del caso CBR durante las búsquedas.

6.3.5.4. Descripción de la fase de reutilización

La fase de reutilización se describe especificando el algoritmo de adaptación usado. Sin embargo, y dado que en este capítulo la idea es hacer los comportamientos lo más simples y elementales posible, se ha decidido no aplicar ninguna adaptación a los comportamientos.

Por otro lado, hay que tener en cuenta que si se aplicase adaptación a los distintos comportamientos resultaría complejo poder depurar la respuesta obtenida a posteriori, debido a que no se podría determinar fácilmente si un problema se origina por la combinación de los comportamientos, por su adaptación, o por una mezcla de ambos efectos.

6.3.5.5. Descripción de la fase de revisión

Para la descripción de la fase de revisión hay que indicar como se realiza la validación de las soluciones adaptadas y, en general, de las soluciones

aportadas por el CBR.

Continuando con el enfoque ya comentado en los capítulos anteriores, la validación de los casos se hará de forma manual mediante observación directa por parte de un experto. Así pues, si durante la ejecución en modo autónomo el sistema se coordina adecuadamente y alcanza la meta, se considerará que los casos aplicados han sido correctos. En caso de observarse alguna conducta extraña o no alcanzar su destino, se estudiarían los ficheros de depuración del sistema para determinar el problema y solucionarlo si es posible.

Sin embargo, y dado que en este caso la salida depende de la combinación de distintos comportamientos entrenados, habrá que tener también en cuenta si las reglas de combinación de comportamientos se han escogido adecuadamente.

6.3.5.6. Descripción de la fase de retención

Por último, para describir la fase de retención hay que indicar los tipos de aprendizajes que se usarán en el CBR.

Como se ha comentado en la fase de reutilización, no se va a aplicar adaptación a los comportamientos aprendidos. Así pues, el único aprendizaje que se usará será aprendizaje por observación, pues es el que permite generar inicialmente la base de casos para que el sistema pueda trabajar de forma autónoma.

El aprendizaje por observación se realiza, como se ha mostrado a lo largo de los distintos capítulos, mediante control remoto mediante un interfaz que resulte cómodo y sencillo al operador.

6.4. Experimentos y resultados

Esta sección resume los resultados de las pruebas realizadas al sistema de coordinación híbrida basado en comportamientos aprendidos.

Las pruebas se han realizado inicialmente con el simulador, pasando posteriormente a pruebas reales usando tres robots AIBO ERS-7 de Sony, en un entorno de similar al campo de fútbol de la Robocup (escalado a 2 x 1'5 metros). Este campo de fútbol dispone de cuatro balizas de colores, tamaño conocido a priori, situadas en posiciones fijas en el entorno. El área de los experimentos está libre de obstáculos y los objetos con los que interactuarán los robots (pelota y porterías) son virtuales a excepción de los otros robots y balizas en el entorno. Esto quiere decir que los objetos virtuales son únicamente visibles a través de las cámaras de los robots, pudiendo éstos interactuar con ellos. La localización se hará mediante la detección visual de las

balizas de colores corrigiendo dicho posicionamiento con un filtro de partículas. Es conveniente recordar que todos los robots comparten su posición via WiFi.

Para probar la navegación coordinada, se han realizado dos combinaciones de comportamientos distintos involucrando a tres robots. Como se ha comentado, dos robots pertenecerán al mismo equipo (R1 y R2) y el otro robot será del equipo contrario (R3). En la primera de las pruebas el robot R3 se encontrará parado, mientras que en la segunda estará en movimiento tratando de bloquear al robot que tenga la posesión de la pelota. Los dos robots que tratan de marcar gol tienen como objetivo acercarse a la portería enemiga por separado, de forma que cuando el robot R3 trate de bloquear a uno de ellos el robot bloqueado pase la pelota al compañero para que finalice la jugada y marque gol.

Por último, y como se comentó anteriormente, el sistema está formado por tres fases, que son: entrenamiento de los comportamientos básicos, configuración de la capa de selección de comportamientos y funcionamiento autónomo. Así pues, los siguientes apartados detallan cada una de estas fases para las pruebas realizadas.

6.4.1. Entrenamiento de los comportamientos básicos

En este apartado se describe el entrenamiento de los comportamientos que no se han implementado mediante aprendizaje, que son: capturar objeto, bordear objeto, moverse por la banda, moverse en paralelo y defender objeto.

El entrenamiento de los robots se hace, como se ha comentado, usando un robot detenido, que envía su posición, mientras se controla remotamente al robot a entrenar. Este modo de entrenamiento resulta más artificial y *aséptico* que el entrenamiento secuencial usado en el capítulo anterior, pero resulta mucho más cómodo y práctico para realizar los entrenamientos.

Durante los entrenamientos se enseña a realizar cada uno de los comportamientos indicados de forma independiente. En los comportamientos en los que se supone un movimiento coordinado con otro robot, como en el comportamiento de moverse en paralelo, durante el propio entrenamiento el operador debe tener en cuenta la posición del robot con el que debe moverse en paralelo, de forma que implícitamente el robot absorba el comportamiento coordinado. Obviamente, el operador debe intentar cubrir la mayor cantidad de situaciones posibles durante los entrenamientos. En cualquier caso, y dado que el enfoque actual es que éstos sean lo más simple posible, es relativamente sencillo cubrir las situaciones más usuales sin mucho problema, siendo para esto importante la correcta definición de los comportamientos, como ya se ha indicado.

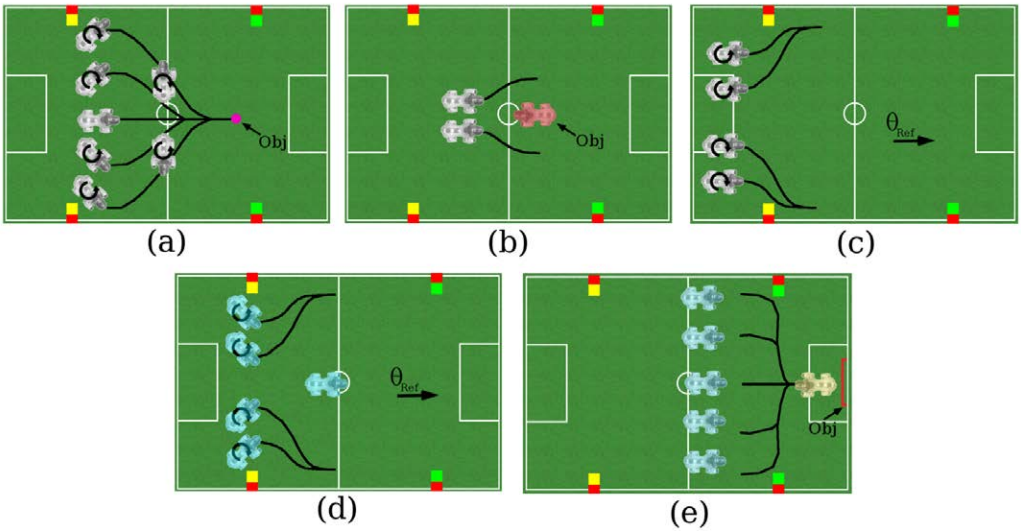

Figura 6.11: Entrenamiento de los comportamientos: a) capturar objeto, b) bordear objeto, c) moverse por la banda, d) moverse en paralelo y e) defender objeto.

Respecto a la tasa de almacenamiento de casos, estos son almacenados cada 100 milisegundos aproximadamente, y el tiempo requerido para los entrenamientos, si bien depende del propio entrenamiento en sí y del comportamiento buscado, pero de forma general se puede considerar que duran entre 5 y 10 minutos.

Los entrenamientos específicos realizados para la obtención de cada comportamiento se muestran en la figura 6.11. Aunque se podrían realizar más entrenamientos incluyendo más situaciones, pero estas trayectorias han demostrado ser suficientes para obtener los comportamientos esperados. Además, hay que tener en cuenta que en casi todos los comportamientos los parámetros de entrada son relativos al robot, y no absolutos respecto al campo, por lo que la posición del robot en el campo es indiferente, cubriéndose realmente muchas más situaciones que las indicadas expresamente en la figura.

Así pues la figura 6.11.a muestra el entrenamiento del comportamiento *capturar objeto*, enseñando al robot a alcanzar el objeto (la pelota) desde diferentes posiciones. La figura 6.11.b muestra el entrenamiento del comportamiento *bordear objeto*, enseñando al robot como evitar un objeto (un robot) que se encuentra en la dirección de movimiento. Por su parte, la figura 6.11.c presenta el entrenamiento del comportamiento *moverse por la banda*, en el que el robot se posiciona para dirigirse hacia el área enemiga siguiendo la banda del campo. En la figura 6.11.d se presenta el entrenamiento del

comportamiento *moverse en paralelo*, donde el robot aprende a moverse en paralelo y a una cierta distancia del objeto de referencia (un robot). Y por último, la figura 6.11.e muestra el entrenamiento del comportamiento *defender objeto*, donde el robot es entrenado para defender un objeto (la portería) bloqueando para ello otro objeto (un robot enemigo), previniendo que éste alcance el primer objeto (la portería).

Tras finalizar los entrenamientos, las bases de casos almacenadas son depuradas, eliminando casos repetidos, estudiando casos conflictivos, etc., reduciendo así las posibles oscilaciones de los comportamientos y evitando que el número de casos de la base de casos crezca demasiado.

Por último, recordar que dos de los comportamientos que se van a usar en las pruebas (*evitar colisión* y *pasar pelota*) son implementados de forma analítica, por lo que no son entrenados. Estos comportamientos, como se indicó en la sección 6.3.1.1, se han implementado de forma analítica porque son suficientemente simples y cerrados como para que no sea práctico su aprendizaje. Concretamente, el primer comportamiento evita que el robot realice movimientos en la dirección en la que se encuentre un obstáculo en caso de colisión inminente, deteniendo al robot por seguridad. El segundo, por su parte, permite lanzar la pelota, ya sea para marcar gol (lanzamiento frontal) o para pasar a un compañero (lanzamiento lateral), activándose cuando el robot tiene la posesión de la pelota y hay un enemigo en las proximidades en la dirección de movimiento. En ese caso, el robot pasa la pelota hacia el lado en que se encuentre el compañero de equipo más cercano. Este movimiento se realiza virtualmente, aunque en la realidad se haría con una secuencia de movimientos de cabeza en la que el robot golpea hacia izquierda o derecha la pelota con la cabeza. Sin embargo, y como se comentó anteriormente, para evitar el deterioro de las articulaciones del cuello se ha evitado realizar este movimiento con los robots reales.

6.4.2. Capa de selección de comportamientos

Tras detallar el entrenamiento de los comportamientos básicos hay que indicar cómo se combinarán estos para cada una de las pruebas a realizar, permitiendo así un comportamiento coordinado más complejo que los comportamientos entrenados.

En general, se considerará que todos los robots tienen siempre activos los comportamientos de evitar colisión y pasar pelota en paralelo con cualquier otra combinación posible de comportamientos. Esto es así para evitar colisiones y, en caso de que el robot tenga la pelota, evitar el bloqueo de ésta. En el caso de que se lance el comportamiento de evitar colisión este

tendrá preferencia sobre el resto de comportamientos por seguridad, impidiendo movimientos en la dirección en la que se encuentre el obstáculo.

Respecto a las pruebas a realizar, la primera de ellas consiste en que los robots atacantes (R1 y R2) deben coordinarse para superar al oponente (R3), el cual estará detenido. Así pues, en esta prueba la configuración de la capa será la siguiente. Para los robots R1 y R2, el robot más alejado de la pelota está configurado para ejecutar el comportamiento *evitar objeto* (entrada: el robot más cercano) y el comportamiento *moverse en paralelo* (entrada: el compañero de equipo más cercano), mientras que el más próximo a la pelota está configurado para ejecutar en secuencia el comportamiento *capturar objeto* (entrada: la pelota) y, tras capturar la pelota, el comportamiento *capturar objeto* (entrada: la portería enemiga). La activación de una secuencia u otra depende, como se ha indicado, de la proximidad de los robots a la pelota. Respecto al robot R3, este estará detenido y funciona únicamente como un obstáculo en esta ocasión, transmitiendo su posición.

En la segunda prueba, los robots atacantes (R1 y R2) están configurados para ejecutar varios comportamientos al mismo tiempo: i) *evitar objeto* (entrada: el robot más cercano); ii) *moverse en paralelo* (entrada: el compañero de equipo más cercano) si el robot no tiene la pelota ni es el robot más cercano a la pelota; iii) *capturar objeto* (entrada: pelota) si el robot no tiene la pelota y es el robot más cercano a la pelota; iv) *moverse por la banda* (entrada: ángulo de referencia de la portería enemiga) si el robot tiene la pelota y está a más de un metro de la portería; y v) *capturar objeto* (entrada: portería enemiga) si el robot tiene la pelota y la distancia a la portería es menor que 1 metro. Por su parte, el robot R3 sólo ejecuta el comportamiento *defender objeto* (entrada: su propia portería y el enemigo más cercano con la pelota), activándose cuando un robot enemigo tiene la posesión de la pelota.

6.4.3. Navegación coordinada autónoma

En esta sección se detallan las pruebas realizadas, una vez comentados tanto los entrenamientos de los comportamientos como sus condiciones de activación.

Así pues, la figura 6.12 muestra el resultado de la primera prueba. Concretamente, la figura 6.12.a muestra la posición de partida de la prueba. La figura 6.12.b muestra como el robot R2 es el más cercano a la pelota, por lo que se mueve directo hacia ésta (comportamiento: *capturar objeto*, entrada: pelota) mientras que el robot R1 trata de colocarse en paralelo (comportamiento: *moverse en paralelo*, entrada Robot R2) con el robot R2. En la figura 6.12.c se puede observar como el robot R2 ha capturado la pelota y se dirige hacia la portería enemiga (comportamiento: *capturar objeto*, entrada: por-

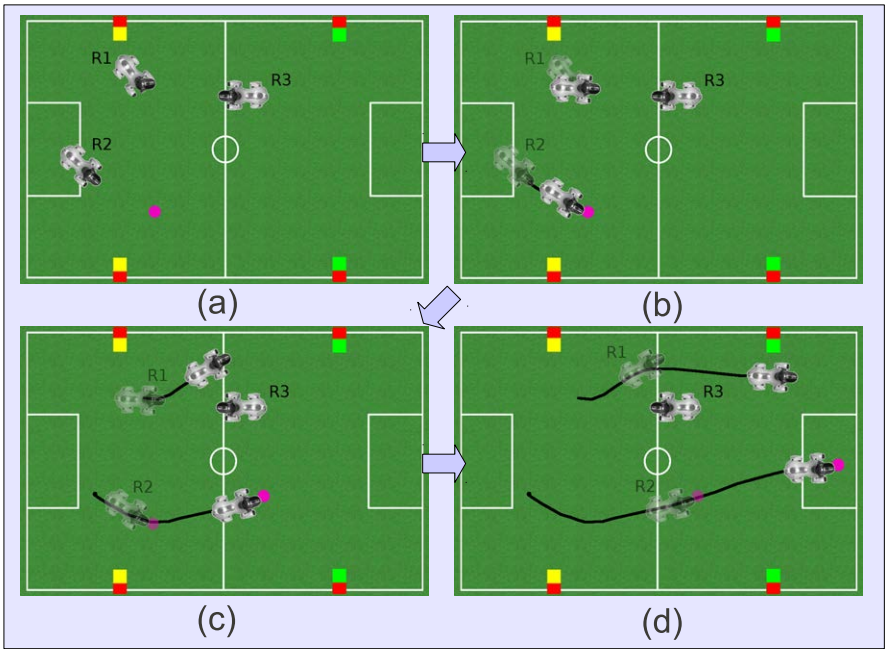

Figura 6.12: Prueba de coordinación con oponente parado.

tería enemiga) mientras que el robot R1 evita el robot R3 (comportamiento: *evitar objeto*, entrada: Robot R3) que está en la dirección de movimiento. Finalmente, en la figura 6.12.d se muestra como el robot R2 alcanza la portería enemiga y el robot R1 vuelve a su posición en paralelo respecto al robot R2. Los resultados de esta prueba son similares desde otras posiciones de inicio de los robots atacantes y defensor, demostrando que los comportamientos adquiridos en las trayectorias de entrenamiento en la figura 6.11 son suficientes para las pruebas a realizar.

Por su parte, la figura 6.13 muestra los resultados de la segunda prueba de coordinación usando los comportamientos aprendidos. Así pues, en la figura 6.13.a se puede ver la posición de inicio, observándose en la figura 6.13.b como el robot R2, el más cercano a la pelota, se dirige directamente hacia ésta mientras que el robot R1 empieza a posicionarse para colocarse en paralelo con el robot R2, manteniéndose mientras tanto parado el robot R3. La figura 6.13.c muestra como el robot R2, tras capturar la pelota, se dirige a la banda del campo, mientras el robot R1 se mueve en paralelo con el robot R2 por la otra banda. En esta misma figura se observa como el robot R3 se mueve para bloquear al robot R2 (el robot enemigo en posesión de la pelota), momento en el que el robot R2 pasa la pelota en la dirección en la que se encuentra el compañero (debido a la proximidad de R3). Por último, la figura 6.13.d muestra como el robot R2 se detiene debido a la inminente

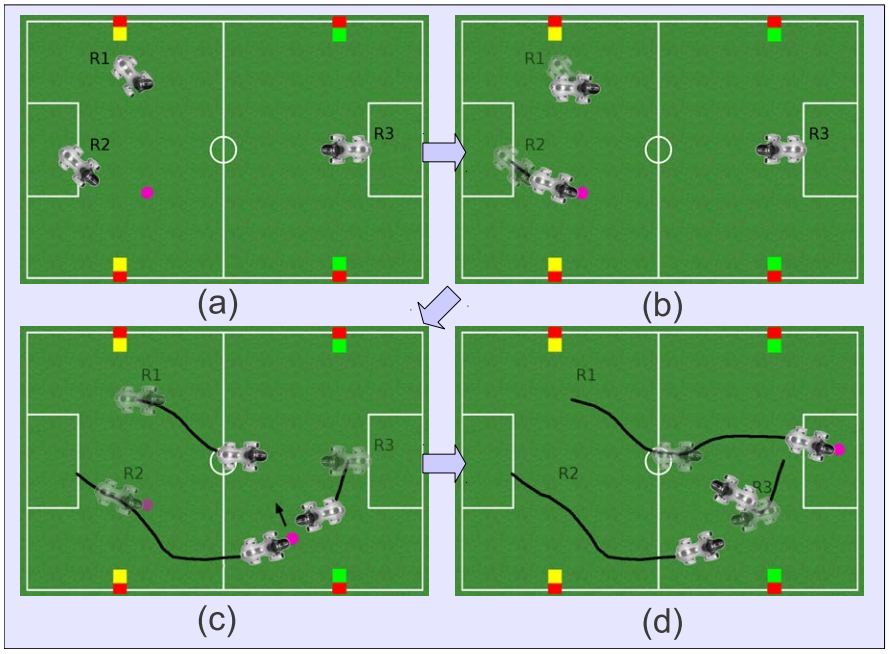

Figura 6.13: Prueba de coordinación con oponente en movimiento.

colisión con el robot R3 (bloqueando a R2). Tras esto, el robot R1 captura la pelota y se dirige hacia la portería, continuando la jugada. El robot R3 trata de detener al robot R1 (el robot más cercano con la posesión de la pelota) en su camino hacia la portería, pero el robot R1 consigue evitar al robot R3 y alcanzar la portería.

Este comportamiento se ha realizado cambiando las posiciones y orientaciones, obteniéndose diferentes comportamientos coordinados, pero en general con un comportamiento global similar, consiguiendo el equipo atacante alcanzar la portería en la mayoría de las ejecuciones.

Estas dos pruebas muestran como la división en comportamientos más simples y la adición de una capa de nivel superior para dispararlos de forma adecuada permite a los robots desarrollar comportamientos coordinados complejos mediante la combinación de un conjunto de comportamientos básicos aprendidos. Concretamente, la primera prueba muestra como se ha podido realizar el comportamiento que en el capítulo anterior resultó excesivamente complejo de entrenar de una forma relativamente sencilla. Por su parte, el segundo experimento muestra como se pueden conseguir comportamientos incluso más complejos mediante la combinación adecuada de comportamientos aprendidos.

6.5. Conclusiones

En este capítulo se ha presentado un sistema de coordinación basado en el uso de comportamientos aprendidos, combinados para obtener comportamientos complejos mediante una capa de alto nivel que los activa en función de la situación a afrontar. El entrenamiento de los robots se ha realizado mediante aprendizaje LfD usando CBR siendo éstos controlados por control remoto.

Al continuar el enfoque presentado en los capítulos anteriores, usando aprendizaje LfD, y gracias a la capacidad de adaptación de los humanos, se consiguen una serie de ventajas, ya comentadas, como la absorción implícitamente durante el entrenamiento de la dinámica y cinemática del robot y de los errores de los sensores (en el caso de realizar los entrenamientos con un robot real). Otra ventaja es que este enfoque permite obtener fácilmente una coordinación implícita entre los robots ya que el propio operador, durante el entrenamiento, tendrá en cuenta a los otros robots con lo que los movimientos dependerán de éstos. Por último, el planteamiento propuesto ofrece la posibilidad de realizar comportamientos concretos que no sean fáciles de codificar o implementar mediante expresiones analíticas.

En este capítulo, se han tratado de resolver la mayoría de problemas que se han ido descubriendo a lo largo de los capítulos anteriores, como la dificultad de realizar entrenamientos de comportamientos complejos, los problemas presentados con el entorno de trabajo Tekkotsu, las limitaciones del servidor CBR usado hasta este capítulo o las limitaciones del uso de marcas ARToolkit para posicionamiento. Para ello, se ha decidido usar comportamientos más simples, unidos mediante una capa de selección de comportamientos de alto nivel, minimizar el uso de Tekkotsu en el programa principal, implementar una nueva librería CBR adaptada a las necesidades del presente trabajo o usar localización basada en balizas de colores corrigiendo el posicionamiento mediante filtros de partículas.

Mediante el uso de estas medidas, se ha comprobado que se ha conseguido solucionar o minimizar la mayoría de problemas encontrados, aunque todavía haya problemas o mejoras posibles. Así, por ejemplo, el sistema de localización usado, si bien ofrece mejores resultados que el basado en marcas ARToolkit, pero hay situaciones en las que los robots pierden su posicionamiento, por lo que habría que intentar mejorar el sistema actual. Otro ejemplo de sistema a mejorar sería la librería CBR, pues no es adecuada para su uso con bases formadas por un número elevado de casos. De hecho, el compañero Ignacio Herrero [48], quien también ha hecho uso de esta librería, la ha mejorado en este sentido, aunque no se ha podido estudiar hasta la fecha los cambios realizados para unir ambas versiones.

Por otro lado, para probar el sistema de coordinación basado en comportamientos aprendidos se han usado tres robots AIBO ERS-7 en un entorno similar al de la competición de fútbol Robocup. El entorno estaba libre de obstáculos y la localización, como se ha comentado, esta basada en balizas de colores usando filtros de partículas para corregir el posicionamiento obtenido. La interacción de los robots tiene lugar con los otros robots (reales) y con objetos virtuales (pelota y porterías) que son únicamente visibles mediante las cámaras de los robots, de forma similar a como funciona la RA para los humanos, pero aplicada a los robots. La razón de usar estos objetos virtuales ha sido evitar en la medida de lo posible el deterioro de los robots AIBO disponibles para la realización de las pruebas, robots que dejaron de fabricarse en 2006 y que, en caso de avería, no podrían ser reparados, por lo que se optó por minimizar los posibles problemas en este sentido. Sin embargo, hay que destacar que, con el sistema de aprendizaje propuesto, si se dispusiese de otros robots equipados con cámaras que pudiesen recibir órdenes de movimiento se podría reutilizar toda la estructura propuesta sin mayores cambios, siendo necesario únicamente implementar el interfaz entre el nuevo robot y el ordenador.

Respecto a las pruebas, se ha dividido a los robots disponibles en dos equipos, estando el equipo atacante formado por dos robots y el defensor por uno. Así pues, el objetivo del equipo atacante es sortear al robot defensor (que tratará de bloquearlos) y conseguir marcar un gol, para lo cual es necesaria la coordinación de los robots atacantes.

Para ello, en primer lugar se ha realizado el entrenamiento de los comportamientos aprendidos. Tras el entrenamiento, se ha configurado la capa de selección de comportamientos de forma que permita realizar el comportamiento coordinado buscado. Por último, se ha puesto a los robots a trabajar en modo autónomo, para comprobar el resultado obtenido.

Finalmente, se realizaron las pruebas, que han mostrado que es posible obtener diferentes comportamientos coordinados dependiendo de la activación o desactivación de los comportamientos aprendidos y de como éstos son combinados.

Capítulo 7

Conclusiones finales y resultados de la tesis

7.1. Introducción

En este capítulo se resumirán las conclusiones del trabajo presentado así como las distintas decisiones que ha sido necesario tomar en función de los problemas encontrados. Tras esto, se detallarán las principales aportaciones del presente trabajo, así como los resultados y publicaciones que se han realizado durante el desarrollo de esta tesis. Por último, se presentarán las líneas de trabajo futuras.

7.2. Conclusiones

En el presente trabajo se ha presentado un sistema de coordinación basado en el uso de comportamientos aprendidos sencillos que se combinan para obtener comportamientos más complejos. La combinación de los comportamientos se realiza mediante una capa de alto nivel que los activa en función de la situación a afrontar.

El aprendizaje de los robots se ha realizado por control remoto a través de un joystick mediante aprendizaje LfD usando CBR. De esta forma, se consigue una asociación fácil y directa entre lo que percibe el robot en cada instante y los movimientos que el operador debe realizar en función de la situación concreta. Se ha escogido una implementación basada en aprendizaje en vez de una implementación analítica porque este enfoque permite una mayor facilidad para definir trayectorias *ad hoc*, que no tienen porque encajar con una fórmula analítica, sin necesidad de realizar ningún modelado del entorno. Además, el aprendizaje propuesto ofrece una serie de ventajas, como

el hecho de que la cinemática y dinámica del robot así como los errores de calibración y los errores sistemáticos son implícitamente tomados en cuenta por el operador, el cual los compensa naturalmente durante el entrenamiento. Esto es debido a que, cuando el operador controla el robot, implícitamente absorbe y asimila la estructura y movimientos del robot, gracias a la capacidad de adaptación innata de las personas. De esta forma, el robot aprende, mediante el entrenamiento, como se movería una persona *si tuviese la misma estructura corporal que el robot.*

La técnica de IA seleccionada ha sido CBR, en vez de otras técnicas como las ANN, por la transparencia y la posibilidad de depuración y acceso a la información aprendida que ofrece esta técnica, lo que permite estudiar que está pasando en cada instante dentro del sistema, algo que en el caso de otras técnicas no es fácil de realizar.

Respecto a la coordinación de los robots, se ha optado por usar coordinación implícita frente a explícita porque el planteamiento propuesto encaja perfectamente con este tipo de coordinación. La razón es que tan sólo es necesario que, durante el entrenamiento, el operador tenga en cuenta a los demás robots en el entorno para que el comportamiento entrenado sea un comportamiento coordinado implícito, ya que el robot se movería acorde a sus propias decisiones pero teniendo en cuenta al resto de robots.

Por otro lado, para realizar la coordinación implícita entre los robots se ha permitido la comunicación entre los robots para que se transmitiesen su posicionamiento. Además, se ha visto la necesidad de usar un sistema de localización explícita para los comportamientos coordinados. De hecho, en las pruebas realizadas con comportamientos individuales se ha hecho uso de una localización implícita mediante la información en *crudo* de la imagen (como la posición de la pelota en la imagen o las líneas detectadas) para obtener los comportamientos a realizar. Sin embargo, para comportamientos coordinados las entradas en crudo no resultan adecuadas ya que la definición del caso CBR se hace demasiado compleja, siendo más adecuado el uso de información de más alto nivel como el posicionamiento del robot respecto a algún punto de referencia. Así pues, tras probar varias alternativas de localización para comprobar cual resultaba más adecuada, finalmente se ha optado por una localización basada en balizas visuales corrigiendo el posicionamiento mediante filtros de partículas.

En lo que respecta a la arquitectura de control, se ha usado una estructura híbrida a pesar de que inicialmente se comenzó con un enfoque reactivo. La razón es que en los experimentos se comprobó que un enfoque reactivo parece ser suficiente para comportamientos simples individuales. Además, los entrenamientos no resultan demasiado complejos ni tediosos siempre y

cuando el comportamiento a realizar esté claro y bien definido. Sin embargo, para comportamientos más complejos, como los realizados en el presente trabajo, el entrenamiento que es necesario realizar usando un enfoque reactivo resultó demasiado complicado. La razón es que la cantidad de situaciones que hay que tener en cuenta en los entrenamientos crece enormemente dependiendo del comportamiento a implementar. Por otro lado, también hay que tener en cuenta que si las trayectorias realizadas están asociadas a situaciones muy concretas, lo cual es posible en comportamientos complejos, sería difícil reutilizar el comportamiento en circunstancias más genéricas. Así pues, se ha concluido que, para solucionar estos problemas, lo más oportuno es descomponer los comportamientos complejos en otros más simples que se combinen mediante una arquitectura híbrida. De esta forma se simplifican enormemente los entrenamientos a realizar, ya que los comportamientos están mejor definidos y son menos ambiguos, resultando más fácil entrenar todas las situaciones que deban afrontar. Además, es más sencillo reutilizar estos comportamientos en una mayor cantidad de situaciones, al ser más simples y genéricos que los comportamientos complejos.

Por otro lado, y pasando a un nivel más técnico, para el desarrollo del sistema se optó inicialmente por el enfoque que parecía más sencillo y prometedor, que era el entorno de alto nivel Tekkotsu, el cual permite, entre otras cosas, el control del robot AIBO mediante comandos de movimiento de alto nivel o aplicar segmentación a la imagen. Sin embargo, se ha comprobado que el entorno Tekkotsu tenía una serie de problemas de estabilidad, así como de excesivo consumo de recursos en el ordenador debido al interfaz Java, lo cual se apreciaba especialmente al usar varios robots simultáneamente.

Para evitar en la medida de lo posible estos problemas se ha optado por usar Tekkotsu únicamente como interfaz con el robot, siguiendo la estructura planteada en [97], realizando el programa principal en C++ y aislándolo de dependencias propias del entorno Tekkotsu. De esta forma se podría reutilizar el sistema desarrollado en cualquier robot que disponga de visión y acepte comandos de movimiento frontal y de rotación como es el caso del robot AIBO.

En lo que respecta a la librería CBR usada, se partió de una implementación en Java desarrollada por el grupo KEMLg de la Universidad Politécnica de Cataluña (UPC), usada en el aprendizaje de comportamientos simples de los primeros capítulos. Sin embargo, al comenzar con los comportamientos más complejos se ha comprobado que esta opción no era la más adecuada, pues requeriría ejecutar un servidor CBR por cada comportamiento en ejecución en cada uno de los robots. Así pues, para solucionar este inconveniente y tener un mayor control sobre el CBR, se ha implementado una librería CBR

específica para las necesidades del proyecto, de forma que se pueda integrar fácilmente con la estructura ya comentada.

Por último, el sistema de coordinación basado en comportamientos aprendidos se ha probado usando tres robots AIBO ERS-7 de Sony en un entorno similar al de la competición de fútbol Robocup, aunque de dimensiones más reducidas. El entorno estaba libre de obstáculos y la localización esta basada en visión usando balizas de colores, corrigiendo el posicionamiento mediante filtros de partículas. Para las pruebas, se han formado dos equipos de robots, uno con dos robots (el equipo atacante) y otro con un robot (el equipo defensor). Así pues, el objetivo del equipo atacante es sortear al robot defensor (que tratará de bloquearlos) y conseguir marcar un gol, para lo cual es necesaria la coordinación de los robots atacantes. Cada robot puede interactuar con los otros robots (reales) y con objetos virtuales (pelota y porterías). Estos objetos virtuales son visibles únicamente a través de las cámaras de los robots, de forma similar a como funciona la RA para los humanos, pero aplicada a los robots. La razón de usar estos objetos virtuales ha sido evitar en la medida de lo posible el deterioro de los robots AIBO disponibles para la realización de las pruebas, robots que dejaron de fabricarse en 2.006 y que, en caso de avería, no podrían ser reparados. Finalmente, en las pruebas realizadas se ha comprobado que es posible obtener diferentes comportamientos coordinados dependiendo de la activación o desactivación de los comportamientos aprendidos y de como éstos son combinados.

7.3. Aportaciones de la tesis

En este apartado se resumen, de forma breve, las principales aportaciones realizadas en la presente tesis. Las principales aportaciones son:

- Aplicación del aprendizaje mediante control remoto por LfD usando CBR a robots con extremidades usando visión. Los ejemplos de aplicación del aprendizaje LfD usando CBR en la literatura están orientados a robots con ruedas usando como sensor de entrada láser o sonar. En el presente trabajo este enfoque ha sido aplicado a robots con extremidades (concretamente robots AIBO ERS-7) usando como sensor de entrada la visión.

- Aplicación de comportamientos aprendidos usando CBR a la coordinación de varios robots. El CBR ha sido aplicado a coordinación en otros trabajos, pero en general a alto nivel para establecer la selección, activación y configuración de los comportamientos de bajo nivel. En el presente trabajo el CBR se ha usado para la implementación de los

comportamientos de bajo nivel que permiten, mediante su combinación, un comportamiento emergente coordinado entre varios robots.

- Uso de objetos virtuales y descomposición en comportamientos más simples para simplificar los entrenamientos. Una de los problemas del aprendizaje LfD usado es la dificultad que puede alcanzar la realización de los entrenamientos para rellenar la base de casos adecuadamente en el caso de realizar comportamientos complejos, lo cual se solventa o simplifica en gran medida mediante estas dos soluciones.

- Creación de una nueva librería CBR implementada en C++ y que permite acceso a los casos de una forma rápida y sencilla. La principal ventaja de esta nueva librería es que permite su inclusión dentro de código C++ sin necesidad de desperdiciar recursos en conexiones cliente-servidor con librerías o aplicaciones desarrolladas en Java, lenguaje que usan la mayoría de aplicaciones CBR. Además, esta librería permite realizar la depuración de las bases de casos gráficamente así como mostrar los casos de la base, aunque con limitaciones.

7.4. Resultados y publicaciones

En este apartado se resumen los resultados y publicaciones que han surgido durante la realización de esta tesis, separando entre las publicaciones relacionadas directamente y las relacionadas parcialmente con la tesis.

Así pues en primer lugar se listarán las publicaciones relacionadas directamente, cuyo contenido está enfocado a la aplicación de aprendizaje LfD usando CBR para la creación de comportamientos sobre los robots AIBO. Estas publicaciones son:

- J. M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato and F. Sandoval. Pure reactive behavior learning using case based reasoning for a vision based 4-legged robot. *Robotics and Autonomous Systems*, 57(6-7):688–699, 2009.

  DOI:10.1016/j.robot.2008.11.003

- J. M. Peula, J. Cebolla, C. Urdiales and F. Sandoval. Explicit coordinated localization using common visual objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4889–4894. IEEE, 2010.

  DOI: 10.1109/ROBOT.2010.5509398

- J. M. Peula, C. Urdiales, I. Herrero and F. Sandoval. Implicit robot coordination using case-based reasoning behaviors. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5929–5934. IEEE, 2013.

  DOI: 10.1109/IROS.2013.6697216

- I. Herrero, C. Urdiales, J. M. Peula, I. Sanchez-Tato and F. Sandoval. A guided learning strategy for vision based navigation of 4-legged robots. *AI Commun.*, 19(2):127 – 136, 2006.

  ISSN: 0921-7126

- I. Herrero, C. Urdiales, J. M. Peula and F. Sandoval. A bottom-up robot architecture based on learnt behaviors driven design. In *Advances in Computational Intelligence*, pages 159–170. Springer, 2015.

  DOI: 10.1007/978-3-319-19258-1_14

Por otro lado se encuentran las publicaciones relacionadas parcialmente con la tesis. En estas publicaciones se hace uso del CBR para aprendizaje y aplicación a la navegación, aunque no expresamente sobre robots AIBO. Estas publicaciones son:

- J. M. Peula, C. Urdiales, I. Herrero, M. Fernandez-Carmona and F. Sandoval. Case-based reasoning emulation of persons for wheelchair navigation. *Artificial Intelligence in Medicine*, 56(2):109 – 121, 2012.

  DOI: 10.1016/j.artmed.2012.08.007

- C. Urdiales, J. M. Peula, M. Fernández-Carmona and F. Sandoval. Learning-based adaptation for personalized mobility assistance. In *Case-Based Reasoning Research and Development*, pages 329–342. Springer, 2013.

  DOI: 10.1007/978-3-642-39056-2_24

- C. Urdiales, J. M. Peula and M. Fernández-Carmona. *Collaborative Assistive Robot for Mobility Enhancement (CARMEN)*. Springer, 2012.

  DOI: 10.1007/978-3-642-24902-0

- C. Urdiales, J. Peula, M. Fernandez-Carmona, R. Annicchiaricco, F. Sandoval and C. Caltagirone. Adaptive collaborative assistance for wheelchair driving via cbr learning. In *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*, pages 731–736. IEEE, 2009.

  DOI: 10.1109/ICORR.2009.5209575

- C. Urdiales, J. M. Peula, C. Barrué, U. Cortés, F. Sandoval, C. Caltagirone and R. Annichiaricco. An adaptive scheme for wheelchair navigation collaborative control. *Asociation for the Advancement of Artificial Intelligence (AAAI'08)*, 2008.

7.5. Líneas futuras

Por último, se mencionarán las líneas futuras de trabajo para continuar la presente tesis. Entre las distintas líneas de trabajo que se pueden considerar estarían:

- Aplicación del sistema propuesto a otros robots basados en visión para sustituir a los AIBOs. Esta alternativa sería interesante para poder prescindir de los robots AIBO y usar robots actuales que no presenten los problemas de soporte de los AIBO. Además, así se podrían comprobar los problemas que surgirían al aplicar este enfoque a otras plataformas.

- Implementar la capa de selección de comportamientos mediante CBR, y no de forma analítica. Como se ha comentado, el CBR ha sido aplicado anteriormente a la selección y configuración de comportamientos. Así pues, una línea futura sería la implementación de la capa de selección de comportamientos usando CBR, de forma que el sistema pueda resultar más flexible y adaptable.

- Mejorar la localización basada en balizas de colores y filtros de partículas añadiendo información de las porterías y de las líneas del campo. Actualmente la localización utiliza únicamente las balizas de colores, las cuales resultan insuficientes en ocasiones siendo necesario que el robot se detenga y haga una panorámica del entorno para poder estimar su posición. Este problema podría reducirse añadiendo más información proveniente de las porterías y líneas de campo.

- Mezclar comportamientos analíticos con comportamientos aprendidos. Si bien el enfoque presentado se ha orientado al uso principalmente de comportamientos aprendidos para comprobar su viabilidad, pero no todas las situaciones requieren del uso de comportamientos aprendidos. Así pues, en situaciones más reales lo ideal sería mezclar comportamientos analíticos, para las situaciones más generales, combinados con comportamientos aprendidos, para aquellas situaciones en las que el enfoque analítico no resulte adecuado o presente problemas.

- Mejorar la librería CBR. La librería desarrollada actualmente dispone de una funcionalidad limitada y ajustada a las necesidades del presente trabajo. Así pues, sería interesante mejorarla, por ejemplo, con las aportaciones del compañero Ignacio Herrero Reder, permitiendo su aplicación a bases con un gran número de casos. Igualmente sería interesante ampliar su funcionalidad añadiendo nuevos algoritmos de búsqueda o la posibilidad de usar estructuras de memoria jerárquicas. Además, el depurador gráfico que dispone la librería también puede ser mejorado, añadiendo por ejemplo la posibilidad de representar más objetos, para lo cual habría que buscar una representación gráfica adecuada.

- Aplicación del sistema a más robots. Esto actualmente no es posible ya que no se dispone de más robots, por lo que la única opción sería usar el simulador. Sin embargo, sería interesante pues plantearía nuevos retos y dilemas, como decidir cuanta información y que robots tener en cuenta a la hora de actuar.

Bibliografía

[1] A. Aamodt. Explanation-driven case-based reasoning. *Topics in Case-Based Reasoning*, pages 274 – 288, 1994.

[2] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodologies variations, and systems approaches. *AI Communications*, 7(1):39 – 59, 1994.

[3] S. Abdallah, K. Shaalan, and M. Shoaib. Integrating rule-based system with classification for arabic named entity recognition. In *Computational Linguistics and Intelligent Text Processing*, pages 311–322. Springer, 2012.

[4] R. Abiyev, D. Ibrahim, and B. Erin. Navigation of mobile robots in the presence of obstacles. *Advances in Engineering Software*, 41(10):1179–1186, 2010.

[5] M. Abramson and R. Mittu. Learning and coordination: An overview. In *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pages 343–350. IEEE, 2011.

[6] Y. Aloimonos. Purposive and qualitative active vision. *Image Understanding Workshop*, 828:816 – 828, 1990.

[7] I. Amundson and X. D. Koutsoukos. A survey on localization for mobile wireless sensor networks. In *Mobile Entity Localization and Tracking in GPS-less Environnments*, pages 235–254. Springer, 2009.

[8] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[9] R. C. Arkin. *Behaviour based robotics*. Cambridge, 1998.

[10] B. Auslander, T. Apker, and D. W. Aha. Case-based parameter selection for plans: Coordinating autonomous vehicle teams. In *Case-Based Reasoning Research and Development*, pages 32–47. Springer, 2014.

[11] G. Baldassarre, A. V. Trianni, A. M. Bonani, A. F. Mondada, A. M. Dorigo, and A. S. Nolfi. Self-organised coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 37(1):224 – 239, 2007.

[12] A. Barrett, G. Rabideau, T. Estlin, and S. Chien. Coordinated continual planning methods for cooperating rovers. *Aerospace and Electronic Systems Magazine, IEEE*, 22(2):27–33, 2007.

[13] I. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31, 2000.

[14] S. Begum, M. Ahmed, P. Funk, N. Xiong, and M. Folke. Case-based reasoning systems in the health sciences: A survey of recent trends and developments. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(4):421–434, 2011.

[15] R. Berger and G. Lämmel. Exploiting past experience–case-based decision support for soccer agents. In *KI 2007: Advances in Artificial Intelligence*, pages 440–443. Springer, 2007.

[16] I. Bichindaritz. Case-based reasoning in the health sciences: Why it matters for the health sciences and for cbr. In K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *Advances in Case-Based Reasoning*, volume 5 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin / Heidelberg, Berlin, Germany, 2008.

[17] I. Bichindaritz and C. Marling. Case-based reasoning in the health sciences: What's next? *Artificial Intelligence in Medicine, Special Issue on Case-based Reasoning in the Health Sciences*, 36(2):127 – 135, 2006.

[18] D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Architecture for hybrid robotic behavior. In *Hybrid Artificial Intelligence Systems*, pages 145–156. Springer, 2009.

[19] J. Biswas and M. Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4379–4384. IEEE, 2010.

[20] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296, 2008.

[21] O. Booij, B. Terwijn, Z. Zivkovic, and B. Kröse. Navigation using an appearance based topological map. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3927–3932. IEEE, 2007.

[22] J. Borenstein, H. Everett, L. Feng, et al. Where am i? sensors and methods for mobile robot positioning. Technical report, University of Michigan, 1996.

[23] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, 1991.

[24] L. K. Branting and D. W. Aha. Stratified case-based reasoning: Reusing hierarchical problem solving episodes. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence*, pages 20 – 25, Montreal, Canada, August 1995.

[25] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[26] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 2061–2066. IEEE, 2000.

[27] F. Calabrese and G. Indiveri. An omni-vision triangulation-like approach to mobile robot localization. In *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation*, pages 604 – 609, June 2005.

[28] S. H. Chen, A. J. Jakeman, and J. P. Norton. Artificial intelligence techniques: an introduction to their use for modelling environmental systems. *Mathematics and Computers in Simulation*, 78(2):379–400, 2008.

[29] H. K. Cigizoglu and M. Alp. Generalized regression neural network in modelling river sediment yield. *Advances in Engineering Software*, 37(2):63–68, 2006.

[30] A. Crowe, C. McClean, and M. Cresser. An application of genetic algorithms to the robust estimation of soil organic and mineral fraction densities. *Environmental Modelling & Software*, 21(10):1503–1507, 2006.

[31] G. Davis, N. Wiratunga, B. Taylor, and S. Craw. Matching smarthouse technology to needs of the elderly and disabled. pages 29 – 38, Trondheim, Norway, 2003.

[32] G. de Souza and A. Kak. Vision for mobile robot navigation: A survey. *Trans. on Patt. Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[33] M. Dekker, P. Hameete, M. Hegemans, S. Leysen, J. Oever, J. Smits, and K. Hindriks. Hactarv2: An agent team strategy based on implicit coordination. In L. Dennis, O. Boissier, and R. Bordini, editors, *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin Heidelberg, 2012.

[34] M. B. Dias and A. A. Stentz. A free market architecture for distributed control of a multirobot system. volume 11, pages 115 – 122, Venice (Italy), 2000.

[35] M. B. Dias and A. A. Stentz. Opportunistic optimization for market-based multirobot control. volume 3, pages 2714 – 2720, 2002.

[36] M. Dorigo, A. E. Bonebeau, and A. G. Theraulaz. Ant algorithms and stigmergy. volume 16, pages 851 – 871, 2000.

[37] S. Enderle, M. Ritter, D. Fox, S. Sablatnög, G. Kraetzschmar, and G. Palm. Vision-based localization in robocup environments. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 291–296. Springer, 2001.

[38] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, 1997.

[39] S. Fox and D. B. Leake. Combining case-based planning and introspective reasoning. Carbondale, IL, April 1995.

[40] S. Garnier, A. C. Jost, A. R. Jeanson, A. J. Gautrais, A. M. Adadpour, A. G. Caprari, and A. G. Theraulaz. Aggregation Behaviour as a Source of Collective Decision in a Group of Cockroach-Like-Robots. volume 3630, pages 169 – 178, 2005.

[41] A. Georgiev and P. Allen. Localization methods for a mobile robot in urban environments. *IEEE Transactions on Robotics*, 20(5):208– 216, Oct. 2004.

[42] D. Goldberg, V. Cicirello, M. Dias, R. Simmons, S. Smith, T. Smith, and A. Stentz. A distributed layered architecture for mobile robot coordination: Application to space exploration. *Proceedings of the 3rd Int. NASA Workshop on Planning and Scheduling for Space*, 2002.

[43] K. Z. Haigh and M. Veloso. Route planning by analogy. In *ICCBR '95: Proceedings of the First International Conference on Case-Based Reasoning Research and Development*, pages 169 – 180, 1995.

[44] S. T. Hansen, M. Svenstrup, H. J. Andersen, and T. Bak. Adaptive human aware navigation based on motion pattern analysis. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 927–932. IEEE, 2009.

[45] K. Hara, M. Inoue, S. Maeyama, and A. Gofuku. Navigation using one laser source for mobile robot with optical sensor array installed in pan and tilt mechanism. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'08)*, pages 257 – 262, Xian, Jul. 2008.

[46] F. Hayes-Roth. Rule-based systems. *Communications of the ACM*, 28(9):921–932, 1985.

[47] I. Herrero, C. Urdiales, J. M. Peula, I. Sanchez-Tato, and F. Sandoval. A guided learning strategy for vision based navigation of 4-legged robots. *AI Commun.*, 19(2):127 – 136, 2006.

[48] I. Herrero, C. Urdiales, J. M. Peula, and F. Sandoval. A bottom-up robot architecture based on learnt behaviors driven design. In *Advances in Computational Intelligence*, pages 159–170. Springer, 2015.

[49] J. Hodál and J. Dvořák. Using case-based reasoning for mobile robot path planning. *Engineering Mechanics*, 15(3):181–191, 2008.

[50] A. Holt, I. Bichindaritz, R. Schmidt, and P. Perner. Medical applications in case-based reasoning. *Knowledge Engineering Review*, 20(2005):289 – 292, 2005.

[51] H. Hu and M. Brady. Parallel processing architecture for sensor based control of an intelligent mobile robot. *Robotics and Autonomous Systems*, 17:235 – 257, 1996.

[52] L. Iocchi, D. Nardi, and M. Salerno. Reactivity and deliberation: a survey on multi-robot systems. In *Balancing reactivity and social deliberation in multi-agent systems*, pages 9–32. Springer, 2001.

[53] M. Jamzad, B. Sadjad, V. Mirrokni, M. Kazemi, H. Chitsaz, A. Heydarnoori, M. Hajiaghai, and E. Chiniforooshan. A fast vision system for middle size robots in robocup. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*, pages 71–80. Springer Berlin Heidelberg, 2002.

[54] J. Jianqiang, C. Weidong, and X. Yugeng. A rule-driven autonomous robotic system operating in a time-varying environment. In *RoboCup 2003: Robot Soccer World Cup VII*, pages 487–494. Springer, 2004.

[55] C. Jones, D. Shell, M. J. Mataric, and B. Gerkey. Principled approaches to the design of multi-robot systems. In *Proc. of the Workshop on Networked Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, 2004.

[56] H. R. V. Joze, S. A. Zonouz, S. Rahbar, M. Valipour, and A. Fathi. Impossibles aibo four-legged team description paper robocup 2006, 2006.

[57] R. Kala, A. Shukla, R. Tiwari, S. Rungta, and R. Janghel. Mobile robot navigation control in moving obstacle environment using genetic algorithm, artificial neural networks and a* algorithm. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 4, pages 705–713. IEEE, 2009.

[58] E. Kalapanidas and N. Avouris. Short-term air quality prediction using a case-based classifier. *Environmental Modelling & Software*, 16(3):263–272, 2001.

[59] E. Kalapanidas and N. Avouris. Feature selection for air quality forecasting: a genetic algorithm approach. *AI Communications*, 16(4):235–251, 2003.

[60] D. S. Kaster, C. B. Medeiros, and H. V. Rocha. Supporting modeling and problem solving from precedent experiences: the role of workflows and case-based reasoning. *Environmental Modelling & Software*, 20(6):689–704, 2005.

[61] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 396–404. Springer New York, 1990.

[62] J. L. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, 1992.

[63] K. Konolige, C. Ortiz, R. Vincent, A. Agno, B. Limketkai, M. Lewis, L. Briesemeister, D. Fox, J. Ko, B. Stewart, and L. Guibas. Centibots: Large scale robot teams. *Proocedings of the 2003 NRL Workshop on Multi-Robot Systems*, 2, 2003.

[64] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562, 2014.

[65] M. Kruusmaa. Global navigation in dynamic environments using case-based reasoning. *Autonomous Robots*, 14(1):71–91, 2003.

[66] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. In D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*, pages 18–33. Springer Berlin Heidelberg, 2005.

[67] T. Laue and T. Röfer. Particle filter-based state estimation in a competitive and uncertain environment. In *Proceedings of the 6th International Workshop on Embedded Systems, Vaasa, Finland*, 2007.

[68] J. B. Lee, M. Likhachev, and R. C. Arkin. Selection of behavioral parameters: Integration of discontinuous switching via case-based reasoning with continuous adaptation via learning momentum. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1275 – 1281, 2002.

[69] M. Likhachev and R. Arkin. Spatio-temporal case-based reasoning for behavioral selection. In W. Kwon and B. H. Lee, editors, *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1627–1634, New Jersey, USA, 2001. IEEE Service Center.

[70] M. Likhachev, A. M. Kaess, A. Z. Kira, and A. R. C. Arkin. Spatio-temporal case-based reasoning for efficient reactive robot navigation.

Mobile Robot Laboratory, College of Computing, Georgia Institute of Technology, 2005.

[71] M. Likhachev, M. Kaess, and R. C. Arkin. Learning behavioral parameterization using spatio-temporal case-based reasoning. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1282–1289. IEEE, 2002.

[72] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.

[73] H. Liu and H. Iba. A layered control architecture for humanoid robot. In *Proc. of Int. Conf. on Autonomous Robots and Agents*, pages 434 – 439, 2004.

[74] Z. Liu, N. Jiang, , and L. Zhang. Self-localization of indoor mobile robots based on artificial landmarks and binocular stereo vision. In *Proceedings of the 2009 International Workshop on Information Security and Application (IWISA09)*, pages 226–231, Nov. 2009.

[75] R. Lopez De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, et al. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3):215–240, 2005.

[76] B. Mahaman, H. Passam, A. Sideridis, and C. Yialouris. Diares-ipm: a diagnostic advisory rule-based expert system for integrated pest management in solanaceous crop systems. *Agricultural Systems*, 76(3):1119–1135, 2003.

[77] W. Manikas, K. Ashenayi, and R. Wainwright. Genetic algorithms for autonomous robot navigation. *Instrumentation & Measurement Magazine, IEEE*, 10(6):26–31, 2007.

[78] F. Mantz, P. Jonker, and W. Caarls. Thinking in behaviors, not in tasks. *RoboCup International Symposium*, 2005.

[79] F. Martín, V. Matellán, J. M. Cañas, and P. Barrera. Visual based localization for a legged robot. In *RoboCup 2005: Robot Soccer World Cup IX*, pages 708–715. Springer, 2006.

[80] A. Martinoli, A. K. Easton, and A. W. Agassounon. Modeling swarm robotic systems: A case study in collaborative distributed manipulation. *International Journal of Robotics Research, Special Issue on Experimental Robotics*, 23(4):415 – 436, 2004.

[81] M. J. Mataric. Interaction and intelligent behavior. Technical report, MIT Artificial Intelligence Laboratory, 1994.

[82] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2992–2997. IEEE, 2011.

[83] J. Meyer, R. Adolph, D. Stephan, A. Daniel, M. Seekamp, V. Weinert, and U. Visser. Decision-making and tactical behavior with potential fields. In G. Kaminka, P. Lima, and R. Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI*, volume 2752 of *Lecture Notes in Computer Science*, pages 304–311. Springer Berlin Heidelberg.

[84] J. C. Mogul. Emergent (mis)behavior vs. complex software systems. *Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems*, 40(4):293 – 304, 2006.

[85] F. Mondada, A. L. Gambardella, A. D. Floreano, and A. M. Dorigo. SWARM-BOTS: Physical Interactions in Collective Robotics. *Robotics and Automation Magazine*, 12(2):21 – 28, 2005.

[86] A. S. Montero, H. Sekkati, J. Lang, R. Laganière, and J. James. Framework for natural landmark-based robot localization. In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*, pages 131–138. IEEE, 2012.

[87] R. Murphy, J. G. Blitch, and J. L. Casper. Robocup/aaai urban search and rescue events: Reality and competition. *AI Magazine, In press.*, 2001.

[88] D. Nakhaeinia, S. Tang, S. M. Noor, and O. Motlagh. A review of control architectures for autonomous navigation of mobile robots. *International Journal of the Physical Sciences*, 6(2):169–174, 2011.

[89] D. Navarro, G. Benet, and M. Martinez. Line based robot localization using a rotary sonar. In *Proceedings of the 12th IEEE Conference on Emerging Technologies and Factory Automation ETFA'07*, pages 25 – 28, Patras (Greece), Sep. 2007. AAAI Press.

[90] M. Nilsson and M. Sollenborn. Advancements and trends in medical casebased reasoning: An overview of systems and system development. pages 178 – 183, 2004.

[91] S. Nissen. Implementation of a fast artificial neural network library (fann). *Report, Department of Computer Science University of Copenhagen (DIKU)*, 31, 2003.

[92] S. Nolfi. Behaviour as a Complex Adaptive System: On the Role of Self-Organization in the Development of Individual and Collective Behaviour. *Complexus*, 2:195 – 203, 2005.

[93] G. M. O'Hare and N. Jennings. *Foundations of distributed artificial intelligence*, volume 9. John Wiley & Sons, 1996.

[94] G. Onkal-Engin, I. Demir, and S. N. Engin. Determination of the relationship between sewage odour and bod by neural networks. *Environmental Modelling & Software*, 20(7):843–850, 2005.

[95] L. E. Parker. The effect of heterogeneity in teams of 100+ mobile robots. *Proocedings of the 2003 NRL Workshop on Multi-Robot Systems*, 2, 2003.

[96] G. J. Pelletier, S. C. Chapra, and H. Tao. Qual2kw–a framework for modeling water quality in streams and rivers using a genetic algorithm for calibration. *Environmental Modelling & Software*, 21(3):419–425, 2006.

[97] E. J. Perez. Arquitectura de navegacion distribuida para agentes roboticos. Master's thesis, University of Malaga (Spain), 2006.

[98] E. J. Pérez Rodríguez. Distributed intelligent navigation architecture for robots. *AI Communications*, 21(2-3):215–218, 2008.

[99] P. Perner, T. Gunther, and H. Perner. Airborne fungi identification by case-based reasoning. In *Workshop on CBR in the Health Sciences (ICCBR'03)*, pages 73 – 79, 2003.

[100] J. M. Peula, J. Cebolla, C. Urdiales, and F. Sandoval. Explicit coordinated localization using common visual objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4889–4894. IEEE, 2010.

[101] J. M. Peula, C. Urdiales, I. Herrero, M. Fernandez-Carmona, and F. Sandoval. Case-based reasoning emulation of persons for wheelchair navigation. *Artificial Intelligence in Medicine*, 56(2):109 – 121, 2012.

[102] J. M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato, and F. Sandoval. Pure reactive behavior learning using case based reasoning for a vision based 4-legged robot. *Robotics and Autonomous Systems*, 57(6-7):688–699, 2009.

[103] J. M. Peula, C. Urdiales, I. Herrero, and F. Sandoval. Implicit robot coordination using case-based reasoning behaviors. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5929–5934. IEEE, 2013.

[104] A. M. G. Pinto, A. P. Moreira, and P. G. Costa. Indoor localization system based on artificial landmarks and monocular vision. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 10(4):609–620, 2012.

[105] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, volume 2, pages 802–807, 1993.

[106] A. Ram, R. C. Arkin, K. Moorman, and R. J. Clark. Case-based reactive navigation: a method for on-line selection and adaptation of reactive robotic control parameters. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 27(3):376–394, 1997.

[107] R. Rocha, A. J. Dias, and A. A. Carvalho. Cooperative multi-robot systems:: A study of vision-based 3-D mapping using information theory. *Robotics and Autonomous Systems*, 53(3-4):282 – 311, 2005.

[108] T. Röfer and M. Jüngel. Vision-based fast and reactive monte-carlo localization. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 856 – 861, Sept. 2003.

[109] T. Röfer and M. Jüngel. Fast and robust edge-based localization in the sony four-legged robot league. *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, pages 262–273, 2004.

[110] T. Röfer, T. Laue, and D. Thomas. Particle-filter-based self-localization using landmarks and directed lines. In *RoboCup 2005: Robot Soccer World Cup IX*, pages 608–615. Springer, 2006.

[111] C. Rohrig and F. Kunemund. Mobile robot localization using wlan signal strengths. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on*, pages 704–709. IEEE, 2007.

[112] R. Ros, R. L. De Màntaras, C. Sierra, and J. L. Arcos. A cbr system for autonomous robot navigation. In *CCIA*, pages 299–306, 2005.

[113] R. Ros and M. Veloso. Executing multi-robot cases through a single coordinator. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 215. ACM, 2007.

[114] R. Ros, M. Veloso, R. L. De Màntaras, C. Sierra, and J. L. Arcos. Retrieving and reusing game plays for robot soccer. In *Advances in Case-Based Reasoning*, pages 47–61. Springer, 2006.

[115] M. A. Ruiz and J. R. Uresti. Team agent behavior architecture in robot soccer. In *Robotic Symposium, 2008. LARS'08. IEEE Latin American*, pages 20–25. IEEE, 2008.

[116] J. Ruiz-del Solar, P. Guerrero, P. Vallejos, P. Loncomilla, R. Palma-Amestoy, P. Astudillo, R. Dodds, J. Testart, D. Monasterio, and A. Marinkovic. Uchile1 strikes back, 2006 team description paper. In *Robotics Symposium, 2006. LARS'06. IEEE 3rd Latin American*, pages 200–207. IEEE, 2006.

[117] A. Rusdinar, J. Kim, J. Lee, and S. Kim. Implementation of real-time positioning system using extended kalman filter and artificial landmark on ceiling. *Journal of mechanical science and technology*, 26(3):949–958, 2012.

[118] J. San Pedro, F. Burstein, and A. Sharp. A case-based fuzzy multicriteria decision support model for tropical cyclone forecasting. *European Journal of Operational Research*, 160(2):308–324, 2005.

[119] Y. Sasaki, S. Kagami, and H. Mizoguchi. Multiple sound source mapping for a mobile robot by self-motion triangulation. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 380–385. IEEE, 2006.

[120] R. Schank. *Dynamic memory.* New York: Cambridge University Press, 1982.

[121] R. Schank and R. Abelson. *Scripts, plans, goals and understanding.* Lawrence Erlbaum, 1977.

[122] T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *Robotics and Automation, IEEE Transactions on*, 18(5):670–684, 2002.

[123] M. Schumacher. Multi-agent systems. *Objective Coordination in Multi-Agent System Engineering: Design and Implementation*, pages 9–32, 2001.

[124] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, and H.-M. Tai. Autonomous local path planning for a mobile robot using a genetic algorithm. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1338–1345. IEEE, 2004.

[125] M. Sharma, M. P. Holmes, J. C. Santamaría, A. Irani, C. L. Isbell Jr, and A. Ram. Transfer learning in real-time strategy games using hybrid cbr/rl. In *IJCAI*, volume 7, pages 1041–1046, 2007.

[126] S. Simhon and G. Dudek. A global topological map formed by local metric maps. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 3, pages 1708–1714. IEEE, 1998.

[127] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First results in the coordination of heterogeneous robots for large- scale assembly. *Proceedings of the Int. Symposium on Experimental Robotics (ISER)*, 2000.

[128] S. Sousa, F. Martins, M. Alvim-Ferraz, and M. C. Pereira. Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. *Environmental Modelling & Software*, 22(1):97–103, 2007.

[129] M. Sridharan, G. Kuhlmann, and P. Stone. Practical vision-based monte carlo localization on a legged robot. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3366–3371, 2005.

[130] M. U. Suleman and M. M. Awais. Learning from demonstration in robots: Experimental comparison of neural architectures. *Robotics and Computer-Integrated Manufacturing*, 27(4):794–801, 2011.

[131] M. Svenstrup, S. Tranberg, H. J. Andersen, and T. Bak. Pose estimation and adaptive robot behaviour for human-robot interaction. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3571–3576. IEEE, 2009.

[132] E. Szelke and G. Markus. A learning reactive scheduler using cbr/l. *Computers in industry*, 33(1):31–46, 1997.

[133] J.-H. Tzou and K. L. Su. High-speed laser localization for a restaurant service mobile robot. *Artificial Life and Robotics*, 14(2):252 – 256, Nov. 2009.

[134] I. Ulrich and J. Borenstein. Vfh*: local obstacle avoidance with look-ahead verification. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2505–2511 vol.3, 2000.

[135] C. Urdiales. *Collaborative Assistive Robot for Mobility Enhancement (CARMEN)*. Springer, 2012.

[136] C. Urdiales, E. Perez, J.Vazquez-Salceda, M.Sanchez-Marre, and F. Sandoval. A purely reactive navigation scheme for dynamic environments using case-based reasoning. *Autonomous Robots*, 39(5):67–78, 2006.

[137] C. Urdiales, E. J. Perez, and F. Sandoval. A time stamp control strategy for cbr based reactive navigation in dynamic environments with priorities. In *IAT '04: Proceedings of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference*, pages 58 – 64, Washington, DC, USA, 2004. IEEE Computer Society.

[138] C. Urdiales, J. M. Peula, M. Fdez-Carmona, C. Barrué, E. J. Pérez, I. Sánchez-Tato, J. Del Toro, F. Galluppi, U. Cortés, R. Annichiaricco, et al. A new multi-criteria optimization strategy for shared control in wheelchair assisted navigation. *Autonomous Robots*, 30(2):179–197, 2011.

[139] C. Urdiales, J. Vázquez-Salceda, E. Perez, M. Sànchez-Marrè, and F. Sandoval. A cbr based pure reactive layer for autonomous robot

navigation. In *Proceedings of the 7th IASTED International Conference on Artificial Intelligence and Soft Computing*, pages 99–104, 2003.

[140] H. Utz, A. Neubeck, G. Mayer, and G. Kraetzschmar. Improving vision-based self-localization. In *RoboCup 2002: Robot Soccer World Cup VI*, pages 25–40. Springer, 2003.

[141] W. Van der Hoek and M. Wooldridge. Multi-agent systems. *Foundations of Artificial Intelligence*, 3:887–928, 2008.

[142] M. Van Lent and J. Laird. Learning hierarchical performance knowledge by observation. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 229–238, June 27-30 1999.

[143] R. Vaugham, A. N. Sumpter, A. J. Henderson, A. A. Frost, and A. S. Cameron. Experiments in automatic flock control. *Robotics and Autonomous Systems*, 31(1):109 – 117, 2000.

[144] R. Vázquez-Martin, E. Pérez, C. Urdiales, J. del Toro, and F. Sandoval. Hybrid navigation guidance for intelligent mobiles. *IEEE ITS Society Newsletter*, 4(8):24–30, 2006.

[145] A. Vellido, E. Martí, J. Comas, I. Rodríguez-Roda, and F. Sabater. Exploring the ecological status of human altered streams through generative topographic mapping. *Environmental Modelling & Software*, 22(7):1053–1065, 2007.

[146] M. Veloso, S. Lenser, D. Vail, M. Roth, A. Stroupe, and S. Chernova. Cmpack-02: Cmu's legged robot soccer team. *Kaminka; Lima; and Rojas., eds., RoboCup*, 2002.

[147] N. Vuković and Z. Miljković. New hybrid control architecture for intelligent mobile robot navigation in a manufacturing environment. *FME Transactions*, 37(1):9–18, 2009.

[148] I. Watson. An introduction to case-based reasoning. In *Progress in Case-Based Reasoning*, pages 1–16. Springer, 1995.

[149] T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel. Cs freiburg: coordinating robots for successful soccer playing. *Robotics and Automation, IEEE Transactions on*, 18(5):685–699, 2002.

[150] A. Wendel, A. Irschara, and H. Bischof. Natural landmark-based monocular localization for mavs. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5792–5799. IEEE, 2011.

[151] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image-retrieval system with monte carlo localization. *IEEE Transactions on Robotics*, 21(2):208– 216, 2005.

[152] Y. Yan and T. Zhenmin. Control architecture for autonomous multi-robot system: Survey and analysis. In *Intelligent Computation Technology and Automation, 2009. ICICTA'09. Second International Conference on*, volume 4, pages 376–379. IEEE, 2009.

[153] Z. Yan, N. Jouandeau, and A. A. Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10:399, 2013.

[154] F. Zetian, X. Feng, Z. Yun, and Z. XiaoShuan. Pig-vet: a web-based expert system for pig disease diagnosis. *Expert Systems with Applications*, 29(1):93–103, 2005.

[155] Q. Zhang and S. J. Stanley. Forecasting raw-water quality parameters for the north saskatchewan river by neural network modeling. *Water research*, 31(9):2340–2350, 1997.

[156] A. Zhu and S. X. Yang. A survey on intelligent interaction and cooperative control of multi-robot systems. In *Control and Automation (ICCA), IEEE International Conference on*, pages 1812–1817, 2010.