Arquitectura de Comportamientos Reactivos para Agentes Robóticos basada en CBR

Tesis Doctoral

UNIVERSIDAD
DE MÁLAGA

Ignacio Herrero Reder

Escuela Técnica Superior de Ingeniería de Telecomunicación

Málaga, 2015

Departamento de Tecnología Electrónica
E.T.S.I. Telecomunicación
Universidad de Málaga

Tesis Doctoral

Arquitectura de Comportamientos Reactivos para Agentes Robóticos basada en CBR

Autor:
Ignacio Herrero Reder
Ingeniero de Telecomunicación

Directora:
Cristina Urdiales García
Dra. Ingeniera de Telecomunicación

Dña. Cristina Urdiales García, Profesora Doctora del Departamento de Tecnología Electrónica de la Universidad de Málaga

Certifica:

Que D. Ignacio Herrero Reder, 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:

Arquitectura de Comportamientos Reactivos para Agentes Robóticos basada en CBR

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 Noviembre de 2015

Fdo: Dra. D^{a}. Cristina Urdiales García

A las niñas de mis ojos, Rosi y Sofía. Si he acabado esta Tesis ha sido por VOSOTRAS

Agradecimientos

Esta Tesis no hubiera sido posible sin la intervención y ayuda de gran cantidad de personas y entidades que me han dado su apoyo, tanto moral como material, en el proceso de realización del presente trabajo.

Mi agradecimiento a los doctores Francisco Sandoval y Antonio Díaz Estrella que confiaron en mí para darme su aval en la petición de una beca FPU del MEC, beca que me permitió adentrarme en el maravilloso mundo de la docencia y la investigación en el ámbito de la Universidad de Málaga. Espero no haber decepcionado su confianza.

Al grupo de investigación ISIS (Ingeniería de Sistemas Integrados) por su soporte económico y material a través de los diferentes proyectos en los que he participado, soporte indispensable para la realización de toda mi investigación.

A mis compañeros del Departamento de Tecnología Electrónica, los "Habituales del Comedor", con sus charlas de desayuno en las que hablar de lo divino y de lo humano, de la política universitaria y de las aficiones personales, de la mejora docente e investigadora, y de los últimos chismes del pasillo.

A mis compañeros de asignaturas: Microcontroladores, Sistemas Empotrados, Microbótica, las asignaturas del Máster SEEI, etc. Por hacer de mi trabajo una tarea más agradable y demostrarme cada día su compañerismo y amistad.

A los sufridores de las Tesis a extinguir: Peula, Carmen, Jose Manuel, Martín...mucho ánimo que ya queda menos!!!

A mis padres, por haberme proporcionado los medios para desarrollar una carrera que me ha llenado plenamente, como ingeniero, como docente, y como investigador. También por su cariño y apoyo en todas las fases de mi vida.

A mi directora, Cristina, mi "jefa", compañera de asignaturas, y amiga. Gracias por tu apoyo y tiempo dedicado, muchas veces sacado de no se sabe donde. Espero seguir disfrutando de tu compañía en futuras aventuras docentes e investigadoras.

Y finalmente a Rosi y Sofía, por los sacrificios realizados para permitirme acabar esta Tesis, y porque lo son TODO en mi vida. Espero poder dedicaros, a partir de ahora, todo el tiempo que os merecéis.

Resumen

Aunque los mecanismos últimos que definen la Inteligencia permanecen aún en la oscuridad, se conoce ya el papel fundamental que juega el aprendizaje en la misma, así como su expresión a través de la capacidad de identificar situaciones y predecir acciones o respuestas a partir de la información adquirida precisamente a través de dicho aprendizaje.

Efectivamente, tanto en su vertiente de aprendizaje por observación o tutelado, como en la del aprendizaje a través de la experiencia, el aprendizaje supone el medio definitivo de adquisición de conocimientos del ser humano. Estos conocimientos le permiten afrontar situaciones ya vividas ante las cuales repetir la misma respuesta exitosa ya probada, o incluso situaciones desconocidas que se puedan asimilar, a mayor o menor nivel conceptual, con la información conocida, para responder de forma similar, e incorporar estos nuevos conocimientos una vez comprobada su utilidad.

Dado este esquema general del funcionamiento de la Inteligencia Humana, parece razonable intentar imitar su estructura y características en un intento por diseñar una arquitectura general de inteligencia, aplicada a la Robótica. Existe bastante controversia acerca de si esa imitación debe centrarse en los aspectos de implementación —imitando la estructura, interconexión, y funcionamiento de las neuronas, células básicas del depósito de la inteligencia humana que es el cerebro—; trabajar a nivel de "caja negra", replicando los diferentes subsistemas que se identifican con las capacidades cognitivas del ser humano; o intentar soluciones híbridas. La elección de una u otra filosofía influye no solo en las características de la arquitectura considerada, sino en las herramientas de *Inteligencia Artificial (IA)* utilizadas en su diseño. En este trabajo, inspirados por las teorías de Hawkins en su obra *On Intelligence*, hemos escogido la segunda vertiente, proponiendo una arquitectura jerárquica de inteligencia en el que los diversos módulos se implementan a partir de Razonamiento basado en Casos —*Case Based Reasoning (CBR)*—, una herramienta de *IA* especialmente apta para la adquisición de conocimiento a través del aprendizaje, y para la predicción basada en similitud de información.

Dentro de esta arquitectura la presente tesis se centra en las capas inferiores de la misma, las de tipo reactivo, expresadas en forma de comportamientos básicos, que implementan conductas sencillas pero indispensables para el funcionamiento de un robot. Estos comportamientos han sido tradicionalmente diseñados de forma algorítmica, con la dificultad que esto entraña en muchos casos por el desconocimiento

de los aspectos intrínsecos que los animan. Además, con este diseño, carecen de la capacidad de adaptarse ante nuevas situaciones no previstas y adquirir nuevos conocimientos a través del funcionamiento del robot, algo indispensable si se pretende que el robot se desenvuelva en ambientes dinámicos y no estructurados. Este enfoque no parece, por tanto, acertado. En su lugar, el trabajo de esta tesis considera la implementación de comportamientos reactivos con capacidad de aprendizaje, como forma de superar los inconvenientes anteriormente mencionados consiguiendo al mismo tiempo una mejor integración en la arquitectura general de Inteligencia considerada, en la cual el aprendizaje ocupa el papel principal.

Así, en esta tesis se proponen y analizan diversas alternativas de diseño de comportamientos reactivos, construidos a través de sistemas *CBR* con capacidad de aprendizaje. Dentro de estos sistemas *CBR* se estudia i) la problemática de selección, organización, y representación de la información como recipiente del conocimiento de los comportamientos; ii) los problemas asociados a la escalabilidad de esta información; iii) los aspectos que acompañan al proceso de predicción a través de la recuperación de la respuesta de una experiencia previa similar a la presentada; iv) la identificación de la respuesta no solo con la acción a tomar por parte del comportamiento sino con un concepto que represente la situación presentada; y v) la adaptación y evaluación de la respuesta obtenida, ante situaciones desconocidas, para incorporar dichas situaciones como nuevo conocimiento del sistema. Por otra parte, también se analiza la organización de comportamientos básicos que permite obtener, a través de sus interacciones, comportamientos emergentes de nivel superior pero aún dentro de un alcance reactivo.

Todo ello se probará con un robot real y con un simulador de dicho robot, en un escenario de aplicación clásico en Robótica, como es la competición Robocup, aunque en nuestro caso utilizaremos una variante ligeramente modificada. El desarrollo de esta tesis ha supuesto, además de los aspectos puramente investigadores, un esfuerzo adicional en el desarrollo de las herramientas y metodología de pruebas necesarias para su realización. En este sentido, se ha programado un primer prototipo de marco de implementación de comportamientos reactivos con aprendizaje, basados en *CBR*, para la plataforma de desarrollo robótico *Tekkotsu*.

Abstract

Although the final mechanisms which define Intelligence remain yet in the dark, it's already known that Learning plays an essential role in its development and, also, that Intelligence expresses through the ability to identify situations and predict actions or responses from the information collected precisely by means of such Learning.

Learning, both in its aspects of learning-by-observation and learning-by-experience, provides the human being the definitive way of knowledge acquisition. This knowledge allow humans to face, not only past experiences in which they can repeat successful known responses again, but also unknown new scenarios which can resemble, at higher or lower conceptual level, the already stored knowledge, so the "right" response could be also similar. Then, if this new experience and response show its utility, they can be also integrated into the knowledge pool.

Bearing in mind this general framework of Human Intelligence, many researchers are considering the imitation of this structure and characteristics in an attempt to develop a general framework for Robotic Intelligence. However, there is some controversy about which is the best way to achieve this objective. Some researchers prefer to focus on implementation issues, mimicking the structure, interconnection, and operation of neurons, the basic elements of the storage medium of knowledge, the human brain. Other scientists rather work at a "black box" level, so they aim their efforts at replicating the different subsystems responsible of human cognitive abilities. Finally, many others take hybrid approaches in order to take advantage of the best features from both views. Choosing one or another approach not only affects the characteristics of the designed architecture, but the AI tools used in its development. In our work, inspired by Hawkins' theories set out in his book *On Intelligence*, we have chosen the second of the above-mentioned approaches. Thus, we propose a hierarchical architecture of robotic intelligence in which the different modules are based on Case Based Reasoning (CBR), an AI tool specially suitable for knowledge acquisition through learning, and also for prediction based in information similarity.

In this architecture, the present thesis focuses on the lower reactive layers which are designed as basic behaviors. These behaviors implement several tasks which, although simple, are essential to achieve a good robot performance. Reactive behaviors have been traditionally designed in an algorithmic way, but this approach often fails, due to the lack of knowledge about the intrinsic mechanisms which control behavior. This approach has also problems related to the inability of the robot to adapt to

unexpected situations and to acquire new knowledge through operation, aspects which are mandatory for a robot if it is intended to operate in dynamic unstructured environments. Hence, this algorithmic approach doesn't seem appropriate. Instead, this thesis proposes the development of reactive behaviors with learning capabilities, as a way to overcome the aforementioned drawbacks, and allow a better integration into the proposed general framework for Robotic Intelligence, in which Learning plays a fundamental role.

Consequently, in this thesis we propose and analyze different alternatives for the design of reactive behaviors, implemented from CBR systems with learning capabilities. Regarding CBR systems, we study i) the difficulties related to selection, organization, and representation of information as a knowledge container for the behaviors;ii) the scalability problems which arise in connection of the information increase; iii) the aspects related to the prediction process, cristallyzed through the recovery of a response attached to a previous experience which is considered similar to the present situation; iv) the identification of this response not only with an action to be executed by the behavior, by also with a concept to depict the present situation; and v) the adaption and further evaluation of the recovered response in the presence of new unknown scenarios, so we can decide if these new situations can be integrated to the behavior knowledge.

Besides, we also study the organization and combination of basic behaviors in order to obtain, through their interactions, emergent higher-level behaviors, but still at a reactive level.

All this work has been tested with a real robot and a simulator of that robot, in a classic application scenario of the Robotic domain, the Robocup competition, though we will consider an slightly modified version. Development of this thesis has required, besides the academic researching aspects, an additional effort in order to implement the tools and methodology of experiments needed for its fulfillment. In this sense, we have programmed a first prototype of a development framework of reactive behaviors with learning capabilities, based on CBR, for the development robotic platform *Tekkotsu*.

Índice general

Acrónimos

3T	Three Tier
AFSM	Augmented Finite State Machine
AIBO	Artificial Intelligence roBOt
ANN	Artificial Neural Networks
BBR	Behavior-Based Robotics
BERRA	Behavior Based Robot Research Architecture
CBR	Case Based Reasoning
CIP	Cognitive Information Process
DBN	Dynamic Bayesian Networks
FL	Fuzzy Logic
FSM	Finite State Machine
GA	Genetic Algorithms
HTM	Hierarchical Temporal Memory
IA	Inteligencia Artificial
IRM	Innate Releasing Mechanism
ISIS	Ingeniería de Sistemas Integrados
KEMLG	Knowledge Engineering and Machine Learning
MPL	Multilayer Perceptron
NHC	Nested Hierarchical Controller

PCA	Principal Component Analysis
PF	Potential Fields
PFA	Potential Fields Approach
RAE	Real Academia Española de la Lengua
RCS	RealTime Control System
RL	Reinforced Learning
ROS	Robot Operating System
SA	Sense-Act
SPA	Sense-Plan-Act
TCA	Task Control Architecture
TCP	Transfer Control Protocol
VHF	Vector Histogram Field
YUV	Luminance-Bandwidth-Chrominance

Simbología

ρ	En la representación de una recta en el espacio de Hough, distancia entre el origen de coordenadas y el punto (x, y) de la recta más cercano al origen de coordenadas.
θ	En la representación de una recta en el espacio de Hough, ángulo del vector director de la recta perpendicular a la recta original y que pasa por el origen de coordenadas.
$\theta_{IN}(i)$	Valor θ de la recta i presente en la imagen de entrada al caso.
$\theta_{CBR}(i)$	Valor θ de la recta i presente en el caso CBR recuperado.
θ_{Prom}	Promedio de las variaciones de θ de todas las rectas consideradas en la entrada del caso CBR.
U_{adap_θ}	Umbral en la variación promedio de θ para todas las líneas, para adaptar el caso recuperado.
F_{adapt}	Factor de adaptación para la rotación propuesta en el caso CBR recuperado.
k_{adapt}	Constante de adaptación para la rotación propuesta en el caso CBR recuperado.
$BallPos$	Indicador binario de posesión de pelota.
c_n	Caso n dentro de una base CBR.
(Cx, Cy)	Coordenadas $[x, y]$ del centroide de un objeto.
a_i	Componente i dentro de un caso CBR.
$d(c_1, c_2)$	Distancia entre dos casos, $c1$ y $c2$, de una base CBR.
$d_i(c_1a_i, c_2a_i)$	Distancia local entre los componentes i de dos casos, $c1$ y $c2$.
F_{obj}	Factor de objetivo como componente de la función de utilidad de un comportamiento.
C_{ap}	Parámetro heurístico que permite ajustar la expresión del factor de utilidad dependiente de la aplicación.
tam_{pel}	Tamaño de la pelota en la imagen de cámara.
$\theta_{rob-pel}$	Ángulo entre el robot y la pelota.

F_{seg}	Factor de seguridad como componente de la función de utilidad de un comportamiento.
C_{seg}	Parámetro heurístico que permite ajustar la expresión del factor de seguridad.
tam_{obst}	Tamaño del obstáculo más cercano al robot.
$\theta_{rob-obst}$	Ángulo entre el robot y el obstáculo más cercano.
F_{suav}	Factor de suavidad como componente de la función de utilidad de un comportamiento.
C_{suav}	Parámetro heurístico que permite ajustar la expresión del factor de suavidad.
Δ_{dir}	Variación de la dirección de avance del robot entre dos consultas CBR consecutivas.
K_{obj}	Constante que fija la importancia relativa del objetivo en la eficiencia del comportamiento.
K_{seg}	Constante que fija la importancia relativa de la seguridad en la eficiencia del comportamiento.
K_{suav}	Constante que fija la importancia relativa de la suavidad en la eficiencia del comportamiento.
Mod_{input}	Información de entrada a un comportamiento/módulo CBR.
$Obji_{desc}$	Descritor del objeto i dentro de un caso.
I	Conjunto de imágenes observadas durante la ejecución o entrenamiento de un comportamiento.
V_I	Volumen de información asociado al conjunto de imágenes observadas durante la ejecución o entrenamiento de un comportamiento.
N	Dimensión del espacio de información de todas las imágenes observadas durante la ejecución o entrenamiento de un comportamiento.
P	Dimensión de un sub-espacio de N con información suficiente para la caraterización de un comportamiento determinado.
λ_{olvido}	Factor de olvido exponencial para el enventanado de imágenes consecutivas.
w_i	Peso de la distancia local del componente i entre dos casos.
g_i	Peso del componente de similitud local dentro de la similitud global.
p_{PF}	Peso del del vector PFA en la adaptación de un caso CBR.
S_i	Saltos entre los intervalos de dicretización del componente i entre el problema y el caso CBR.
$sim(c_1, c_2)$	Similitud entre dos casos, c1 y c2, de una base CBR.
sim_i	Medida de similitud local entre los atributos de un caso y un problema.
$sim(q, p)$	Medida de similitud entre un caso y un problema.
$umbral_{CmpCaso}$	Umbral en el número de saltos totales de diferencia entre un problema y un caso recuperado, para adaptar el caso.

$umbral_{CmpIndiv}$	Umbral en el número de saltos de diferencia entre componentes individuales de un problema y un caso recuperado, para adaptar el caso.
$umbral_{CmpObj}$	Umbral en el número de saltos totales de diferencia entre objetos de un problema y un caso recuperado, para adaptar el caso.
η	Utilidad del caso CBR adaptado tras su aplicación.
$\vec{V_{Atr_i}}$	Vector PFA de atracción del objeto i.
$\vec{V_{CBRCase}}$	Vector de acción del caso CBR recuperado.
$p_{RD}(t)$	Peso a aplicar al vector de respuesta del comportamiento de evitación de obstáculos en la composición del comportamiento emergente.
$dAtr_i$	Distancia estimada de un objeto atractor al robot.
α_{Atr_i}	Constante de caida del campo atractor.
β_{Rep_j}	Constante de caida del campo repulsor.
ϕ_{head}	Ángulo de la cabeza del robot con respecto a su dirección de avance.
ϕ_{img}	Ángulo del centro de los objetos visualizados respecto del punto inferior medio de la imagen de cámara.
ϕ_{obj}	(Estimación del) Ángulo los objetos visualizados respecto a la dirección de avance del robot.
Aa_i	Área visible del objeto atractor i.
Ar_j	Área visible del objeto repulsor j.
k_{rep}	Número de objetos repulsores/factor de ponderación del componente atractor en la generación del vector PF.
$\vec{V_{PF}}$	Vector generado mediante PFA para la adaptación de un caso CBR.
$\vec{V_{Rep_j}}$	Vector PFA de repulsion del objeto j.

Índice de figuras

Índice de tablas

1 Introducción: Vision general de la tesis

Un camino de mil millas comienza con un paso.
Proverbio chino

1. Introducción

En el presente capítulo se describen los objetivos perseguidos con la realización de esta tesis, orientada hacia el desarrollo y análisis de las capas de bajo nivel o reactivo de una arquitectura de inteligencia robótica según un modelo memoria/predicción, y basado en el paradigma de *Razonamiento Basado en Casos —CBR—*. Esto nos permitirá implantar en un robot comportamientos de bajo nivel o nivel reactivo que, en lugar de ser programados, puedan aprenderse a través de las experiencias "vividas" por el robot y supervisadas por un operador humano.
Este capítulo se organiza tal como se indica a continuación. En el apartado 1 se presenta la motivación que ha llevado al desarrollo de la tesis presentada; en el apartado 2, se indican los objetivos que se pretenden alcanzar a la finalización de la tesis; finalmente, en el apartado 3, se describe la estructura y organización de la tesis.

2. Motivación de la tesis

Uno de los grandes anhelos del ser humano ha sido, desde siempre, la creación de seres artificiales, criaturas nacidas de la mezcla de lo humano y la tecnológico. Ya en la mitología de la antigua Grecia se encuentran referencias como la del gigante de bronce Talos, o los autómatas del dios Hefestos. Durante muchos siglos estos ingenios solo fueron plasmados en la mitología, los cuentos, y la imaginación de los hombres; pero finalmente los avances en ingeniería permitieron la creación de los primeros autómatas, como las ingeniosas máquinas a vapor de Herón de Alejandría, en el siglo I d.C; o las figuras mecánicas del inventor musulmán Al-Jazarií en el siglo XIII. En los siglos siguientes, inventores como Leonardo Da Vinci, Müller von Königsberg, John Dee, o Jacques de Vaucanson, desarrollaron y en algunos casos implementaron diversos autómatas, a imitación de hombres y animales, que podían realizar algunas de sus

funciones, como volar o incluso ejecutar piezas musicales. A finales del siglo XIX y hasta finales del XX se construyeron varios ingenios de apariencia humana, capaces de realizar únicamente funciones muy específicas; todos ellos se basaban en servomecanismos. El término *Robot* data de 1921, y apareció por primera vez en la obra de teatro *Los Robots Universales Rossum* del escritor checo Karel Capek, para describir a servidores de apariencia humana. Desde este momento se empieza a extender la idea de que podría ser posible crear robots con comportamiento y apariencia que imitase a los seres humanos de una forma amplia: primero en la ciencia ficción, con autores como Isaac Asimov; y, poco después, con visos a una implementación real, con la formulación de los principios de la cibernética por parte de Norbert Wiener.

Es a partir de aquí que se toma conciencia de que, si se desea diseñar y construir robots con características similares a los seres humanos y que se comporte de forma *autónoma*, necesitamos dotarles de los medios para adquirir conocimientos acerca del entorno en el que se desenvuelve, procesar y comprender esta información, y tomar las decisiones adecuadas según la tarea que le haya sido encomendada, para influir sobre el mundo a partir de las órdenes a sus componentes motores y actuadores. En resumen, más allá de los componentes mecánicos y electrónicos; de la variedad de sensores; de la forma y estructura del robot, uno de los aspectos fundamentales para su diseño se debe centrar en dotarle de *Inteligencia* similar a la que tenemos los seres humanos.

La *Real Academia Española de la Lengua (RAE)* define la *Inteligencia*, en su diccionario de la lengua española, como la "capacidad para entender y comprender" y "la capacidad para resolver problemas". Detrás de esta aparentemente sencilla definición, nos encontramos algo mucho más complejo: la inteligencia nos permite comprender el mundo que nos rodea, asimilar su información a través de nuestras percepciones, y desenvolvernos en el mismo mediante las respuestas desencadenadas ante las percepciones que tenemos de nuestro entorno. La inteligencia es lo que nos hace seres humanos, lo que nos ha llevado a situarnos por encima del resto de seres vivos en la cadena trófica y lo que marca el acento en la evolución de nuestro futuro. Sin embargo los aspectos clave de la inteligencia permanecen aún envueltos en la bruma del misterio; sabemos que la tenemos, la usamos a diario para resolver problemas o realizar nuestras tareas cotidianas, pero no conocemos en toda su profundidad como se desarrolla, como funciona, que factores que una persona pueda llegar a alcanzar un potencial distinto de otra, etc.... Diversos investigadores de distintas áreas (Psicología, Biología, Computación,...) han reconstruido pequeñas partes aisladas del gran puzle, pero aún estamos lejos de completarlo.

La *IA* es un área multidisciplinar que estudia la creación y diseño de entidades artificiales con capacidad de razonar y resolver problemas de manera similar a como lo hacemos los seres humanos. Aunque la mayoría de los trabajos se desarrollan en las áreas de Ciencias de la Computación (Informática, Robótica, Electrónica,..), la *IA* se apoya en gran medida en los conocimientos propios de otras áreas, como la Psicología, la Lógica, la Filosofía, o la Biología. En resumen, trata de trasladar los conocimientos que tenemos de la Inteligencia Humana [1] al campo de la computación. Por ello, resulta fundamental el campo de origen a partir del cual se desarrollan las teorías e intentos de la *IA*: los nuevos avances y teorías acerca de la Inteligencia Humana constituyen un buen punto de partida para intentar desarrollar nuevos modelos de *IA*.

En este sentido, esta tesis parte del análisis un nuevo modelo de Inteligencia

[1] Nos referimos a Inteligencia *Humana* en contraposición a la Inteligencia *Artificial*

Humana, el modelo memoria/predicción desarrollado por el profesor Jeff Hawkins en su libro *On Intelligence* [1], para tratar de establecer una posible arquitectura de *IA* aplicable a la robótica. Dentro de esta arquitectura nos interesa especialmente las capas de nivel más bajo, que permitirían que un robot pudiese aprender comportamientos a un nivel reactivo, a partir de las experiencias vividas en situaciones similares durante su funcionamiento. Para la implementación de estas capas nos basaremos, en principio, en uno de los tipos de sistemas expertos que en nuestra opinión mejor se ajustan al modelo original, como es el razonamiento basado en casos o *CBR*.

3. Objetivos de la tesis

En la actualidad, las arquitecturas híbridas de modelado de inteligencia son, debido a su competencia y aplicación exitosa, las preferidas para el diseño de sistemas de control inteligente de los robots modernos. La mayoría de estas arquitecturas constan de una capa deliberativa de alto nivel, que gestiona los objetivos globales del robot a largo plazo, manejando para ello conocimiento más abstracto; de una capa reactiva que determina la respuesta del robot de forma local y a corto plazo, y está constituida por *comportamientos* rudimentarios; y de una o varias capas intermedias que actúan como interfaz o árbitro entre las capas deliberativa y reactiva. Las diversas implementaciones existentes pueden diferir en la organización de sus capas; en las herramientas o técnicas empleadas para implementar estas capas; o en las mecanismos para seleccionar o combinar los módulos que las integran. Sin embargo y, por lo general, los comportamientos se suelen diseñar en las etapas iniciales de desarrollo de la arquitectura y de un modo algorítmico inmutable con poca o ninguna capacidad de adaptación a posibles cambios en el entorno o en la evolución del problema. Hay ciertas evidencias que sugieren que estos componentes reactivos deberían tener capacidad de cambio o adaptación mediante un aprendizaje derivado de las situaciones experimentadas por el robot. Por ejemplo, el campo de la Etología nos enseña como existen ciertos comportamientos presentes en algunos animales, los llamados comportamientos reflejos o innatos, que son adquiridos desde el nacimiento, y que podrían "imitarse" en un robot mediante la programación de patrones sensación-acción, o usando Máquinas de Estado Finito —*Finite State Machine (FSM)*—. Por otra parte, las conductas y habilidades aprendidas deben ser entrenadas y desarrolladas mediante repeticiones y experimentación. Finalmente, tenemos los comportamientos innatos-con-memoria, una variante de los comportamientos innatos, que necesitan de un ajuste o puesta a punto a través de la experiencia: es así como, por ejemplo, una abeja-infante aprende, a partir del reflejo innato de vuelo. a reconocer el aspecto de su panal y como navegar desde y hacia él [2].

Adicionalmente, las teorías más actuales acerca de la inteligencia humana afirman que, incluso en una respuesta deliberativa compleja, solo hay un número pequeño de neuronas implicadas en la transmisión de impulsos nerviosos que constituye el reflejo del pensamiento, transmisión que tampoco se realiza a una velocidad muy elevada y que está, en todo caso, muy alejada de la velocidad de procesamiento de cualquier computador moderno. En vez de en la velocidad o capacidad de procesamiento, la inteligencia humana parece estar basada en el aprendizaje por experiencia, en almacenar los eventos diarios, los conocimientos adquiridos, las respuestas dadas, en

la potente base de datos constituida por nuestro cerebro, para recuperarlas cuando se presente una situación similar [1].

A día de hoy ya existen multitud de arquitecturas robóticas híbridas que siguen estas pautas de adaptación o aprendizaje [3] [4] [5],pero la mayoría se centra en el aprendizaje en los niveles deliberativos. Según estudios recientes, el cerebro humano parece seguir una estructura cíclica o "de nido", de forma que sus diferentes áreas deberían operar según los mismos principios de funcionamiento sin importar a que aspectos se dediquen. Solo se diferenciarían en la naturaleza de la información tratada y, especialmente, en el nivel de abstracción de los conceptos manejados que determinaría su posición en la cadena de razonamiento [1]. Así, parece razonable proponer una arquitectura de inteligencia robótica en la cual sus diferentes capas tendrían mecanismos de operación equivalentes, al menos a un nivel funcional y de procesamiento de la información. De este modo, todas las capas y sub-capas a cualquier nivel deberían fundarse en algún tipo de aprendizaje por experiencia, y esto *incluye a las capas inferiores de tipo reactivo*. Además de mejorar la flexibilidad y adaptabilidad del nivel reactivo, la arquitectura ganaría en homogeneidad y escalabilidad, especialmente si se emplean los mismo métodos o herramientas de *IA*. En este sentido, proponemos *CBR* como el método que consideramos más adecuado para acometer esta tarea, como veremos a continuación.

En un arquitectura robótica híbrida, la capa reactiva se basa en establecer una correspondencia entre la información sensorial y la respuesta de los actuadores, mediante un conjunto de módulos rudimentarios de bajo nivel denominados "comportamientos". Estos módulos no necesitan un modelo complejo del entorno o del problema a resolver, ya que trabajan únicamente con información local y a corto plazo, relacionada con los objetivos o campo de acción del comportamiento. Esto los hace especialmente adecuados para una toma de decisiones rápida a bajo nivel, algo fundamental en entornos dinámicos y no estructurados. Además, y mediante su combinación, debería ser posible obtener comportamientos emergentes más complejos —de nivel "más alto"—, lo cual redundaría positivamente en la escalabilidad del sistema. Esta emergencia se puede conseguir de diferentes formas, desde conmutación entre comportamientos hasta arquitecturas de subsunción [6], pasando por el uso de campos de potencial —*Potential Fields (PF)*— [7], pero en todas estas técnicas es necesario ajustar u optimizar un alto número de parámetros conforme al problema a resolver, con lo que existe una gran dependencia de aspectos como la cinemática y dinámica del robot empleado, la calibración de los sensores, o la aparición de errores de origen mecánico. Por este motivo, gran número de arquitecturas o capas reactivas incluyen herramientas de *IA* tales como reglas de lógica difusa —*Fuzzy Logic (FL)*— [8] [9], o redes neuronales artificiales —*Artificial Neural Networks (ANN)*— [10] que permiten un "ajuste fino" de los comportamientos a través de su ejecución. No obstante, el desarrollo de las expresiones analíticas que definan un comportamiento suele ser una tarea ardua, ya que algunos comportamientos son difíciles de expresar y se adaptan mejor a una función u otra dependiendo de las circunstancias específicas de operación. Los métodos basados en lógica difusa definen un conjunto de reglas para establecer correspondencias entre las lecturas de los sensores del robot y las instrucciones motoras, reglas cuyos parámetros pueden ajustarse a través del funcionamiento del robot. Sin embargo, resulta complejo diseñar correctamente estas reglas, ya que para ello se requiere un gran dominio tanto del

campo de la lógica difusa como del dominio del problema a resolver. Por su parte las *ANN* no presentan estos problemas ya que, una vez se ha escogido la estructura de capas y neuronas de la red neuronal, el proceso de aprendizaje resulta transparente para el usuario. Por contra, aunque el uso de *ANN* nos va a proporcionar un sistema de control mejor o peor, no nos proporciona ninguna pista acerca de que se podría hacer para mejorar su comportamiento, ya que las *ANN* son, para el usuario, similares a "cajas negras". Frente a estas herramientas tenemos *CBR* que, combinado con una adquisición de conocimientos a través del aprendizaje nos permite, no solo obtener un sistema totalmente funcional, sino comprender también el proceso de aprendizaje y adquisición de conocimientos, lo cual resulta extremadamente útil para la depuración de errores o funcionamientos incorrectos del sistema. Por este motivo, en esta tesis elegiremos *CBR* como el elemento fundamental sobre el que se basa el diseño de la arquitectura de inteligencia robótica propuesta. A partir de bases CBR especialmente modificadas y adaptadas a las necesidades de la arquitectura, desarrollaremos módulos que implementen comportamientos básicos a nivel reactivo, cuya funcionamiento se aprenderá a través de un entrenamiento y de la experiencia del robot. Posteriormente, y siempre a un nivel reactivo, se estudiará la forma de combinar entre sí varios de estos comportamientos para conseguir obtener un comportamiento emergente de nivel superior.

4. Estructura de la tesis

Este trabajo se enmarca dentro de una de las líneas de investigación que ha venido desarrollando, en los últimos años, el grupo de investigación *Ingeniería de Sistemas Integrados (ISIS)*, dentro del Departamento de Tecnología Electrónica, en el campo de la Inteligencia Computacional aplicada a la Robótica. *ISIS* ha participado en varios proyectos de investigación (TIN2004-05961 TIN2004-07741-C02-01, TIN2008-06196/TIN, o TIN2011-27512-C05-01, entre otros) en los cuales se han implementado y probado diversos paradigmas de IA en el entorno de la Robótica.
El trabajo se entronca también con otras tesis desarrolladas por miembros del grupo [11] [12]. Una gran parte de estos trabajos previos ya utilizan *CBR* como paradigma fundamental del modelo de inteligencia artificial desarrollado, si bien la gran mayoría se centra en los niveles deliberativos o de planificación a alto nivel en las arquitecturas de comportamiento desarrolladas. Además, estos trabajos no utilizan la visión artificial como fuente principal de información del robot. Por último, todos ellos trabajan con robots rodantes, por lo que se eluden una serie de aspectos que cobran especial relevancia con el uso de robots de tipo bípedo o cuadrúpedo.

La organización seguida en la presente tesis es la que se describe a continuación:

- **Capítulo 2. Arquitecturas y Comportamientos** . Este capítulo se centra en el concepto de arquitectura, entendida como el modelo de organización y procesamiento de información que hace posible la inteligencia en un robot. En él se repasarán las principales familias o paradigmas de arquitectura robótica, indicando algunos ejemplos de implementación de las mismas. De todas ellas, nos fijaremos seguidamente en las arquitecturas y capas reactivas, para las cuales se define el concepto de comportamiento, como elemento fundamental constituyente. Se realizará un breve repaso a los comportamientos, desde su origen biológico

hasta su aplicación en el ámbito de la robótica, revisando sus aspectos asociados más importantes que serán tratados posteriormente de forma práctica.

- **Capítulo 3. Inteligencia y aprendizaje**. En este capítulo se realizará una revisión del concepto de inteligencia, inicialmente referida a la inteligencia humana, para su posterior trasposición a la inteligencia artificial o robótica. De esta última se realizará un rápido repaso de sus herramientas y métodos más relevantes. A continuación, se incidirá en el papel que tiene el aprendizaje en el desarrollo de la inteligencia partiendo, de nuevo, desde la visión de la inteligencia humana. Por último, se estudiará y analizará el modelo "memoria/predicción" de Inteligencia Humana, propuesto por J. Hawkins que, trasladado al campo de la Robótica, constituirá la base teórica de la arquitectura de inteligencia propuesta en esta tesis.

- **Capítulo 4. CBR: razonamiento basado en Casos**. Se estudiará de forma detallada las características y aspectos más relevantes de esta herramienta *IA*, prestando especial importancia a los aspectos de adquisición y gestión del conocimiento en la misma. Se tratará de buscar analogías entre sus diferentes aspectos y componentes, y los asociados al modelo "memoria-predicción" analizado en el capítulo anterior, como medio de implementar dicho modelo en el ámbito de la Robótica.

- **Capítulo 5. Arquitectura híbrida de aprendizaje basada en *CBR***. En este capítulo se propondrá una arquitectura de *IA* basada en el modelo de inteligencia "memoria-predicción", y fundamentada en *CBR* [2]. De esta arquitectura se estudiará en detalle la implementación de sus capas inferiores (capas reactivas) en un robot cuadrúpedo modelo *Artificial Intelligence roBOt (AIBO)* cuyo funcionamiento se probará en la resolución de un problema en un entorno simulado. En dicho problema, la fuente principal de información para el robot provendrá de una cámara integrada en el mismo, por lo que tendremos que resolver varios problemas relacionados con la Visión Artificial.

- **Capítulo 6. Pruebas experimentales**. Este capítulo está dedicado a presentar las pruebas realizadas para analizar la viabilidad de la capa de arquitectura propuesta en los capítulos anteriores. Para ello se presentará un escenario de pruebas —campo de fútbol de Robocup para AIBOs— para el cual se desarrollarán módulos básicos CBR que implementen comportamientos o aspectos de comportamientos a través de un entrenamiento previo complementado con un aprendizaje por experiencia. Se detallará el proceso de diseño e implementación de los módulos justificando las decisiones tomadas. Finalmente se propondrán algunas pruebas de combinación de módulos básicos para la obtención de comportamientos emergentes más complejos.

- **Capítulo 7. Conclusiones y Lineas Futuras**. Para finalizar en este capítulo, y a la luz de los resultados obtenidos en el capítulo de Pruebas, se establecerán las adecuadas conclusiones respecto a la capacidad de la arquitectura propuesta como base para implementar la capa reactiva de una arquitectura híbrida de

[2] Se propondrán también alternativas que incluyan otros paradigmas de *IA* como, por ejemplo, reglas de decisión

inteligencia robótica. Así mismo, se detallarán posibles líneas de acción para subsanar debilidades o errores de la misma, y/o buscar nuevas propuestas basadas en el paradigma *CBR* o en variantes del mismo que añadan otros métodos o herramientas complementarias de *IA* .

2 Arquitecturas y comportamientos

El comportamiento es un espejo en el que cada uno muestra su ser
Johann Wolfgang von Goethe (1749-1832) Escritor y científico

1. Arquitecturas de control

Los sistemas robóticos se diferencian de muchas otras aplicaciones tecnológicas en la complejidad de sus objetivos de funcionamiento y en la necesidad de desenvolverse en entornos dinámicos no estructurados. En esta situación, la combinación de la propia dinámica del robot con la de su entorno, la aparición de errores e incertidumbre, y la necesidad de reaccionar en tiempo real ante situaciones inesperadas o no perfectamente definidas, determinan que los aspectos de diseño, operación, y validación de estos sistemas requieran un tratamiento especial y un conjunto de técnicas adecuadas. En el ámbito de la Robótica, una arquitectura de control define un marco de implementación y un conjunto de herramientas que permiten afrontar la complejidad del diseño y realización de un robot completo o un sistema o subsistema del mismo. La arquitectura de un robot engloba diferentes aspectos o puntos de vista. La *estructura arquitectónica* se refiere a como un sistema complejo se puede dividir en subsistemas, y a la forma en que estos interactúan; un ejemplo sería establecer una serie de capas o niveles, cada uno de los cuales trata con un aspecto específico de funcionamiento del robot. En cambio, el *estilo arquitectónico* se centra a los conceptos computacionales o algorítmicos que subyacen en el sistema, por ejemplo, si el intercambio de información entre los módulos del sistema se realiza mediante *sockets* o memoria compartida, o si se utiliza el entorno *Tekkotsu* o el *Robot Operating System (ROS)* para el desarrollo software de la estructura.

Si bien todo robot tiene una arquitectura con una estructura dada y un estilo, a menudo resulta complicado separarla de la implementación particular sobre la que se ha desarrollado inicialmente. Por lo general la elección de la arquitectura está muy influenciada por el tipo de aplicación para la que se diseña el sistema o subsistema o, en el caso de los robots, a la funcionalidad que se trata de conseguir en su operación. Por este motivo, la búsqueda de una arquitectura con una orientación más generalista como el diseño de un marco general de inteligencia robótica, resulta

una tarea realmente difícil. En todo caso, la estructura de la arquitectura estará más orientada a la implementación particular, e implica la realización de una filosofía que se puede obtener a través de diversas herramientas. En esta tesis consideraremos ambos aspectos, si bien dentro de la filosofía —en nuestro caso, la importancia del aprendizaje como elemento clave de la inteligencia—, nos centraremos en la implementación de una determinada estructura. Para ello seguiremos una estrategia de descomposición en la cual dividiremos el sistema a implementar en **módulos**, unidades funcionales menores interconectadas entre sí que, al operar conjuntamente, desarrollarán la funcionalidad total del sistema. Esta descomposición nos permite afrontar el diseño de cada módulo de forma independiente lo cual redunda en un diseño mas fácil de abordar. Por el contrario, la división da lugar a una necesaria fase de *integración* de componentes en la cual se deposita gran parte de la complejidad eludida a través del diseño individual de los módulos. En los siguientes apartados se analizarán someramente los principales estilos de arquitectura de control existentes en el estado de la ciencia actual, en lo referente a su aplicación al diseño de una arquitectura de inteligencia robótica. Estos son los denominados *Paradigmas de inteligencia robótica*, y son el paradigma jerárquico o deliberativo; el paradigma reactivo; y, por último, el paradigma híbrido.

2. Paradigmas de inteligencia robótica

Aunque hay una enorme variedad de robots, con diferentes capacidades y entornos o campos de utilización, todos ellos comparten unas mismas características en lo relacionado a su estructura y componentes. Todo robot tiene un armazón o estructura mecánica con una forma física, orientada generalmente a la realización de una tarea en particular. Esta estructura está muy relacionada con las necesidades sensoriales de adquisición de información, y las necesidades motoras asociadas al tipo de movimiento y manipulación que es capaz de desarrollar el robot, que determinan la inclusión de los sensores y actuadores adecuados a dichos requerimientos. Además, los robots contienen algún tipo de elemento de control que determina que debe hacer el robot en cada momento según las circunstancias en que se encuentre, tanto a nivel del estado del entorno, como de su propio estado actual interno, y el objetivo —o sub-objetivo final— que determina su razón de ser. Este elemento de control es lo que constituye lo que podríamos denominar la Inteligencia del robot.

En este sentido, se suele considerar la existencia de tres grandes familias o *paradigmas* que estudian posibles formas de organización de la inteligencia en los robots: *jerárquico*, *reactivo*, e *híbrido deliberativo/reactivo*. Para diferencia entre estos paradigmas, partimos de la existencia de tres primitivas básicas que definen el funcionamiento de un robot: *sensar*(SENSE), *planificar*(PLAN), y *actuar*(ACT) [13]. La operación de *sensar*, nos permite recolectar información del entorno del robot a través de sus diversos sensores; *planificar*, se centra en construir un modelo del mundo en el que se desenvuelve el robot, y determinar las acciones a realizar según el estado de dicho entorno y la funcionalidad o motivación que rige al robot; *actuar*, se refiere a como realizar las acciones planificadas en el paso anterior, así como supervisar su cumplimiento. Es a través de estas primitivas como establecemos las diferencias entre los paradigmas y sus implementaciones: en particular, según la forma en que se relacionan las tres primitivas; pero también según como se procesa y distribuye la

información sensorial a través de todo el sistema que integra al robot, y en que lugar —o lugares— del sistema se toman las decisiones correspondientes a la planificación.

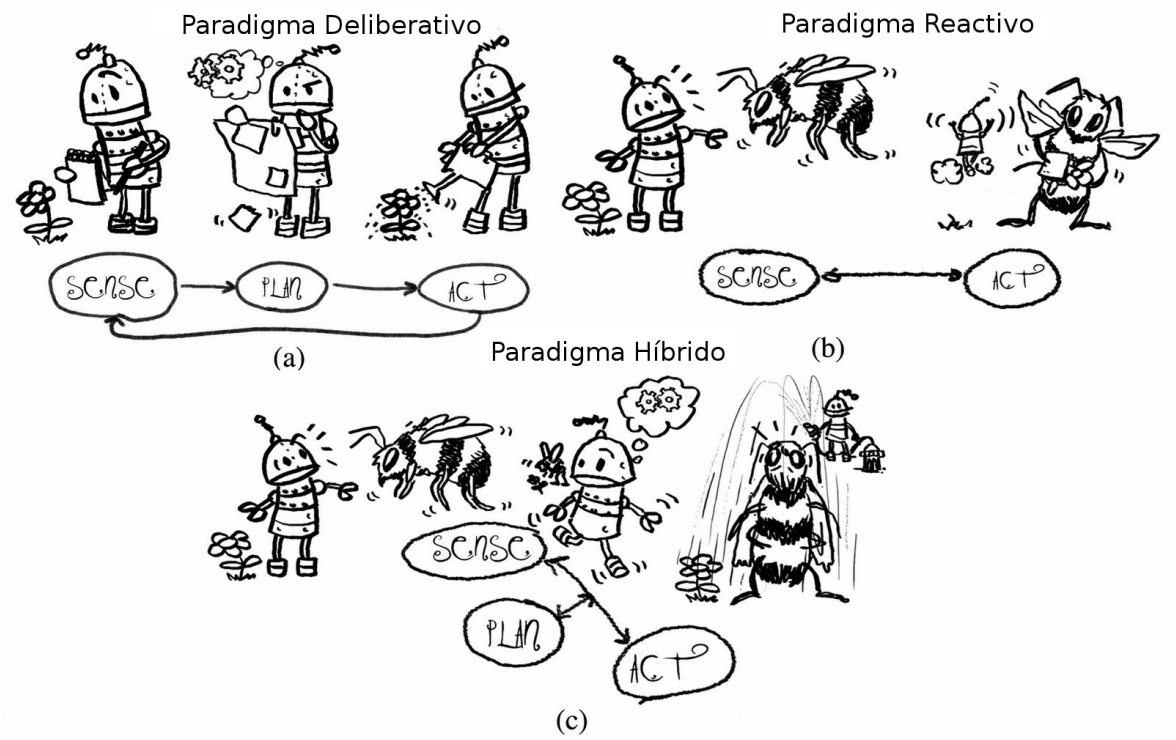


Figura 2.1: Primitivas Sense-Plan-Act en los diferentes paradigmas(C. Urdiales)

2.1. Paradigma Jerárquico

El paradigma *Jerárquico* (también llamado *Deliberativo*) es el más antiguo de los tres y fue el más empleado hasta principios de los 90. Bajo este paradigma, las tres primitivas de la Robótica se engarzan de forma secuencial y ordenada, en una filosofía descendente (*top-down*). En primer lugar, el robot obtiene información de su entorno a través de sus sensores (SENSE) y construye un modelo de su entorno y del estado actual del mismo; en una segunda fase, el robot computa las órdenes y acciones necesarias para alcanzar o al menos acercarse a su objetivo (PLAN); y en una tercera fase, se trasladan esas órdenes a comandos debajo nivel sobre los actuadores que permiten llevarlas a cabo (ACT, figura 2.1.a) [14] [15]. Shakey, el considerado como primer robot móvil autónomo con *IA* utilizaba una variante de este paradigma.

A medida que aumentó la complejidad de los robots y de las tareas a realizar por estos, empezaron a mostrarse las limitaciones de este modelo. La información necesita atravesar todos los módulos de la arquitectura, por lo que cualquier fallo en alguno de ellos provoca un fallo total del sistema. La construcción de un modelo del mundo, que incluya no solo las características mas o menos estables del entorno, sino las cambiantes, e incluso las del propio robot, se convirtió en una tarea realmente complicada. Además, se reconoció rápidamente el problema de no poder rectificar ante una información errónea, incompleta, o cambiante, durante las fases de planificación y actuación. Para paliar este problema, se introdujeron sub-objetivos dentro del objetivo general del robot, así como la capacidad de evaluar en que medida se iban cumpliendo

esos sub-objetivos y el objetivo final. De esta forma, el ciclo secuencial *Sense-Plan-Act (SPA)* se ejecutaba de forma mas frecuente y con variables adicionales que modificaban o afectaban principalmente a la forma en que se realizaba la planificación. Por otra parte, este paradigma asume que la información que obtiene el robot del entorno es siempre completa y que por tanto no se va a encontrar con situaciones desconocidas ante las que no tenga prevista una respuesta —hipótesis de *mundo cerrado* [6]—. Esta suposición no es asumible por ejemplo para la inteligencia humana, y parece razonable que tampoco lo sea para una *IA* basada en la humana. Los seres humanos nos enfrentamos cada día a sorpresas que desafían lo que creíamos conocer del mundo y aún así somos capaces de encontrar soluciones, mejores o peores. Lo mismo deberíamos buscar en los robots. Como alternativa o solución a este problema, los investigadores en Robótica plantearon la posibilidad de trabajar en diferentes niveles de abstracción, empezando por modelos del mundo sencillos de construir y actualizar para intentar generalizar los resultados de sus investigaciones a mundos cada vez mas abiertos y complejos y, finalmente, al mundo real.

2.2. Paradigma Reactivo

El paradigma reactivo surgió, a finales de los 80, como una respuesta al paradigma jerárquico más orientada hacia la inteligencia biológica y la psicología de los seres vivos. Su principal diferencia es la eliminación completa de la fase de planificación, quedando únicamente las fases de "sensado", y de actuación (*Sense-Act (SA)*, figura 2.1.b) [13]. En este caso, la información de los sensores provoca directamente una respuesta en los actuadores del robot.

Para conseguir conductas más complejas, el funcionamiento del robot se basa en la superposición de módulos *SA* denominados *comportamientos* [13], que trabajan con información local para obtener la mejor salida para el módulo, independientemente del resto de módulos (figura 2.2). De esta forma, la salida suele ser algún tipo de combinación/selección de las salidas individuales de los distintos comportamientos, lo que se suele denominar un *comportamiento emergente*. La eliminación de un modelo interno del entorno evita la necesidad de mantener una representación actualizada del mismo, y permite que cada módulo trabaje únicamente con información local directamente relacionada con el cometido del módulo. Así, se eliminan muchos de los problemas manifestados en los sistemas deliberativos. Por otra parte, la utilización de una filosofía ascendente (*bottom-up*) en la descomposición del sistema, frente a la división horizontal típica de las arquitecturas deliberativas permite que cada comportamiento tenga un flujo de información propio entre la parte sensora y la parte actuadora, frente a la existencia de un único flujo lineal de los sistemas deliberativos. Esto da lugar a importantes ventajas, como i) la independencia y, por tanto, reducción del tiempo de respuesta de cada módulo, que redunda en un tiempo de respuesta más rápido de todo el sistema al completo; ii) la mayor resistencia a errores, debido a la —relativa— independencia entre los comportamientos; o iii) la mayor facilidad de diseño de cada uno de los módulos independientes que favorece una mayor escalabilidad del sistema, si bien a medida que aumenta el número de componentes la fase de integración también puede llegar a complicarse bastante.

Como esta tesis se centra en la implementación de las capas reactivas de una

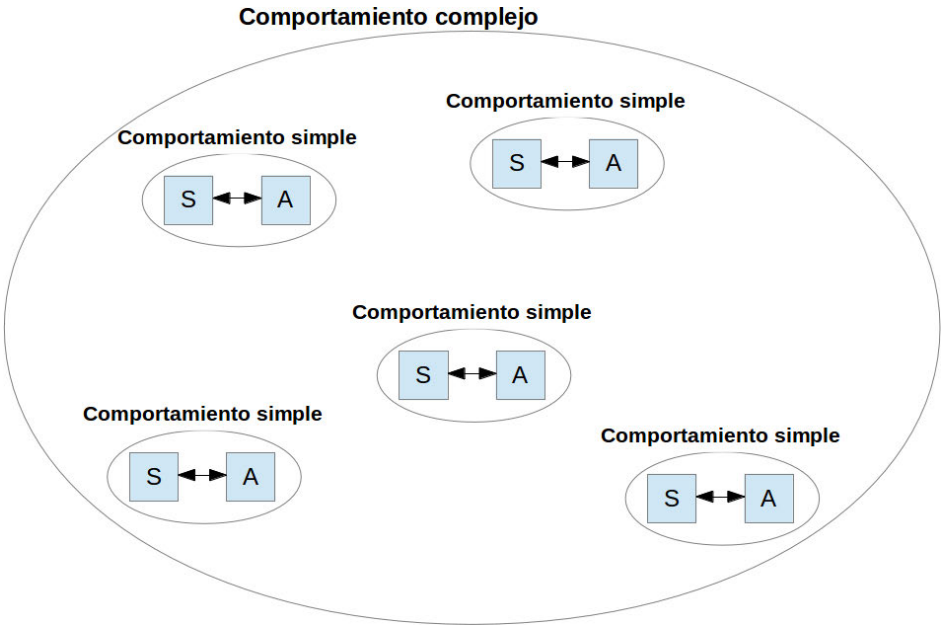


Figura 2.2: Superposición de comportamientos reactivos para generar uno más complejo

arquitectura a través de comportamientos, dedicaremos un apartado más adelante para tratar este paradigma con mas detalle.

2.3. Paradigma Híbrido

El paradigma Híbrido Reactivo/Deliberativo surgió a mediados de los 90, partiendo de los dos anteriores y es el paradigma que se suele seguir en la actualidad en la investigación de *IA* Robótica. Trata de complementar sus respectivas fortalezas suavizando, al mismo tiempo, sus debilidades. Así, en estas arquitecturas suelen existir 3 capas básicas: una capa reactiva de bajo nivel; una capa deliberativa de alto nivel; y una capa intermedia que actúa de interfaz entre ambas. Estas capas se pueden descomponer, a su vez, en un mayor o menor número de módulos. A este tipo particular de variante de arquitectura híbrida se le denomina también, por este motivo, "de 3 capas" —*Three-layer*— La mayoría de las soluciones que siguen esta filosofía son muy particulares y adaptadas a la resolución de un problema concreto, por lo que resulta difícil compararlas a nivel de prestaciones. En [16] podemos encontrar una revisión de las arquitecturas híbridas mas influyentes en los últimos años, entre las que podemos destacar la arquitectura *Three Tier (3T)* [17]; la *Task Control Architecture (TCA)* [18] ; la *LASS* [19]; y la *Behavior Based Robot Research Architecture (BERRA)* [20]. En la arquitectura *3T*, por ejemplo, el robot primero planea o delibera como descomponer su objetivo principal en una serie de sub-objetivos o tareas, así como cuales de sus comportamientos implementados serían los más adecuados para cumplir cada una de esta tareas. A continuación, los comportamientos escogidos comenzarían a ejecutarse según el paradigma reactivo. Pasamos, pues a una organización de las tres primitivas de la Robótica en la forma *planificar*, y a continuación *sensar-actuar* (P,S-A, figura 2.1.c). En la fase de adquisición de información, los sensores proporcionan a cada comportamiento la que éste necesita, pero también proporcionan la información al

modulo planificador de manera que se trabaja simultáneamente con modelos locales mas simples y detallados en los comportamientos; y con un modelo global mas abstracto.

Hay que tener en cuenta que la planificación, o porción deliberativa, tiene un horizonte a largo plazo y necesita de un conocimiento global del entorno, generalmente trabajando con representaciones más abstractas de la información; por contra la necesidad de una gran potencia computacional y la visión global del problema, hacen que esta parte no sea adecuada a una ejecución rápida, en tiempo real. La parte reactiva también sufre variaciones respecto al modelo reactivo "puro". Los comportamientos no funcionan solo como acciones reflejas ante sensaciones captadas, sino que se incluyen también otro tipo de comportamientos, como los innatos —no dependen de la información "sensada"— [2]; o los *aprendidos* [21] [22]. La combinación de salidas de los comportamientos es también más variada, y puede ir más allá de, por ejemplo, una suma vectorial de las trayectorias determinadas por los comportamientos individuales.

3. Arquitecturas de IA en Robótica

Los paradigmas descritos en el apartado anterior establecen una filosofía de organización y de trabajo, pero son las *arquitecturas* las que permiten una formalización de las mismas que se pueda llevar a su implementación física en el mundo real. Las arquitecturas proporcionan un medio apropiado de organización de un sistema de control y, al mismo tiempo, imponen ciertas restricciones sobre la forma en que el problema concreto de control puede ser resuelto [23]. La arquitectura permite definir también un conjunto de elementos o componentes de la misma, y unas reglas de interacción entre dichos componentes [24]. Estudiando arquitecturas en Robótica y los casos donde fueron usadas para implementar una aplicación robótica, podemos conocer diversas formas de emplear las herramientas e ideas asociadas a un paradigma para construir robots inteligentes.

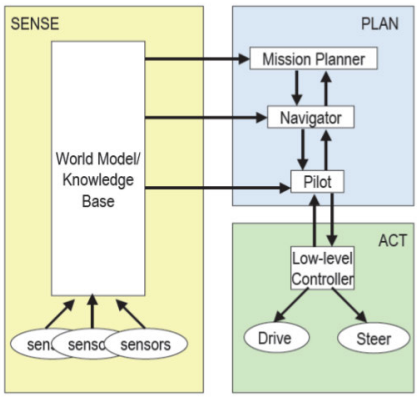
En este sentido, existen una serie de criterios para la evaluación de una arquitectura diseñada para implementar una *IA* Robótica [13]:

- Soporte a la modularidad.
- Orientación a un nicho o parcela. Evaluación de buen funcionamiento en la aplicación para la que fue ideada.
- Portabilidad a otros campos de aplicación u otros robots.
- Robustez en su funcionamiento.

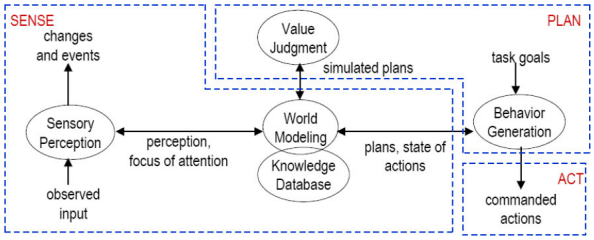
3.1. Arquitecturas Deliberativas

Entre las arquitecturas robóticas basadas en el paradigma jerárquico o deliberativo, destacamos las dos siguientes arquitecturas clásicas (figura 2.3), ya que exponen perfectamente la idea que subyace en esta categoría:

- el *Controlador Jerárquico Anidado*(*Nested Hierarchical Controller (NHC)*) [25], cuya principal contribución fue la descomposición de la capa de planificación en tres subcapas, según el nivel de abstracción de la funcionalidad del robot



(a) **Arquitectura NHC [25]**



(b) **Arquitectura RCS-NIST [14]**

Figura 2.3: Ejemplos de arquitecturas Jerárquicas

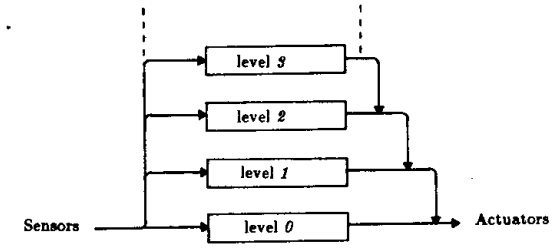
(Planificación de Misión, Navegador, y Piloto), y que admite la posibilidad de recoger información de los sensores para modificar su curso de acción.

- Otra arquitectura muy conocida de este periodo fue el *Sistema de Control en Tiempo Real*(*RealTime Control System (RCS)*) desarrollada por Albus [14]. Con una arquitectura basada en la *NHC*, destaca por la descomposición de cada uno de los bloques de primitivas (*sense-plan-act*) para asociarlos a los sub-bloques mas apropiados, incorporando un pre-procesamiento de la información sensorial o *extracción de características*; por otra parte, incorpora un modulo que simula y valora el plan establecido antes de llevarlo a cabo. Una variante de *RCS*, la arquitectura *NASREM*, se ha seguido empleando hasta hace muy poco en sistemas de control semi-autónomos, como por ejemplo el control de un brazo robótico en el espacio.

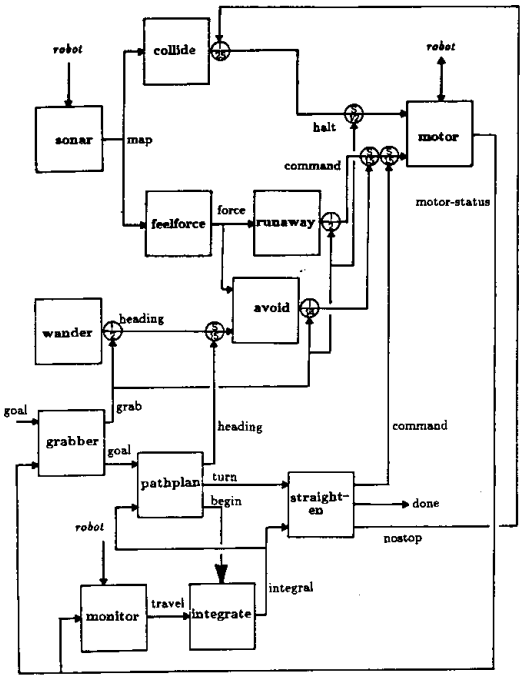
En general estas arquitecturas suelen estar orientadas a la resolución de un tipo de problema particular —en concreto, relacionado con la navegación o desplazamiento de un robot— y presentan, como principal inconveniente, su incapacidad de adaptación ante entornos no completamente conocidos, y su poca velocidad de respuesta ante situaciones que lo requieran. No obstante, a ellas se deben conceptos aún usados en arquitecturas mas modernas de *IA*, como la descomposición de sistemas en módulos dedicados a diferentes niveles de abstracción, o la evolución del comportamiento desde un modo de control semi-autónomo hacia la obtención de una autonomía cada vez mayor.

3.2. Arquitecturas Reactivas

En cuanto a las arquitecturas reactivas debemos destacar principalmente, la arquitectura de *subsunción* o inclusión de Rodney Brooks [26,27] (figura 2.4), así como los trabajos de investigadores posteriores basados en la misma, como Arkin [13, 28], o Mataric [23, 29]. En el trabajo original de Brooks, se define cada comportamiento a través de una red de módulos de "sensado" y activación motora orientados a la realización de una tarea específica; los módulos se implementan a través de máquinas de estados finitos aumentadas (*Augmented Finite State Machine (AFSM)*), que pueden



(a) Diagrama general

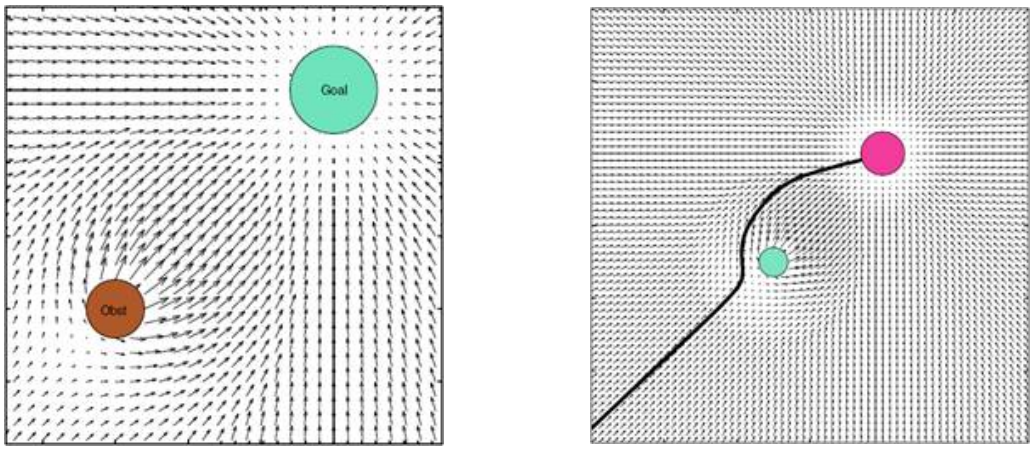


(b) Ejemplo de implementación

Figura 2.4: Arquitectura reactiva de subsunción [26]

interactuar con otros módulos. Los comportamientos se activan en respuesta a estímulos, sin ningún tipo de control o coordinación externa realizada por un módulo de nivel superior. Los módulos se agrupan en capas de competencia, de forma que las capas de nivel inferior incluyen funciones de "supervivencia" del robot, como evitar colisiones o daños, y las capas de niveles mas altos se relacionan con niveles mas abstractos o dirigidos a una funcionalidad compleja, como construir el mapa del camino recorrido. Las capas de un nivel superior pueden modificar, completar, e incluso sustituir las salidas de los comportamientos de su nivel inferior, pero debe existir algún tipo de módulo de arbitraje de determine que salida definitiva aplicar en caso de que exista un conflicto entre diferentes salidas de varios comportamientos a diferentes niveles; en la arquitectura de subsunción original, se utilizaba un mecanismo *winner-take-all* en el que se daba prioridad a la salida de la capa mas alta, a través de mecanismos de supresión e inhibición. En esta arquitectura se trata de minimizar cualquier representación interna del entorno; en todo caso se usa algún tipo de estado interno para activar ciertos comportamientos (como comportamientos tipo "hambre" y "miedo", en el modelo original). Por último, para la ejecución de una tarea existe una cierta jerarquía de activación de las capas relacionadas, a diferentes niveles, aunque este modelo es difícil de generalizar, y suele necesitar una reprogramación para poder ser capaz de realizar otra tarea distinta.

La arquitectura de subsunción permitió solucionar algunos de los inconvenientes de las arquitecturas jerárquicas, como la posibilidad de trabajar a distintas escalas temporales según la necesidad de rapidez de la respuesta; una mayor robustez ante la posibilidad de fallos en alguno de los módulos; o su flexibilidad para modificar y redimensionar los sistemas a partir de la modificación o añadido de nuevos módulos. Sin embargo siguieron presentando una serie de limitaciones importantes relacionados sobre



(a) Representación de *PF* **(b) Comportamiento basado en *PF***

Figura 2.5: Campos de Potencial [30]

todo con la interacción entre componentes al mismo y a distintos niveles, especialmente en la activación de comportamientos a un mismo nivel, y los mecanismos de cooperación entre dichos comportamientos.

Un ejemplo de implementación de arquitecturas reactivas es el basado en la técnica de *campos de potencial*(*Potential Fields Approach (PFA)*) [28]. En este esquema se representan los comportamientos a través de vectores, y se combina la acción de varios comportamientos mediante sumas de vectores, que dan lugar a comportamientos emergentes desde varios comportamientos básicos. En las arquitecturas de campos de potencial las acciones motoras asociadas a los comportamientos se representan mediante campos potenciales, *arrays* de vectores que representan fuerzas de algún tipo, con una magnitud y una dirección (figura 2.5).

Estos *arrays* indican —generalmente— la evolución de una magnitud en una región del espacio, de manera que los objetos del entorno producen "campos de fuerza" de atracción o de repulsión, similares a los campos magnéticos o gravitacionales. Los robots sufren el efecto de la combinación de los campos generados por los distintos objetos, tanto en fuerza como en dirección de atracción/repulsión. La estructura de campos aplicadas a los objetos depende del comportamiento particular del robot; así, un mismo objeto —por ejemplo, otro robot— puede ejercer un campo de atracción cuando el cometido del comportamiento es interceptarlo; y un campo de repulsión si el comportamiento tiene por objeto evitarlo. Cuando el robot se ve sometido a varios campos, se realiza una suma vectorial de cada uno de ellos, y se aplica el vector resultante. En caso de que la suma de 0 —problema de *mínimos locales*— se suelen añadir componentes adicionales de ruido que produzcan vectores aleatorios de pequeña magnitud.

Por lo general, las arquitecturas de campo de potencial se usan en problemas relacionados con la navegación del robot, por lo que la fuerza ejercida por los campos de potencial suele depender de la distancia del robot y de la magnitud del campo; sin embargo, también es posible establecer la fuerza del campo en función de otros estímulos, como por ejemplo el tamaño del objeto generador del campo, o su color. Se suele considerar a los robots como partículas que pueden cambiar instantáneamente de velocidad y dirección lo cual, no solo no es cierto, sino que es especialmente complejo en cierto tipo de robots, como los que tienen patas. Por otra parte, los comportamientos no

se agrupan en capas, y no hay especificado un elemento de control coordinado (aunque puede existir).

Las arquitecturas reactivas presentan, no obstante, algunos problemas evidentes. Tomando el ejemplo de implementación mediante *PFA* se suele observar la aparición de oscilaciones en la trayectoria de los robots, especialmente en el recorrido de corredores estrechos, cuando no se produce la caída en situaciones de mínimos o trampas locales que impiden una correcta ejecución del comportamiento [31]. Incluso en ausencia de los problemas anteriormente mencionados, se observa una cierta falta de eficiencia en el funcionamiento, como por ejemplo la dificultad para encontrar y ejecutar el camino más corto y directo en una determinada trayectoria.

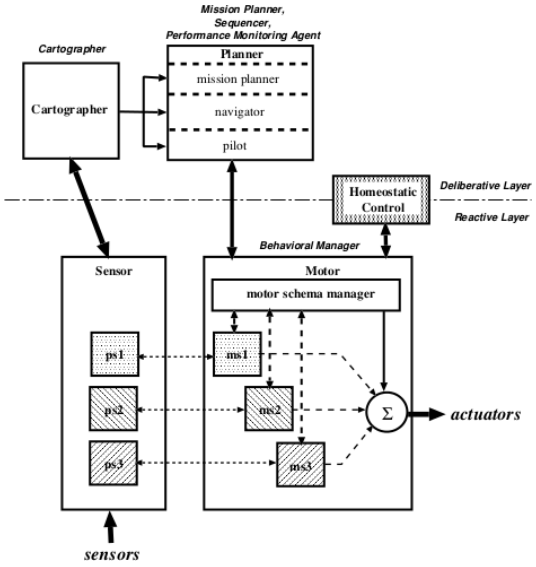
El principal inconveniente de este tipo de arquitecturas es su limitación a aplicaciones que puedan ser implementadas con comportamientos basados en reflejos. Por tanto, no proporcionan soluciones adecuadas para casos en los que el robot necesita, por ejemplo, razonar acerca de la localización de un recurso, planear un curso de acción, etc...Un robot con esta arquitectura hará siempre algo consistente con su percepción actual del entorno, pero esa acción no tiene porqué ser las más correcta o indicada. Un factor crucial será también la capacitación del sistema sensorial del robot para proporcionar la información mas adecuada a cada comportamiento, y el pre-procesado que le permita dar un formato tratable y manejable por parte de los comportamientos.

3.3. Arquitecturas Híbridas

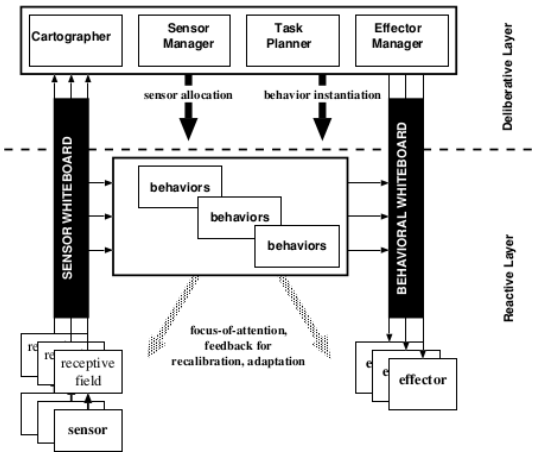
A partir de la década de los 90, comenzaron a aparecer numerosas arquitecturas híbridas de control que presentaban variantes con respecto a las ya existentes. En las arquitecturas híbridas, sus implementaciones se diferencian principalmente en la forma en que se separan las partes reactivas y deliberativas; la manera en que se asignan responsabilidades por parte de la porción deliberativa; y el mecanismo de emergencia del comportamiento global. Las arquitecturas Híbridas más típicas suelen incluir varios módulos funcionales comunes [2]:

- *Un agente secuenciador*, que determina el conjunto de comportamientos a usar en la realización de una sub-tarea, y su orden y condiciones de activación. Puede implementarse, por ejemplo con una *FSM*.
- *Un gestor de recursos*, que asigna recursos a los comportamientos e incluso selecciona entre diferentes planes o esquemas.
- *Un cartógrafo*, módulo que se ocupa de crear, almacenar, y mantener un modelo del entorno, y una representación de la información necesaria para llevar a cabo su tarea.
- *Un planificador de misiones*, que interactúa con los operadores humanos, transfiere sus órdenes a comandos de robot, y genera un plan adecuado a la misión encomendada.
- *Un módulo de monitorización y resolución de problemas*, que permite al robot comprobar el progreso de su operación.

Las arquitecturas híbridas se suelen encuadrar en una de entre tres posibles categorías:



(a) Arquitectura AuRA [32]



(b) Arquitectura SFX [33]

Figura 2.6: Ejemplos de arquitecturas Híbridas de Gestión

- Arquitecturas de Gestión (*managerial*), que subdividen la porción deliberativa en capas relacionadas con su alcance de control, de forma similar a como se hace en en un organigrama empresarial: a alto nivel estarían los agentes que establecen un plan global, que pasan a sus subordinados, los cuales se ocupan de concretar los aspectos del mismo, establecer los recursos necesarios, y dar las órdenes adecuadas a los trabajadores del nivel inferior, que en este caso serían los comportamientos reactivos. Cada capa de la arquitectura se comunica únicamente con sus vecinas, superior e inferior; además de indicar órdenes, es posible supervisar el desempeño de las capas inferiores y dar indicaciones adicionales. Ejemplos de esta arquitectura son *AuRA* (*Autonomous Robot Architecture*) [32] y *SFX* (*Sensor Fusion Effects*) [33] (figura 2.6).

- Arquitecturas de Jerarquía de Estados (*state-hierarchical*). Organizan sus actividades por el alcance temporal de la información o conocimiento del entorno mediante 3 capas —relacionadas con el pasado, presente, y futuro—. Hay una jerarquía de capas que operan sobre las capas inmediatamente inferiores, y tienen acceso a las salidas de dichas capas. Dentro de cada capa existen varios agentes que se interrelacionan para lograr el objetivo de cada capa. La arquitectura más característica de este tipo es la *3T* o de "tres hileras" [34] (figura 2.7), que es la empleada de forma predominante en los robots de la NASA. En esta arquitectura se establecen 3 capas, una deliberativa (*Planner*), otra reactiva (*Skill Manager*) y una que sirve de interfaz entra ambas (*Sequencer*). Esta última capa recibe los objetivos y estrategias de la capa deliberativa, y utiliza una técnica de planificación reactiva (*RAP*) [35], para la selección de un subconjunto de comportamientos primitivos reactivos (*skills*), que se ejecutan según la secuencia indicada por una red de tareas. Estos comportamientos o *skills* son mas similares a los de las arquitecturas *AuRA* o *SFX*, que a a los de las arquitecturas reactivas puras. Asociados a los *skills* se incluyen eventos (*events*) que sirven como puntos de verificación de la corrección en las acciones realizadas.

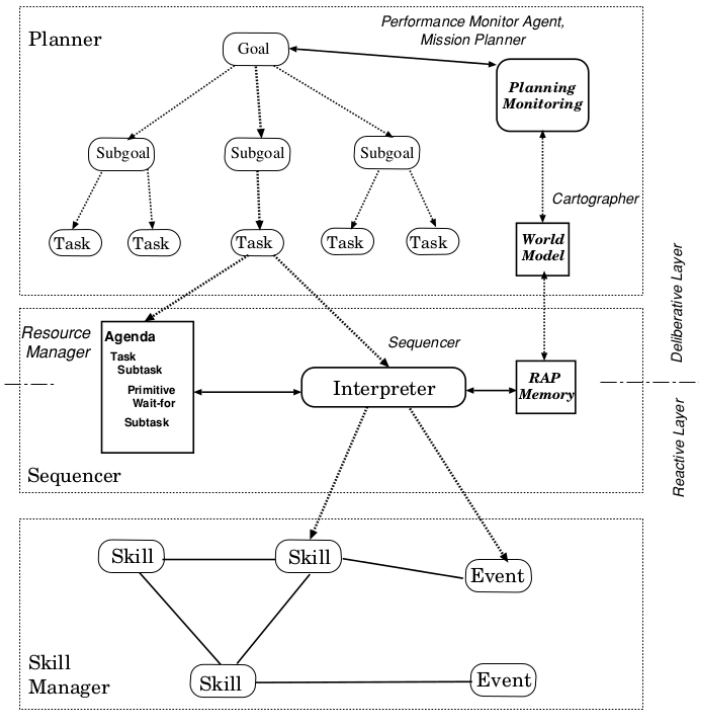
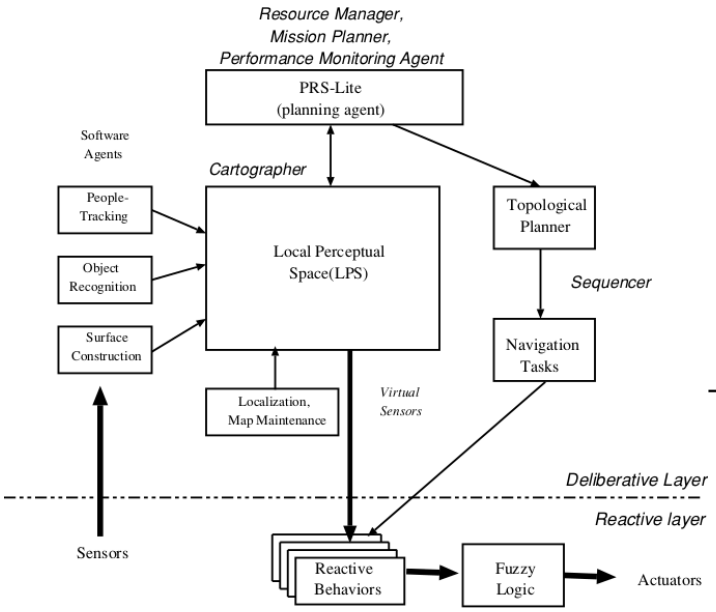


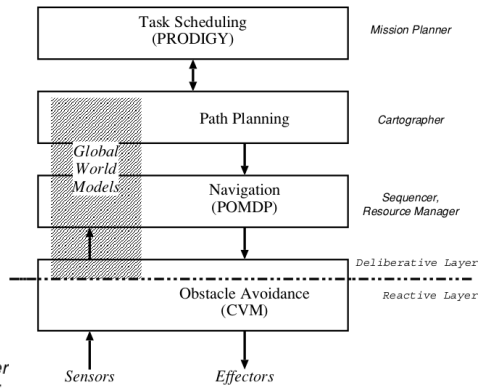
Figura 2.7: Arquitectura 3T [34]

- Arquitecturas orientadas a Modelos (*model-oriented*). Las dos anteriores categorías de arquitectura tenían una clara componente ascendente (*bottom-up*): partían de las arquitecturas reactivas y trataban de incorporar funcionalidades de más alto nivel. Sin embargo, existe otra categoría de arquitecturas que siguen una componente descendente (*top-down*): se centran en la creación de un modelo del entorno que, a la vez, funciona como un "sensor virtual", en la medida en que proporciona información a los comportamientos (o sus módulos equivalentes). Estos últimos no disponen de sensores específicos asociados, sino que toman su información de dicho modelo. Una ventaja importante de este modelo es la posibilidad de realizar una fusión de la información de los sensores a lo largo del tiempo, de forma que la información que se proporciona a los comportamientos de bajo nivel está mas estructurada, simplificada, y filtrada ante la posibilidad de errores e inconsistencias. Entre las arquitecturas más representativas de esta categoría destacamos *Saphira* [36], presente en la línea de robots Pioneer; y *TCA* [18]. *Saphira* busca establecer una coordinación, no solo en los sensores y actuadores del robot, sino también en el cumplimiento de sus objetivos a lo largo del tiempo: se debe mantener una coherencia entre los mismos, si se desea que el robot sea capaz de desempeñar correctamente su función. *Saphira* también incide en la necesidad de que el robot sea capaz de comunicarse para interactuar con los seres humanos, y en las dificultadas relacionadas con la búsqueda de un marco común de referencia para esa comunicación. En su implementación original, *Saphira* realiza la fusión de las salidas de los comportamientos mediante lógica difusa, obteniendo un resultado parecido al de los campos potenciales. Podemos ver un esquema de la arquitectura de *Saphira* en la figura 2.8a.

  En cuanto a *TCA* (figura 2.8b), no tiene comportamientos como tales, aunque las



(a) Arquitectura Saphira [36]



(b) Arquitectura TCA [18]

Figura 2.8: Ejemplos de arquitecturas Híbridas orientadas a Modelos

capas inferiores son muy similares a estos; la inteligencia se distribuye en capas y los errores en los módulos de nivel inferior se propagan hacia las capas superiores para encontrar una posible solución, de manera similar a lo que se propone en el modelo de inteligencia *Memoria Jerárquica Temporal —Hierarchical Temporal Memory (HTM)—* de Hawkins [37].

Hay que destacara que, a menudo, se encajan en esta categoría de Híbridas, aquellas arquitecturas que no encajan entre las Jerárquicas o Reactivas Puras, la gran mayoría emplean módulos o elementos que se pueden ajustar al concepto de comportamiento, aunque a menudo se usan otros nombres. También suele ser aceptada el establecimiento de capas de abstracción de la inteligencia tanto a nivel conceptual como temporal, existiendo capas de nivel superior de tipo deliberativo y capas de nivel inferior de tipo reactivo puros; lo que diferencia a unas arquitecturas híbridas de otras suele ser la existencia de capas intermedias de "acoplo" de las deliberativas y reactivas, y la forma de organizar, estructurar, y controlar cada una de las capas, así como las técnicas o métodos empleadas para su implementación (*ANN*, *CBR*, lógica difusa, reglas de decisión,etc...).

4. Comportamientos

Tras la revisión general a los paradigmas y arquitecturas existentes en *IA*, en este apartado se tratará con más detalle los aspectos relacionados con los *Comportamientos*, concepto que apareció por primera vez en las arquitecturas reactivas, y que también suele formar parte de las capas inferiores de las más modernas arquitecturas híbridas. La motivación de este apartado es el enfoque de la tesis a la implementación, a través de comportamientos, en las capas reactivas de la arquitectura híbrida propuesta, basada en el modelo *HTM*.

El concepto de *Comportamiento* básico aplicado a *IA*, aparece por primera vez en la tesis de Maja Mataric [23], si bien este trabajo se puede considerar heredero de los trabajos previos de Brooks a mediados de los 80 [26]. Los comportamientos permitirían establecer un modelo descentralizado de la inteligencia, en el que de podría obtener una funcionalidad compleja e inteligente a partir de interacciones locales, estructuradas en dichos comportamientos, y mediante el uso de reglas sencillas. Un comportamiento establecería unas leyes de conducta que aspirarían a lograr un determinado objetivo, cumpliendo una serie de restricciones. En este sentido, para muchos autores la *Robótica basada en Comportamientos* (*Behavior-Based Robotics (BBR)*) constituye una evolución de la implementación primitiva del paradigma reactivo.

4.1. Orígenes biológicos de los comportamientos: Agentes Inteligentes

Muchos investigadores en *IA* y en Robótica toman como modelo para sus trabajos, aspectos relacionados con ciencias biológico-humanas como la Psicología, o la Biología. Aunque existen ejemplos de logros tecnológicos que han divergido completamente de su equivalente biológico —siendo el más evidente el vuelo a reacción de los aviones, frente al aleteo de los pájaros— se pueden obtener importantes consecuencias observando la conducta de los seres vivos. Parece razonable acudir al modelo más perfecto que conocemos de la inteligencia, la Inteligencia Humana, como punto de partida, a pesar de lo mucho que nos queda todavía por conocer acerca de ésta. Pero esta aproximación no tiene porqué hacerse necesariamente desde un punto de vista“ mecanicista“ como hacen, por ejemplo, los ”conexionistas“ con sus modelos de *ANN*; en su lugar, se puede buscar una equivalencia funcional. Para conseguir esta equivalencia se define el concepto de *agente*, como sistema capaz de interactuar con el mundo, para obtener información y actuar sobre este, siguiendo unas reglas de conducta. Este concepto se puede aplicar tanto a un animal como un insecto o un perro, a una persona, pero también a seres artificiales. En última instancia. la única diferencia entre estas realizaciones de un agente radica en como se implementan y relacionan los diferentes sub-componentes del sistema. Así, por ejemplo en [38] se establece una correspondencia entre los diversos comportamientos-agentes asociados a la navegación de animales y robots, a través de una jerarquía de navegación. Los comportamientos animales se basan en establecer una correspondencia entre un conjunto de entradas sensoriales y unas acciones motoras que permitan realizar la tarea u objetivo asociado al comportamiento. En el campo de la Etología, se distinguen 3 tipos de comportamientos animales: *reflejos*, que están ”grabados“ en los circuitos neuronales, y generalmente se corresponden con comportamientos con necesidades de respuesta rápida; *reactivos*, que se aprenden inicialmente y se consolidan para ser ejecutados de forma no consciente; y *conscientes*, que se ejecutan de manera deliberativa. Nótese que el término ”reactivo“ tiene unas connotaciones distintas que las utilizadas generalmente en Robótica, que se corresponderían mas con ”reflejo“. En esta tesis utilizaremos la acepción etológica de ”reactivo“, puesto que los comportamientos que forman la base de nuestro modelo serán aprendidos.

Además de la identificación de los propios comportamientos en sí mismos, otros aspectos claves estudiados en la Etología son la forma en que se adquieren dichos

Figura 2.9: Definición gráfica de un comportamiento

comportamientos —cuando no son innatos— y como se seleccionan y coordinan conjuntos de comportamientos que se ejecutan simultáneamente.

4.1.1. Tipos de comportamientos según su adquisición

En cuanto a la adquisición de los comportamientos, se diferencian cuatro formas [39]:

- Comportamientos innatos de tipo reflejo, como el reflejo de succión de un recién nacido ante la sensación de hambre. En Robótica se corresponderían con comportamientos pre-programados.
- Secuencia de comportamientos innatos. Está compuesta por comportamientos similares al del punto anterior, pero que desencadenan la aparición de un nuevo comportamiento. Existe una especie de estado interno que va modificándose a medida que van ejecutándose los comportamientos de la secuencia como , por ejemplo, todos los pasos necesarios para que una avispa se aparee, construya un nido, y ponga huevos en el mismo, pasos que van sucediéndose según la combinación de las condiciones del entorno y ese estado interno. Este tipo de conductas se puede imitar fácilmente en Robótica usando *FSM*.
- Comportamientos innatos con memoria. Son comportamientos innatos que necesitan de una cierta "personalización". Un ejemplo es el comportamiento de las abejas jóvenes para aprender a reconocer la forma de su colmena: de forma innata vuelan alrededor de la colmena pero , al mismo tiempo. aprenden a reconocerla y pueden así encontrar su apertura de entrada. En Robótica se podrían implementar mediante algoritmos pre-programados con parámetros ajustables a través de la operación del robot.
- Comportamientos aprendidos, que son los más comunes en animales superiores como los mamíferos y primates, como podría ser el aprendizaje de caza en las crías de león. Este tipo de comportamiento suele ser más complejo, pero suele estar compuesto de comportamientos más simples (en el caso de la caza, por ejemplo, buscar a la presa, acecharla, rodearla,etc...), muchos de ellos innatos. En muchos casos el aprendizaje de estos comportamientos tiene por objeto determinar cuando desencadenar un comportamiento y que acciones llevar a cabo en el mismo.

4.1.2. Control y coordinación de comportamientos

En cuanto al control y coordinación de comportamientos, Lorenz y Tinbergen definieron [40, 41] el concepto de *mecanismo de liberación innata* (*Innate Releasing*

Mechanism (IRM)). En el *IRM* un comportamiento se activa en respuesta a un estímulo específico, que cumple una determinada condición (liberador o *releaser*) sin la cual el comportamiento no se ejecuta (figura 2.10). A diferencia de los comportamientos reflejos, no basta con que se da la entrada sensorial adecuada para que el comportamiento produzca una salida. A menudo el estímulo liberador procede de un preprocesado de la información sensorial, o de aspectos más abstractos de la misma. Así mismo, puede ser necesaria una combinación de varios estímulos liberadores, tanto internos (motivaciones como el hambre o el miedo), como externos (del entorno, como la presencia de comida o de un predador), para activar el comportamiento. La activación de varios comportamientos se puede producir en paralelo, siempre que se cumplan los estímulos liberadores correspondientes, dando la impresión de un comportamiento conjunto más complejo.

Algunos comportamientos pueden entrar en conflicto y/o afectar a la ejecución de otros. Se pueden dar situaciones de *equilibrio* entre los comportamientos que lleven a que ambos se ejecuten simultáneamente —o no lleguen a ejecutarse del todo!—; de *dominancia*, por el que solo uno de los comportamientos llega a ejecutarse; o incluso de *cancelación*, en cuyo caso, los comportamientos se anulan, y puede llegar a ejecutarse un tercer comportamiento completamente distinto.

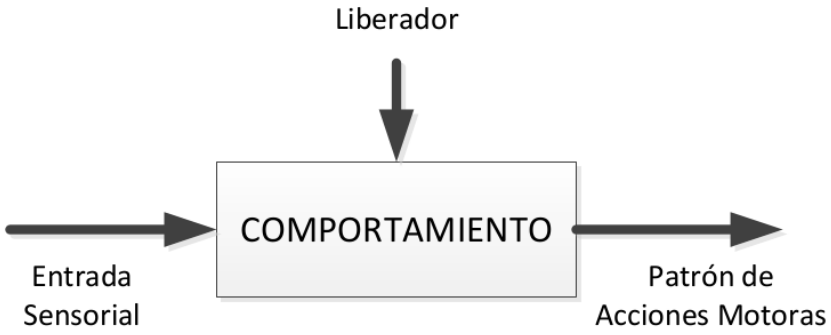


Figura 2.10: Mecanismo de liberación de comportamientos

4.1.3. Recolección de información: tipos de percepción

Otro aspecto clave de los comportamientos —refiriéndonos aún a los animales— es el relacionado con la forma en que se obtiene la información sensorial y como afecta esta información a los comportamientos: en este sentido, los agentes inteligentes participan de un ciclo de acción-percepción en el que la propia actuación del agente provoca cambios en el entorno y, por tanto, en la percepción que tiene el agente del mismo (figura 2.11). Este proceso puede llevar a una estimación de posibles errores o consecuencias de las acciones, que pueden modificar futuras actuaciones; o a predicciones de lo que se espera encontrar en futuras percepciones del mundo.

Aunque la percepción del mundo es algo continuo y, a menudo, automático, a veces el agente necesita dirigir sus acciones a la obtención de una determinada información en concreto, necesaria u orientada a la realización de sus objetivos; es lo que se conoce como *percepción activa*. Las acciones o comportamientos se activarán cuando se detecten los estímulos liberadores asociados a los mismos. Pero las percepciones, además de activar

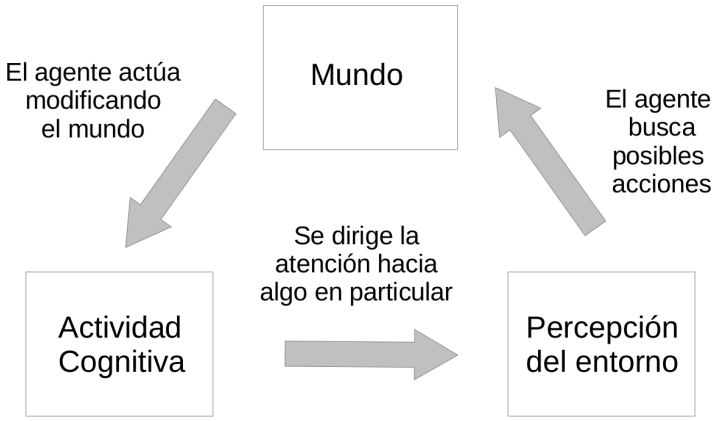


Figura 2.11: Ciclo Percepción-Acción

los comportamientos, también proporcionan información necesaria para su realización (percepción orientada a acción).

Las percepciones del mundo nos permiten obtener información de varias maneras: por una parte,existe una *percepción directa* que permite reconocer potencialidades (*affordances* [42]) del entorno para la realización de una acción, sin necesidad de memoria, inferencia, o interpretación. Un ejemplo es nuestra conducta de esquivar un objeto lanzado por sorpresa contra nuestra cara: en este caso, el flujo óptico del objeto nos permite establecer el tiempo de contacto y esquivarlo a tiempo, sin que existan una etapa consciente deliberativa que determine la acción a tomar. Por otro lado, existen otras formas mas complejas de percepción en los animales: para reconocer instancias particulares de un objeto —por ejemplo, el coche de uno mismo— si que es necesaria la memoria; en general, para la resolución de tareas más complejas que impliquen inferencia o interpretación, no basta con la percepción directa, sino que es necesario un modelo interno, y una percepción descendente [43]. Este es un aspecto importante a la hora de diseñar los comportamientos de un robot: si el comportamiento se puede diseñar mediante percepción directa únicamente, su arquitectura de *IA* y programación resultará más sencilla que si requiere reconocimiento.

Todos los conceptos analizados en los apartados anteriores,nacidos en el ámbito de la Etología, se pueden trasladar al ámbito de la Robótica, dando lugar a la aparición de arquitecturas de control y modelos de inteligencia robóticos basados en comportamientos (*BBR*). En el siguiente apartado se tratarán los aspectos fundamentales para conseguir esta trasposición de conceptos, muchos de los cuales se tendrán en cuenta para el diseño de la arquitectura que es objeto de estudio en esta tesis.

4.2. Robótica basada en comportamientos

BBR propone una implementación de la *IA* que difiere considerablemente de los paradigmas más clásicos de *IA*, en los cuales ya hemos visto que suele existir un flujo de información secuencial: primero se obtiene información del entorno a través del ”sensado“ (Sense); a continuación se construye un modelo del mundo, por lo general bastante complejo, a partir del cual se intenta evaluar el efecto que tendría la realización de las posibles acciones sobre el entorno (Plan); y finalmente se decide la acción o acciones a tomar, y se ejecutan (Act). Este tipo de modelos difiere

bastante de la forma de trabajar del cerebro humano que, como veremos en un capítulo posterior, tiene una orientación más distribuida. Por ello, el paradigma *SPA* presenta una respuesta bastante lenta y difícil de aplicar en entornos dinámicos y poco estructurados. Por contra,*BBR* propone una alternativa a la *IA* clásica, mediante la construcción de comportamientos inteligentes de forma ascendente (*bottom-up*), y partiendo de comportamientos simples que se ejecutan simultáneamente en el cerebro robótico, proporcionando sugerencias acerca de las posibles acciones que podría llevar a cabo un robot ante una determinada situación (figura 2.12). Mientras que en el diseño *top-down* se parte del sistema completo, y se descompone ésta en sub-sistemas cada vez más pequeños y sencillos, en la filosofía *bottom-up* se trabaja con porciones aisladas del problema o sistema o resolver, que se implementan mediante componentes simples conectados en red. La visión *top-down* suele hacer énfasis en el análisis formal y la caracterización de requisitos, buscando que la abstracción esconda los detalles de bajo nivel en el alto nivel. Por contra, el diseño *bottom-up* se apoya en la experimentación y la adaptación para realizar un mapeo directo entre percepciones locales a un módulo y acción como respuesta del mismo. Un aspecto clave es la consecución de comportamientos mas complejos a través de la combinación de los simples. Aunque *BBR* puede parecer exactamente igual a las arquitecturas reactivas clásicas —o a las porciones reactivas de arquitecturas híbridas como la *3T*—, las redes de comportamientos pueden tener un estado, y establecer una representación del problema, algo que no es posible en los esquemas reactivos puros. Esta característica permite expandir enormemente las capacidades de expresión y aprendizaje de estos sistemas. Además, en las arquitecturas híbridas se establece una clara diferenciación entre la operación a corto plazo de las capas reactivas, y a largo plazo de las capas deliberativas, mientras que en *BBR* se trata de usar una representación y una escala temporal uniforme, buscando que cada parte se acomode en tiempo real a las necesidades de otras partes del sistema [44].

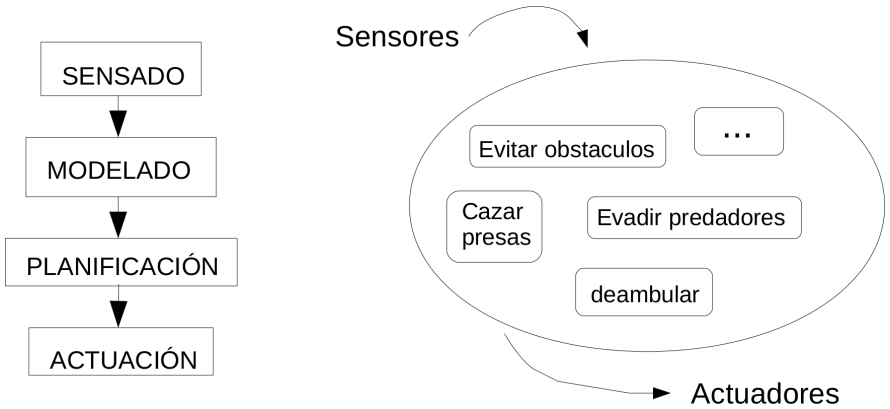


Figura 2.12: Flujo de información en la IA clásica (izquierda) y en BBR (derecha)

En este ámbito, un comportamiento se define a través de una secuencia de acciones que se ejecutan en un orden determinado y que permiten alcanzar un determinado objetivo. Este tipo de comportamiento inteligente no necesita de razonamiento o deliberación, y está mas relacionado con la capacidad de sobrevivir y luchar por alcanzar una serie de objetivos mientras se desenvuelve en un entorno abierto y no estructurado. Los robots basados en comportamientos presentan, por lo general, una serie de características comunes:

- Inicialmente presentan o se les proporciona comportamientos muy básicos, como la evitación de obstáculos o la búsqueda de fuentes de energía, que son indispensables para poder desenvolverse en un entorno no estructurado.
- Algunos de estos comportamientos van a operar de forma simultánea, de forma que la acción final ejecutada por el robot va a implementarse a partir de las sugerencias obtenidas de los diferentes comportamientos, bien por elección de una de ellas, bien por composición, o utilizando técnicas más complejas;
- *BBR*, como arquitectura de control, suele estar más asociado a la implementación de robots autónomos, que se pueden mover libremente sin supervisión humana.
- En *BBR* no se construyen modelos complejos y abstractos del entorno; en su lugar, los robots operan en el mundo real y muchos de sus comportamientos son reactivos, si bien también pueden existir estados internos que proporcionen motivaciones al robot, o una memoria a corto plazo. Pese a esta asunción de contextualización (*situatedness*) del robot, suele ser aconsejable usar simulaciones en la implementación preliminar de los comportamientos, si bien también es necesario probar posteriormente el funcionamiento en robots reales.

El cerebro de un robot con arquitectura *BBR* está constituido por un conjunto de comportamientos básicos, el *repertorio de comportamientos*, y su construcción suele realizarse a través de un proceso de dos etapas: en primer lugar, se deben definir o evolucionar de alguna manera los comportamientos individuales; y, una vez realizada la etapa anterior, se debe diseñar un sistema que seleccione que comportamiento o comportamientos usar en una determinada situación. A medida que aumenta la complejidad de la aplicación, crece también la importancia del diseño de un mecanismo de organización adecuado. En los siguientes apartados analizaremos aspectos relacionados con cada una de las dos etapas referidas.

4.3. Generación de comportamientos

Cualquier robot con arquitectura *BBR* debe contar con comportamientos fundamentales para la supervivencia, como por ejemplo los que le lleven a evitar daños en su estructura, y a mantener un nivel de energía suficiente para su operación. Así mismo puede ser importante, especialmente en robots de mayor tamaño, incorporar comportamientos que eviten dañar a las personas, incluso por encima de dañarse el robot mismo. En cierto sentido, la arquitectura *BBR* trata con comportamientos relacionados con las *leyes de la Robótica* de Isaac Asimov [45].

Antes de diseñar un comportamiento es necesario elegir un marco general de implementación, o *filosofía de diseño*. Existe una gran variedad de posibilidades en cuanto a métodos y modelos para esta implementación, aunque en la práctica la elección se suele realizar entre ciertos tipos de métodos o herramientas, en función de la orientación de la arquitectura o filosofía elegida.

4.3.1. Comportamientos cableados

Un modelo muy sencillo para la generación de comportamientos es el seguido por los *Vehículos de Braitenberg* [46], criaturas artificiales construidas mediante un mapeo

directo sensor-actuador, de forma que la respuesta motora pueda estar determinada, por ejemplo, por una ecuación diferencial que tenga como entrada la intensidad de los estímulos sensoriales. Alguno de los parámetros de la ecuación se puede modificar experimentalmente para ajustar las características del comportamiento. Los robots con este tipo de comportamientos son sencillos, pero también limitados. Los algoritmos de mapeo empleados son simples, y no van más allá de una expresión matemática que relacione los parámetros de entrada y salida del sistema. Una variante más avanzada del modelo anterior se consigue mediante una programación "cableada" de los comportamientos [47] [48]. Esta tarea resulta frecuentemente muy complicada ya que las características de operación del robot en entornos reales ruidosos y poco estructurados, hacen muy difícil predecir y planear la diversidad de situaciones que el robot se pueda llegar a encontrar. Además, no siempre está clara la división de funciones asignadas a cada comportamiento: la navegación directa hacia un objetivo podría combinarse con la evitación de obstáculos en un mismo comportamiento, o codificarse cada una en un comportamiento separado. Por ejemplo, en el diseño de un comportamiento para "deambular" por un escenario con vistas a la exploración o vigilancia del mismo, se establecerían unas reglas que decidiesen el giro del robot ante la aparición de unos determinados umbrales procedentes de los sensores. En general, para el diseño de comportamientos "cableados", se suele emplear reglas de decisión [49], o Autómatas Finitos (*FSM*) [26] [50]. Esta aproximación cableada adolece, no obstante, de importantes inconvenientes: a la dificultad de diseñar los algoritmos en comportamientos complejos, se añade la incapacidad para adaptar los comportamientos ante posibles cambios o errores en la previsión del escenario en el que se va a desenvolver el robot. Por este motivo, la mayoría de las filosofías de diseño de comportamiento con cierto grado de complejidad, suelen incluir, actualmente, cierta componente de evolución o aprendizaje a partir de la operación del robot en el escenario del problema [51] [52].

4.3.2. Comportamientos aprendidos

Dentro de las opciones de diseño de comportamientos adaptables o evolutivos, existen multitud de variantes en función de la herramientas de *IA* empleada para dotar de la capacidad de adaptación/evolución al comportamiento. En el diseño de comportamientos reactivos simples se puede emplear *Aprendizaje por refuerzo —Reinforced Learning (RL)—* [53] [54] [55]; *Algoritmos Genéticos —Genetic Algorithms (GA)—* [56] a menudo en combinación con redes neuronales (*ANN*s) [57] [58]; *ANN*s de forma independiente [59] [60] [61]; o *CBR* [62] [3] [63].

Aunque las técnicas de aprendizaje por refuerzo han dado muy buenos resultados en varios experimentos, para su uso correcto se necesita un gran dominio, no solo de los aspectos teóricos propios de *RL*, si no también del campo de la aplicación en cuestión. En [53] se necesita tener muy bien caracterizado todos los aspectos de la dinámica del movimiento del brazo robot, así como los aspectos clave que determinan la trayectoria de la pelota al ser golpeada. Solo así es posible determinar sobre que parámetros hay que actuar o ajustar para conseguir el comportamiento correcto buscado. En cierto sentido, este tipo de métodos representa una versión muy mejorado de los comportamientos cableados, pero con la capacidad de adaptarse ante incertidumbres, errores, y variantes de las situaciones conocidas. En cuanto a los *GA* y la programación genética en general, en su aplicación en el campo del diseño de comportamientos, de nuevo nos encontramos

con el problema de tener que ajustar un gran número de parámetros y funciones hasta conseguir un buen funcionamiento, como la forma de los datos, los operadores genéticos, o los algoritmos de ajuste [64]. Estos parámetros permitirían optimizar el método para la resolución de un problema particular, pero su naturaleza en cuanto a dependencia del problema tratado, dificulta medir el rendimiento del método en general, y obliga a emplear grandes cantidades de tiempo en la fase de ajuste paramétrico. Adicionalmente, hay cierta propensión a caer en mínimos locales lo cual podría hacer que el comportamiento diseñado no llegase a ser el óptimo, si bien seguramente se podría considerar "suficientemente bueno". Frente a los métodos ya mencionados, las redes neuronales, además de parecerse estructuralmente a los elementos biológicos que determinan el aprendizaje en los seres vivos, no suelen necesitar un conocimiento tan intensivo del proceso que se desea modelar. Suele bastar con tener cierta noción de cuales son las variables que influyen en el mismo para poder introducirlas como entradas a la red, y de que respuesta se asocia a cada situación representada por esas entradas. A partir de aquí, la evolución de la red neuronal es transparente a los usuarios hasta converger —en la mayoría de los casos— hasta una solución válida. Un aspecto importante en el uso de *ANN*s es la elección de la estructura de la red y las características de las neuronas que la integran: aspectos como el número de capas, el número de neuronas por capas, las funciones de evolución de las conexiones entre neuronas y los niveles de realimentación entre capas, resultan clave en las capacidades de aprendizaje de la red. De hecho, uno de las críticas más comunes a la *ANN*s es que las estructuras mas frecuentemente elegidas suelen estar muy alejadas, por su sencillez, de la estructura biológica real del cerebro humano, con lo cual se considera que nunca serán capaces de alcanzar los niveles propios de los seres vivos [1]. Como importante problema adicional, las *ANN*s ofrecen poca información de depuración acerca del proceso de aprendizaje [65]. Por este motivo, la implementación de un comportamiento medianamente complejo mediante *ANN*s resultaría muy abstruso en la medida que no tendríamos conocimiento de "que está pasando" en la red y solo podríamos determinar el posible éxito o fracaso a través de la puesta en marcha del mismo; por otra parte el diseño "cableado" o "a mano" de un comportamiento complejo a sufriría el peligro de olvidar alguna consideración que hiciese que el sistema funcionase correctamente en la mayoría de las ocasiones pero no en todas: es muy difícil tener en cuenta, a priori, todas las alternativas e imprevistos asociados a comportamientos complejos. Finalmente, los sistemas *CBR*, aunque estructuralmente lejanos a la implementación biológica del cerebro humano, si que parecen un buen candidato en cuando a que se adaptan muy bien a las características de funcionamiento de modelos de inteligencia como el de memoria- predicción: en ambos casos la inteligencia, —como motor de definición del comportamiento— se basa en el almacenamiento de la experiencia en una forma canónica y generalizable, con diferentes niveles de abstracción y la capacidad —al menos teórica— de combinar los conceptos almacenados para construir otros mas elaborados o de más alto nivel, en una forma muy similar a la "emergencia" que constituye uno de los aspectos fundamentales en las arquitecturas *BBR*. Por estos motivos esta será la opción elegida en la presente tesis como herramienta o método de implementación de la arquitectura reactiva propuesta. En el capítulo 3 veremos una revisión más detallada de las técnicas y métodos de *IA* existentes y determinaremos cuales pueden ser más útiles para nuestra filosofía de diseño de comportamientos.

4.4. Repertorio de comportamientos

Una vez elegido el marco de implementación, la elección de los comportamientos particulares a implementar está muy relacionada con el dominio del problema, es decir, con el objetivo o aplicación para la que se ha diseñado el robot. En este sentido, es preferible diseñar comportamientos simples, aunque el número final de estos sea mayor, y procurar que no haya solapamientos o interacciones importantes entre los comportamientos diseñados, ya que esto último dificultará la selección o composición de las acciones que deberá ejecutar el robot a partir de las sugerencias de los comportamientos individuales. Esta tarea a menudo no es sencilla y se apoya en la experiencia y conocimientos del diseñador del sistema.

En [23] se analizan los aspectos claves para la selección y evaluación de un repertorio de comportamientos básicos. Si bien en dicho trabajo se particulariza el estudio para un comportamiento conjunto de un grupo de robots, hay que destacar que se puede hacer una analogía con el problema que estamos tratando, ya que en ambos casos se trata de establecer una coordinación en sistemas multiagente: en [23] referida a grupos de robots que colaboran para lograr un objetivo común; y en la tesis actual en grupos de comportamientos básicos que deben coordinarse y combinarse para lograr un comportamiento más complejo. En el trabajo citado se indican dos criterios básicos para elegir los comportamientos básicos que deben formar parte del robot:

- *Necesidad*. Un comportamiento básico será necesario si permite lograr un objetivo requerido por el robot para el cumplimiento de su tarea o de alguna de sus subtareas, objetivo que no puede obtenerse mediante otros comportamientos básicos o sus combinaciones.

- *Suficiencia*. El repertorio de componentes básicos será suficiente para realizar *las tareas asignadas al robot en el dominio de su aplicación*, si no se necesitan comportamientos básicos adicionales para ello. A través de los comportamientos básicos y sus combinaciones debe ser posible generar todos los comportamientos de alto nivel requeridos.

Se suelen añadir también unos criterios adicionales, especialmente en el casos de que los comportamientos se diseñen "a mano" y no sean implementaciones de comportamientos observados en sistemas ya existentes:

- *Simplicidad*. El comportamiento básico se debe implementar de la forma más simple posible.

- *Localidad*. Dentro del marco de trabajo, el comportamiento debe guiarse por reglas locales, utilizando información sensorial disponible localmente.

- *Corrección*. En el modelo en que sea testado, el comportamiento debe acercarse o mantener el objetivo para el que se creó, dentro del conjunto de restricciones para el que fue diseñado.

- *Estabilidad*. Se debe procurar evitar que el comportamiento básico sea sensible a perturbaciones en las condiciones externas para las que fue diseñado.

- *Repetibilidad*. El comportamiento básico se debe ejecutar de acuerdo a las especificaciones en cada prueba o ensayo, dentro de unas condiciones y márgenes de error razonables.

- *Robustez*. El rendimiento del comportamiento no se debe degradar significativamente dentro de unos límites de error y ruido en el plano sensorial y motor/actuador.

- *Escalabilidad*. El comportamiento se debe escalar correctamente independientemente del número de comportamientos totales del repertorio.

A pesar de todas estas recomendaciones, es complejo establecer una métrica exacta para la selección de un repertorio óptimo de comportamientos básicos, ya que en esta selección hay una alta dependencia de la aplicación o tarea que debe resolver el robot. Por ello, muchos de los aspectos de corrección de los métodos y algoritmos propuestos son difíciles de comparar con otros, incluso si se trabaja en el mismo dominio y/o problema a resolver. En [66] se establece una métrica del grado de uso de los comportamientos de un robot y de su rendimiento, acorde al propósito del comportamiento; este mecanismo puede permitir analizar las interacciones entre comportamientos y ayudar a la selección del más adecuado en cada momento. En [67], se establece una métrica basada en parámetros similares a los mencionados en el párrafo anterior, para un comportamiento a nivel global del robot, y se analizan las variaciones en la misma al realizar modificaciones en comportamientos individuales o al añadir o eliminar comportamientos. En [68] se estudia la generación de comportamientos en un robot para la resolución de un juego mediante aprendizaje por demostración; la evaluación de la bondad de los comportamientos generados se realiza mediante una "condición del Mago de Oz", es decir, comparando el comportamiento que realizaría un operador humano con ese mismo robot en circunstancias similares con el desarrollado por el robot. Por tanto, se observa como los criterios de evaluación para discriminar si la implementación de un comportamiento es o no correcta tampoco están definidos de forma amplia y, de nuevo, deben ser definidos de forma empírica por el diseñador del experimento; en este sentido se recomienda establecer de antemano unos criterios que permitan definir la bondad del comportamiento en la realización de su objetivo.

4.5. Organización de comportamientos

Como ya hemos visto, la mayoría de las arquitecturas *BBR* con cierta complejidad suelen incluir todo un repertorio de comportamientos de entre los cuales hay que elegir la activación o combinación de alguno o algunos de forma dinámica en función de las circunstancias del momento. Por ello, un aspecto central de *BBR* es, precisamente, la organización de comportamientos, también conocida como *arbitración de comportamientos* o *selección de acciones*, que trata de decidir que comportamiento o comportamientos activar en un momento dado. Este aspecto es especialmente relevante para los comportamientos motores de alto nivel asociados, por ejemplo, a la locomoción del robot, pero también para otros comportamientos no-motores de tipo cognitivo.

Existe una gran variedad de métodos para la organización de comportamientos, si bien se suelen distinguir dos categorías principales, basados en el procedimiento que usan para determinar la participación de unos u otros comportamientos en la respuesta del robot: métodos de *arbitración* [69] y métodos *cooperativos* [70] [71] [72]. En los métodos de arbitración/selección solo hay activo un comportamiento en cada momento, y la selección del mismo es una función de la información actual leída por los sensores y del estado interno del robot; en cambio, en los métodos cooperativos,

la acción realizada por el robot es una combinación de las acciones sugeridas por los distintos comportamientos ya sea de forma simultánea, mediante una aportación ponderada de cada uno de ellos, o por alternancia temporal de los mismos. También se consideran sistemas en los que se mezclan ambas aproximaciones, de forma que se incluyen condiciones de exclusión mutua de comportamientos dentro de capas que combinan los efectos de varios de estos comportamientos básicos.

4.5.1. Selección exclusiva de comportamientos. Método de la función de utilidad

En algunos sistemas se definen comportamientos que se consideran mutuamente exclusivos, en el sentido de que actúan sobre los mismos elementos motores y actuadores generando ordenes incompatibles para los mismos. Se hace patente la necesidad de establecer de alguna forma cual de ellos se va a seleccionar en cada momento. Este problema ha sido ampliamente estudiado en el campo de la Etología, a partir del cual podemos obtener interesantes conclusiones para el diseño de arquitecturas robóticas basadas en comportamientos. Veamos el ejemplo de un animal que debe decidir entre 3 comportamientos incompatibles de "beber", "comer", o "huir". Existen diversos factores que podrían impulsar la ejecución de alguno de ellos sobre el resto, algunos intrínsecos, como los niveles de energía del animal, y otros dependientes del entorno, como la presencia de predadores, o de comida: la magnitud de estos estímulos permitiría aumentar o reducir la predisposición a la ejecución de un determinado comportamiento. Un aspecto fundamental es que los criterios de selección deben tener una componente de aprendizaje a partir de la experiencia, ya que el comportamiento se ejecutará dentro de un entorno dinámico y cambiante.

En una arquitectura completa a todos los niveles, se establecería una jerarquía de selección a diferentes niveles de abstracción, de forma que la selección de cada uno de los niveles desencadena un nuevo mecanismo de selección en niveles inferiores, y así sucesivamente. Los niveles inferiores estarían relacionados con la actuación directa sobre los elementos motores y actuadores del sistema.

Entre las arquitecturas de selección exclusiva de comportamientos podemos destacar:

- La arquitectura de subsunción [26] ya comentada en apartados anteriores, utiliza un esquema de prioridades fijas en el que los comportamientos de capas superiores pueden inhibir las respuestas de las de capas inferiores, sustituyéndolas por las propias. Los aspectos mas complejos en esta arquitectura se relacionan con el establecimiento de la prioridades. Los criterios de selección suelen estar asociados a máquinas de estado y reglas de decisión, y a la aparición de señales activadores (*releasers*) de posibles comportamientos implicados.

- También existen arquitecturas de selección en las que todos los comportamientos competidores están interconectados con el objeto de enviarse señales inhibitorias recíprocas, además de señales excitadoras con los recursos compartidos [73]. Así, un incremento de actividad de un comportamiento aumenta la inhibición del resto de de comportamientos y también la acción inhibidora de éstos hasta que, finalmente, alguno de los comportamientos alcanza el control sobre el sistema. En este caso la dificultad estriba en definir la potencia de las señales excitadoras e

inhibitorias, y estudiar su dinámica de variación por la interacción de las mismas. Como inconveniente adicional, la arquitectura tiene problemas de escalabilidad ya que la inclusión de nuevos comportamientos en el sistema exige el ajuste de tantas conexiones inhibitorias como adicionales como comportamientos existentes en el sistema y, posiblemente, un re-balanceo de la dinámica de interacción entre los mismos. Hay que destacar que, si bien este tipo de arquitectura distribuida puede permitir la selección de comportamientos en un estilo *"winner-take-all"*, también sería posible aprovechar la dinámica asociada a los cambios constantes debido a la realimentación y a la variación de los estímulos de entrada para obtener comportamientos emergentes con mecanismos similares a los de organización de comportamientos por combinación.

- En contraste con la opción anterior, otras arquitecturas [74] [75] utilizan mecanismos de selección centralizados en los que un módulo central están conectado con todos los comportamientos del sistema reduciendo el número de conexiones totales, lo cual favorece la escalabilidad y modularidad del sistema. El módulo central actuaría de árbitro y selector del comportamiento ganador, y debería recibir del resto de comportamientos la información adecuada que le permitiese establecer un criterio de decisión. Esta información estaría únicamente relacionada con los aspectos de coordinación de los comportamientos del sistema por lo que de facilitaría el diseño del elemento de control. Como variante, en [76], el módulo central de arbitraje recibe información global del sistema, además de desde los comportamientos, y realiza la selección mediante reglas de Lógica Difusa.

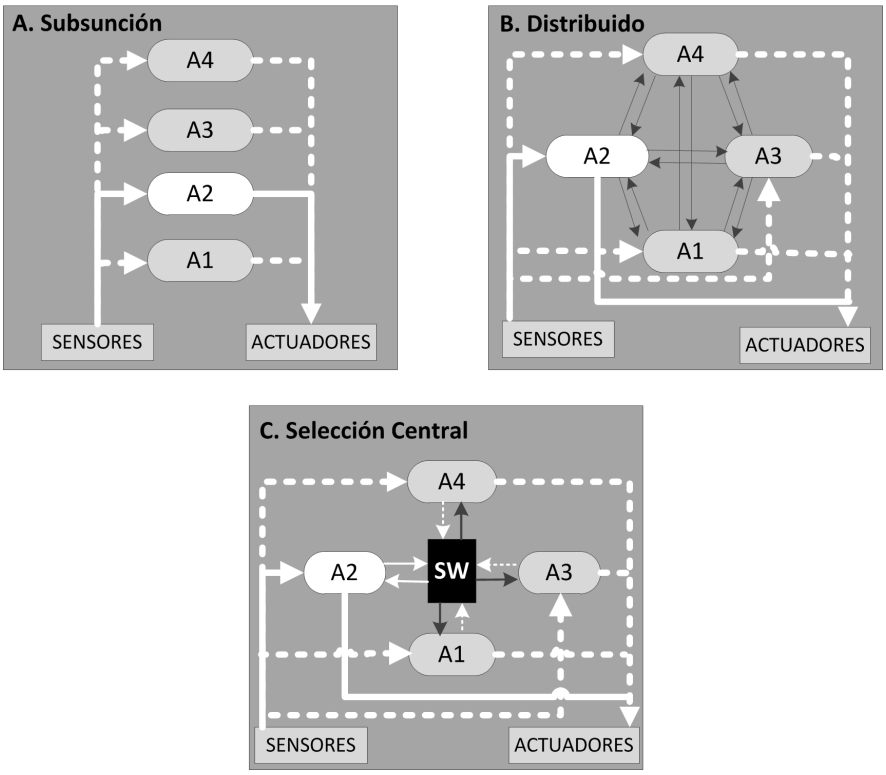


Figura 2.13: Arquitecturas de selección exclusiva de comportamientos

También es posible combinar alguna de las arquitecturas anteriores: en [77] se combinan aspectos de arquitecturas de selección distribuidas y centralizadas, para explicar el funcionamiento del ganglio basal como un componente importante en la arbitración o selección de comportamientos en animales vertebrados. En la figura 2.13 podemos ver un modelo conceptual del sistema de selección propuesto para el ganglio basal, para un ejemplo de control de recursos motores compartidas por 3 comportamientos o sistemas de comandos, denominados "canales" en la imagen. Observamos como cada uno de estos canales genera señales excitadoras(gris claro) e inhibidoras (gris oscuro) hacia los recursos motores y el elemento central de arbitraje -el ganglio basal-, el cual genera a su vez nuevas señales de inhibición o excitación hacia los canales de forma que, finalmente, se selecciona un único canal o comportamiento para la ejecución de las acciones motoras. Este mismo esquema podría adaptarse a métodos de combinación emergente de comportamientos eliminando el paso final de inhibición de los "perdedores" y permitiendo una aportación de cada uno de ellos proporcional al nivel de la señal enviada desde el elemento central.

Función de utilidad

Un aspecto clave en las arquitecturas de selección exclusiva presentadas es establecer un criterio o medida que determine la fuerza excitadora o inhibidora de cada uno de los comportamientos intervinientes en la selección. Para ello se puede acudir al concepto de *Utilidad* [78]. La Utilidad nos proporciona un medio de sopesar el resultado de diferentes situaciones, comparando unas con otras, para decidir finalmente la acción a tomar. Von Neumann y Morgenstein demostraron que, bajo ciertas condiciones [79], existe siempre una *función de utilidad* que nos permite mapear los miembros c_i de un conjunto de posibles resultados a un valor numérico $u(c_i)$, la *utilidad* de C_i, que cumple las siguientes condiciones:

1. $u(c_1) > u(c_2)$ si y solo sí la persona o el criterio prefiere c_1 a c_2.
2. u es una transformación *afín*, es decir:

$$u(p \cdot c_1 + (1-p) \cdot c_2) = p \cdot u(c_1) + (1-p) \cdot u(c_2), \tag{2.1}$$

   para cualquier valor de $p \in [0, 1]$.

Para que la función de utilidad exista, se deben de cumplir los siguientes axiomas:

Axioma 1
Ordenación. Dados dos posibles resultados, c_1 y c_2, un individuo puede decidir de forma consistente acerca de sus preferencias entre ambos, es decir, si prefiere c_1 a c_2 ($c_1 > c_2$) ; si por el contrario prefiere c_2 a c_1; o si la elección le resulta indiferente (c_1 c_2).

Axioma 2
Transitividad. Si $c_1 \geq c_2$ y $c_2 \geq c_3$, se debe cumplir que $c_1 \geq c_3$.

Axioma 3
Axioma de Arquímedes. Si $c_1 \geq c_2 \geq c_3$, $\exists p \in [0, 1]$ tal que $pc_1 + (1-p)c_3 > c_2$ y $\exists q \in [0, 1]$ tal que $c_2 > qc_1 + (1-q)c_3$

Axioma 4
Independencia. Para todo c_1, c_2, c_3, $c_1 \geq c_2$ *si y solo si* $pc_1 + (1-p)c_3 > pc_2 + (1-p)c_3$ $\forall p \in [0,1]$.

Hay que destacar que no hay un único conjunto de preferencias válido para todas las personas ante un problema; ante una determinada situación, una persona podría preferir una consecuencia c_1 frente a otra c_2, mientras que otra persona podría pensar al contrario. La función de utilidad lo que proporciona es una manera de sopesar las situaciones, comparándolas entre sí. En el caso de que exista cierta incertidumbre sobre las consecuencias que puedan derivarse de una situación se puede acudir al concepto de *Utilidad esperada*, $U(c)$ de una consecuencia promedio $c = p_1c_1 + p_2c_2 + ... + p_nc_n$ donde p_k, la probabilidad de una consecuencia c_k se obtiene a partir de:

$$U(c) = \sum_k p_k u(c_k), \tag{2.2}$$

Esto, unido a los axiomas anteriores, nos permitiría estimar una utilidad esperada de una *consecuencia mixta*, c_m que incluyese las probabilidades de las diferentes consecuencias ante una situación.

En la organización de comportamientos basada en utilidad, a cada comportamiento del repertorio, B_i, se le asigna una función de utilidad U_i, dependiente de ciertas variables de estado del robot. Estas variables pueden estar relacionadas con las lecturas de los sensores —se les denomina variables externas, s—; variables físicas internas, p, como pudiera ser el nivel de energía del robot; y variables abstractas internas, x, que serían las equivalentes a las *hormonas* en los sistemas biológicos:

$$U_i = U_i(s_1, ...s_{n_e}, p_1, ...p_{n_p}, x_1, ...x_{n_i}) \tag{2.3}$$

Aunque n_e, n_p y n_i denotarían el número total de variables de cada tipo, no es indispensable que la función de utilidad de cada comportamiento incluya todas las variables: solo aquellas que resultan relevantes en la medida de la utilidad del comportamiento.

En este método se suele establecer una división de comportamientos en dos categorías: comportamientos de Tarea, que están orientados a acercarse al objetivo del robot; y comportamientos auxiliares, que no modifican la eficiencia del robot hacia su tarea, pero son indispensables para su funcionamiento.

A cada comportamiento se le asocia una variable de *tiempo de comportamiento*, t_i. Esta variable se pone a 0 cuando un comportamiento se activa, permite medir el tiempo que permanece activado éste, y afecta a la función de utilidad a través de las variables abstractas internas . En intervalos regulares de tiempo se estiman las funciones de utilidad de los diferentes comportamientos, y se selecciona el comportamiento con mayor valor de utilidad. El aspecto clave de este método se centra en la elección de la función de las variables asociadas al comportamiento y de la forma de la función de utilidad. Para ello se suelen usar a menudo aproximaciones polinómicas, donde los parámetros se obtienen mediante técnicas de computación evolutiva [80, 81] como los algoritmos genéticos de los que se hablará en el capítulo 3.

Cuando los comportamientos básicos se ordenan según una jerarquía o un comportamiento incluye varios posibles sub-comportamientos, se realiza una comparación de valores de utilidad a cada nivel, empezando por los niveles superiores o más abstractos.

En la arquitectura que propondremos en el capítulo 5 consideraremos el uso de funciones de utilidad para medir la bondad de la operación del robot en la ejecución de los diferentes comportamientos, especialmente en lo relacionado con la incorporación de nuevos conocimientos a través de la experiencia. Sin embargo no se considera una selección única de un comportamiento dominante ya que tales tipos de comportamiento tendrían demasiada complejidad y serían muy difíciles de diseñar.

4.5.2. Combinación de Comportamientos

A menudo no basta con seleccionar uno de entre varios comportamientos, sino que es necesario considerar a los comportamientos básicos como los cimientos de comportamientos compuestos más complejos en el mismo dominio. Para generar estos comportamientos compuestos es necesario aplicar algún tipo de operador de combinación con propiedades definidas, y que nos permita obtener la salida deseada. Dos de los operadores más empleados son la combinación directa de comportamientos básicos, y la combinación temporal de los mismos.

- Combinación directa de comportamientos (figura 2.14.a). Consiste en aplicar una función de combinación a las salidas de un subconjunto de comportamientos escogido del repertorio en un instante determinado. Por ejemplo, cuando las salidas de los comportamientos son vectores de dirección y velocidad relacionados con la navegación del robot, se puede realizar una suma ponderada de dichos vectores para obtener un vector que sería la salida del comportamiento de alto nivel equivalente [10]. La elección de los comportamientos a agregar suele ser de tipo intuitiva, aunque también sería posible —aunque más complejo— considerar inicialmente todos los posibles comportamientos del repertorio, y ajustar su peso o influencia de forma forma adaptativa, mediante diversas técnicas como Algoritmos Genéticos (*GA*) [82], lógica difusa [8] [83], *ANN*s, o *CBR* [84].

- Combinación temporal de comportamientos [85]. A menudo la consecución de un determinado objetivo conlleva la realización secuencial de varios sub-objetivos asociados a comportamientos simples o combinaciones directas de estos, de forma que para conseguir el objetivo global se debe realizar una combinación temporal apropiada de dichos comportamientos (figura 2.14.b). La transición de un comportamiento a otro —o a una combinación directa de varios— se produce cuando se cumplen unas determinadas condiciones, que deben ser percibidas o identificadas por el agente. Una vez establecidas las condiciones que determinan la activación de un comportamiento simple o compuesto, se puede construir un módulo de selección mediante diversos métodos, como por ejemplo una máquina de estados finitos(*FSM*). Estas condiciones pueden venir dadas por diversos medios: *RL* [86] [87] [88]; por una base CBR con aprendizaje por demostración [89];mezclando los dos métodos anteriores, a través de una función de "confianza" en la actividad de los comportamientos, adaptada a través de un sistema CBR [90]; o aprendidas por una *ANN* [91] [92].

Aunque estas son las bases de organización de comportamientos a través de combinaciones, existen diversos problemas que es necesario resolver. No siempre están claros las condiciones que indican los comportamientos a activar en cada momentos; ni

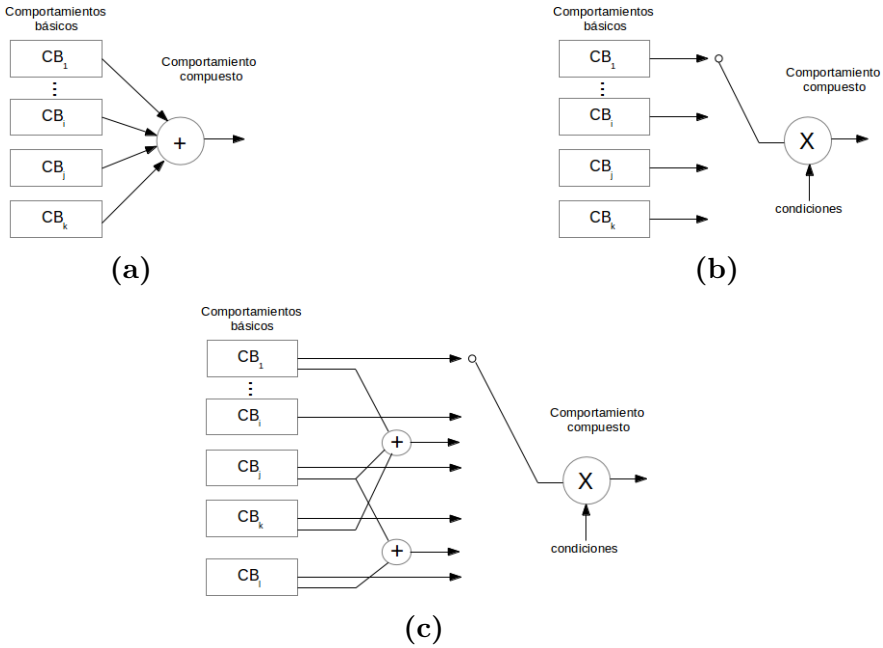


Figura 2.14: Combinación de comportamientos directa (a), temporal simple (b), y temporal-directa (c)

los comportamientos específicos a combinar o la función de composición de las salidas de los comportamientos escogidos. Muchos de estos aspectos dependerán, en todo caso, de las herramientas y métodos empleados para implementar los comportamientos y los módulos de arbitración.

4.5.3. Otros métodos

Además de los métodos indicados en los apartados anteriores, existen otros métodos de organización de comportamientos en la literatura que mezclan o combinan los indicadores en las secciones anteriores. La mayoría de ellos se identifican con arquitecturas específicas, como las redes de activación [93], o la arquitectura DAMN (*Distributed Architecture for Mobile Navigation*) [94]. En muchos de los métodos, se debe especificar la arquitectura de organización de comportamientos "a mano", lo cual hace la tarea especialmente compleja al tener que lidiar generalmente con entornos no-estructurados en los que resulta complicado hacer predicciones. Además, suele ser especialmente difícil estimar la relevancia de algunos comportamientos —sobre todo de forma cuantitativa—, en particular aquellos comportamientos que no están directamente relacionados con el objetivo del robot. Por último, la codificación manual de comportamientos se aleja bastante del modelo biológico-psicológico que se puede encontrar en la naturaleza, en el que el *aprendizaje* y la adaptación cumplen un papel fundamental. En este sentido es más apropiado acudir al aprendizaje, por medio de algoritmos evolutivos o de sistemas expertos adaptables mediante la experiencia. Este enfoque, defendido por algunas de las más novedosas teorías acerca de la inteligencia humana, tal como veremos en el capítulo 3, es el que predominará en el diseño de comportamientos dentro de la arquitectura de control/inteligencia que proponemos en la presente tesis.

3 Inteligencia y Aprendizaje

"Se mide la inteligencia de un individuo por la cantidad de incertidumbre que es capaz de soportar"
Emmanuel Kant (1724-1804) Filósofo

1. Inteligencia Humana e Inteligencia Artificial: evolución histórica y estado de la Técnica

Como ya hemos comentado en capítulos anteriores, si queremos desarrollar una arquitectura de *IA* aplicable a la Robótica, es indispensable partir de un buen modelo que describa el funcionamiento, estructura, y características de la Inteligencia Humana, en la que nos queremos apoyar para nuestro diseño. Nos interesa en particular una descripción de la Inteligencia Humana de tipo conceptual: no es tan importante —además de que complicaría la tarea en gran medida— imitar a la perfección los componentes biológicos que constituyen el cerebro humano, soporte de nuestra inteligencia, si bien es verdad que de la observación de muchos de estos comportamientos biológicos se pueden obtener interesantes conclusiones para la realización de nuestro modelo de *IA* (no obstante, más adelante se hablará de la vertiente *conexionista* de la *IA*, y de las redes neuronales artificiales o *ANN*).

1.1. Pioneros de la IA

Mil años antes de Cristo, antiguos filósofos griegos, chinos, e indios ya estudiaban el desarrollo de métodos de razonamiento mecánico o deducción formal. Pero no fue hasta el siglo IV a.C. cuando Aristóteles, a través de su Lógica Deductiva y el concepto de Silogismos [95], realizó la primera investigación sistemática acerca de los principios del razonamiento válido o correcto. Aristóteles, además, destacó ya el papel de la memoria y de la experiencia o aprendizaje en el desarrollo de la inteligencia: de como las percepciones y acciones de las personas quedan impresas en la memoria [96], y son exploradas durante los procesos cognitivos, de manera que pueden ser recuperadas a través de relaciones de similitud, contraste, o contigüidad [97]. Esta visión de la inteligencia, y en particular el papel que juega la memoria en el proceso, está muy

relacionada con la propuesta en el modelo que seguiremos como referencia en esta Tesis, como veremos en apartados posteriores.

En el año 1.315 Ramon Llull expresó, en su *Ars Magna* la idea de que el razonamiento podía implementarse de manera artificial en un artefacto mecánico con el que poder mostrar las verdades de la fe cristiana de una manera tan clara que no hubiese lugar a discusión. Inspirado en los trabajos de Roger Bacon, Llull pretendía construir una máquina que demostrase que los dogmas de la fe cristiana eran correctos y una tesis, en forma de libro, que pusiera sobre la mesa los errores que cometían los infieles, sin dejar lugar a ningún tipo de incertidumbre. El *Ars Magna* establece una teoría formal del conocimiento, y al mismo tiempo una mecanización formal del mismo, dando soporte físico u operativo a la lógica aristotélica. Por este motivo se le considera uno de los precursores del razonamiento automático, ya entendido como procedimiento mecánico que razona por sí mismo [98].

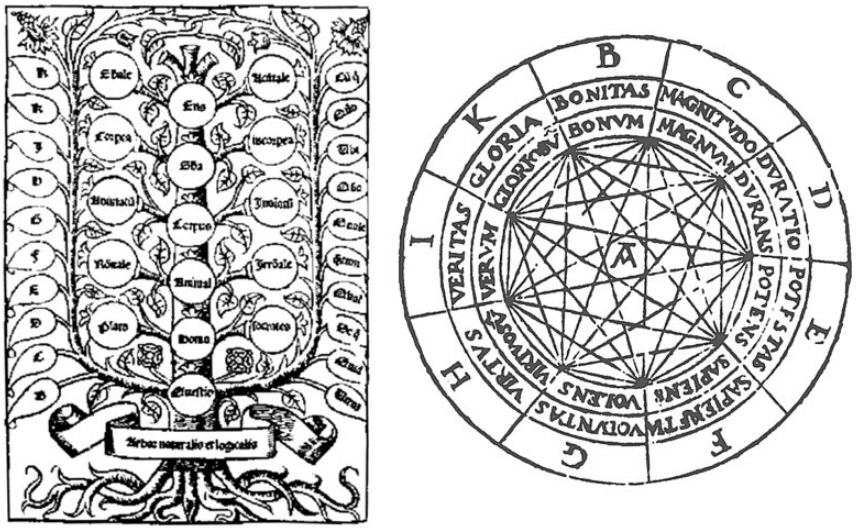


Figura 3.1: Ars Magna: árbol del conocimiento [99]

En el siglo XVII Gottfried Leibniz, fuertemente influenciado por la obra de Llull, postula la posibilidad de que las maquinas puedan generar ideas automáticamente, es decir, por sí mismas [100]. Según su punto de vista, el pensamiento se reduce a la realización de cálculos, por lo que propone construir un lenguaje simbólico y formal, la *característica universal* (*Characteristica Universalis*), capaz de expresar sin ambigüedad todos los pensamientos humanos. Este lenguaje guiaría el razonamiento de filósofos y científicos, que podrían reducir sus investigaciones y discusiones a una traslación de sus enunciados o problemas a dicho lenguaje simbólico —codificación del pensamiento—, seguido de una aplicación de cálculos matemáticos sobre los mismos, que llevaría a la solución correcta. Leibniz escribió el compendio de su teoría en su tesis, *De Arte Combinatoria* [101]. Aunque la concepción de la mente humana como una calculadora o computadora parece superada hoy día, sus conceptos de codificación de experiencias y conocimientos son un aspecto clave —y aún no resuelto— en las más modernas teorías sobre la inteligencia. De esta misma época, destacamos también los trabajos del filósofo Thomas Hobbes, en su concepción del hombre como máquina; y de Blaise Pascal en el diseño y construcción de calculadoras mecánicas. En la segunda mitad del siglo XIX y primera del XX, científicos y matemáticos como Charles

Babbage, Ada Lovelace, o George Boole lograron romper, a través del estudio de la lógica matemática, las barreras que impedían una realización plausible y tangible de la Inteligencia Artificial. El matemático David Hilbert planteó a la comunidad científica de las décadas de 1920-1930, la necesidad de responder a una cuestión fundamental: "¿Puede el razonamiento matemático ser formalizado?" [102]. En respuesta a esta pregunta Gödel, con su *Teorema de Incompletitud*, estableció los límites de lo que la lógica matemática era capaz de lograr; y Turing y Church, con su *Test de Turing* y *Calculo Lambda* respectivamente, establecieron que, dentro de estos límites, cualquier tipo de razonamiento matemático podía ser mecanizado o plasmado de forma artificial [103] [104]. Para Turing, la conciencia y el pensamiento se identifican con estados de

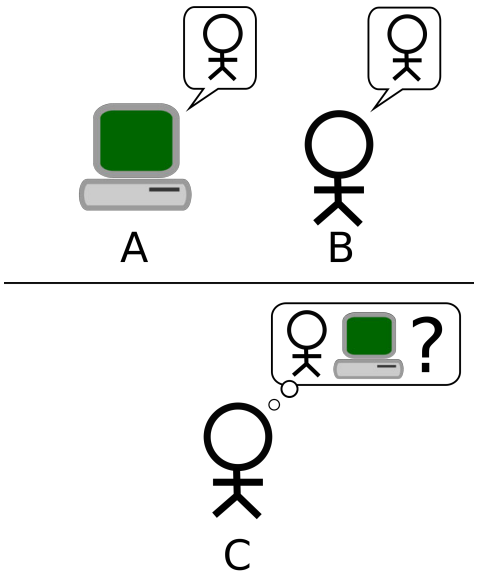


Figura 3.2: Test de Turing [105]

un sistema, definidos principalmente por su papel en la generación de nuevos estados y salidas en el sistema. Por primera vez se admitía completamente la certeza de que un dispositivo mecánico, a través del manejo de un lenguaje simbólico binario, pudiese imitar cualquier proceso de deducción matemática. Estos descubrimientos marcaron un hito en la historia de la *IA* e hicieron soñar a los científicos con la posibilidad de desarrollar máquinas pensantes [106]. En las siguientes décadas se empezarían a construir máquinas cada vez mas avanzadas, con capacidades para realizar operaciones complejas de cálculo matemático, aplicables a multitud de áreas de la sociedad humana. Es en 1956 cuando, en la famosa Conferencia de Darmouth, John McCarthy acuña el término *Inteligencia Artificial*, entendido como "la ciencia e ingeniería de hacer máquinas inteligentes". En la actualidad, la *IA* se suele definir como el "estudio y diseño de *agentes inteligentes*", entendiendo por agentes inteligentes, a aquellos sistemas capaces de percibir y obtener información de su entorno, y realizar acciones basadas en esta información, con objeto de maximizar la probabilidad de éxito en la realización de una determinada tarea o función [107].

1.2. Funcionalismo

De cualquier forma, la mayoría de las teorías que hemos ido mencionando establecen un paralelismo entre el cerebro humano y los computadores; es una teoría de la mente denominada *Funcionalismo* [108]. Esta visión de la inteligencia, aparecida en la segunda mitad del siglo XX, considera que los estados mentales se identifican por su rol funcional o relación causal con otros estados mentales, entradas sensoriales, y salidas conductuales. Por ello, se podrían implementar a través de estados físicos en diferentes tipos de criaturas o sistemas como por ejemplo, computadores o robots, siempre y cuando estos sistemas implementen las funciones apropiadas. Desde este punto de vista, el cerebro no sería mas que un computador que maneja otra simbología distinta de la binaria, y ejecuta algoritmos aún no conocidos totalmente por los investigadores. Si esto fuese así, y dada la potencia creciente de cálculo de los computadores, en muy poco tiempo sería posible que las máquinas superasen al ser humano en inteligencia [109]; dentro de la *IA* esta visión es la que se denomina también de *Inteligencia Artificial Fuerte*. El problema es si este concepto de cálculo algorítmico es realmente —y únicamente— lo que determina la inteligencia y el pensamiento humano, y parece ser que no es así. Se han planteado muchas objeciones a la visión Funcionalista de la inteligencia [107]. Algunas de ellas son de naturaleza filosófica o subjetiva: la identificación del pensamiento con al alma; el temor ante la posibilidad de que puedan existir seres superiores a los humanos; predicciones de lo que una máquina "nunca será capaz de hacer", muchas de las cuales han sido refutadas en la actualidad; supuestos sentidos extrasensoriales de los seres humanos; necesidad de una creatividad;...Aunque la gran mayoría no suponen objeciones importantes a la teoría hay, sin embargo, algunas que si han puesto seriamente en duda las bases del Funcionalismo:

- Objeciones matemáticas. El teorema de incompletitud de Gödel apela a la existencia de límites a la capacidad de un sistema computador basado en lógica para responder ante cualquier tipo de problema o cuestión. Ante esto, Turing responde que los seres humanos también son propensos a cometer errores e incapaces de realizar cierto tipo de tareas, pero no tiene en cuenta que llega un momento en que los humanos somos capaces de resolver problemas que parecían imposibles en el pasado.

- Informalidad del comportamiento. Todo sistema gobernado por leyes es predecible y, por tanto, no es realmente inteligente. Turing replica que con el tiempo, el comportamiento de las máquinas podría llegar a ser tan complejo que sería muy complicado de predecir. Sin embargo, si se conoce en profundidad la programación algorítmica de la máquina si que sería posible establecer al menos probabilidades de observar un determinado comportamiento dado una situación o información manejada por el sistema artificial.

- Naturaleza biológica del cerebro, en especial, comportamiento analógico del funcionamiento de la neuronas, las unidades fundamentales que componen el cerebro humano. Turing admite esta importante diferencia con respecto a la lógica binaria de las máquinas pero defiende que, dada la creciente potencia de los computadores, el comportamiento analógico de los sistemas nerviosos humanos podría llegar a ser simulado por una máquina. Sin embargo, Turing no tiene en cuenta que en la actividad cerebral humana las neuronas son capaces de realizar

"solo" unas 200 operaciones por segundo [110], mientras que hay gran cantidad de actividades que deben ser resueltas en medio segundo o incluso menos tiempo. Es verdad que el cerebro humano se asemejaría mas a una red de computadores paralelos, que a su vez realizan ciertas actividades de forma secuencial, en vez de a un solo computador muy rápido; pero aún así, el argumento se caería por la "regla de los 100 pasos" [111]: no importa cuantos computadores paralelos se conecten entre sí, ya que estos, funcionando con lógica de Turing, no podrían ser capaces de realizar la mayoría de procesos considerados mas simples en el ser humano.

- Necesidad de Consciencia o Entendimiento. Para que un sistema o entidad sea considerada como inteligente, no solo basta con que sea capaz de resolver problemas o demostrar comportamientos considerados como inteligentes; es necesario que el sistema sea consciente de lo que está realizando, y comprenda la forma de resolver ese problema. Esta objeción fue presentada por Searle en su "Experimento de la Habitación China" [112]; y por Block, con su "Argumento de la Nación China" [113]. Ante esta objeción, Turing arguye la imposibilidad filosófica de saber, a través únicamente de la observación de un comportamiento considerado como inteligente, si al agente —incluido una persona— es consciente de dicho comportamiento inteligente. Esta objeción parece una de las más débilmente contestadas y constituye, en gran medida, la base de la teoría de inteligencia que pretendemos trasladar a la *IA*.

De esta forma, se intuye que la computación algorítmica (la *IA* fuerte) no parece ser la clave de la inteligencia humana, y probablemente, tampoco lo será nunca para la *IA*.

1.3. Conductismo (inteligencia)

Otra importante corriente en el campo de teoría de la Inteligencia, aparecida en la primera mitad del siglo XX, es la denominada *Conductismo* [114]. De acuerdo con esta escuela de pensamiento, el comportamiento de los seres humanos (y en general de cualquier animal) se explica a través de sus inclinaciones conductuales, es decir sus tendencias a comportarse de cierta manera, según su naturaleza, y en respuesta a la aparición de determinados estímulos. La inteligencia no se explicaría a través de estados mentales, o procesos internos, sino a través de la observación de comportamientos considerados como inteligentes, en la medida que resuelven problemas o implementan tareas. or tanto interesa más replicar los resultados de lo inteligencia que como funciona en realidad ésta. Esta corriente, englobada dentro de la conocida como Inteligencia Artificial Tradicional (*GOFAI, Good Old Fashioned IA*), representa la información de forma simbólica y la manipula mediante una serie de reglas orientadas a la resolución de un determinado problema en particular, sea mover a un robot a través de una fábrica, reconocer la escritura humana, o localizar rostros en una imagen. Tiene por tanto importantes puntos en común con el Funcionalismo, ya que defiende la implementación de *IA* mediante algoritmos ejecutados en potentes computadoras. Y cae, por tanto, en las mismas "debilidades" ya que, al ignorar el funcionamiento intrínseco de los procesos cognitivos en el cerebro humano, descarta la presencia de importantes características atribuidas a la inteligencia humana, como la creatividad, la capacidad de generalización,

o la flexibilidad y adaptabilidad ante errores o situaciones cambiantes. Como Searle mostró en el "Experimento de la Habitación China", la ejecución correcta de un conjunto de instrucciones, sin ningún tipo de conocimiento de su significado, ni del significado global de la tarea a resolver, no es prueba suficiente de inteligencia. En realidad, las teorías más actuales de la *IA* consideran otros aspectos, como la capacidad de predicción o anticipación, como elementos fundamentales de la inteligencia. No obstante, hay que admitir que, la combinación de métodos derivados de la *IA* clásica con otros métodos tales como las *ANN*, los sistemas expertos, la lógica difusa, o los algoritmos genéticos, entre otros, para desarrollar *sistemas híbridos*, ha demostrado un cierto éxito, en particular a lo que se refiere a los niveles más altos o deliberativos de la inteligencia [115].

1.4. Conexionismo

El Conexionismo [116] es, de todas las corrientes de investigación en la teoría de la inteligencia, la más cercana a una visión de tipo biológico. Los modelos y arquitecturas presentadas por los investigadores "conexionistas" se basan en los conocimientos actuales acerca de la neurofisiología del cerebro, intentado trasladar a la *IA* las características funcionales que se consideran necesarias para el pensamiento. A diferencia de las escuelas anteriores, los "conexionistas" inciden en un procesamiento paralelo de información, obtenida a través del análisis de sus propiedades estadísticas, en vez de usar reglas lógicas. Las arquitecturas propuestas se basan en la agrupación de unidades simples, las *neuronas artificiales*, para construir redes interconectadas denominadas *Redes Neuronales Artificiales*(*ANN*) [117]. Existen diversas variantes de neuronas artificiales , así como de métodos de interconexión entre las mismas pero, en todas ellas, el conocimiento y la memoria se distribuye a través de todos los componentes y de sus interconexiones, de manera muy similar a lo que ocurre en el cerebro humano. Las *ANN* han demostrado una gran robustez con respecto a la

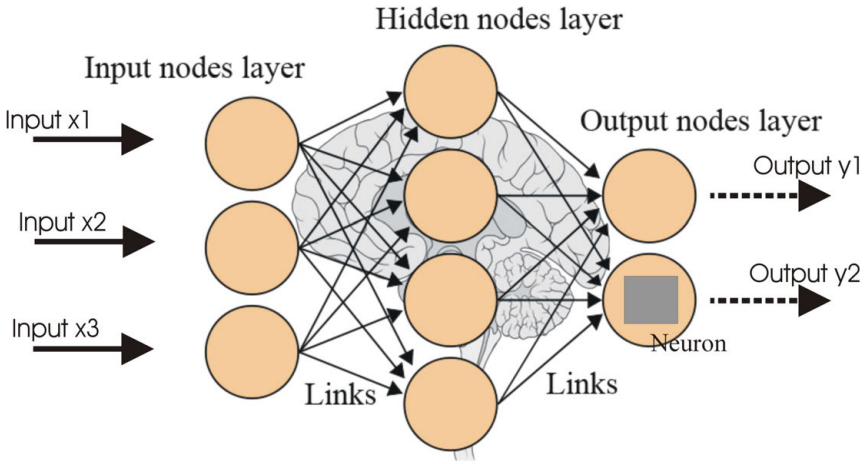


Figura 3.3: Redes Neuronales Artificiales [118]

información ruidosa o incompleta, así como ante las inexactitudes o degradación de su

estructura global, obteniendo unos resultados excelentes en problemas de clasificación o categorización. Estas redes son capaces de capturar patrones estadísticas muy sutiles, que serían muy difíciles de deducir mediante reglas lógicas. Como cualidad adicional son el primer paradigma de *IA* de entre los que hemos visto hasta ahora, que incluye la noción del *aprendizaje* como parte del proceso cognitivo, algo que se ha demostrado fundamental en la inteligencia humana, en contraposición a la simple programación de reglas. Sin embargo, las *ANN* también demuestran ciertas debilidades en la búsqueda de una respuesta definitiva a la *IA*, especialmente relacionadas con la inexactitud de los modelos propuestos para imitar el funcionamiento de las redes neuronales del cerebro humano.

- Por ejemplo, las *ANN* clásicas no suelen considerar la variedad de clases de neuronas existentes en un cerebro biológico, ni los efectos de las hormonas y neurotransmisores (aunque existen algunas excepciones en el campo de la Neurociencia Computacional, una sub-variante dentro de la corriente Conexionista).

- Los modelos de *ANN* más comunes como el perceptrón multicapa (*Multilayer Perceptron (MPL)*) [119] emplean una técnica de aprendizaje supervisado llamada *retropropagación (backpropagation)*, en la cual se propaga el error de las salida obtenidas solo hacia las neuronas que han contribuido en la formación de cada salida. Este modelo no tiene en cuenta la enorme cantidad de conexiones de realimentación entre las diferentes áreas del cerebro humano [120]. Aunque la funcionalidad de esas conexiones no está del todo clara, su inclusión en el modelo sería aconsejable.

- Las *ANN* tradicionales no suelen incluir una *componente temporal* tendiendo, por contra, a trabajar con patrones estáticos. Teniendo en cuenta que la capacidad de predicción es una característica fundamental de la inteligencia humana, sería razonable que los modelos incluyesen características temporales o, al menos, secuencias de patrones.

- Por último, y desde un punto de vista de investigación, las *ANN* presentan el problema de que funcionan demasiado como una "caja negra" que, tras el proceso de aprendizaje, solo proporcionan el resultado como elemento de análisis. La encriptación de la información a través de las interconexiones entre las neuronas artificiales, y de la propia estructura de red planteada, presenta muchas dificultades de análisis y complica la tarea de comparar el desempeño de diferentes realizaciones y de encontrar posibles puntos débiles en las mismas.

En conclusión, aunque la filosofía postulada por las *ANN* era muy prometedora y ha demostrado bastante éxito en la solución de problemas específicos, la proposición de un marco de referencia de la inteligencia basado únicamente en *ANN* no está claro, al menos con los modelos actuales. Sin embargo, hay que destacar la existencia de una segunda variante de modelo de *ANN*, las conocidas como *redes auto-asociativas* o *de Hopfield* [121]. Estas redes *si* presentan un importante grado de realimentación entre sus unidades componentes, y si tienen en cuenta la componente temporal, ya que evolucionan a lo largo del tiempo. Estas redes pueden obtener resultados correctos con información incompleta, y almacenar secuencias temporales, algo muy parecido a

lo que hacen realmente las redes neuronales del cerebro humano. Pese a todas estas características positivas, las redes auto-asociativas suelen ser utilizadas en la resolución de problemas concretos, pero no se han propuesto como elemento fundamental de un modelo completo y generalizado de *IA*, quizás debido a la excesiva complejidad de su análisis en el caso de redes de mayor tamaño y con mayor número de interconexiones.

1.5. Inteligencia artificial débil: Sistemas Expertos

Las filosofías de diseño de *IA* que hemos ido viendo en los apartados anteriores aspiran, en gran medida, al desarrollo de un modelo o marco de inteligencia general, que permita emular la forma en que los seres humanos resuelven los problemas y afrontan las situaciones encontradas en el día a día de su vida. Existen, sin embargo, otras corrientes en el campo de la investigación en *IA*, que consideran este objetivo demasiado ambicioso y que defienden, por contra, el desarrollo de procedimientos y técnicas enfocadas a la resolución de problemas circunscritos a áreas muy concretas de la inteligencia. Este conjunto de métodos y algoritmos se engloban dentro de la conocida como *Inteligencia Artificial Débil o Reducida*. Y, si bien no coinciden en la visión de construir arquitecturas globales que nos permitan desarrollar *IA* similares a las humanas, si que forman parte, a menudo, de las arquitecturas más elaboradas y completas, en muchos casos como método de implementación de alguna de esas etapas. Por este motivo hemos creído interesante realizar una breve revisión de las principales técnicas de *IA* débil, que presentamos a continuación. Las técnicas conocidas en su conjunto como *Sistemas Expertos*, tratan de incorporar a un computador la experiencia humana en un determinado campo, buscando que el sistema artificial sea capaz de cumplir su tarea al mismo nivel que el experto humano. Un problema importante para lograr este objetivo es que a menudo los expertos humanos no tienen claras las reglas utilizadas para tomar sus decisiones, basando muchas de ellas en reglas generales según su experiencia (*heurísticos*), e incluso en intuiciones. Algunos primeros ejemplos de este tipo de sistemas fueron *DENDRAL* [122], considerado como el primer sistema experto, y aplicado a la química; *MYCIN* [123], en el campo de la medicina; y *PROSPECTOR* [124], en el campo de la geología. Todos ellos fueron añadiendo nuevos avances en el área, especialmente relacionados con los métodos de capturar, analizar, y expresar información y reglas para tomar sus decisiones: esto es lo que se conoce actualmente como *Ingeniería del Conocimiento*. Poco a poco, se añadieron aspectos como la separación de las reglas y el mecanismo de razonamiento en bloques distintos; o la incertidumbre, en muchos casos, del conocimiento, y la forma de afrontar dicho problema. Entre las características comunes a este tipo de sistemas tenemos i) su aplicación en un dominio o área muy específica; ii) el uso de conocimientos basados en experiencia; iii) la capacidad de explicación de las decisiones tomadas; y iv) el razonamiento simbólico, que permite representar diferentes tipos de conocimientos, como conceptos, evidencias, o reglas, mediante datos cualitativos en vez de numéricos.

Los sistemas expertos, a diferencia de los algoritmos programados, no siguen una secuencia predefinida de pasos; por contra, permiten un razonamiento inexacto, y pueden trabajar con datos incompletos, inexactos, o difusos. No obstante, y pese a su buen desempeño en tareas muy específicas, los sistemas expertos presentan algunos problemas que los hacen inviables como base única para un marco de *IA*:

- No tienen en cuenta que en la ”vida real“ no siempre es posible aislar

completamente y de forma independiente las tareas o problemas. Aspectos pertenecientes a otra área distinta del problema que se pretende resolver pueden influir en la resolución de éste.

- Estos sistemas no son robustos ni flexibles; si la definición de los problemas a resolver no es muy estricta y clara, no suelen tener un buen desempeño. Así mismo, es difícil validar que el conocimiento almacenado en las reglas es suficientemente completo y correcto.
- Aunque es posible seguir la secuencia de razonamientos que han llevado a una solución, esta cadena no suele incluir conocimientos heurísticos que permitan entender mejor el problema.
- Si el número de reglas es alto, se puede tardar mucho en encontrar la respuesta, por lo que este tipo de métodos no parece adecuado para sistemas en tiempo real.
- La mayoría de los sistemas expertos no son capaces de evolucionar o aprender a medida que se ejecutan. Las reglas de conocimiento se "graban" una vez definidas y suelen tener poco margen de capacidad de variación a partir del análisis de su desempeño (y esto solo los mas modernos).

En ocasiones se han combinado sistemas expertos con *ANN*, de forma que éstas últimas ayudasen a extraer conocimientos no evidentes en las decisiones de los expertos, e incluso corregir y revisar las reglas de los sistemas [125] [126].

1.5.1. Reglas de decisión

Los sistemas expertos más representativos y conocidos son los basados en *reglas de decisión* [127]. En estos sistemas el conocimiento se codifica a través de unas reglas, mas o menos complejas, que unen unas premisas o condiciones con unas conclusiones o acciones. En este sentido se asemeja en parte al modelo *SA* de las arquitecturas reactivas, si bien en este caso las condiciones engloban tanto a elementos lingüísticos como a sus valores, relacionados por unos operadores; con las conclusiones ocurre algo similar. En la figura 3.4 podemos ver la estructura y componentes de un modelo general de sistema experto basado en reglas. Se distinguen varios componentes:

- Una base de conocimientos, que contiene el conocimiento del dominio asociado a la resolución del problema, en este caso representado por un conjunto de reglas que se "activan" cuando se cumplen sus condiciones.
- Una base de datos que incluye un conjunto de evidencias en las cuales se busca una correspondencia con las condiciones de las reglas.
- Un motor de inferencia, que enlaza las reglas y las evidencias de las dos bases comentadas, permitiendo realizar el razonamiento que lleva al sistema a encontrar una solución.
- Unos componentes de explicación/justificación, que permiten determinar como se ha llegado a una conclusión y qué evidencias fueron las responsables de la misma. Este es un aspecto clave de cualquier sistema experto, la capacidad de, no solo llegar a una conclusión, sino de indicar el proceso hasta llegar a la misma, justificando de este modo la decisión tomada.

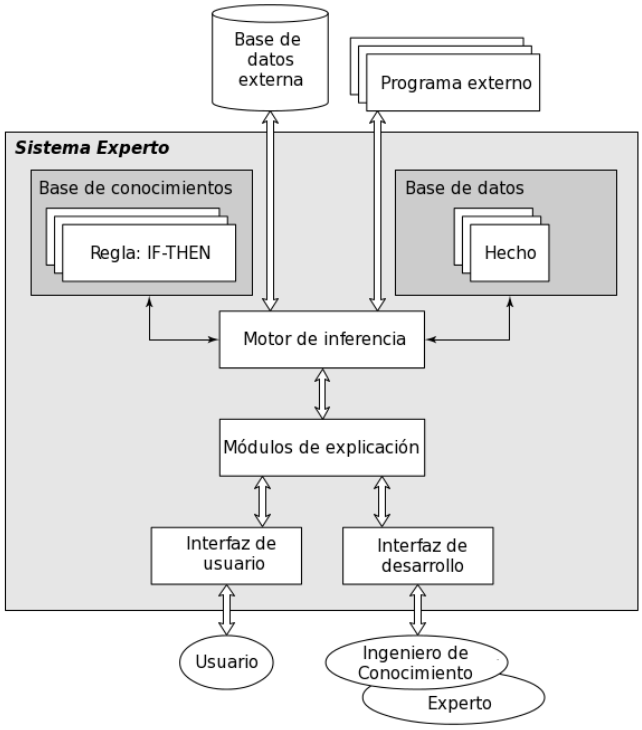


Figura 3.4: Estructura y componentes de un Sistema Experto

- Un interfaz de usuario, que permita a un usuario plantear un determinado problema y acceder a la solución y al razonamiento de la misma.
- Opcionalmente, se puede incluir un interfaz de acceso a datos y programas de apoyo, externos al sistema;
- Un interfaz de desarrollo, para la edición y depuración de la base de conocimientos y el acceso a los componentes de explicación. Este componente permitiría monitorizar los cambios en dicha base, o el proceso de activación de las reglas en una ejecución; o incluso que el sistema pidiese nuevas evidencias y reglas en caso de que se considere necesario.

Esta estructura, con algunas modificaciones, es extensible a otros sistemas expertos distintos de los basados en decisiones.

1.5.2. Redes Bayesianas

Como hemos indicado anteriormente, uno de los aspectos clave en cualquier sistema experto es su capacidad para manejar información incompleta, inconsistente, con cierto grado de incertidumbre y, a pesar de ello, ser capaz de llegar a una conclusión válida. En la medida en que es muy plausible que la información de que dispongamos no sea exacta, esta propiedad de los sistemas expertos resulta especialmente interesante. Para afrontar la incertidumbre, se emplea la Teoría de Probabilidad, especialmente probabilidades condicionadas enunciadas en la forma de reglas *Bayesianas*. A diferencia de las reglas de decisión "clásicas", las reglas bayesianas establecen que, ante una condición o condiciones, se cumplen unas conclusiones o acciones, pero solo con un

cierto nivel de probabilidad. Esto permite que los expertos pueden establecer varias hipótesis a partir de un mismo conjunto de evidencias, y determinar las probabilidades de ocurrencia de cada una de ellas; de este forma sería posible clasificar las hipótesis y escoger la más probable. Estas probabilidades asociadas a las reglas se pueden obtener mediante probabilidad condicionada "clásica"; o bien siendo el experto el que asigne un *factor de certidumbre* a cada regla según su nivel de creencia en la misma [123].

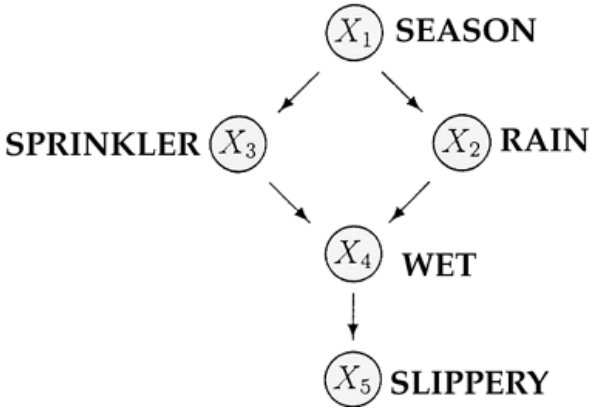


Figura 3.5: Ejemplo de una red Bayesiana simple [128]

A partir de esta teoría de reglas bayesianas, se pueden construir sistemas más complejos de representación del conocimiento a través de, por ejemplo, grafos dirigidos como las *Redes Bayesianas* [129]. Dependiendo del dominio del problema y de la implementación de la aplicación, los nodos de la red pueden representar variables, parámetros, hipótesis, o estados del sistema; las transiciones y desplazamientos entre los nodos de la red vendrán determinados por las probabilidades o certidumbres asociados a los mismos. Entre los tipos mas empleados de redes bayesianas tenemos [107] las "ingenuas" (*naive*), con estructura en forma de "V"; redes bayesianas dinámicas (*Dynamic Bayesian Networks (DBN)*), que evolucionan a lo largo del tiempo; redes gaussianas, que siguen una distribución de probabilidad gaussiana; y, por último, las cadenas de Markov.

1.5.3. Lógica difusa (*Fuzzy Logic*)

En los sistemas expertos clásicos, para afrontar la posible incertidumbre del conocimiento se utilizaba teoría de probabilidad. Sin embargo, es evidente que la mayoría de los seres humanos no calculamos probabilidades para determinar la ocurrencia de algún evento, sino que pensamos en concepto del tipo "esto ocurre a menudo, esto a veces, esto ocasionalmente, rara vez," etc... Por otra parte, tampoco solemos emplear —en la mayoría de los casos— valores numéricos exactos para definir los objetos que nos rodean, sino que utilizamos conceptos más indefinidos como "grande" frente a "pequeño", o "rápido" frente a "lento". La Lógica Difusa [130] trata de imitar esta forma de razonamiento humano y codificar su conocimiento en una forma más cercana la que aplicaría un experto a la hora de resolver un problema. Una de sus principales rupturas con los métodos anteriores es el uso de variables conceptuales o lingüísticas en lugar de numéricas.

Los sistemas de Lógica Difusa utilizan menos reglas, ya que incluyen la posibilidad de *fusionarlas*; esto hace que sean, por lo general, más rápidos que otros sistemas

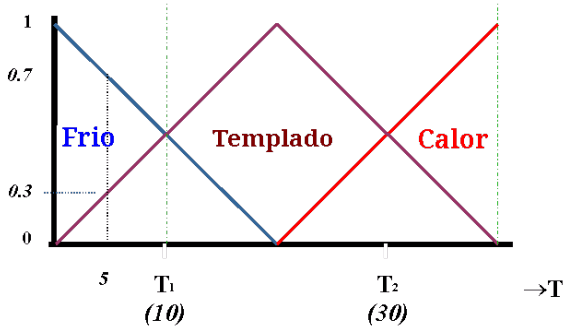


Figura 3.6: Ejemplo de reglas de lógica difusa

expertos. De la misma manera, en esta técnica es posible fusionar distintos expertos, cada uno especializado en un problema o área, de forma que se consiga expandir el dominio de aplicación de los problemas sin necesidad de recurrir a un experto con un dominio muy amplio de varios campos o a un nivel muy abstracto, lo cual resultaría mucho mas complicado que trabajar con sistemas enfocados a un dominio mas preciso y reducido. Un aspecto clave para el buen desempeño de este tipo de sistemas es la definición o extracción de reglas de los expertos humanos, algo que depende no solo de las capacidades del experto, sino también de la complejidad del problema. Por otra parte, puede ser muy costoso en tiempo y esfuerzo ajustar y probar las reglas hasta que el sistema diseñado logre un buen desempeño[1]. En este sentido, existen actualmente sistemas de lógica difusa adaptativos que, apoyándose en otros tipos de técnicas o métodos como las *ANN*, permiten encontrar nuevas reglas o adaptar y mejorar las existentes a partir del procesamiento de los datos de entrada y la observación de los resultados.

1.5.4. Marcos de representación del Conocimiento (*Frame-based*)

Los sistemas expertos basados en Reglas de Decisión y Redes Bayesianas representan el conocimiento mediante reglas de uno u otro tipo, pero esta no es la única representación posible: Minsky [131] propuso una representación de conocimiento basada en *marcos*. Un marco permite representar un concepto mediante un conjunto de atributos (*slots*), con unos valores asociados, o los procedimientos necesarios para calcularlos. En contraste con los métodos basados en reglas, esta representación evita que la información se disperse por toda la base de conocimientos, acelerando la búsqueda de soluciones. El concepto es similar al de los objetos en la programación orientada a objetos: los marcos contienen todos los atributos que los describen, así como los métodos necesarios para gestionarlos. Además, y de la misma manera que en programación orientada a objetos, distinguimos entre marcos de clase, que representan un concepto o grupo de objetos similares, e instancias de clase, que se refieren a un objeto particular dentro de la clase.

El nivel de descomposición de un problema en marcos depende del problema en cuestión a resolver y de las indicaciones del experto, pero como los marcos admiten herencia, es posible establecer diferentes niveles de abstracción. La interrelación entre

[1]Como ejemplo, el metro de la ciudad japonesa de Sendai se controla con un sistema que utiliza 54 reglas de lógica difusa, pero los ingenieros que lo diseñaron tardaron varios años en ajustar el sistema hasta lograr un funcionamiento correcto

los marcos puede establecerse a partir de la generalización, que relacionaría una superclase más general con sus subclases (como por ejemplo, la clase "coche" como parte de la clase "vehículo"); agregación, por el que las subclases son componentes de una superclase que representa un todo (la subclase "rueda" como parte de la clase "coche"); o la asociación, que describe una relación semántica entre dos clases independientes (por ejemplo, una persona puede ser dueña de tres instancias de las clases "coche", "perro", y "casa"; aunque estas clases son independientes, se relacionan a través de ese concepto de posesión).

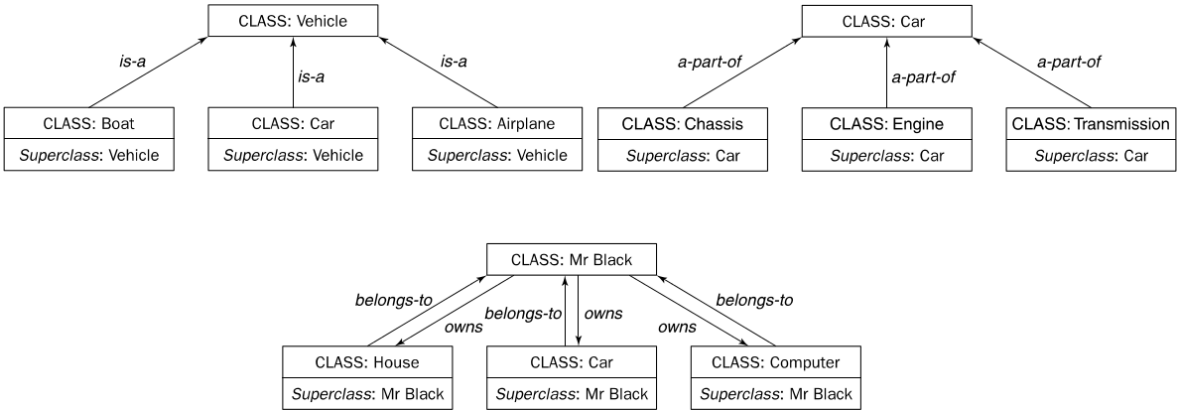


Figura 3.7: Ejemplos de clases en un sistema basado en marcos [132]

Pese a que conceptualmente este tipo de sistemas presenta aspectos muy interesantes para una teoría de inteligencia, al igual que los niveles de abstracción entre conceptos, también presentan algunos inconvenientes. A menudo es muy difícil diferenciar las propiedades esenciales de los objetos de las accidentales, por lo que suele ser complicado construir conceptos compuestos que presenten herencia múltiple. No obstante, los sistemas expertos mas actuales combinan tanto marcos como reglas, usando reglas para la evaluación de la información contenida en marcos; para ello se usan claúsulas de búsqueda de patrones. A diferencia de los sistemas basados en reglas, aquí las reglas juegan un papel auxiliar.

1.5.5. Computación Evolutiva

La Computación Evolutiva es otra aproximación a la *IA* basada en la realización de modelos computacionales tomados de la selección natural y la genética. No en vano, la inteligencia define también la capacidad de un sistema para adaptar su comportamiento a un entorno abierto y cambiante. La evolución está relacionada con la habilidad de una población para sobrevivir y reproducirse en un entorno, pero también con la habilidad de un organismo para anticipar cambios en dicho entorno y adaptarse a los mismos. Esta cualidad, denominada *aptitud evolutiva* (*evolutionary fitness*), es lo que se trata de optimizar en la Naturaleza. En una población, el cruce de dos individuos con mayor aptitud evolutiva tiene mayor probabilidad de obtener un individuo con alta aptitud evolutiva; y como estos individuos son los que tienden a sobrevivir, la población evoluciona hacia la existencia de dicha aptitud. Pero si las condiciones del entorno cambian, favoreciendo otra aptitud distinta, la población evolucionará con el tiempo hacia esta nueva aptitud. Las principales técnicas de la computación evolutiva

son los algoritmos genéticos, las estrategias evolutivas, y la programación genética, que simulan la evolución mediante procesos de selección, mutación, y reproducción.

- Los algoritmos genéticos (*GA*) [133] manipulan cromosomas artificiales —en realidad, cadenas de dígitos binarios— mediante operaciones como selección, cruce, o mutación. Para ello, realizan una serie de etapas que permiten moverse de una población de cromosomas artificiales a otra distinta. Dos aspectos básicos para la resolución de un problema mediante *GA*s, son i) la forma en que se codifica el problema; y ii) cómo se evalúa. Para la codificación cada cromosoma está compuesto por genes, generalmente binarios. En cuanto a la evaluación, se define una función de aptitud de un cromosoma en la resolución de un problema; esta aptitud determina las bases de selección de los cromosomas durante la fase de reproducción. En la reproducción se utilizan los operadores de *cruce* para intercambiar partes de dos cromosomas; y de *mutación*, para modificar el valor de un gen en una posición aleatoria. Este proceso se realiza de forma iterativa, siendo cada iteración una *generación* de la población. Tras un cierto número de iteraciones se espera obtener una población de cromosomas con gran aptitud. En un problema real se suele usar una población de miles de cromosomas en un proceso de 50 a 500 generaciones, hasta obtener una solución casi óptima. Para estudiar la evolución de la población se suelen emplear gráficas de rendimiento, que representan la aptitud media de la población.

- Las estrategias evolutivas [134] suelen estar enfocadas a la resolución de problemas de optimización en ingeniería. Se basan en la introducción de cambios aleatorios imitando las mutaciones naturales. En su versión mas simple, se aplican mutaciones normalmente distribuidas con media cero y una desviación predefinida, en cada rango de variación de las variables. En cada generación se calcula la solución obtenida con respecto a la población original; si la solución con los parámetros mutados mejora la de los parámetros originales, estos son reemplazados. Este proceso continúa hasta encontrar una solución satisfactorio o ejecutar un determinado número de generaciones. Una ventaja de este método respecto del anterior, es que no es necesario codificar el problema, es decir establecer una representación del mismo adecuada a la manipulación de los algoritmos evolutivos.

- La programación genética [135] tiene como objetivo hacer que las computadoras solucionen problemas sin ser programadas explícitamente para ello. Esta teécnica se considera una extensión de los *GA*, pero el resultado final no es una representación codificada de un problema, sino un programa que resuelve ese problema. Para ello se manipulan los programas aplicando operadores genéticos, lo que hace necesario que los programas en si mismos se puedan representar como datos manipulables; esto se consigue utilizando programas en lenguaje LISP, que tiene una estructura muy orientada a símbolos, y capacidad para automodificarse y escribir nuevos programas. Los elementos básicos de un programa LISP —átomos y listas— se pueden representar en estructuras de árbol, lo cual permite separar los elementos en ramas, sobre los que aplicar los operadores genéticos de cruce o mutación. Sin embargo, estas técnicas no han obtenido grandes resultados en problemas complejos, que necesiten programas de gran tamaño.

Las técnicas de computación evolutiva parecen mas orientadas a la resolución de problemas concretos más que a un marco de inteligencia artificial general. Sin embargo, puede ser interesante incluirlas como parte de esquema más complejos como en el caso de las soluciones híbridas, especialmente para la resolución de aspectos concretos que no presenten excesiva complejidad.

1.5.6. Razonamiento basado en casos (*CBR*)

El razonamiento basado en casos(*CBR*) [136] es otra aproximación de la *IA* a la resolución de problemas a través del aprendizaje. En vez de apoyarse en un conocimiento general en el dominio del problema, o buscar correspondencias en las relaciones generales entre los descriptores del problema y las soluciones, *CBR* utiliza conocimientos específicos obtenidos a partir de la experimentación de diversas situaciones concretas de dicho problema y las soluciones empleadas para ellas. La idea es poder resolver problemas nuevos a partir de situaciones semejantes experimentadas y resueltas, adaptando las soluciones a las nuevas circunstancias. Este tipo de sistemas son capaces, además, de mantener un aprendizaje incremental y continuado a lo largo del tiempo añadiendo soluciones ante nuevas experiencias, optimizando las ya existentes, y mejorando, en suma, su rendimiento a medida que continúa ejecutándose.

La operación de un sistema *CBR* se realiza a través de un *ciclo CBR*, que implica operaciones de: i) recuperación de experiencias almacenadas parecidas a la que se presenta en el momento actual; ii) reutilización de la información y el conocimiento recuperado para resolver el problema actual; iii) evaluación de la corrección en la solución alcanzada; y iv) almacenamiento de la nueva experiencia en la base de información del sistema. Cualquier sistema *CBR* se define a través de un modelo de funcionamiento o procesado de este ciclo *CBR*: y de una estructura de tareas y métodos para la realización del razonamiento basado en la información almacenada a través de los casos.

Aunque *CBR* nació como una herramienta enfocada a la resolución de problemas en un dominio concreto, su estructura , características, y funcionamiento parecen muy apropiadas para utilizarla como componente básico de un modelo y arquitectura de *IA*; aspectos como el aprendizaje continuo a través de la experiencia; la resolución de problemas por analogía; o la capacidad para representar información a diferentes niveles de abstracción se ajustan a las características supuestamente presentes en la inteligencia humana. Por este motivo, hemos considerado este paradigma como la base para el modelo de *IA* que propondremos en esta Tesis. En el capítulo 4 se analizará con mayor profundidad los aspectos fundamentales de *CBR*, aspectos que deberemos considerar a la hora de implementar los niveles reactivos de la arquitectura propuesta, que constituyen la base de nuestro trabajo.

2. El proceso de Aprendizaje

Tan importante como el concepto o modelo de inteligencia es el proceso de como surge y se desarrolla tal inteligencia. El proceso de aprendizaje es fundamental en el desarrollo de la inteligencia humana y, consecuentemente, se supone que ha de tener también un papel fundamental en el desarrollo de las *IA*. Existen gran cantidad de Teorías de Aprendizaje que intentan describir como se realiza este proceso, tanto

en humanos como en otros animales, así como su impacto en el desarrollo de la inteligencia. La mayoría de ellas están muy interrelacionadas con algunas de las teorías sobre la inteligencia que hemos visto en el apartado 1 de este capítulo. Se distinguen principalmente tres perspectivas con respecto a las teorías sobre el aprendizaje: *Conductismo*, *Cognitivismo*, y *Constructivismo*.

2.1. Conductismo (aprendizaje)

El Conductismo, como teoría sobre el aprendizaje [137], está íntimamente relacionada con su teoría homónima en el campo de la inteligencia. Según esta teoría, el aprendizaje es resultado de ejercer un condicionamiento sobre un sujeto que está adquiriendo un nuevo comportamiento, mediante la aplicación de una recompensa o un castigo, que incrementan o decrementan, respectivamente, la probabilidad de ocurrencia de ese comportamiento en presencia de estímulos o circunstancias similares. Según esta concepción, no se necesita un entendimiento del proceso de aprendizaje, más allá de una asociación entre un estimula de entrada y una respuesta de salida. Muchos métodos de *IA*, tales como las *ANN*, los algoritmos genéticos*GA*, o los sistemas expertos, se basan o aplican un aprendizaje reforzado. Una explicación más detallada acerca del aprendizaje reforzado en la *IA* se puede consultar en [138].

2.2. Cognitivismo

El Cognitivismo [139] o Procesamiento Cognitivo de la Información —*Cognitive Information Process (CIP)*— se inspira en el desarrollo de computadoras con una arquitectura de procesamiento estricto tipo "entrada-procesado-salida". Según las teorías cognitivas, el conocimiento está compuesto por construcciones simbólicas mentales en el cerebro de los individuos, y el aprendizaje se basa en trasladar estas representaciones simbólicas a la memoria, para que allí puedan ser procesadas. El aprendizaje, en el Cognitivismo, se centra más en asimilar una solución particular a un problema particular, que en adquirir ciertas habilidades que harían posible, no solo resolver dicho problema, sino un conjunto de problemas de características similares. En este modelo de aprendizaje, se programarían algoritmos específicos en potentes computadoras, que no tendrían ninguna capacidad de adaptación mas allá de las posibles variaciones programadas.

2.3. Constructivismo

Para el Constructivismo [140], el aprendizaje es un proceso activo, en el que los aprendices construyen nuevas ideas o conceptos, a partir de su experiencia y conocimiento pasado y actual. El aprendiz toma un papel activo en la selección y transformación de la información, construye hipótesis y toma decisiones, apoyado en un modelo mental o estructura cognitiva. Esta teoría incluye muchos conceptos y características deseables en un modelo de inteligencia, como la flexibilidad en los conceptos a aprender; la asimilación de los mismos a través de la experiencia; la adaptación ante diferencias entre los conocimientos adquiridos y nuevas situaciones; y un cierto grado de autonomía respecto del instructor o programador, en el proceso de aprendizaje. Sin embargo, muchas de estas características (como se toma una decisión,

como se selecciona y codifica la información, como se modela una experiencia,..) no están definidas con precisión, por lo que queda mucho trabajo de investigación por delante para plasmar este modelo en el campo de la *IA*.

2.4. Otros modelos de inteligencia y teorías de aprendizaje

Finalmente, algunas de las últimas teorías acerca de la inteligencia humana y el proceso de aprendizaje, se alimentan de los mejores aspectos e ideas de los que hemos analizado en las secciones anteriores, de forma que unos puedan resolver las debilidades y carencias de otros, hasta llegar al mejor rendimiento posible. Esto no nos debe sorprender, ya que los sistemas de *IA* con mayor éxito son, en la actualidad, *Sistemas Híbridos* [115] [141], los cuales combinan muchas de las tecnologías anteriormente indicadas. Una de estas teorías, que ha causado un gran impacto especialmente en el campo de la *IA*, es la enunciada por Jeff Hawkins en [1], y su modelo derivado llamado *Memoria Jerárquica Temporal*(*HTM*), propuesto por Hawkins y George en [37]. En realidad, la mayoría de los aspectos que integran el modelo de Hawkins no son conceptos nuevos en los campos de *IA* o Neurociencia; ya existían como fragmentos diseminados aquí y allí, como parte de diferentes interpretaciones de la inteligencia y el aprendizaje. Quizás el principal mérito de Hawkins ha sido agruparlos en un marco común. El modelo *HTM* se tratará en mayor profundidad en la siguiente sección, ya que este trabajo se centra en la implementación de algunos aspectos de este modelo aplicados a un robot cuadrúpedo orientado a través de la visión.

3. El modelo de Inteligencia como Memoria Jerárquica Temporal (HTM)

El modelo de Memoria Jerárquica Temporal *HTM* [37] es un modelo de aprendizaje computacional desarrollado por Jeff Hawkins y Dileep George, que intenta modelar algunas de las propiedades algorítmicas y estructurales del neocortex en forma de redes con una estructura similar a la de las redes Bayesianas [142]. La idea principal del modelo *HTM* es que el cerebro humano no actúa como un procesador o una red de procesadores distribuidos, ejecutando programas que permiten obtener una respuesta ante un conjunto de entradas; ni el aprendizaje consiste en escribir un conjunto de instrucciones suficientemente amplio como para tener respuesta ante cualquier situación posible. En lugar de esto, y de acuerdo al modelo *HTM*, el cerebro humano se parece mas a un sistema de memorias que almacena, de forma distribuida entre sus diferentes componentes, información relacionada con las experiencias vividas por el individuo en el pasado. Desde este punto de vista, la inteligencia estaría directamente relacionada con la abundancia y riqueza de las experiencias disfrutadas, y con la capacidad de asimilar nuevas experiencias de una forma útil, que permitiese responder a experiencias similares futuras o adaptar los conocimientos almacenados a experiencias distintas.

3.1. Organización jerárquica de la memoria

Según el modelo *HTM*, una posible modelo de cerebro artificial, estaría organizada como un árbol jerárquico e interconectado de módulos idénticos en estructura y

funcionalidad, cada uno de los cuales implementaría una funcionalidad común de aprendizaje y memoria —o parte de ella—, de manera similar a como hacen las diferentes áreas y sub-áreas especializadas del cerebro humano [143]. Los módulos de las capas inferiores manejan conceptos muy sencillos, incrementando la abstracción y complejidad de estos conceptos a medida que ascendemos en la jerarquía. Los módulos inferiores recibirían su información, en principio, de entradas sensoriales, y proporcionarían los conceptos mas simples a los módulos de su capa inmediatamente superior, los cuales combinarían estos conceptos para formar otros cada vez mas elaborados a medida que la información fluye hacia las capas más altas. Existen además gran cantidad de conexiones de realimentación entre los módulos de las diferentes capas, como ocurre con las neuronas de un área en el cerebro humano (figura 3.8).

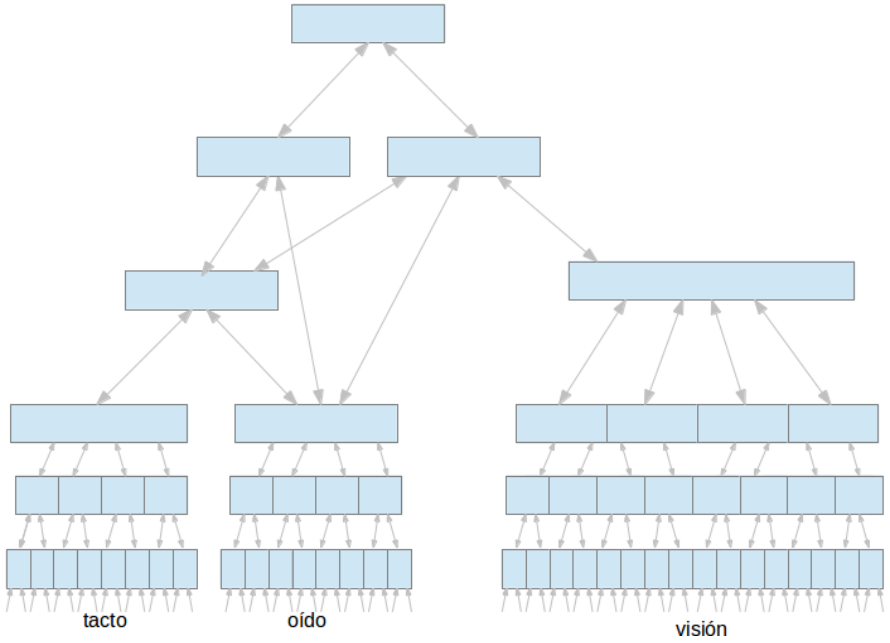


Figura 3.8: Organización jerárquica HTM [1]

Estas conexiones de realimentación permiten reforzar y consolidar los conceptos identificados inicialmente por los diferentes módulos al afrontar una determinada situación, a medida que dichos conceptos son confirmados por los componentes de las capas superiores. Así, por ejemplo al encontrarnos a una persona y verle la cara, las capas de los niveles inferiores de la jerarquía obtienen información de las áreas visuales que les permiten reconocer la existencia de unos ojos, una nariz, o una boca; esta información sería reconocida en módulos o áreas de la capa superior como una cara; y esa cara en particular se pasaría como información a otras áreas superiores relacionadas con el almacenamiento de ”rostros conocidos“, para acabar identificando a esa persona como tu jefe, tu esposa, un actor famoso...o alguien que se parece a alguno de estas personas! Esta organización jerárquica y distribuida presenta ventajas incuestionables, como por ejemplo que cualquier fragmento de información, a cualquier nivel de la jerarquía, en cuanto a complejidad o abstracción del concepto, se puede compartir con cualquier otro módulo de la jerarquía para participar en la creación de nuevos conceptos o consolidar los ya almacenados. Además, y tal como hemos visto en el ejemplo anterior, esta estrategia presenta una gran ventaja respecto a la generalización del conocimiento, evitando largas etapas de aprendizaje pero permitiendo, al mismo

tiempo, la incorporación de nueva información incluso en la fase operativa del sistema. Adicionalmente, se consigue una gran eficiencia en cuanto al almacenamiento de información, en cuanto se evita la redundancia de datos.

Con respecto a la implementación de los módulos o componentes de la arquitectura *HTM*, y aunque en la propuesta original se modelaban como redes similares a las Bayesianas [37], creemos que éste no es un aspecto esencial del modelo, sino más bien una cuestión de implementación. Otras aproximaciones de la *IA*, como las *ANN*, los sistemas de lógica difusa, un sistema *CBR*, o incluso una combinación de estos, u otros paradigmas de la *IA* en la forma de un sistema híbrido, podrían modelar los diferentes módulos de la jerarquía, si bien se deberían realizar los cambios apropiados en cuanto al proceso de aprendizaje y el procesamiento de la respuesta inteligente, con respecto a la propuesta original.

3.2. Importancia del tiempo: predicciones

Otro aspecto clave de la teoría de inteligencia de Hawkins, y que está presente en su modelo *HTM*, es el reconocimiento de la importancia de los aspectos temporales en los procesos cognitivos y de aprendizaje. En cada objeto que vemos, se reconoce fácilmente una estructura jerárquica espacial, al considerarlo compuesto de objetos o elementos mas simples y pequeños, los cuales a su vez están formados por otros elementos más simples, y así sucesivamente. Este modelo del mundo y de sus objetos, como una *estructura nido*, ya ha sido ampliamente utilizado en muchos sistemas de *IA*, especialmente los relacionados con el reconocimiento de patrones, y puede ser trasladado al modelo propuesto, considerando que los módulos inferiores se ocuparían de las partes menores y mas simples —conceptualmente hablando— de la jerarquía espacial, subiendo hacia elementos cada vez mas complejos manejados en capas mas altas. Así un determinado patrón espacial, a cualquier nivel de complejidad, sería el resultado de una coincidencia en la activación de estados o respuestas implantados en diversos módulos de la jerarquía en un determinado instante de tiempo. Por ejemplo, la memoria de cómo es nuestra casa no se almacena en una única región, si no en una jerarquía de regiones que representa en sí la estructura jerárquica de nuestra casa, las relaciones entre sus componentes y el reconocimiento de los mismos (a alto y a bajo nivel).

Pero es precisamente la inclusión de una *estructura jerárquica temporal* la que da su importancia real a este modelo *HTM*. Los módulos, en cualquier capa de la jerarquía, establecen también una correlación temporal entre los patrones que se encuentran cercanos, no solamente a nivel espacial, como los que representan una nariz y una boca en una cara, sino también a nivel temporal, como varias notas musicales consecutivas, que se reconocen como una o mas posibles secuencias de ocurrencia. A un nivel superior de la jerarquía, estas secuencias se identifican como conceptos más complejos, que a su vez se agrupan con otros conceptos de complejidad similar, en nuevas secuencias tanto temporales como espaciales que, de nuevo, se identifican con conceptos aún mas complejos o globales, y así sucesivamente. Por ejemplo, los órganos sensoriales auditivos reconocen una nota a partir de unas variaciones temporales de presión en el tímpano; módulos de niveles superiores podrían reconocer varias notas consecutivas como parte de un *riff* y, finalmente, módulos todavía más arriba en la jerarquía reconocerían este *riff* como parte de una melodía o canción. Se llega así a una conclusión que

puede sorprender en principio, y es que todo lo que nos rodea en el mundo posee una estructura jerárquica dual, tanto en el tiempo, como en el espacio, no importa cuales sean los sentido implicados en captar la información principal asociada a ese concepto; a veces, no obstante, es complicado identificar alguno de estos componentes, el espacial o el temporal, en el mismo.

La principal consecuencia de esta inclusión de una componente de correlación temporal asociada al modelo de inteligencia, es la capacidad de *predecir* una respuesta en los niveles mas bajos, algo que resulta fundamental para evaluar las consecuencia de las posibles acciones que se van a llevar a cabo, y evaluar los posibles desconocimientos o éxitos en la reconstrucción mental de los conceptos a partir de unos patrones de entrada, mediante la comparación de la respuesta esperada con la realmente obtenida.

Por tanto es la capacidad de *predicción*, y no la observación de un comportamiento, la verdadera expresión de la inteligencia, si bien el comportamiento va a ser, generalmente, el resultado o respuesta ante una predicción. La predicción va a ser posible gracias al *aprendizaje* que nos permitirá almacenar, a lo largo del tiempo, patrones de experiencia en el ámbito del tiempo y el espacio, organizados en una estructura jerárquica, y distribuidos a lo largo de los diferentes elementos de las sucesivas capas que forman la estructura del cerebro humano o de la arquitectura de *IA* que se proponga.

Por último, la *Creatividad*, otra de las características típicas atribuidas a la inteligencia humana, estaría relacionada con la capacidad de inferir predicciones para una situación no experimentada ni almacenada en la memoria, por analogía o reconstrucción a partir de datos en principio diferentes y no relacionados.

3.3. Funciones básicas del modelo de inteligencia/aprendizaje HTM

Con respecto al proceso de aprendizaje, el aprendizaje necesario para el desarrollo de la inteligencia supone, según el modelo *HTM*, al menos cuatro pasos o funcionalidades básicas [1] que pueden ser implementadas de muy diversas maneras:

1. *Identificar conceptos en el mundo.* Relacionado con la forma en que la información se reconoce, codifica y almacena en la base de conocimientos. Los principales aspectos a resolver en esta etapa, están relacionados con la forma en que la entrada primaria se particiona y sub-muestrea para ser distribuida entre los diferentes módulos y niveles de la jerarquía; como se representa la información en cada módulo, teniendo en cuenta el nivel de la jerarquía donde está integrado el mismo; y cuanta información se debe proporcionar para un buen aprendizaje.

2. *Inferir conocimiento de las nuevas entradas.* En un proceso similar al reconocimiento de patrones, las nuevas entradas se deben mapear respecto a las almacenadas, o al menos se debe medir su similitud para establecer cuales de las almacenadas son las mas afines, y así decidir una determinada respuesta en función del conocimiento almacenado. Los patrones muy distintos a los existentes y considerados como nuevos, se deberán añadir a la base de conocimiento, haciéndola cada vez más completa. En teoría, si la representación interna del conocimiento se ha establecido de forma acertada, no será necesario

añadir muchos nuevos patrones, ya que la información o conceptos subyacentes serán equivalentes a la mayoría de las entradas.

3. *Realizar predicciones*. Para ello se combinarían las entradas actuales y las más recientes para ajustarse a una secuencia similar que estuviese ya almacenada en la base de conocimiento, de manera que pudiese predecirse cuales son las siguientes entradas más probables, y realizar las acciones o tomar las respuestas adecuadas en consecuencia. Este mecanismo se realizaría en paralelo en cada módulo de un nivel de la jerarquía, y sus salidas alimentarían a los módulos de las capas inmediatamente superiores para donde se realizarían acciones similares. Además, las salidas o respuestas de cada módulo también se realimentarían hacia los módulos de las capas inferiores, permitiendo comparar las predicciones en las respuestas con las entradas reales que se produzcan a continuación. Si las predicciones resultan ser acertadas, se tomarán las acciones asociadas y se reforzarán los conceptos relacionados; en caso contrario, se ascenderá en la jerarquía hacia un concepto más abstracto que permita ajustar la predicción con la nueva situación. Esto nos permitiría, por ejemplo, reconocer un rostro con algún tipo de alteración como un golpe u ojo morado, a pesar de que no hayamos visto nunca ese rostro en dicha situación. La combinación de ciertos elementos reconocibles a bajo nivel —el resto de elementos de la cara que si se reconocen— junto con la generalización en niveles más altos —el cambio de aspecto que sufre cualquier rostro cuando tiene un golpe u ojo morado— nos permitiría reconocer ese rostro como perteneciente a alguien familiar. Estas diferencias entre las predicciones y las entradas reales se incorporarían a la base de conocimientos en los niveles adecuados —ya almacenaríamos el aspecto que tiene ese familiar en particular cuando sufre un golpe en el ojo—.

4. *Dirigir los comportamientos*. Finalmente, las predicciones obtenidas como resultado del proceso de reconocimiento cognitivo, nos permiten realizar ciertas acciones relacionadas con la respuesta predicha. Algunas de estas acciones estarán directamente relacionadas con los estados alcanzados en algún módulo o módulos de forma que estos módulos, además de enviar su salida hacia arriba y hacia abajo en la jerarquía, activarán la ejecución de determinadas acciones o comportamientos, en sentido *"sensar"* $\rightarrow$ *actuar*. Una combinación de varias de estas acciones imbricadas en los módulos podría dar lugar a comportamientos conjuntos o emergentes realmente complejos. No obstante, parece razonable que estos comportamientos directos o reactivos deberían combinarse con comportamientos de alto nivel de tipo deliberativo, que modulasen los de bajo nivel, dando mas peso a unos u otros según las circunstancias. En los siguientes capítulos de esta Tesis se explorará en mas detalle este subnivel dentro del modelo de memoria/aprendizaje, mediante la propuesta de un marco similar de aprendizaje reactivo basado en la experiencia, que se implementará en un robot *AIBO*, y que se probará en un entorno controlado.

El modelo *HTM* ha sido utilizado en varias aplicaciones del campo de la inteligencia artificial, como *Vitamin-D*, para la detección y reconocimiento de personas en imágenes de vídeo; el sistema de análisis energético, *EDSA*, para la detección de eventos en yacimientos petrolíferos [144]; diversas aplicaciones de la empresa aeronáutica Lockheed

Martin, relacionadas con integración de información de sensores y reconocimiento de objetos en entornos urbanos [145]; la *"app"* fotográfica para *iPhone*, *iResemble*, para categorización de fotografías; o *Grokk*, una aplicación de predicción en el ámbito de *"big data"* [146]. El ejercito de Estados Unidos está desarrollando un sistema experto de ciber-defensa que permitiría detectar posibles amenazas a partir del análisis de la información existente en Internet; dicho sistema esta fundado también en la arquitectura *HTM* de Hawkins [147] [148]. En el ámbito de la robótica *HTM* se ha utilizado como base en arquitecturas híbridas basadas en redes neuronales [149] y procesos de Markov con aprendizaje por refuerzo [150], aunque no tenemos noticias de que se haya empleado aún combinado con *CBR* como herramienta de aprendizaje, tal como se hará en nuestra propuesta.

3.4. Críticas al modelo HTM

El modelo *HTM* parece muy prometedor como marco de explicación de la inteligencia humana y un buen punto de partida para el desarrollo de modelos similares aplicables a la *IA*. Muchos de sus aspectos fundamentales han sido reconocidos como válidos e interesantes [151], aunque también ha recibido críticas por parte de otros prestigiosos investigadores en el campo de la *IA* [152] [153], debido a la existencia de ciertos aspectos desconocidos y "agujeros" que no se deben ignorar.

Una de las primeras objeciones está relacionada con la selección de información significativa para el problema o área a la cual se adscribe cada uno de los módulos componentes de la arquitectura, y la representación de dicha información en forma invariante que capture la esencia de la información relevante [109]. Este aspecto se corresponde con lo enunciado en el primer punto de la sección 3.3, "Identificar conceptos en el mundo". Hawkins, en su concepción original del modelo [37], sugiere la existencia de un proceso de aprendizaje no supervisado, en el cual no hay un supervisor que proporcione una información detallada acerca de la estructura de la información de entrada a cada módulo. La información se obtiene mediante técnicas entroncadas con la teoría de la probabilidad o el análisis estadístico de los patrones de entrada, muy similares al *Análisis de Componentes PrincipalesPrincipal Component Analysis (PCA)* tan ampliamente utilizado en el reconocimiento de patrones [2]. Aunque es verdad que mucho del conocimiento humano se obtiene a través de la auto-experimentación y la experiencia, también hay un gran componente de aprendizaje supervisado o dirigido, el cual suele acelerar y facilitar bastante el proceso de aprendizaje. Así que ambos, tanto el aprendizaje no supervisado, como el supervisado, se deben considerar como parte del proceso de aprendizaje tanto en el caso de la inteligencia humana, como en el de la inteligencia artificial.

[2]Se propondrán también alternativas que incluyan otros paradigmas de *IA* como, por ejemplo, reglas de decisión

4

CBR: razonamiento basado en Casos

La experiencia no es lo que sucede, sino lo que haces con lo que te sucede.
Aldous Huxley (1894-1963) Escritor

1. Introducción

CBR (*Case Based Reasoning* o Razonamiento Basado en Casos) es una herramienta del ámbito de los Sistemas Expertos, ampliamente utilizada en muchos entornos de aprendizaje basado en experiencia del campo de la Robótica [154] [155] [156] [157] [158].

CBR es una técnica de adaptación, razonamiento, y aprendizaje, que resuelve un problema actual mediante la recuperación y adaptación de experiencias pasadas [159] [136]. *CBR* propone la catalogación de la información como la clave para ser capaces de usar la experiencia para comprender un problema. Su premisa fundamental es considerar la memoria como un ente dinámico, cuyos cambios son resultado de la experiencia. Los sistemas *CBR* se basan en recordar las soluciones que se dieron en el pasado a problemas similares al actual, y adaptarlas al nuevo problema (figura 4.1). El nuevo problema y su correspondiente solución se convertirá en un nuevo "caso" a almacenar en la base de casos para poder ser también usado posteriormente en nuevos problemas, de forma que, a medida que se obtienen nuevas soluciones, estaremos más capacitados para resolver nuevos problemas aún cuando no dispongamos de demasiada experiencia acerca de ellos. Como se puede comprobar, la filosofía del *CBR* se acerca bastante al modelo de inteligencia que considera a ésta como predicción basada en el aprendizaje por experiencia.

En cuanto a la estructura de una base de casos, es bastante similar a una base de datos ordinaria, con la diferencia de que en estas últimas se realizan búsquedas de registros en particular, mientras que en *CBR* se recuperan registros similares al buscado, en caso de que no se encuentre aquel. Esto nos obliga a establecer una medida de la similitud de los casos, que va a depender mucho de la tarea particular a resolver. Aunque *CBR* pueda presentar cierto parecido con las Redes Neuronales, se diferencia principalmente en que los casos que representan el problema están almacenados de forma explícita, lo cual permite un análisis de los mismos, que nos puede permitir comprender como se almacena el conocimiento. Además, es posible añadir casos iniciales

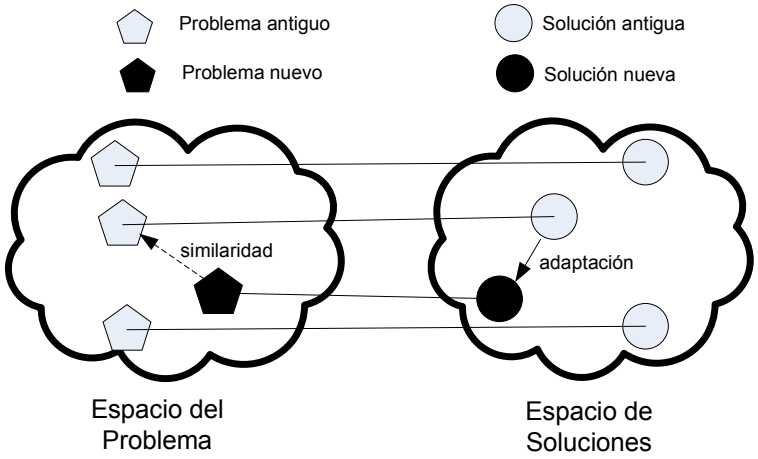


Figura 4.1: Fundamentos de CBR

a la base, obtenidos no por experiencia, si no previa a ella —casos a priori—, que permitan controlar en cierta medida el proceso de aprendizaje, y facilitar la adquisición inicial de un conocimiento preliminar.

Entre las ventajas principales que se han identificado en el modelado de sistemas a partir de *CBR* tenemos:

- *CBR* no necesita partir de un modelo explícito del problema o sistema al que se pretende aplicar, si no únicamente de una colección de casos que pueden ser obtenida a partir de ejecuciones anteriores de dicho sistema u observaciones del problema a resolver, y que describan la experiencia acumulada por el mismo a lo largo del tiempo.

- No realiza una generalización global del dominio sobre el que se aplica, constituyendo una alternativa muy adecuada para aquellas aplicaciones donde coexisten diferentes generalizaciones y métodos que se combinan para alcanzar una solución. Esta coexistencia permite la aplicación de soluciones muy distintas a situaciones aparentemente similares, pero que no tienen ninguna relación entre si, lo cual redunda en una mayor flexibilidad y capacidad de *CBR* para la resolución de problemas.

- Para su implementación solo es necesario ser capaces de describir adecuadamente las experiencias del sistema o problema a resolver en forma de casos. Esto implica un estudio previo para la selección de los atributos más significativos en relación con la experiencia que se desea almacenar.

- Al apoyarse en una colección de estructuras de información, *CBR* permite aplicar aquellas técnicas de búsqueda en bases de datos que permitan gestionar un volúmenes masivos de información de forma eficiente.

- En *CBR* no solo es posible almacenar un conjunto de experiencias predefinido, si no que también se puede aprender de forma incremental a medida que se experimenten nuevas situaciones. Esto permite además una mayor sencillez en el mantenimiento de la base de casos.

- Debido al aprendizaje incremental anteriormente comentado, es posible aplicar *CBR* contando únicamente con un reducido número de casos iniciales que representen un conocimiento limitado del problema o sistema a modelar, en contraste con otras técnicas que requieren una exploración exhaustiva del mismo. A medida que se incorporan nuevas experiencias a través de casos, el conocimiento almacenado por el sistema *CBR* también se incrementará.

- Es relativamente sencillo extraer y analizar, en todo momento, el conocimiento completo que posee el sistema *CBR*, recuperando los casos almacenados en la base de datos. Esto facilita las labores de depuración durante la fase de desarrollo, al poder conocer exactamente con cual caso de los almacenados ha sido identificado un nuevo problema y cual es la solución que se le va a dar. Además, permite un procesamiento de la información de la base de casos a través de herramientas estadísticas para introducir un mayor nivel de abstracción en el conocimiento que posee el sistema, o algún tipo de optimización de la experiencia acumulada en dichos casos.

2. Historia y campos de aplicación

El razonamiento basado en casos, o *CBR* (*Case Based Reasoning*), evolucionó a partir de las teorías propuestas en 1977 por Schank y Abelson, en las que se postulaba que el ser humano se basaba en las experiencias acumuladas en el pasado para resolver los problemas planteados en el presente, aunque estos no hubiesen sido afrontados con anterioridad [159]. Schank planteó el concepto de "guión"(*script*) como una estructura de memoria conceptual para la descripción de eventos. Schank desarrolló sus teorías iniciales para elaborar un modelo cognitivo de aprendizaje basado en experiencia [160], e introduciendo el concepto de "paquetes de organización de memoria"(*MOP*). Este nuevo modelo proponía una teoría de Memoria Dinámica, que permitía relacionar el recuerdo, la compresión, la experiencia, y el aprendizaje en el ser humano, cuya inteligencia estaba basada en una memoria que variaba dinámicamente a medida que iba acumulando más experiencia. Otros autores [161] [162] defendieron también la importancia de las soluciones ante experiencias pasadas para la resolución de problemas actuales y futuros. Posteriormente, el modelo analítico de Schank fue ampliado con la inclusión del razonamiento por analogía [163] [164], y estudios anteriores y posteriores acerca de la formación de conceptos, y el aprendizaje en el ser humano desde el punto de vista psicológico [165] y filosófico [166].

CYRUS [167] [168] es considerado como el primer sistema *CBR* desarrollado a partir del modelo propuesto por Schank. Fue desarrollado por el propio grupo de investigación de Schank en la Universidad de Yale, y sirvió de base para sistemas similares, como *MEDIATOR* [169], *PERSUADER* [170], *CHEF* [171], y *JULIA* [172].

A mediados de los 80, el grupo de investigación de Porter, en la Universidad de Texas, Austin, desarrolló una variante del modelo y sistema *CBR* propuesto por Schank, orientado al aprendizaje de conceptos para tareas de clasificación. Esta rama llevó al desarrollo de sistemas como *PROTOSS* [173], y *GREBE* [174]. Por otra parte, en la Universidad de Massachussets, el grupo de Rissland desarrolló *HYPO* [175] y *CABARET* [176] con vistas a la aplicación de precedente legales en jurisprudencia. En

el MIT, Koton desarrolló el sistema *CASEY* [177] enfocado al modelado de procesos biológicos complejos.

En Europa, los primeros sistemas expertos basados en *CBR* aparecieron a finales de los 80 y principios de los 90. En Abeerden, Escocia, el grupo de Sleeman desarrolló el sistema *REFINER*, para la adquisición de conocimientos a través de casos [178], y posteriormente, *IULIAN* [179]. Richter y Althoff, de la Universidad de Kaiserlautern, desarrollaron en primer lugar *MOLTKE* [180], y mas tarde *PATDEX* [181] y la herramienta *S3-CASE*, con aplicación al diagnostico médico complejo. Aadmot, en la Universidad de Trondheim, Noruega, investigando aspectos relacionados con el proceso de aprendizaje y la integración de casos de experiencia y conocimientos a priori, implementó *CREEK* [182]. En el Reino Unido existen varios grupos de investigación trabajando en la aplicación de *CBR* en el campo de la ingeniería civil, en la Universidad de Salford [183]; en Edimburgo, con el sistema *KICS* [184]; o en pais de Gales [185].

Por último, en España destacamos al grupo IIIA-CSIC de Plaza y López de Mantaras, con sus estudios de aplicación de *CBR* a la diagnosis médica (*BOLERO*) [186] [187] y a la cooperación entre robots [188]; y al grupo KEMLG de la Universidad Politécnica de Catalunya, con la participación en el desarrollo de sistemas como A-TEAM, que combinan *CBR* con otros dominios de sistemas expertos, o E-TOOLS, para la aplicación de *CBR* a la asistencia de discapacitados [189].

CBR tiene una enorme diversidad de campos de aplicación; a partir de sistemas como los que hemos descrito anteriormente y herramientas de desarrollo asociadas a los mismos, ha sido posible aplicar este paradigma en campos tan variados como el asesoramiento legal, control de tráfico de equipos, diagnosis médica, inversión bursátil, gestión de emergencias, etc.

2.1. CBR en el campo de la Robótica

Uno de los campos más prometedores para la aplicación de *CBR* es la robótica. En este área, *CBR* se ha usado principalmente en navegación de robots, principalmente para aprender caminos para *path-planning* global en entornos estáticos [190] [191]. Se han realizado también intentos para *planning* global en entornos dinámicos, a partir de mapas topológicos de une entorno conocido a priori [192]. Sin embargo, Kruusmaa [154] concluye que una navegación global basada en *CBR* solo es interesante cuando existe una gran densidad de obstáculos de cierto tamaño y pocas soluciones al problema. De otro modo, el espacio de soluciones posibles puede resultar demasiado grande.

Otros métodos basados en *CBR* se centran en una navegación reactiva [62] [193] [194], acumulando experiencia en una ventana de tiempo mientras navegan en un entorno buscando conseguir un comportamiento emergente global enfocado hacia un determinado objetivo. *CBR* se ha empleado también en auto-localización basada en marcas en [195]. El autor de la tesis ya propuso en [196] la utilización de *CBR* aplicado a la navegación basada en imagen en un robot tipo *AIBO*, para resolver errores de visión y proporcionar acciones motoras genéricas adaptadas a las condiciones de entrada y la cinemática del propio robot. Otros autores [197] [198] [188] también usaron *CBR* para la resolución de estrategias de juego cooperativo entre varios robots, basándose en la recolección de información de las posiciones de los diferentes robots durante el juego, y representando dicha información a través de casos. Es importante destacar que en ninguna de estas aproximaciones se consideró la separación de la información en una

jerarquía; se utilizó únicamente una base *CBR* en la que cada uno de los casos incluía todos los parámetros a considerar. Esta aproximación puede dar lugar a problemas de escalabilidad del modelo.

Este trabajo propone la organización del modelo de conocimiento en una jerarquía similar a la que se establece en las distintas áreas del cerebro humano, tal como se postula en [37]. Para ello se establecerán diversos módulos de adquisición de conocimientos a partir de *CBR*, de diferentes niveles de complejidad. En las capas inferiores los módulos procesarán entradas sensoriales y motores, generando respuestas a escala temporal reactiva. Dichas respuestas podrán ser directamente aplicadas a los elementos motores/actuadores del robot o constituir la entrada de otros módulos *CBR* de nivel superior que procesarán, por tanto, información mas compleja, y cuyas respuestas se alejarán cada vez más del plano reactivo, hacia el deliberativo, a medida que vayamos subiendo en la jerarquía. Con esta estructura pensamos que se conseguirán las siguientes ventajas:

- Se evitarán problemas de escalabilidad, que impiden aumentar demasiado la complejidad del problema. Cuanto más complejo sea el problema a resolver, mayor número de parámetros o atributos habrá implicados en el mismo, y mayor será también el espacio de posibles soluciones con respecto a las cuales podríamos comparar un caso. Esto daría lugar a una mayor dificultad y lentitud en la búsqueda del caso más adecuado de entre los almacenados en la base de casos *CBR*, por lo que es posible que el robot no pudiese reaccionar adecuadamente ante los cambios en sus entorno para la realización de sus tareas.

- Se facilita la identificación de los atributos fundamentales a incluir en cada uno de los módulos, especialmente en los de bajo nivel. Al tratar con tareas simples, es mas sencillo para el experto el identificar cuales han de ser dichos parámetros, que rangos de valores se deben considerar, e incluso el tipo de distancia a utilizar para estimar la similitud entre varios. A medida que se sube de nivel, se trabaja con información más elaborada con lo que, si las decisiones tomadas en niveles inferiores son correctas, no debería ser complicada deducir nuevos rangos y distancias.

- Se aumenta la eficiencia y velocidad en las búsquedas y recuperaciones de casos más parecido al consultado, al disponer de un espacio de casos menor en cada módulo *CBR*, y tener cada caso un número de atributos más reducido.

- Capacidad de abstracción o descomposición ajustable al problema a resolver; según las características del problema a resolver, podemos partir de módulos *CBR* en las capas inferiores (por ejemplo, que simplemente reconozcan líneas) o abstraer varios de esos módulos en uno solo mas elaborado, para simplificar la arquitectura del sistema. Todo dependerá de las necesidades de la aplicación, y del grado de identificación de los conceptos necesarios para resolver un problema.

- Mayor facilidad para considerar diferentes escalas temporales en el modelo. En las realizaciones actuales en las que se considera un único módulo *CBR* que trabaja con todos los parámetros del sistema, el tiempo se introduce multiplicando la entrada para diferentes instancias de los parámetros en diferentes instantes de tiempo, o construyendo secuencias de los patrones obtenidos como respuesta de la

base *CBR* a lo largo del tiempo. Si bien esta segunda opción puede ser interesante de considerar, la estructura jerárquica lleva inherente en si misma las diferentes escalas de tiempo a considerar a la hora de resolver un problema. Los módulos de nivel inferior pueden realizar consultas *CBR* ante cada instancia de valores de los parámetros sensoriales y motores; sin embargo, es posible construir módulos de nivel superior que vayan coleccionando, no solo los patrones de salida de módulos asociados a diferentes áreas sensoriales/motoras, sino a secuencias de patrones procedentes de un mismo módulo, e incluso a varias secuencias de patrones, cada una de ellas correspondientes a módulos distintos.

- Finalmente, las capas inferiores a nivel reactivo absorberán la dinámica y cinemática del robot, así como los errores sistemáticos de los sensores, a través del aprendizaje. Así, sustituyendo estas capas, se podría reutilizar el resto de la jerarquía cuando se cambie de robot, de una forma ágil e intuitiva.

3. Organización de la información: Bases del conocimiento

Un aspecto fundamental para el uso correcto de *CBR* como herramienta de aprendizaje, se centra en la adquisición de la información necesaria representativa de la experiencia que se desea aprender, así como en la codificación adecuada de dicha información. Richter [199] considera una organización de los sistemas basada en módulos, que manejan el conocimiento o la parte del conocimiento asociada a la tarea o subtarea que es gestionada por ese módulo en particular. A su vez, este conocimiento se clasifica y divide en elementos de conocimiento a los que llama **Recipientes de Conocimiento**(*Knowledge Containers*), que contribuyen a un proceso de aplicación del conocimiento completo, y que pueden gestionarse a través de diferentes mecanismos.

Cada recipiente de conocimiento, al contener únicamente partes o aspectos particulares del conocimiento total del módulo y sistema, no es capaz de resolver individualmente la subtarea asociada al módulo que los contiene; necesita para ello de la contribución del resto de elementos de conocimiento. Otro aspecto clave en este modelo de conocimiento es la forma de expresar los conocimientos en sí mismos, a través de estructuras de datos elementales y operaciones de inferencia que manipulan dichas estructuras. Estas estructuras de datos no tienen porqué identificarse directamente con los recipientes de conocimiento (suelen ser más sencillas en ese caso), pero si que forman parte de estos. De hecho, una misma estructura u operación puede formar parte de del conocimiento encerrado en más de un recipiente de conocimiento y, habitualmente, un recipiente de conoclimiento englobará a varias estructuras de datos o/y operaciones, como base de conocimiento.

La representación y caracterización adecuada de los recipientes de conocimiento que constituyen parte de un módulo asociado al conocimiento o aprendizaje de una subtarea, constituyen aspectos clave en el diseño del sistema de conocimiento. La **compilación** se ocupa de estos aspectos. con el objeto de disponer de un conocimiento completo de todo lo relacionado con la tarea o subtarea a resolver.

3.1. Compilación en sistemas CBR

Para *CBR*, la organización de la información es importante, pero no tan crucial como en otros sistemas expertos. En *CBR* es posible "funcionar" con un conjunto de información incompleto, y con representaciones del conocimiento incompletas o no "perfectas". Aún en estas condiciones el sistema *CBR* funcionará, puede que no de una forma totalmente correcta, o de la manera más eficiente, pero lo hará, a diferencia de otros sistemas expertos "tradicionales" en las que estas carencias e inconcreciones suelen provocar un comportamiento absolutamente incorrecto del sistema. Una ventaja adicional de *CBR* es su capacidad para modificar sobre la marcha el contenido de la base de conocimientos, añadiendo nueva información o modificando la ya existente mediante nuevos procesos de compilación en los que se trasvasa información entre los diferentes recipientes de conocimiento. Esto ocurre, por ejemplo, cuando se evalúa la eficiencia de un caso y se determina que dicho caso no es tan interesante.

3.2. Recipientes de conocimiento en CBR

En [199] se identifican 4 clases de recipiente de conocimiento que, a través de su contenido e interacción, comprimen la información necesaria para el buen funcionamiento de un módulo *CBR*. Estas 4 clases son: i)el **Vocabulario**; ii)**la medida de similitud o distancia**; iii)**la base de casos**; y iv)los **métodos de transformación de soluciones**, que dan lugar a una adaptación del conocimiento existente en la base *CBR*.

Figura 4.2: Recipientes de conocimiento CBR y su interacción [199]

A su vez, cada recipiente de conocimiento se puede subdividir en diferentes sub-recipientes, más relacionados con las estructuras de datos y algoritmos que implementan el soporte de conocimiento; por ejemplo, las distintas partes de un problema pueden estar asociadas a diferentes bases de casos; se pueden emplear diversas funciones de distancia para la medida de similitud de distintos elementos; pueden utilizarse varios métodos de adaptación de soluciones, etc... Como se ve en la figura 4.2, todos los recipientes dependen unos de otros para la representación de la información completa que nos permita llegar a una solución. Por este motivo, una decisión inadecuada en la integración o caracterización de un recipiente puede afectar al resto, dando lugar a soluciones que no sean totalmente correctas, a una menor eficiencia

en la resolución de los problemas, a una menor competencia general del sistema, o a dificultar el mantenimiento del mismo. Un aspecto clave de la forma de organización de la información en un sistema *CBR*; será la capacidad de transferir información de unos recipientes a otros, para buscar la representación del conocimiento más eficiente y sencilla en cada uno de ellos. Esto nos permite construir sistemas *CBR* que inicialmente tienen un bajo desempeño, pero que mejoran su calidad a medida que se refina su estructura a lo largo del tiempo.

3.2.1. El Vocabulario

Cuando hablamos de *Vocabulario* en el campo del conocimiento, nos referimos a las estructuras de datos y elementos de éstas que se emplean para representar aspectos básicos del conocimiento. Por ejemplo, en *CBR* se suele usar una representación *atributo/valor*, aunque se pueden utilizar otras representaciones, como una taxonomía o una jerarquía, a partir de los atributos; y también es posible añadir *predicados*.

En la representación *atributo/valor*, tan importante es escoger los atributos adecuados, como su semántica o forma de expresión. Esta elección de los atributos es una de las tareas más complejas e importantes en las fases iniciales del diseño de un sistema *CBR*, ya que nada nos garantiza que la elección realizada contenga el número y tipo adecuado de atributos que definen el conocimiento necesario para resolver un problema en particular. A este proceso de elección los atributos se le conoce como **indexación** (*indexación*). Algunos autores [200] [171] han propuesto guías para esta labor que, aunque orientadas al campo del vocabulario, pueden ser aplicables a otros campos. A la hora de escoger cuales éstos índices o atributos para un caso *CBR* se debe tener en cuenta que:

- Los índices han de ser predictivos o deterministas.
- Deben refleje los aspectos fundamentales del problema a resolver
- Deben ser lo suficientemente abstractos, como para ampliarse en un futuro...
- ...pero lo suficientemente concreto como para ser reconocido como tal en una ocasión futura.
- El número de índices o atributos no debe ser muy grande, por motivos computacionales, ya que un número elevado de atributos implica un mayor tiempo de comparación entre el caso ofrecido y los almacenados, y por tanto mayor tardanza en la recuperación. Además, a mayor número de atributos por caso, mayor espacio de casos posibles para explicar el problema, con lo que se necesita más memoria y más tiempo para encontrar el caso más adecuado de entre la base. Hay que tener en cuenta que puede que la información elegida tenga cierta correlación convirtiendo alguno de los parámetros en redundantes.

Por lo general se suelen establecer dos criterios para la elección de los atributos:

- *Plenitud principal*. Se consigue cuando todas las propiedades relevantes del módulo o tarea pueden ser formuladas. Si el conjunto de atributos escogido es incompleto, algunas propiedades o características relevantes para el problema no podrán representarse en el mismo, disminuyendo la corrección o eficiencia del sistema. Aún así, el sistema *CBR* será capaz de funcionar.

- *Eficiencia*. Este aspecto está relacionado con la creación de *atributos virtuales* que expresan un conocimiento relevante para el problema que viene dado por una relación entre varios atributos, pero no por la inclusión individual de los mismos. Estos atributos virtuales permiten mejorar el comportamiento de la base de conocimiento.

La forma más usual para escoger estos índices suele ser por medio de decisiones tomadas por un experto humano, que analiza el problema y determina cuales son los atributos más adecuados del mismo, de forma subjetiva. El problema es que la decisión puede no ser correcta u óptima y, en este sentido es adecuado apoyarse en análisis estadísticos y minería de datos para analizar la idoneidad de la elección.

Existen también métodos alternativos automatizados para la elección de los atributos, como la **indexación basada en listas de control**(*checklist-based indexing*) [201] empleada en *MEDIATOR* y *CHEF*, y consistente en analizar el dominio y obtener las dimensiones o descriptores que demuestren gran influencia en la resolución del problema; cada caso sería indexado en función de los valores tomados con respecto a los atributos de esta lista. En *CYRUS*, la selección se realiza en función de las características que permiten diferenciar unos casos de otros. Otras técnicas se basan en el aprendizaje inductivo [202] para identificar aquellos atributos que sean predictivos, y que se emplearán como índices. Por último existen técnicas basadas en "explicaciones" (*explanation-based*), que determinan atributos relevantes para cada caso, investigando cuales de ellos son predictivos [203].

Finalmente, es posible dividir los recipientes de conocimientos en partes o sub-recipientes, como por ejemplo *atributos de recuperación* útiles para la computación de similitudes entre casos de la base CBR; *atributos de entrada*, para crear reglas de conclusión, o de generación de nuevos conocimientos; o *atributos de salida*, que proporcionen información útil para el usuario para por ejemplo, depuración del comportamiento de la base de conocimientos.

3.2.2. La medida de Similitud o Semejanza. Utilidad

La medida de similitud como recipiente de conocimiento, $sim(q, p)$, tiene por objeto el "mapeo" de pares de descripciones de un problema para establecer un nivel de "parecido" entre los mismos. La similitud se obtiene como una función de medidas locales de similitud sim_i en el dominio (o dominios) de los atributos A.

$$sim(q, p) = f(sim_i(q_i, p_i) \mid i \in I) \tag{4.1}$$

En general, la función f suele ser una suma lineal ponderada de las similitudes entre atributos locales de forma que la ecuación anterior queda:

$$sim(q, p) = \sum(g_i \times sim_i(q_i, p_i) \mid 1 \leq i \leq n) \tag{4.2}$$

donde g_i es un conjunto de pesos con coeficientes reales no negativos (generalmente en $[0, 1]$ y normalizados de forma que $\sum g_i = 1$).

Así, en el recipiente de conocimiento que es la medida de similitud, estamos considerando varios sub-recipientes, como son las medidas o funciones de similitud locales, g_i; y la función de similitud global, f (en la ecuación (4.2), un sumatorio). E incluso dentro de estos sub-recipientes encontraríamos otros elementos, como los pesos

asociados a las funciones de similitud locales. Nótese como el conocimiento asociado a las funciones de similitud locales pertenece más al dominio de conocimiento general del problema, mientras que la función de similitud global está mas orientada a la tarea y comprime aspectos de conocimiento relacionados con su utilidad para la tarea, y la relevancia de las similitudes locales en la resolución de la misma.

Uno de los aspectos que consideraremos en capítulos posteriores, una vez hayamos planteado en detalle el problema a resolver, es como establecer las funciones y estructuras más adecuadas para la implementación tanto de las medidas de similitud locales, como una mediad de similitud global.

En la elección de la medida de similitud global deberemos procurar siempre que se cumpla el siguiente *axioma de monotonicidad*:

Axioma 5
Si $sim(q,p) > sim(q,r)$ entonces debe existir al menos un $i \in I$ (y, por tanto, algún atributo) que cumpla que $sim(q_i,p_i) > sim(q_i,r_i)$ [204]

Si este axioma falla en el problema planteado, esto significa, probablemente, que la descripción del problema no es correcta y, por tanto, no hay garantías de encontrar una solución adecuada. Por eso es clave organizar el sistema de forma que el vecino más cercano (atendiendo a la medida de semejanza) a un problema p, permita obtener de la base de casos la solución más **útil** para el problema. Stahl, en [205], introduce una nueva variable, la **utilidad**, como elemento de conocimiento que forma parte de la información almacenada en la base de casos. El concepto de utilidad tiene dos versiones, una relacional, y otra funcional. Si planteamos un problema (o caso de entrada a una base *CBR*), p, y tenemos dos posibles soluciones, s_1 y s_2, en el espacio de soluciones del dominio de conocimiento, vamos a encontrar:

- Una relación de preferencia entre las soluciones,s_1 y s_2, $pref(s_1,s_2)$, que representa que, ante el problema p, es más útil la solución s_1, que la s_2 (definición relacional de utilidad).
- Una función real $u(p,s_1)$, que determina un grado de utilidad de la solución s_1 asociada a p, dentro de una escala absoluta. Todas las posibles soluciones s_i tendrán un determinado valor de utilidad en esa escala para dicho caso (definición funcional de utilidad).

Dependiendo del escenario de aplicación, la utilidad puede venir determinada por diferentes aspectos, como la corrección de las soluciones sugeridas; el grado de aplicabilidad o reutilización de las acciones o decisiones asociadas a esas soluciones; un grado de satisfacción del usuario del sistema; o una ganancia de información acerca del sistema para el usuario, a partir de las soluciones, o salidas.

En *CBR*, en una primera aproximación, se puede identificar la función de utilidad con la medida de similitud, de forma que para un problema q y un caso (p,s), la ecuación $sim(q,p) = u(q,s)$ se puede considerar válida. No obstante, en una aproximación más compleja y evolucionada, la medida de similitud incluirá, no solo el "parecido" entre el problema presentado y cada uno de los casos almacenados, sino también la eficiencia de la solución asociada a dichos casos para la resolución del problema presentado. En este sentido, más adelante veremos como incluir el concepto de eficiencia de la solución en la fase de recuperación en la base CBR, y como dicho concepto está muy ligado a las características y condicionantes de la tarea a resolver por parte del módulo CBR.

3.2.3. La base de casos

El papel de esta base es contener las experiencias en el formato indicado por el vocabulario. Las experiencias se obtienen a partir de los casos almacenados en la base o de variaciones de los mismos. Entre los aspectos más importantes para el diseño de una "buena" base de casos tenemos:

- Tratar de contener solo casos (p, s) para los que la utilidad de s es máxima o al menos la mejor posible. Esto significa que se deberían evaluar los casos almacenados tras la fase de adquisición o aprendizaje por experiencia realizada por el entrenador, a través de una función de utilidad/recompensa, y eliminar aquellos que no aporten ventajas reales.
- Almacenar el mayor número de casos posibles para aumentar la competencia del sistema.
- Obtener una base de casos de menor tamaño posible para disminuir el tiempo de búsqueda. Para ello, de nuevo, tendremos que estimar la eficiencia de los casos almacenados.

3.2.4. La transformación de la solución

Este aspecto del conocimiento hace referencia a como el conocimiento obtenido a partir del vecino mas cercano, puede no ser suficiente para resolver el problema planteado, bien porque la medida de similitud no es la más adecuada, o porque la base de casos no contiene una solución mejor, y la obtenida no es suficientemente correcta. La adaptación de soluciones suele seguir una serie de reglas que son el componente principal de este recipiente de conocimiento Por tanto, el conjunto de problemas que puede resolver el sistema está determinado por los casos almacenados en la base sometidos a la aplicación de las reglas. Un problema es que la aplicación de las reglas no tiene porque dar lugar, necesariamente , a un aumento de la similitud original obtenida tras la consulta a la base *CBR*. De nuevo, en la transformación de las soluciones jugará un papel importante la medida de la eficiencia de las mismas en la resolución de la tarea asociada al módulo.

3.3. Adquisición y Trasvase de conocimiento entre recipientes

En los apartados anteriores hemos visto como el conocimiento de un sistema experto, en nuestro caso, un sistema *CBR*, se va a organizar en la forma de diferentes recipientes de conocimiento que responden a distintas estructuras, representaciones, y relaciones. Teóricamente podrían existir recipientes de conocimiento "ideales" que, en sí mismos, y sin colaboración de otro tipo de recipientes comprimiesen todo el conocimiento necesario para el aprendizaje de una tarea.

- En el caso del vocabulario, el recipiente ideal e único estaría constituido, para cada descripción del problema, p, por un solo atributo virtual en el dominio de todas las soluciones, *sol*, y construido a partir de todos los atributos significativos del problema. Este atributo ideal cumpliría que $s = sol(p)$ nos permitiría obtener la solución correcta, para cualquier problema p.

- Una medida de similitud ideal sería aquella que nos diese una similitud absoluta $sim_{ideal}(q, p) = 1$ si el caso (p, s) nos permite obtener la mejor solución, s; y nos daría $sim_{ideal}(q, p) = 0$ en caso contrario.

- Una base de casos ideal sería aquella que contuviese **todos** los casos posibles en el dominio del problema.

- Y la transformación de soluciones ideal sería aquella que pudiese ignorar completamente los casos, y construir una solución desde 0, simplemente aplicando las reglas de adaptación asociadas.

Esto claro que estos recipientes de conocimiento ideales *no existen*, pero su simple proposición nos plantea la cuestión de *como y cuando repartir el conocimiento del sistema* a lo largo de los diferentes tipos de recipientes. Respecto al cuándo, se van a considerar generalmente dos momentos o etapas para establecer este reparto: antes de poner en marcha el sistema; y una vez el sistema está ejecutándose.

En un sistema *CBR* se puede incrementar el conocimiento a lo largo de su ejecución, especialmente en lo que respecta al contenido de la base de casos, lo cual implica que no es necesario partir de un conocimiento inicial completo y comprendido del problema para poner en marcha el sistema. Esto es una gran ventaja respecto a los tradicionales sistemas de lógica programada, en los que el conocimiento será el codificado antes de la puesta en marcha del sistema, y permanecerá inmutable durante su funcionamiento. Será necesario, no obstante, antes de la puesta en marcha del sistema, resolver aspectos como la definición de los atributos a considerar, y la recogida de (algunas) muestras de los mismos; la elección de una medida de similitud adecuada; y el establecimiento de unas reglas de adaptación eficientes y operativas. Una buena elección en todos estos aspectos, permitirá aumentar la eficiencia del sistema, no solo la inicial, sino también la mejora que se logrará con la incorporación de nuevos conocimientos durante su funcionamiento. Para optimizar las características de los recipientes en cuanto a su aportación al conocimiento completo del problema, se puede optar por mejorar individualmente el conocimiento comprendido dentro de cada recipiente (y sub-recipiente) de conocimiento; o traspasar conocimiento entre los diferentes recipientes, dado el elevado grado de acoplamiento entre los mismos. Tanto en uno como en otro tipo de mejora, existen tanto técnicas basadas en intervención humana como por aprendizaje automático de máquina (*machine learning*).

3.3.1. Mejora de información en los recipientes individuales

- Vocabulario. Para mejorar o refinar los atributos que conforman los elementos básicos del conocimiento, se tratará de eliminar aquellos atributos redundantes o con dependencias funcionales, y añadir atributos virtuales útiles (lo cual conlleva a menudo la eliminación de los atributos desde los que se forman). La inclusión de atributos virtuales suele ser necesaria debido a fallos en el axioma de monotonicidad visto más arriba; este método es inductivo y bastante complejo en la actualidad, no existiendo demasiado progreso en cuanto a métodos automáticos para conseguirlo.

- Medida de Similitud. No hay métodos sistemáticos para determinar cuál es la mejor medida de similitud para un dominio concreto. Las principales

investigaciones se centran en determinar los pesos de la suma ponderada en la función global [206] [207]. En [208] podemos encontrar un estudio comparativo de medidas de similitud para dominios en el que los atributos del problema son categóricos. La elección de la medida de similitud está muy condicionada por el dominio del problema que se desea resolver, y necesita de la intervención de un experto. Es posible refinar algunos aspectos mediante estudios comparativos, aunque en todos ellos hay que tener en cuenta que el elevado acoplamiento de los recipientes de conocimiento puede dificultar la interpretación de los resultados.

- Base de Casos. Es conveniente tratar de eliminar aquellos casos que no contribuyan a la resolución del problema, aunque esto no es siempre obvio, ya que un caso que no sirve por ahora puede hacerlo en el futuro. Es importante establecer un baremo de calidad de cada caso en la base. Esta calidad se puede medir en relación a la competencia general del sistema [209] [210], o a una orientación utilitarista hacia el problema a resolver [211]. Este apartado está muy relacionado con la denominada fase de *mantenimiento* del proceso *CBR*

- En cuanto a la transformación de soluciones, no hay demasiados trabajos que estudien la mejora sistemática de las reglas de adaptación en el dominio *CBR*. En la práctica, se suelen emplear las técnicas estándar de los sistemas basados en reglas. Podemos ver algunas aplicaciones de estas técnicas a *CBR* en [212] [213] [214] y [215].

3.3.2. Trasvase de información entre recipientes de conocimiento

Es otro aspecto clave en la mejora de la información contenida en los recipientes. Nótese que un trasvase de información de un contenedor a otro no siempre supone que se elimine la información del contenedor original. Se recomienda el siguiente procedimiento de diseño del conocimiento de un sistema:

1. Comenzar con un vocabulario dado u obvio.
2. Establecer un conjunto de casos
3. Considerar una medida de similitud simple y/u obvia.
4. No utilizar, inicialmente, reglas de adaptación.

A continuación se procedería a mejorar el sistema mediante técnicas de compilación horizontal y vertical:

- Compilación Vertical. Trasvase entre diferentes niveles de abstracción. En [216] se muestra un ejemplo de trasvase de conocimiento entre la base de casos y la transformación de las soluciones.

- Compilación Horizontal. Dentro del mismo nivel de lenguaje, buscar una formulación mas compacta. Se busca mejorar la eficiencia, el tamaño de la base, o su inteligibilidad. Por ejemplo, reemplazar características por variables, omitir detalles, o introducir nuevos niveles de abstracción. En CBR se podría, por ejemplo, añadir casos generalizados que engloben conjuntos de casos con soluciones similares [217], apoyándose en técnicas de *clustering*.

Adicionalmente se puede mejorar el conocimiento de un recipiente apoyándose en el contenido en otro recipiente distinto, pero sin trasvase efectivo del mismo. Por ejemplo, se pueden optimizar las medidas de similitud a partir de las reglas de adaptación como en [218].

4. Aspectos básicos de un sistema CBR

A la hora de utilizar *CBR* para la resolución de cualquier tipo de problema, o para el modelado de algún sistema complejo, hay una serie de decisiones importantes que es necesario tomar, y pasos a seguir para el diseño e implementación de la estructura *CBR*. Los principales aspectos y elementos a tener en cuenta son:

1. Estructura y características del caso a considerar
2. Estructura u organización de la base de casos
3. Operaciones sobre la base *CBR*: el ciclo *CBR*

A continuación realizaremos una breve introducción a estos aspectos. En capítulos posteriores profundizaremos sobre los mismos,indicando las decisiones tomadas para la implementación del modelo CBR empleado en nuestro sistema.

4.1. El caso CBR

Un aspecto básico para el funcionamiento correcto del *CBR* es la descripción adecuada del problema que queremos resolver o, más bien, de las experiencias que hemos tenido con respecto a dicho problema, y que servirán de base para afrontar futuras experiencias expresadas en términos similares. Como hemos comentado anteriormente, las experiencias que constituyen la base de conocimiento del sistema se almacenan como una colección de **casos**. El concepto de caso entronca con el "paquete de organización de memoria" o *MOP* (*Memory Organization Packet*) de la teoría de Schank; un caso se puede definir como una unidad de conocimiento dentro de un contexto, que describe una determinada experiencia [219]. Un caso debe contener, al menos, un conjunto de atributos o parámetros que permitan describir adecuadamente el problema a resolver, y una solución que indique la respuesta a ejecutar ante esa instancia del problema, o permita clasificar/categorizar la misma (*CASEY* [177]). A menudo los casos suelen incluir también una métrica del grado de la eficiencia resultante, con respecto a un objetivo, al aplicar dicha solución (*MEDIATOR* [169]). De esta forma se espera solucionar nuevos problemas a partir de las soluciones existentes ante problemas parecidos, evaluar nuevas situaciones, y prevenir problemas que puedan surgir de aplicar la solución almacenada ante un problema parecido al actual.

La forma exacta de representación de la información es muy variable, y abarca un amplio rango de formalismos típicos del campo de la Inteligencia Artificial, como por ejemplo reglas y redes semánticas, predicados, objetos o tramas (*frames*, [131]). Estas dos últimas representaciones son las mas usadas en la mayoría de los sistemas CBR actuales.

Aparte de la propia información incluida en el caso, otro aspecto fundamental es la estructura final de dicha información dentro del caso, la cual estará muy ligada a

la estructura de la base de casos empleada para su almacenamiento y a los métodos utilizados para explorar dicha base a la búsqueda de la solución más adecuada a un nuevo problema. Esta elección a menudo depende de las características o dominio del conocimiento del problema a resolver. En [136] y [220], se propone una clasificación del caso, dependiendo de las características de la información que encierre, respecto al dominio del conocimiento al que pretende identificar:

- casos **concretos**, que representan experiencias concretas; o **abstractos** para episodios más generalizados.
- casos **completos**, que representan unidades de conocimiento separadas; o **parciales**, en el que la información de la experiencia está distribuida en varias sub-unidades.
- casos **aislados** entre si; o **relacionados** unos con otros.

Podemos distinguir tres representaciones básicas del caso: i)plana; ii)jerárquica(estructurada); o iii)generalizada. Esta representación va a influir en la organización de los casos dentro de una base de casos.

4.1.1. Representación Plana

En la representación plana, cada caso es entrada o fila de una tabla, donde cada columna se corresponde con un atributo del caso. Por tanto, cada caso se representa mediante un vector $c = (a_1, a_2, ..a_n) \in D_1 \times D_2 \times ... \times D_n$, siendo D_i el dominio del atributo en la posición $i-sima$. Cada atributo representaría una dimensión en el espacio multidimensional que representa al caso, y por ende a la experiencia. Desde este punto de vista, un caso se puede representar como un punto en el espacio n-dimensional, donde sus coordenadas vienen dadas por los valores particulares de sus atributos. En el caso de que el valor de algún atributo no se conozca, no es posible identificar esta posición, pero si un objeto n-dimensional que incluya el caso buscado, con todos los valores posibles para el atributo desconocido.
La representación plana de una base de casos constituye la alternativa mas sencilla de implementación, con la ventaja de que siempre se va a obtener el caso más parecido dentro de la base, al realizarse una búsqueda exhaustiva en la fase de recuperación. Además, la incorporación de nuevos casos es muy simple: basta con incluir el caso como una nueva entrada al final de la tabla. Su principal inconveniente es el crecimiento exponencial en el tiempo de respuesta del sistema si el número de casos almacenados es muy elevado, al necesitar realizar un mayor número de comparaciones para encontrar el caso más cercano. Por otra parte, esta estructura no es capaz de capturar las dependencias y relaciones entre atributos, con lo que la similitud de casos no siempre se establece de la forma más adecuada. Por estos motivos, este tipo de representación no es la más aconsejable para problemas complejos con gran número de atributos.

4.1.2. Representación Estructurada o Jerárquica

Para subsanar los problemas antes mencionados de la representación plana, Watson y Pereira [221] sugirieron en la división de un problema complejo en subproblemas más simples, algo realizable cuando no existe un alto grado de dependencia entre los

subproblemas. Cada subproblema tendría su propia base de casos y se le aplicaría el método CBR más adecuado al dominio del problema, obteniendo un mejor resultado conjunto.

Los casos de cada subproblema o sub-casos, representarían ejemplos específicos del subproblema a solucionar, y la combinación de los mismos representaría un caso del problema global. Desde este punto de vista la información de un caso del problema global se organizaría de forma jerárquica, con una serie de rasgos comunes o contexto en el "tronco" del árbol y con los rasgos propios de cada subproblema en las "ramas" y "hojas". Plaza defiende también en [222] una representación estructurada del caso, argumentando una mayor riqueza en expresividad en relación con la representación plana. Plaza define el concepto de *feature terms* o términos característicos, que representan una generalización de los términos de primer orden, permitiendo definir conceptos de más bajo nivel que, agrupados, construyan conceptos de nivel superior, de forma jerárquica.

Otro problema interesante de este tipo de representación, se centraría en tratar de expresar las interrelaciones entre atributos y conceptos dentro de cada caso. Sanders [223] llama *feature-based representations* a las representaciones de casos que no expresan interrelaciones entre atributos, y determina que, aquellas que si expresan interrelaciones, *graph-structured representations* ,deben cumplir las siguientes características:

- Flexibilidad para expresar las relaciones entre dos componentes de un caso.
- Diversidad. Posibilidad de que estas relaciones cambien de caso a caso.
- Ampliación del conjunto de relaciones definidas en caso necesario.

En caso de querer expresar relaciones entre atributos, deberemos expresar éstas a través de valores que toman los atributos, no del propio atributo en sí (representación orientada a Objetos).

El principal inconveniente de este tipo de representaciones es el aumento, a veces excesivo, en la complejidad de los algoritmos de exploración de las bases de casos, la mayor dificultad para incluir nuevos casos dentro de la base, y la posibilidad de no recuperar el caso óptimo en algunas situaciones.

4.1.3. Representación Generalizada

La representación Generalizada de casos surge como posible solución a los problemas de crecimiento de la base de casos para problemas complejos. Ya hemos comentado anteriormente que un caso se puede ver como un punto en un espacio n-dimensional, donde n es el número de atributos que lo definen. En principio cada uno de esos puntos tendría definida una solución individual al problema o experiencia que representa. Por lo que el espacio total de representación de casos vendría definido por el producto de el espacio del problema por el espacio de soluciones. Para reducir este espacio del caso, se introduce [217] el concepto de *Caso Generalizado*, como aquel que asigna soluciones a varios problemas en vez de a una solo. La soluciones que forman parte de un Caso Generalizado representan pequeñas variaciones de una solución fundamental. El caso generalizado incluiría todos los casos individuales cuyos atributos satisfacen unas restricciones de valores determinadas. Esta organización permitiría, por ejemplo,

que un caso individual perteneciese a varios Casos Generalizados, siempre y cuando cumpliese las restricciones fijadas para los mismos. Para realizar comparaciones de un nuevo caso respecto al contenido de la base, se mediría la similitud del mismo respecto al Caso Generalizado, en vez de hacerlo con todos los casos que lo integren.

Este *clustering* de casos permitiría hacer una primera búsqueda más rápida, para poder a continuación examinar solo los casos individuales que forman parte del Caso Generalizado escogido. Es posible emplear métodos generales de *clustering* para la obtención de estos Casos Generalizados [224], ya que cada caso representa a un individuo de un espacio, y existe una noción de similitud o distancia inherente al *CBR*.

En el modelo que proponemos se considerará una representación híbrida en la que se empleará, siempre que sea posible, una estructura jerárquica navegada a través de una información "de contexto". Esta información permitirá separar, de forma unívoca, diferentes tipos de escenarios, evitando los problemas típicos de la representación jerárquica pura que pueden llevar hacia ramas equivocadas de la estructura. Al final de cada rama tendremos una base casos con representación generalizada, construida a través de la discretización del espacio de casos en representantes asociados a conceptos abstractos. Estos conceptos serán elegidos en función de las características del comportamiento que se desea aprender. Aspectos mas detallados de esta estrategia se podrán ver en el ejemplo de diseño de comportamiento propuesto en el apartado 3.2.

4.2. Estructura de la base de casos

Una **base de casos** es una colección de casos que representan un conjunto de experiencias acerca de un determinado dominio o problema, y que incluyen un conocimiento de similitud entre todos los casos incluidos. La estructura de almacenamiento de casos de la base de casos está muy relacionada con la estructura de los propios casos, e influye de manera importante en las prestaciones del sistema *CBR* para el campo de aplicación. Debe buscarse un equilibrio entre dos objetivos en conflicto: el almacenamiento de información con **suficiente riqueza semántica**, para reflejar los conceptos que encierran los casos; y facilidad de manipulación de los casos, lo que implica búsquedas rápidas y eficientes en la base y facilidad para añadir nuevos casos. Dos de los modelos de bases de casos más empleados son el **modelo de memoria dinámica** [160] [167], y el **modelo de categorías y ejemplares** [136].

- El modelo de memoria dinámica fue propuesto por Schank y empleado por Kolodner en su sistema *CYRUS*. Se basa en la organización de los casos (MOPS) en **episodios** (*General Episodes*,*GE*). Un episodio es una estructura jerárquica e **indexada**, que facilita la manipulación de los casos dentro de la base *CBR*. Un episodio agrupa tres tipos de elementos: casos; **normas**, que representan características comunes a todos los casos que forman parte del episodio; e **índices** que representan características que distinguen unos casos de otros dentro del episodio, y que permiten apuntar a un caso particular o a un episodio más especifico y de nivel inferior en la jerarquía, cuyos casos comparten esta característica. La recuperación de un caso particular exige una búsqueda descendente, partiendo de episodios generalizados de nivel superior, hacia nuevos episodios de nivel inferior, hasta llegar a un caso particular dentro de estos. El modelo se dice dinámico, porque la aparición de una nueva característica

común a varios casos o distintiva de uno en particular, provoca la creación de nuevos episodios en la memoria, permitiendo adquirir nueva información, no solo específica, si no general. Un aspecto clave en este modelo es la **indexación** o creación de **índices** de la que hablaremos más adelante.

- En el modelo de categorías y ejemplares fue empleado en el sistema *PROTOS* [173], a los casos se les denomina **ejemplares**, y se agrupan en **categorías**, pudiendo pertenecer a varias a la vez. La pertenencia a una u otra categoría viene dada por la existencia de determinados rasgos; pero aunque no todos los rasgos estén presentes, un caso puede pertenecer a una categoría, aunque se considerará que tiene una pertenencia débil o, al menos, menor que otro caso que sí incluya todos los rasgos de la categoría. Además, los casos y categorías se encuentran relacionados entre si mediante grafos que pueden indicar diferencias entre casos, grado de pertenencia a una categoría, o inclusión de un rasgo dentro del prototipo de una categoría. La búsqueda de casos se realiza examinando las categorías que incluyan varios de los rasgos propios del caso, incorporando el caso a la categoría que tenga mayor semejanza en los mismos.

Otro aspecto fundamenta en el buen funcionamiento de CBR es el **mantenimiento** de la base de casos. En general, a medida que se "viven" nuevas experiencias, éstas han de ser incorporadas a la base de casos, para mejorar el conocimiento del dominio del problema, pero evitando el almacenamiento de información redundante para impedir un crecimiento innecesario en el tamaño de la base [209]. En otras ocasiones, la consideración ante un caso almacenado cambia, por lo que éste debe ser modificado, o incluso eliminado del sistema [225]. Esto implica la necesidad de un análisis periódico de la base de casos. Existen guías para ayudar al mantenimiento de la base de casos, especialmente en el campo de análisis conversacional [226]. En [227] se propone un test para determinar si el contenido inicial de la base de casos es el adecuado para resolver un problema; se basa en dividir los casos en un grupo de entrenamiento y un grupo de control y observar como se desenvuelve el sistema, modificando el contenido de ambos grupos. También es importante analizar la similitud entre los casos almacenados, para eliminar los demasiado parecidos entre sí, o agruparlos a través de un prototipo; en [228] se propone un análisis de probabilidad de similitud entre casos aleatorios. Riesbeck [229] propone establecer dos grupos de casos: un grupo estándar de casos prototipo, de manera que la mayoría de casos del dominio sean variaciones de los mismos; es posible que existan varios casos ejemplo que representen un prototipo, para abarcar el mayor número posible de atributos. y un grupo de excepciones, casos que caerían en la descripción de un prototipo, pero son manejados de otra forma debida a alguna circunstancia especial. Lieber [230] ligó el mantenimiento de la base de casos, con el coste computacional en la fase de adaptación (se verá posteriormente), y McKenna [231] estableció una competencia del sistema en función de el número y el tipo de problemas que fuese capaz de resolver, indicando que era necesario evitar la existencia de casos redundantes. De cualquier forma, la evaluación de las mejoras en el sistema al añadir nuevos casos, se debería realizar de forma empírica, mediante la ejecución de experimentos con dicha base de datos aumentada,cuyo rendimiento se analizaría de forma estadística. Otro aspecto de mantenimiento de la base de casos se refiere a la eliminación de casos ya existentes, como consecuencia del añadido de otros casos [225] [232]. En la medida en que todos los casos no contribuyen por igual

al conocimiento del sistema, en caso de eliminar alguno deberían ser los que menos lo hacen [225]. Finalmente, Racine y Yang definen en [233] unos criterios de calidad de una base de casos, en función de:

- **Consistencia**. No debe haber casos que contradigan el conocimiento a priori del dominio u otros casos en la base.
- **Corrección**. Es la frecuencia media de recuperación de casos que pueden ser usados mas eficientemente en el sentido de calidad a la solución del problema, y facilidad de adaptación.
- **Redundancia**. Tiende a incrementarse a medida que crece el tamaño de la base de casos, algo que es necesario evitar.
- **Coste de Adaptación**. Debe reducirse al máximo para incrementar el rendimiento del sistema. Suele entrar en conflicto con la redundancia, ya que el coste de adaptación es menor cuanto mayor es la base de datos.
- **Cobertura**. Se define con el conjunto de casos del dominio que pueden resolverse mediante la adaptación de los casos de la base. Idealmente debe tender al espacio del problema a resolver.
- **Coste de Recuperación**. Número medio de accesos a la base, necesarios para recuperar un caso apropiado.
- **Relevancia**. Número de casos relevantes recuperados, respecto al número de casos irrelevantes, en un conjunto de casos recuperados. Depende de la elección adecuada de la función de similitud.

La base de casos propuesta seguirá, en principio, una variante del modelo de memoria dinámica, en el que se recorrerá una estructura jerárquica según una información de contexto, hasta alcanzar una base en la parte final de la estructura. Aquí se realizará un primer filtrado de los casos mediante una indexación de objetos en escena, que permite una recuperación de situaciones parecidas al problema en cuanto a los objetos que aparecen en el escenario. Posteriormente se seleccionará el caso mas adecuado mediante una función de distancia. En trabajos futuros se explorará la posibilidad de categorizar los casos a través de una descripción del escenario que representan, lo cual nos permitiría añadir también aspectos propios de la representación mediante categorías y ejemplares. En cuanto al mantenimiento, se contemplará la adquisición de nuevos casos a través de la experiencia, para aquellas situaciones o problemas demasiado distintas a las almacenadas en la base de conocimientos. Sin embargo, actualmente no se contempla en el modelo propuesto un mantenimiento de la base de casos basado en el uso de los casos o en la eficiencia medida en la aplicación de los mismos, ni tampoco la eliminación de casos considerados poco útiles. En el capítulo 5 se indicarán con más detalle las decisiones tomadas en la características y estructura de la base *CBR* que integrarán los comportamientos desarrollados, así como la línea de razonamiento y motivaciones que han llevado a la elección de dichas alternativas.

5. Operación de un sistema CBR: El ciclo CBR

Todo sistema*CBR* desarrolla su operación a través de un ciclo esquemático(Figura 4.3) denominado **ciclo CBR**, y que se compone de cuatro fases [136]:

- **Recuperación** del(os) caso(s) de la base más similar(es) al presentado en la situación actual.
- **Reutilización** de la información almacenada en el caso recuperado para resolver el problema.
- **Revisión** de la solución obtenida para determinar su viabilidad en la resolución del problema y adaptación de la misma en caso de ser necesario.
- **Retención** de la nueva solución obtenida como parte de un nuevo caso en la base de datos.

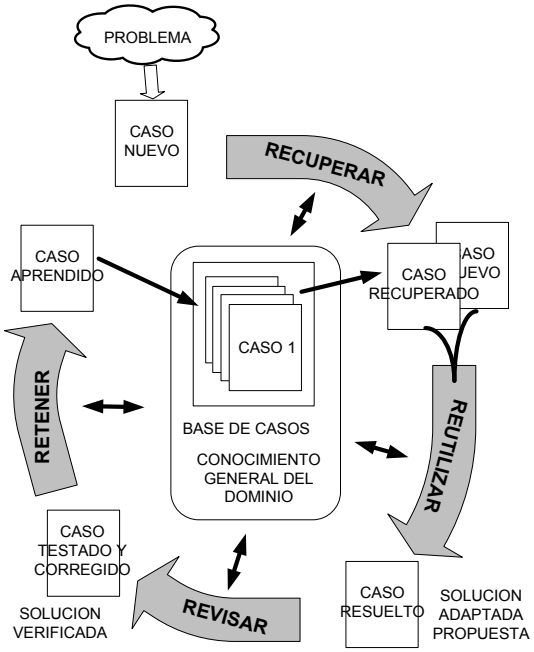


Figura 4.3: El ciclo CBR

En general, ante la llegada de un nuevo problema a resolver, primero se describe éste a través de la estructura definida para un caso en la base *CBR*. A continuación se compara este nuevo caso con respecto a los casos almacenados en la base de casos, y se obtiene el(los) mas parecido(s); en principio los casos recuperados deberían representar situaciones que fueron experimentadas en el pasado y que son similares a la que se está presentando en la actualidad. Por tanto, la solución que se aplicó en los casos recuperados de la base, parece en principio bastante adecuada para la resolución del problema o caso actual. En algunas situaciones, el caso recuperado no va a ser tan parecido al propuesto como cabría desear, por lo que la solución recuperada deberá ser adaptada a la nueva situación/experiencia y, tras su aplicación, deberá evaluarse si resuelve correctamente el problema planteado. Si esto es así, el nuevo problema, junto con la solución adaptada, se convierte en un nuevo caso a añadir a la base de casos *CBR*.

Hay que destacar que cada una de las fases conlleva una serie de sub-fases o sub-tareas que detallaremos a continuación. Es frecuente que se necesiten conocimientos generales del dominio de la aplicación durante todo este proceso, a menudo a través de una intervención humana (por ejemplo,para evaluar si la adaptación de un caso ha resuelto correctamente un problema).

Algunos autores [234] añaden una nueva fase de evaluación (*review*) de la solución obtenida tras la fase de revisión, antes de la aplicación al problema o de una nueva revisión.

5.1. Fase de Recuperación

La fase de recuperación se inicia con una descripción total o parcial del problema a resolver mediante el sistema *CBR*, y finaliza con la localización del caso almacenado que mejor se ajuste a las características del presentado. Esta operación se realiza a través de una serie de sub-fases que incluirían: **Identificación de características** , **Emparejamiento inicial**, y **Selección**.

La identificación de características consiste simplemente en expresar el problema en los mismos términos de descripción utilizados para almacenar la información en la base de casos, permitiendo la construcción de un caso de entrada (suponemos que las decisiones de elección de índices o atributos que representarán el dominio del problema, ya se han tomado). Para ello se utilizan descriptores capaces de convertir unos datos de entrada en información formateada según la estructura de representación del caso. En esta fase se realiza el filtrado de posibles datos ruidosos o inválidos, y se solucionan problemas de ausencia de características que deberían estar presentes en la información de entrada, pero cuyos valores no se conocen. Una vez realizada la identificación la recuperación se lleva a cabo en dos pasos; en el primero, el emparejamiento inicial, se buscan una serie de candidatos posibles, y en el segundo, la selección, se selecciona el mejor de entre ellos, a través de una medida de similitud con respecto al caso de entrada. En algunas variantes, los dos pasos se realizan en uno solo. A diferencia de otras bases de datos convencionales, donde se busca un dato en concreto que puede no ser localizado, en las bases *CBR* siempre se recupera algún caso, que puede ser más o menos parecido al caso de entrada. Entre las técnicas consideradas para esta función de búsqueda del candidato más adecuado dentro de la base de casos tenemos [219]:

- **Vecindad**. Esta técnica es la más empleada actualmente. Implica establecer una medida de similitud entre el caso de entrada y los casos almacenados, mediante un sumatorio ponderado de una función de distancia entre los valores de cada uno de los atributos [201]. Es necesario elegir una función de distancia adecuada [235], y establecer el peso que va a tener cada atributo en la medida de similitud del caso. A medida que aumenta el número de casos almacenados, la búsqueda tardará más tiempo; una posible solución consistiría en realizar búsquedas paralelas dividiendo la base de casos entre varios procesadores. Aspecto claves para el buen funcionamiento de esta técnica son la elección de una medida de similitud adecuada,y la asignación adecuada de los pesos, en función de la importancia de cada uno de los atributos para el problema a resolver.

- **Inducción o búsqueda en árbol**. Se utiliza cuando el caso tiene una estructura jerárquica, donde alguno de los atributos permite una identificación inicial de un

grupo de casos, y el resto discrimina entre estos [236]. Se realiza una búsqueda de tipo árbol. Es una técnica interesante cuando hay algún atributo que es claramente más decisivo a la hora de caracterizar un caso.

- **Inducción guiada por Conocimiento**. Es una variante del anterior, en la que se identifican manualmente aquellos atributos que se piensa que pueden afectar a los atributos principales. Se suele emplear en conjunción con otras técnicas para acotar el conjunto inicial de casos y acelerar el proceso de búsqueda.

- **Recuperación de plantillas**. Consiste en recuperar todos aquellos casos que encajan dentro de unas ciertas restricciones; se suele combinar con otras técnicas como las de vecindad [227].

- **Recuperación guiada por adaptación**. En este caso, no solo se tiene en cuenta la similitud entre el casos de entrada y el recuperado, si no la utilidad de éste último para resolver el problema [237]. En este sentido, en [210] se considera más útil un caso que se más fácil de adaptar combinando medidas locales y globales de adaptabilidad. Esta estrategia favorece la fase posterior de adaptación.

- otros métodos menos utilizados están basados en la medida de la competencia de los casos en una base organizada según la cobertura de los mismos respecto al domino del problema (recuperación basada en 'huella' [238]); en el cumplimiento de unos compromisos en un subconjunto de índices del caso [239]; en una organización de los casos de la base en función de unas relaciones de orden [240]; o en la vecindad entre casos, considerando a estos como una red donde los casos están interconectados en función de su distancia respecto a un aspecto, para realizar la búsqueda del caso a recuperar solo entre casos vecinos (recuperación por 'pesca y disminución' —*fish-and-shrink*— [241]).

El método de vecindad es muy adecuado para dominios en los que predominan valores numéricos en los atributos, y los casos se pueden ver como puntos en un espacio [242]. Lo atributos simbólicos resultan más complicados de evaluar; para simplificar se suele buscar simplemente que tomen un valor exacto. En [242] se emplearon matrices de distancias para calcular las distancias entre valores simbólicos.

A menudo, el proceso de recuperación se divide en varias sub-fases que implican algunos de los métodos indicada más arriba, en especial una primera fase de recuperación por vecindad, con objeto de limitar el número de posibles candidatos; y una segunda que tiene en cuenta una representación estructural o jerárquica de los casos recuperados en la primera fase.

La recuperación propuesta en nuestro modelo toma aspectos de varias de las técnicas indicadas, combinándolas en pro de una operación más eficaz. Así, se partirá de una recuperación por búsqueda en árbol, que nos permitirá discriminar un escenario más concreto dentro de todos los posibles escenarios del problema para, posteriormente, aplicar una recuperación por plantillas que considere únicamente aquellas situaciones almacenadas con los mismos objetos presentes que el problema presentado. Finalmente, se acudirá a una búsqueda por vecindad en la que se compararán los atributos de los casos seleccionados para encontrar el caso almacenado más similar al problema propuesto.

5.2. Medidas de Similitud

Como se ha visto en el apartado anterior, una medida de similitud es necesaria para recuperar el caso más adecuado de la base de casos, y también como estimación, a priori, del rendimiento tras la fase de adaptación. Algunos autores prefieren utilizar el termino **relevancia** para definir a los casos escogidos, término que incluye, pero no se limita a la similitud.

Las medidas de similitud se pueden dividir en dos tipos: **ordinal** y **cardinal** [243]. Las ordinales devuelven un orden de parecido del conjunto de casos respecto al caso de entrada; las cardinales devuelven un valor numérico para cada par de casos comparados, una puntuación, que puede utilizarse también para establecer un orden en el conjunto de casos. En las medidas cardinales no se considera la posibilidad de que dos casos sean incomparables, algo que es posible en el caso ordinal.

5.2.1. Similitud basada en distancia

Es una medida de similitud típicamente usada en la representación plana, cuando los casos, c_n, se representan mediante vectores de características, en los que todos ellos se representan mediante valores numéricos, y la recuperación de casos se realiza por vecindad. La distancia $d(c_1, c_2)$ entre dos casos, c_1 y c_2, vendría dada por la ecuación:

$$d(c_1, c_2) = \sum_{i=1}^{n} w_i \times d_i(c_1 a_i, c_2 a_i) \tag{4.3}$$

donde, $d_i(c_1 a_i, c_2 a_i)$ es una distancia local entre los componentes a_i de ambos casos, y w_i un peso a aplicar a dicha distancia local dentro del cómputo de la distancia global.

Asimismo, la medida de similitud entre los mismos, $sim(c_1, c_2)$, en un intervalo $[0, 1]$, vendría dada por:

$$sim(c_1, c_2) = \frac{1}{1 + d(c_1, c_2)} \tag{4.4}$$

Este concepto de similitud no siempre es el más apropiado: no tiene en cuenta las relaciones estructurales entre los casos; y aparecen problemas cuando alguno de los atributos del caso no tiene un valor definido. Esto se debe a que no siempre es adecuado considerar los casos como puntos en un espacio multidimensional; esto dependerá del dominio del problema y de la representación del mismo. Por ese motivo, los conceptos de 'recuperación del caso más ajustado' y 'medida de similitud' no siempre son sinónimos, aunque se puedan basar en las mismas técnicas. En [244], se diferencia el concepto de **distancia de percepción**, que se correspondería con el concepto clásico de distancia en la representación espacial, y el concepto de **distancia juzgada**, que sería una función de la anterior, y es la que debería permitir recuperar el caso más adecuado a otro en una base *CBR*. Una de las propiedades que distinguen a una distancia de otra es la inecuación triangular; en la distancia de percepción se cumple que $d(A, C) \leq d(A, B) + d(A, C))$ lo cual no tiene porqué cumplirse en la distancia juzgada. Si el cuarto axioma se cumple para una distancia d, y ésta es aditiva en cada línea recta del espacio de atributos, a dicha distancia se le llama **distancia de Minkowski** y cumple, para cada par de puntos $A = (a_1, ...a_n)$ y $B = (b_1...b_n)$, la ecuación:

$$d(A,B) = (\sum_{i=1}^{n} |a_i - b_i|^p)^{\frac{1}{p}} \tag{4.5}$$

Otras técnica de medida de similitud, el **modelo de contraste de características**, proponen abandonar la representación en un espacio multidimensional de los casos, en favor de una representación basada en características binarias [245]. La similitud entre dos casos, A y B vendría medida por una función de sus características comunes y de sus características distintivas $sim(A,B) = f(A \cap B, A \setminus B, B \setminus A))$. Esta función, siempre que satisfaga unas propiedades de monotonicidad e independencia [245], se podría representar mediante una función lineal de las características comunes y distintivas.

Leake [246] añadió al concepto de similitud entre dos casos la adaptabilidad del nuevo caso respecto al antiguo. Ante una descripción incompleta del caso, se realiza un preproceso del mismo para establecer hipótesis sobre el posible valor a partir de los casos almacenados. Relacionado con el tema de la descripción incompleta de casos, en [247] se defiende una representación del caso a diferentes niveles de abstracción de manera que la similitud se comienza a medir a un alto nivel de abstracción, exigiendo un alto grado de coincidencia, que se irá relajando a medida que bajemos el nivel de abstracción del caso al añadir nuevos detalles y características. Esta representación reduce el número de casos en la base de casos, y aumenta la flexibilidad del sistema al ser capaces de reutilizar los niveles bajos de abstracción en varios de los niveles altos; sin embargo se debe incorporar un mayor conocimiento del problema para establecer las relaciones entre los diferentes niveles, lo que aconseja contar con la ayuda de un experto humano.

5.2.2. Similitud para representación estructurada de casos

Muchos autores defiende una representación estructurada de los casos, criticando la incapacidad de la representación plana de capturar las características relevantes de un problema y, sobre todo, sus interrelaciones. Sin embargo la representación estructurada exige una reconsideración del concepto clásico de similitud.

Frecuentemente se representa la información de los casos estructurados a través de grafos dirigidos, donde los nodos son conceptos, objetos o relaciones entre ellos, y las flechas permiten relacionar entre sí los nodos. Wang e Ishii [248] establecieron una medida de similitud para este tipo de grafos a partir de un mapeo *uno-a-uno* entre los nodos y flechas de dos casos distintos, estableciendo una medida de similitud local entre los mismos. La similitud entre dos grafos-casos se obtendría agregando las similitudes locales de sus elementos. El problema surge ante la necesidad de establecer correspondencias entre los nodos y flechas de uno y otro caso, lo cual es altamente ineficiente desde un punto de vista computacional, teniendo en cuenta que los casos pueden tener bastantes elementos, y pueden existir gran cantidad de casos almacenados en la base de casos. En [248] se intentó solucionar el problema mediante algoritmos genéticos con resultados eficientes, aunque no óptimos.

En nuestro modelo se probaron algunas medidas de similitud "clásicas" , como la distancia Euclidea, Manhattan, y Chebychev [249]. No se encontraron diferencias significativas en su comportamiento por lo que, atendiendo a experimentos pasados, y al proceso seguido de conceptualización de la información a través de una discretización, se decidió utilizar finalmente una distancia Manhattan para la búsqueda final por

vecindad, y una distancia Jaccard para el filtrado previo de casos según los objetos presentes en el escenario. Hay que tener en cuenta que no se realizó un estudio intensivo de la aplicación de cada una de las distancias, en el que se modificasen los atributos aplicados a cada caso, o los intervalos de discretización de los mismos. Sin embargo se programaron las estructuras adecuadas para permitir modificar la medida de similitud, y se introdujo la posibilidad de usar diferentes medidas de similitud a aplicar a atributos individuales o grupos de atributos del caso. En trabajos posteriores se espera explorar esta línea de investigación para analizar si se mejora el comportamiento de la recuperación de casos por similitud, aplicando dichas medidas.

5.3. Fase de Reutilización: Adaptación

Tras haber obtenido el caso almacenado más parecido al caso de entrada, es necesario emplear el mismo para resolver el problema planteado. Si el caso devuelto coincide o es muy similar al de entrada, se puede aplicar directamente la solución propuesta para éste. En caso contrario será necesario adaptar la solución obtenida por el sistema para que sea aplicable al problema a resolver. La fase de adaptación se concentra principalmente en las discrepancias entre el caso de entrada y el recuperado; y en las porciones reutilizables del caso recuperado en el problema definido por el nuevo caso [136].

La adaptación se suele apoyar bastante más en el conocimiento general del dominio del problema que en el conocimiento específico contenido en el caso. Por ello los métodos específicos empleados para la adaptación en un determinado sistema *CBR*, pueden no ser válidos para sistemas distintos.

Existe una gran variedad de técnicas de adaptación, pero básicamente se clasifican en tres grupos. La **adaptación nula** propone directamente la solución aportada en el caso devuelto, sin realizar ningún tipo de modificación en la misma [250]. Esta técnica es muy útil para sistemas que pudiendo ser más o menos complejos tienen soluciones simples. La familia de técnicas de **adaptación estructural** o **transformacional** [201] [183] reutiliza la solución asociada al caso devuelto, aplicando una serie de **operadores transformacionales** como borrado, adición, reemplazo,etc [250], o por medio de funciones de transformación dependientes de la calidad de las soluciones adaptadas [251]. Para ello es necesario tener un conocimiento experto acerca de como las diferencias entre las descripciones del problema afectan a la diferencia entre las soluciones. Por último los métodos de **adaptación derivativa** no almacenan solo casos, si no también los métodos que permitieron llegar a las soluciones de dichos casos. Ante un nuevo problema, simplemente se aplica el método asociado al caso más cercano para encontrar la nueva solución. Estas técnica solo se pueden emplear en aquellos casos en los que el dominio del problema está bien comprendido. Es posible combinar técnicas estructurales y derivativas para lograr mejores resultados [183].

El grado de adaptación necesario está muy relacionado también con la estructura y complejidad de la solución de los casos [250]. Por ejemplo, la existencia de interacción entre partes de la solución tiene un impacto importante en la complejidad de la adaptación ya que, de no existir dicha interacción, cada porción de la solución se puede tratar independientemente con lo que la adaptación será más sencilla.

Otra posibilidad de adaptación se basa en la recuperación de más de un caso como posible solución, de manera que todos ellos puedan contribuir a una solución final

adecuada [228], mediante diferentes técnicas, como la **aproximación evolutiva** [250] [252].

En nuestro modelo hemos considerado la recuperación del caso más similar únicamente. Una vez recuperado, se medirá si el caso es lo suficientemente similar al problema presentado a través del número de "saltos" en la representación discreta o conceptual del problema, lo que significaría un mayor o menor alejamiento en el concepto particular que representa un atributo o grupo de atributos del caso. El número de "saltos" considerado como límite que determina la necesidad de adaptación se obtendrá de forma heurística teniendo en cuenta el significado de los conceptos asociados a los diferentes atributos y objetos de los casos. En cuanto a la adaptación en sí, se realizará a través de la combinación de la respuesta asociada al caso obtenido, con la generación de un vector de movimiento a través de la técnicas de campos de potencial o *PFA*.

5.4. Fase de Revisión

A menudo suele ocurrir que la solución generada en la fase de reutilización-adaptación no da los resultados esperados. En este caso, es necesario aprender de los fallos cometidos para mejorar el sistema, mediante una fase de Revisión. Esta fase suele sub-dividirse en otras dos, **evaluación** y **reparación de fallos**.

En la fase de evaluación se determina la viabilidad de la solución adaptada para resolver el problema planteado por un nuevo caso. Esto se suele llevar a cabo fuera del ciclo *CBR*, ya que es necesario aplicar la nueva solución y las consecuencias de la misma pueden tardar algo de tiempo en aparecer. La evaluación de la nueva solución se pude realizar manualmente, mediante la intervención de un experto humano; automáticamente, mediante un algoritmo que simule el resultado de la aplicación de la nueva solución; o experimentalmente, observando el funcionamiento del sistema tras la incorporación de la nueva solución.

Si se detectan errores en la solución adaptada, es importante ser capaces de determinar cuales fueron las causas de los mismos y, en función de éstas, realizar una nueva adaptación del caso para lograr una nueva solución que deberá ser evaluada. Para ello suele ser necesario tener conocimientos del dominio del problema, por lo que esta fase suele llevar el apoyo de un experto humano.

En nuestro modelo se definirá, para cada comportamiento desarrollado, una función de utilidad o eficiencia, que determinará el grado de cumplimiento en la realización de los objetivos del comportamiento. Esta función incluirá aspectos relacionados con la suavidad en las acciones de movimiento del robot, y con la seguridad en su desplazamiento en el escenario de pruebas.

5.5. Fase de Retención: Aprendizaje

La fase de retención la última fase del ciclo *CBR* y atañe a la incorporación de todos aquellos nuevos casos obtenidos tras la fase de adaptación, que han sido evaluados satisfactoriamente. Añadir un nuevo caso a la base de casos *CBR* implica enriquecer el conocimiento total del problema al añadir las experiencias almacenadas en el nuevo caso.

La información que se añade a la base de casos representa el aprendizaje del

sistema acerca del problema que se intenta solucionar o "comprender" mediante *CBR*. Esta información puede haber sido introducida "a priori" por un operador humano o por observación directa de casos reales, en lo que se denomina un **aprendizaje por observación**, o puede incrementarse con la ejecución del sistema tras añadir un nuevo caso adaptado y evaluado positivamente; en este último caso se habla de **aprendizaje por experiencia**.

El aprendizaje por observación [253] forma parte del desarrollo inicial de cualquier sistema que implementa el paradigma *CBR*, de forma similar a la fase de entrenamiento de las redes neuronales. Se basa en la introducción de un conjunto de casos iniciales dentro de la base de casos, que constituirán el conocimiento inicial con el que el sistema empezará a operar

El aprendizaje por experiencia [160] permite aumentar gradualmente el conocimiento que posee el sistema al ir incorporando, mediante su funcionamiento normal, nuevas experiencias en la resolución de nuevos problemas. Para añadir un nuevo caso a la base *CBR* es necesario que el caso haya sido adaptado (de lo contrario ya existiría en la base) y su aplicación al dominio del problema haya sido valorada positivamente. En algunas ocasiones, un nuevo caso no será añadido a la base de casos, pero modificará la estructura de casos existentes o su valoración. Existen dos grande familias de métodos que implementan el esquema de aprendizaje por experiencia de un sistema *CBR*:

- **Aprendizaje Positivo**, que se realiza en caso de evaluación positiva del nuevo caso adaptado. En este caso, es necesario incorporar el nuevo caso a la base de casos, operación que tendrá mayor o menor complejidad según la estructura de casos y de base de casos considerada.

- **Aprendizaje Negativo**. Ocurre cuando la evaluación del caso adaptado (de su solución), es negativa. Es necesario averiguar si la causa de la valoración negativa es la recuperación de un caso inadecuado en el sistema, debido a que, a pesar de ser el más similar, no lo es lo suficiente (esto suele ocurrir cuando la base de casos no contiene los suficientes, a consecuencia de un entrenamiento escaso); o si el sistema no ha devuelto el caso más parecido al de entrada, algo más común en los sistemas y casos con representación jerárquica o estructurada. En este último caso puede ser necesario una re-organización de la estructura del caso y/o de la base de casos [254]. De cualquier forma, es necesario evitar que esta solución adaptada con evaluación negativa vuelva a aparecer en el sistema. Para ello, se puede almacenar el caso fallido en la base de casos, añadiendo una nueva etapa de filtrado de casos no deseados previa a la búsqueda (etapa de **Anticipación**); o almacenar el caso en otra base *CBR* exclusivamente dedicada a situaciones fallidas, de manera que, antes de buscar en la base asociada a la resolución del problema, se buscaría en esta para descartar el caso entrante como fallido [171]; o corregir el error mediante un procesamiento externo a la base *CBR* —por ejemplo, un experto humano— e incluir la solución en la base cuando se evalúe como correcta.

En el modelo propuesto emplearemos la función de utilidad referida en el apartado anterior para evaluar la bondad del nuevo caso en cuanto a la consecución de los objetivos del comportamiento. Se considerará únicamente un aprendizaje positivo, por lo que el nuevo caso se incorporará a la base en caso de resultar eficiente; si el caso

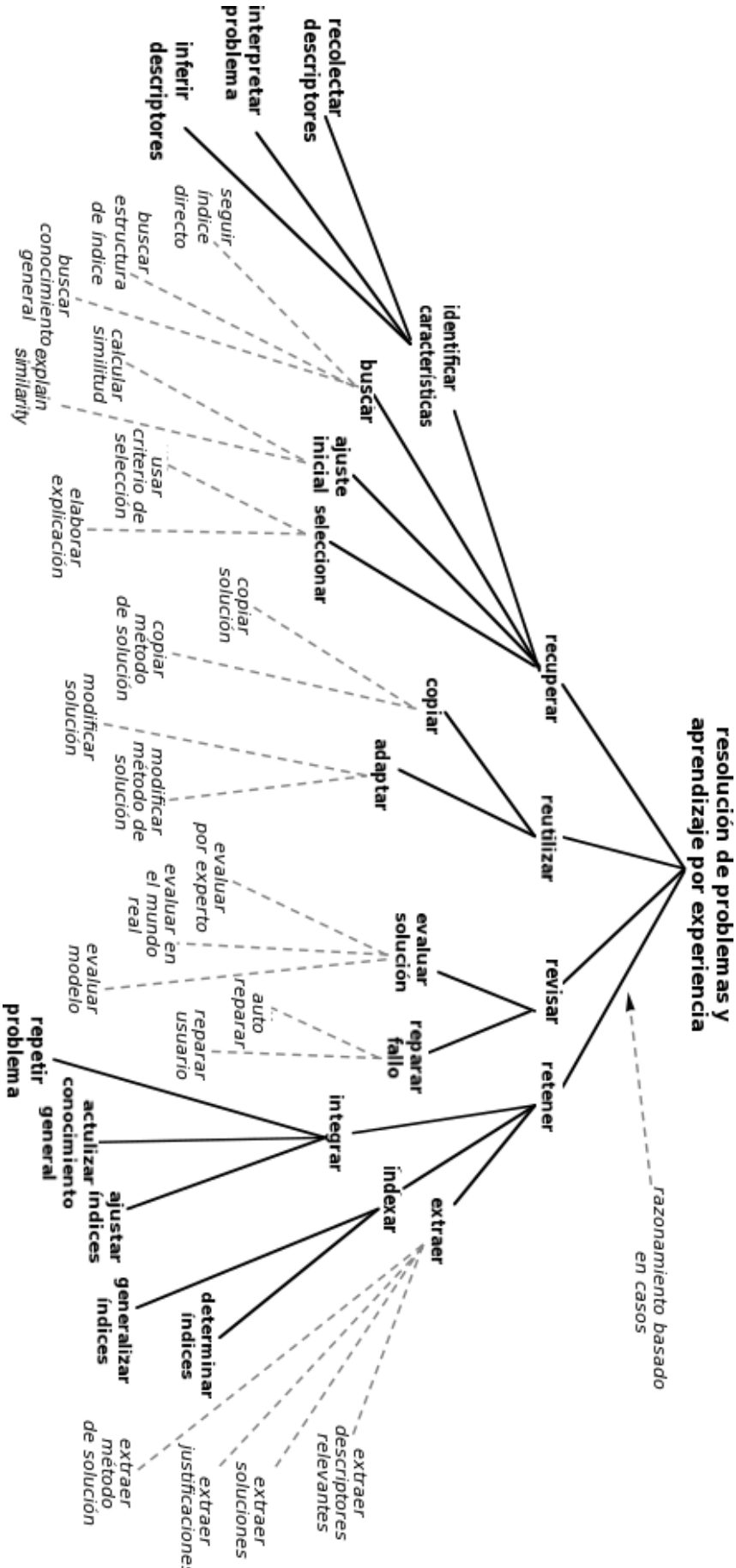


Figura 4.4: Descomposición de las fases y sub-fases CBR [136]

no es considerado adecuado, no se conservará ninguna información en este sentido del mismo. En el siguiente capítulo se tratarán con más detalle este y otros aspectos relacionados con la implementación concreta del sistema *CBR* utilizado en el desarrollo de comportamientos según el modelo que proponemos.

5

Arquitectura híbrida de aprendizaje de comportamientos robóticos basada en CBR: implementación de etapas reactivas

"La clave de la Inteligencia Artificial ha sido siempre la representación"
Jeff Hawkins (1957-) Ingeniero

1. Introducción

En este capítulo, y tras haber realizado en los capítulos anteriores una revisión de los aspectos pertinentes, presentamos la arquitectura propuesta para la implementación de comportamientos robóticos reactivos a bajo nivel, a través del aprendizaje por observación y experiencia. Para ello se describirán en detalle las diferentes características que tiene y los componentes que la integran, justificando las decisiones tomadas en cada caso. En ocasiones estas decisiones estarán relacionadas con las propias pruebas de funcionamiento, si bien la descripción de estas últimas se realizará en un capítulo posterior. Así mismo, se inscribirá la arquitectura propuesta dentro del marco global de arquitectura de inteligencia robótica inspirada en la teoría de inteligencia de Hawkins [1], en la cual ocupa los niveles mas bajos correspondientes a un horizonte reactivo de respuesta.

2. Modelo general de inteligencia robótica basada en la arquitectura HTM

Tal como vimos en el capítulo 3, el modelo de inteligencia y aprendizaje humano propuesto por Hawkins constituye un buen punto de partida para tratar de establecer un modelo similar aplicable a la inteligencia robótica. Los puntos principales a considerar en nuestra arquitectura serían:

- capacidad para almacenar información que describa de forma adecuada las situaciones a las que se enfrenta un robot para resolver un problema. La

representación deberá ser compacta para no exceder en su necesidad de recursos, pero a la vez suficientemente precisa para permitir al robot discriminar entre las situaciones de interés.

- adquisición de conocimientos de una forma sencilla y que no necesite una descripción profunda del problema a resolver con dichos conocimientos. Dicha adquisición debe poder realizarse de forma incremental, permitiendo una adaptación continua.
- inferencia de conocimientos nuevos a través de conceptos parecidos ya existentes en la "memoria" de robot.
- predicción de respuesta del robot ante las situaciones experimentadas, sean ya conocida o no.
- estructura de "nido" en los diferentes componentes y capas de la jerarquía de la arquitectura, de forma que todos ellos compartan unos fundamentes estructurales y de funcionamiento, y se diferencien en la información a gestionar, especialmente en su nivel de abstracción. Este aspecto es especialmente importante para conseguir escalabilidad en la arquitectura.
- realimentación de información entre los diferentes módulos y niveles, incluyendo una realimentación temporal. Este es quizás el aspecto más complejo de conseguir en la arquitectura propuesta, como veremos más adelante, y constituye un campo de investigación abierto y poco explorado.

Debido a estos condicionantes, se ha optado por escoger *CBR* como herramienta a emplear para la implementación de los módulos básicos que integran la arquitectura propuesta, que se puede observar en la figura 5.1.

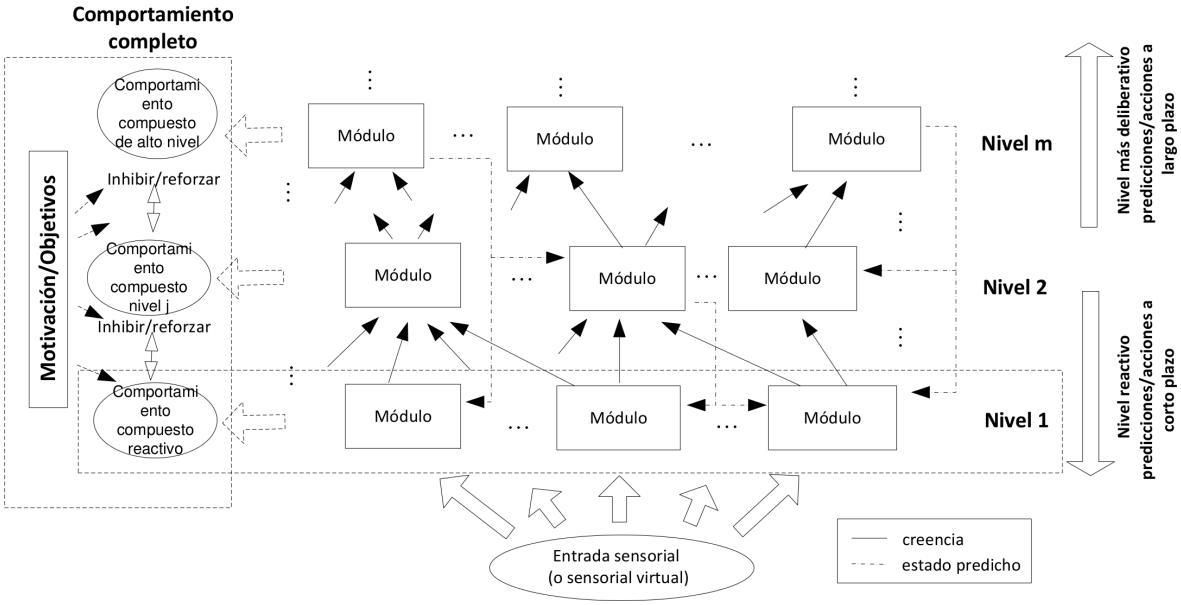


Figura 5.1: Marco general de aprendizaje basado en experiencia

2.1. Componentes del modelo

En esta arquitectura, los módulos se organizarían en una jerarquía completamente conectada, en la que las capas superiores manejan conceptos complejos y más abstractos, que se simplifican y hacen más precisos y definidos a medida que se desciende en la jerarquía. Estos módulos desempeñarían los roles de las diferentes áreas del cerebro humano: reconocer colores (área V4); audición primaria (área 41 de Broadmann); funciones motoras primarias (área M1); etc. En principio los módulos podrían conectarse con otros módulos muy lejanos en la jerarquía, para el intercambio de información, pero lo más habitual sería que solo los módulos contiguos estuviesen interconectados. Los módulos (figura 5.2) almacenarían no solo información conceptual, sino también temporal y sus salidas se conectarían a otros módulos más altos en la jerarquía. De este modo, los módulos a bajo nivel se asocian con predicciones a corto plazo y los de alto nivel con estrategias a largo plazo. En esta jerarquía, los módulos realimentarían predicciones hacia sus módulos inferiores, las cuales se compararían con las siguientes entradas —temporalmente hablando— para confirmar la exactitud de la predicción realizada. En caso de que las predicciones no fuesen acertadas, se deberán tomar las medidas adecuadas para añadir nuevos conocimientos o modificar los ya existentes. Cada módulo de la jerarquía almacenaría información relacionada con ciertos conceptos u objetos individuales del mundo real, en la forma de una **creencia** (*belief* [255]). Esta información se adquiriría mediante una etapa de aprendizaje "tutelado", o bien a través de la experiencia. La principal ventaja de definir los módulos a través de estos conceptos, es que cada módulo puede diseñarse y probarse de forma relativamente independiente del resto, y sus respuestas son predecibles hasta cierto punto.

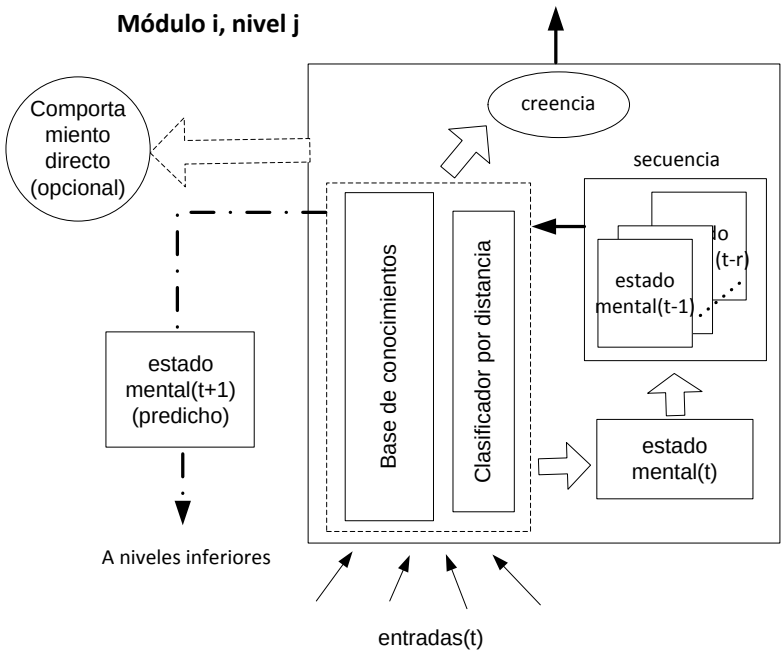


Figura 5.2: Estructura de un módulo

La información obtenida en el aprendizaje se reúne en una base de conocimientos (figura 5.3), que almacenaría instancias de los datos de entrada seleccionados como una

colección de "estados mentales". Cada uno de estos "estados mentales" representaría el modelado actual de un aspecto del mundo real, ya sea un objeto físico, como el concepto "pelota"; una acción, como "la pelota se está alejando"; o incluso conceptos muchos más complejos como "estoy jugando con la pelota". La complejidad del concepto dependería de la situación del módulo en un determinado nivel de la jerarquía.

2.2. Representación de la información

Un aspecto clave de este modelo será la representación de la información que constituye el conocimiento almacenado en el módulos; esta información debería ser invariante, de forma que el concepto o "creencia" pueda ser identificado independientemente de la instancia o realización concreta que tome. Además, se debería seleccionar únicamente el conjunto de datos necesarios para construir el invariante del concepto, evitando incluir información redundante o innecesaria. Sin embargo, procesos como encontrar la representación invariante de una "creencia" asociada a un módulo, o determinar la distribución de la información a través de varios módulos interconectados son aspectos muy complejos y que todavía están lejos de resolverse. Algunas aplicaciones actuales de *IA*, como el reconocimiento de rostros en una imagen, se apoyan en métodos estadísticos como el Análisis de Componentes Principales (*PCA*) [256]; en reconocimiento de habla se han utilizado filtros y Transformadas Rápidas de Fourier (*FFT*) [257]; y en muchas otras aplicaciones, se han empleado redes neuronales artificiales y sistemas expertos con lógica difusa [258] [259]; pero aún no se ha logrado encontrar un método aplicable de forma general, en especial el método concreto que los seres humanos empleamos para almacenar conceptos en nuestro cerebro. En la actualidad la información asociada a cada módulo suele ser escogida por un experto externo —un humano especialista en el dominio de la aplicación—, que filtra la información innecesaria, y elige solo la más esencial. En muchos casos este experto se apoya también en un análisis estadístico del problema a resolver, pero la decisión final suele ser suya. Otro aspecto fundamental de la base de conocimientos es que se debe almacenar también la evolución de los "estados mentales" en una secuencia temporal; de esta forma sería posible predecir la siguiente instancia del "estado mental", la cual, además de contribuir a la construcción de la "creencia", podría realimentarse a los módulos de niveles inferiores y compararse con las siguientes entradas, como una confirmación de la certeza en al predicción. La salida del módulo sería una instancia o realización temporal de la "creencia", la cual se enviaría a los

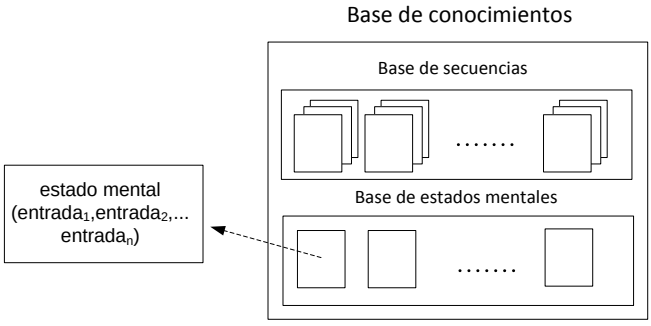


Figura 5.3: Base de conocimiento en cada módulo

módulos de niveles superiores para formar parte de la construcción de conceptos más complejos y abstractos.

2.3. Identificación de conceptos

Tras el aprendizaje, y durante a fase de operación, los datos de entrada, procedentes del sistema sensorial artificial del robot, alimentarían a los módulos inferiores, repartiéndose y distribuyéndose entre los mismos, según las necesidades de cada módulo —las cuales habrían sido establecidas por un experto y/o mediante métodos estadísticos, tal como se indicó anteriormente—. Estos datos pueden preprocesarse para extraer características fundamentales relacionadas con una representación invariante [260] que cada módulo recogería como una agrupación de valores representando una instancia o patrón del concepto que contiene el módulo —o una parte correspondiente a una secuencia temporal—. Esta instancia se compararía con las almacenadas en la base conocimientos para encontrar las más parecidas y establecer un grado de similitud. A partir de aquí, se pueden implementar diversas estrategias para decidir que estado del concepto será identificado por el módulo: coger el patrón más similar entre los almacenados; ajustar este patrón de máxima similitud según su distancia al patrón de entrada; obtener una agrupación patrones similares con una probabilidad de ser los que representen al indicado por la entrada, etc. Una vez se haya escogido un estado del concepto correspondiente a un patrón de entrada, se procedería a mapearlo dentro de alguna de las secuencias temporales almacenadas, para contribuir a la construcción de un concepto con evolución temporal. Por ejemplo, diferentes instancias de un patrón de entradas describiendo a una pelota —posición en el campo visual, distancia, tamaño, ...— se podrían mapear a través de un secuenciamiento en un concepto del tipo "esquivar hacia la izquierda", como respuesta motora a dicho concepto. O diferentes patrones de entrada correspondientes a sonidos con una frecuencia, tono, y duración, se identificarían individualmente con notas musicales que, al formar una secuencia, se identificarían como una melodía determinada. El almacenamiento de la evolución temporal del concepto sería también útil para poder predecir un siguiente estado esperado para el patrón de entradas del módulo, como el siguiente estado del concepto dentro de la secuencia seleccionada. A través de la realimentación hacia módulos inferiores, los patrones esperados en el futuro podrían compararse con las siguientes entradas reales, para determinar si nuestra predicción ha sido correcta. Además, aunque hemos considerado las fases de entrenamiento y de operación como etapas separadas, ambas podrían superponerse, ya que patrones de entrada muy distintos de los almacenados, o respuesta adaptadas muy cambiadas, obtenidas durante la fase de operación, se incorporarían a la base de conocimientos del módulo. Sin embargo, sería aconsejable partir de un mínimo de entrenamiento para dotar al robot de la suficiente habilidad como para explorar el mundo y así adquirir nuevos conocimientos ante esas nuevas situaciones encontradas.

2.4. Respuestas asociadas a conceptos identificados

Finalmente, los procesos cognitivos asociados a los módulos, además de construir e identificar conceptos y realizar predicciones, deberían permitir ejecutar acciones y "liberar" comportamientos que determinen el curso de interacción del robot con

su entorno, para la resolución de un determinado problema o cumplimiento de una aplicación. Estos comportamientos directos se derivarían de la "creencias" identificadas por cada módulo a partir de la entrad actual, y podrían ser "disparadas" por la acción de distintos módulos a diferentes niveles de la jerarquía. Cada comportamiento directo asociado a un módulo se aprendería y almacenaría durante la fase de entrenamiento, y se activaría en la fase de operación cuando se detectasen patrones de entrada similares a los asociados. Los comportamientos asociados a los módulos inferiores serían más reactivos, actuando a corto plazo, mientras que los de los niveles superiores de corresponderían con las componentes deliberativas de la conducta del robot. Por ejemplo, un módulo de bajo nivel identificando una creencia de "intenso calor delante" dispararía un comportamiento del tipo "sepárate de la fuente de calor", mientras que un módulo de alto nivel identificando el concepto "soy un bombero y estoy apagando un fuego" que también incluiría el concepto de "calor delante" del módulo de bajo nivel, podría activar un comportamiento con una estrategia de acercar una manguera al origen del calor, que implicaría acercarse a la fuente, en vez de huir de ella. En nuestro marco general asumimos que los módulos a un determinado nivel activarían comportamientos similares en complejidad y asociados, a partir de los cuales se construiría un comportamiento compuesto *emergente* de mayor nivel de abstracción y complejidad(figura 5.4).

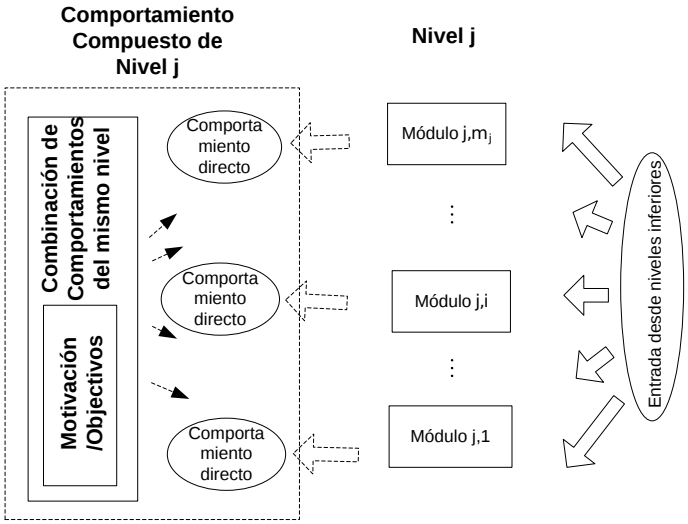


Figura 5.4: Comportamiento compuesto en un nivel

En la medida en que las directrices marcadas por los diferentes módulos podrían entrar en conflicto —por ejemplo, que un módulo indique la realización de un acción y otro de justo la acción opuesta—, la combinación de los comportamientos se debería realizar de acuerdo con un objetivo o motivación [260], el cual podría ser indicado por un experto o inferido de una respuesta conjunta satisfactoria en una experiencia previa similar. Así, a los comportamientos directos que más contribuyesen a lograr los objetivos, se les otorgaría un mayor peso en la respuesta conjunta emergente. Por otra parte, los comportamientos emergentes obtenidos a diferentes niveles de la jerarquía, podrían entrar también en conflicto o, por el contrario, reforzar la respuesta seleccionada por cada uno de ellos. Una alternativa a usar una combinación, tal como se indica en el párrafo anterior, podría ser dar una mayor predominancia a la salida de los niveles superiores respecto a los inferiores, en la medida en que tendrán en cuenta

una respuesta relacionada con el objetivo a más largo plazo. Solamente en caso de encontrar situaciones peligrosas o arriesgadas, se permitiría a los comportamientos de bajo nivel imponerse a los de alto nivel, aunque esto implicaría la implementación de mecanismos para detectar dichas situaciones.

Hay que destacar que esta arquitectura que acabamos de describir a grandes rasgos no está completamente definida, ya que muchos de sus aspectos se han planteado únicamente a nivel teórico, especialmente a niveles más altos de la jerarquía, correspondientes a las capas deliberativas de la inteligencia robótica. Esta tesis se va a centrar en el estudio y desarrollo de los niveles bajos, que tratan con los comportamientos reactivos básicos, y el desarrollo de comportamientos más complejos a partir de los mismos. En los siguientes apartados desarrollaremos los aspectos concretos de implementación de esta porción inferior del modelo propuesto y plantearemos un escenario en el que aplicaremos nuestras propuestas para demostrar su funcionamiento a través de un ejemplo de aplicación.

3. Arquitectura *bottom-up* de aprendizaje reactivo para un robot AIBO jugador de fútbol

Como ya hemos planteado en apartados anteriores, la conducta de los seres humanos no siempre está perfectamente calculada o planeada. A menudo, en presencia de un estímulo, el cuerpo reacciona con una actividad o movimiento automático. Esta respuesta de tipo "sensación-acción" (*sense-act*), a menudo está relacionado con situaciones simples y/o peligrosas, que desencadenan comportamientos sencillos, como cuando se toca una tubería muy caliente y se retira la mano inmediatamente. Los niveles deliberativos o de planificación suelen tener menos peso en estas circunstancias, dando una mayor importancia a los niveles reactivos. Frecuentemente, diferentes respuestas simples de tipo reactivo, producidas individualmente desde módulos sencillos de bajo nivel se pueden combinar para crear comportamientos conjuntos más complejos. Si bien el concepto de "comportamientos emergentes" se aplica generalmente a la interacción del comportamiento de diferentes individuos para formar una suerte de "inteligencia colectiva" [29], este mismo concepto se puede aplicar a la interacción de diferentes módulos de la arquitectura de un único robot [261]. Esto se puede ver claramente a través de un ejemplo: en el juego del fútbol se utilizan estrategias de tipo deliberativo que tienen en cuenta aspectos de alto nivel, como la identificación de la táctica del rival, y sus puntos fuertes y débiles; o el estado del marcador y el tiempo que queda para la finalización de un partido. Estos aspectos influyen en cierta medida en la propia estrategia o comportamiento de los jugadores, especialmente a nivel colectivo. Sin embargo, aún influida por estos criterios, la actuación individual de cada jugador contiene muchos otros aspectos que prescinden del alto nivel: la localización de la pelota; la aproximación hacia ella; su control; la decisión de pasarla o chutar; la evitación de contrincantes en el transcurso de todas las acciones anteriores,etc. La combinación de todos estos aspectos permite construir comportamientos complejos a partir de comportamientos mucho mas simples de tipo reactivo, sin necesidad —o con menor necesidad— de los niveles deliberativos.

Siguiendo esta línea de razonamiento, en el trabajo de esta tesis trataremos de demostrar que, contrariamente a la visión clásica de diseño de las capas reactivas, en

la que su comportamiento es inmutable y predefinido, también es posible y deseable diseñar los comportamientos de bajo nivel a través de un proceso de aprendizaje. Para ello estableceremos un modelo de arquitectura ascendente —*bottom-up*— de diseño de comportamientos reactivos a través del aprendizaje. En esta arquitectura los comportamientos reactivos de bajo nivel se implementarán mediante un serie de módulos, cada uno de los cuales incluirá un sistema *CBR* que le dotará de la capacidad de aprender a través de la observación y la experiencia, y predecir las acciones más adecuadas a partir del conocimiento almacenado. Además se estudiará la construcción de comportamientos emergentes más complejos a partir de los módulos reactivos, considerando su combinación y/o alternancia dirigidas inicialmente por un sistema de reglas de decisión y proponiendo, para trabajos futuros, su sustitución por un sistema *CBR* que decida el peso de cada uno de los módulos componentes en la respuesta conjunta, a partir de la experiencia acumulada en la realización de los objetivos globales. Finalmente se incluirá una demostración de diseño e implementación de esta arquitectura en un escenario de aplicación y robot concreto, a través del cual se mostrarán las criterios a seguir para la toma de ciertas decisiones de diseño y las ventajas y debilidades de la arquitectura propuesta, así como su rendimiento y eficiencia.

3.1. Estructura general del sistema propuesto

Dentro del modelo general de arquitectura de inteligencia propuesto en el apartado 2 del presente capítulo, nuestro trabajo se centrará en la capa inferior , correspondiente a los niveles reactivos más simples del robot. Para la implementación de un comportamiento reactivo con cierto grado de complejidad, se tendrán que definir una serie de comportamientos directos que implementen diferentes aspectos o facetas del comportamiento de alto nivel que se desea construir. Cada uno de estos comportamientos directos tendrá su propia base *CBR* para el almacenamiento de la información que caracterice el comportamiento, información que será incorporada a través de una fase previa de aprendizaje por observación, complementada con una fase posterior de aprendizaje por experiencia durante la operación del robot.

Un menor nivel de complejidad de los comportamientos directos de partida permitirá que la adquisición de la base de conocimientos asociada al comportamiento sea más sencilla y la cantidad de información menor, al estar implicados un menor número de variables en la definición del caso, si bien el número de módulos componentes aumentará. Sin embargo la obtención del comportamiento emergente se complicará ya que la interrelación entre las salidas de los módulos componentes no será —en principio— aprendida durante las fases de entrenamiento y deberá ser establecida *a posteriori*. En una fase inicial de nuestro trabajo se tomó esta aproximación (figura 5.5.izq.), trabajando con módulos muy sencillos asociados a conceptos simples y poca información de entrada, de forma que la caracterización de cada comportamiento directo se conseguía a través de aprendizajes relativamente cortos con una base de conocimientos pequeña. La implementación de comportamientos emergentes a partir de pocos módulos de estas características, y mediante conmutación de comportamientos dio buenos resultados para la obtención de comportamientos emergentes no demasiado complejos, que sin embargo eran relativamente fáciles de entrenar y obtener como un único comportamiento con un número de entradas mayor pero todavía manejable.

Sin embargo, al incorporar un mayor número de módulos simples a combinar, la combinación hacia la emergencia presentaba problemas relacionados con la orientación independiente del entrenamiento de cada módulos. Se realizaron pruebas adicionales para realizar la combinación de las respuestas de los diferentes módulos mediante la incorporación de un módulo superior de arbitración/combinación que determinaba la influencia de cada submódulo componente a través de un entrenamiento, pero no se lograron resultados concluyentes. Estos ejemplos se muestran en el apartado 2 del capítulo 6.

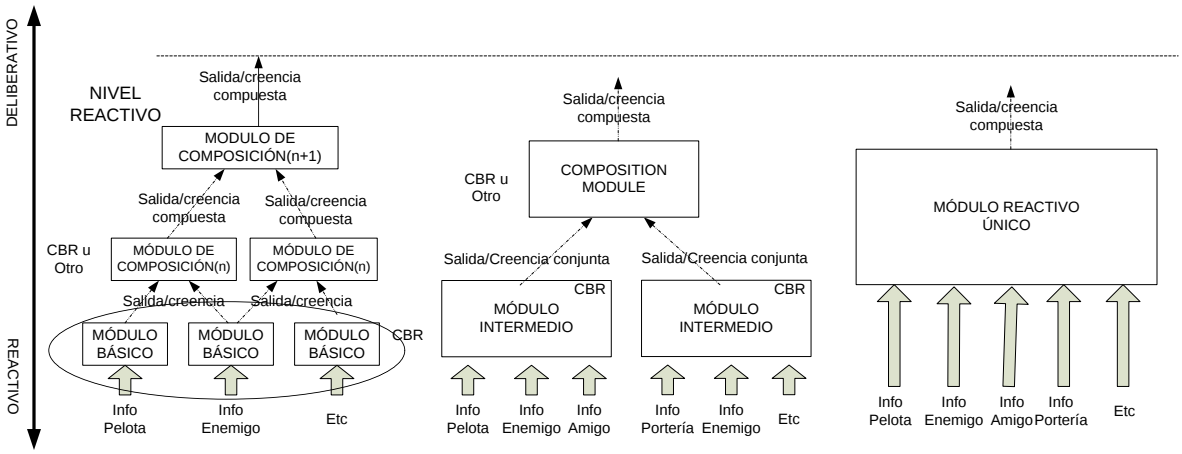


Figura 5.5: Diferentes estrategias de diseño de comportamientos reactivos emergentes mediante CBR

Por otra parte la aproximación opuesta, que consideraría un único comportamiento muy complejo con un enorme número de entradas(figura 5.5.der.), se descartó de entrada por problemas de escalabilidad, si bien esta opción permitiría incluir en la base de conocimientos y mediante aprendizaje, no solo la respuesta del robot en presencia de determinados objetos/conceptos individuales, sino la variación de esta respuesta ante combinaciones y relaciones de estos objetos. Un único comportamiento de "jugar al fútbol" teniendo en cuenta todas las posibles variables y elementos de influencia, incluso en un escenario limitado como el propuesto, necesitaría de una cantidad absurda de casos de un tamaño demasiado grande como para ser manejable en tiempo real. Recordemos que la potencia del diseño de comportamientos a través del aprendizaje radica en la incorporación automática de la "reglas" que determinan dichos comportamientos sin tener un conocimiento detallado de los algoritmos y relaciones que definen dichas reglas.

Finalmente, la arquitectura propuesta (figura 5.5.cent.) considera módulos básicos de un nivel de complejidad intermedio, más orientados a la incorporación de acciones o porciones del comportamiento global que a respuestas ante objetos o conceptos concretos. Estos módulos incorporan información de varios elementos del mundo donde se desenvuelva la aplicación del robot, pero solo aquellos que se considere que influyen realmente en la respuesta de dicho comportamiento, de forma que aunque el número de entradas y el tamaño de la base de conocimientos aumenta con respecto a la propuesta original, sigue dentro de unos márgenes razonables. La principal ventaja de esta decisión de diseño es que es posible capturar, a través del entrenamiento, la influencia de los componentes de entrada en el comportamiento, tanto de forma individual como conjunta, facilitando la etapa posterior de agrupación de módulos para obtener el

comportamiento emergente de alto nivel. En el capítulo 6 se mostrará un ejemplo de diseño de un comportamiento reactivo con estas características así como la obtención de comportamientos más complejos mediante emergencia.

3.2. Escenario de trabajo

Uno de los aspectos recurrentes de las arquitecturas y métodos empleados en *IA*, es que suelen estar orientados —sobre todo en sus primeras realizaciones— a un dominio en particular, que determina la toma de muchas de las decisiones de diseño. Aunque esta orientación a la aplicación choca con la propuesta de una teoría o arquitectura de aplicación general para la inteligencia/aprendizaje robótico, suele ser necesaria en las fases iniciales del desarrollo para, una vez comprobada su corrección y utilidad, tratar de extender la teoría a una generalización mas amplia en otros dominios y aplicaciones. El trabajo desarrollado en esta tesis no será una excepción a esta pauta: muchas de las decisiones de diseño tomadas vendrán determinadas por el dominio de la aplicación que se desarrollará como ejemplo de la arquitectura, si bien trataremos de indicar, en la medida de lo posible, las alternativas que se podrían seguir para lograr una mayor generalidad. De esta forma, antes de empezar a describir las diferentes características de diseño de los componentes y etapas de desarrollo de nuestra arquitectura, pasaremos a describir el escenario de trabajo en el que se desarrollarán los experimentos de prueba de la arquitectura.

3.2.1. Sistema hardware de testeo

Como soporte de implementación de nuestra arquitectura, usaremos un robot *AIBO* cuadrúpedo modelo ERS-7 (figura 5.6), que cuenta con una cámara "a-bordo" situada en el morro del robot, así como diversos sensores infrarrojos de distancia. La elección de un robot cuadrúpedo no es casual, ya que esta clase de robots experimentan un control más impreciso que los robots con ruedas, debido a la propia mecánica de control de su movimiento. En este tipo de robots el control odométrico interno resulta muy complejo, por lo que es necesario realizar continuas correcciones fundadas en el propio estado del robot y el modelo del entorno percibido. Esta característica lo hace ideal para probar como la reactividad de los comportamientos absorbe fácilmente las pequeñas imprecisiones de control de los motores, de posibles deslizamientos de las extremidades y también de imperfecciones en el propio modelo del mundo. Recordemos que tampoco la inteligencia humana tiene siempre un conocimiento completo y perfecto de su entorno, y es su capacidad de adaptación ante estas lagunas de conocimiento lo que le dota de su enorme capacidad y versatilidad. Por otra parte, una descripción algorítmica de un comportamiento, por ejemplo de navegación del robot, resulta más compleja en este tipo de robots que en los robots con ruedas, por lo que quedarán mas de manifiesto las ventajas de absorber las características del comportamiento a aprender a través del entrenamiento,en vez de utilizar dichos algoritmos. Se podría decir que un funcionamiento adecuado de la arquitectura propuesta en este tipo de robot implicaría también un funcionamiento correcto, casi con toda seguridad, en la mayoría de robots con ruedas de un nivel de complejidad similar. Otro tipo de robots, como hexápodos o bípedos, se han descartado inicialmente por su mayor lentitud y la no disponibilidad física del robot, aunque se plantea intentar adaptar la arquitectura propuesta en un futuro a otros modelos.

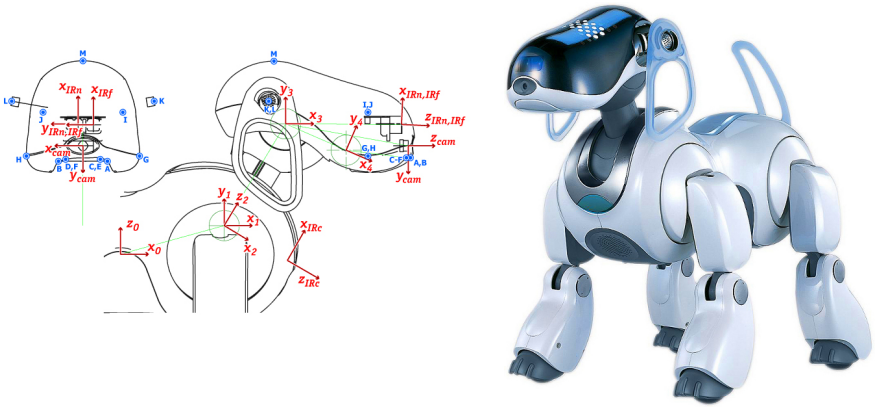


Figura 5.6: Robot AIBO ERS-7

En los seres humanos la visión constituye el sentido más desarrollado y que aporta mas información para modelar el mundo y determinar su conducta, por lo que parece razonable que fuese también la principal fuente de información de nuestro robot. *AIBO* dispone de una cámara a color en espacio *Luminance-Bandwidth-Chrominance (YUV)* de una resolución máxima de 208x160 píxeles con un ritmo de captura de 30fps. Para simplificar la operatividad de nuestro modelo, partiremos de un cierto grado de elaboración de la información prescindiendo, desde el punto de vista comparativo con un cerebro humano, de algunas de las capas inferiores, como las que reconocerían luz/oscuridad, orientación, agrupación de puntos (píxeles) de un mismo color, etc. Estas funciones "primitivas" serán realizadas conjuntamente por módulos no basados en aprendizaje sino en algoritmos clásicos de segmentación (figura 5.7). También usaremos como información de entrada a nuestros módulos la lectura del sensor de infrarrojos alojado en el pecho del robot y que le permitirá detectar la presencia de objetos tales como el suelo o una pelota, indicando la distancia a los mismos.

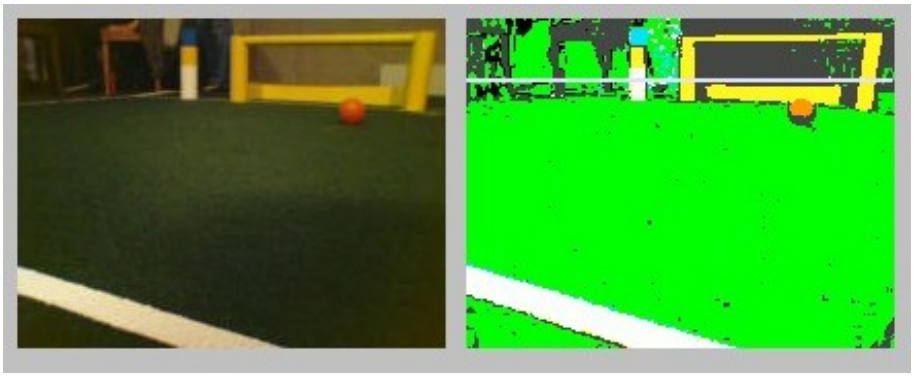


Figura 5.7: Segmentación de imágenes de la cámara de AIBO

Finalmente, y pese a contar con robots *AIBO* reales , gran parte de las pruebas se realizarán sobre el simulador 3D MIRAGE (figura 5.8), el cual está integrado en la herramienta de desarrollo Tekkotsu [262], que es la que usaremos para el control del robot, y sobre la que desarrollaremos nuestra arquitectura. Los principales motivos también están relacionados con la discontinuidad de *AIBO* por parte de Sony, que hace difícil la reparación o sustitución de robots que puedan averiarse. Algunos de los robots empleados ya han sufrido de ciertas averías asociadas principalmente a causas

mecánicas ("cojera" o desequilibrio motriz en las extremidades; "tembleque" en la unión del cuello; agotamiento y ruptura de baterías,...) lo cual nos hace especialmente cautos a la hora de realizar pruebas con estos robots reales. Sin embargo, el simulador escogido reproduce con una gran fidelidad los aspectos mas importantes de nuestro robot al contar con un motor físico —*Bullet Physics* [263]— muy completo. Como ventajas adicionales, el uso del simulador nos permite abstraernos de problemas adicionales como fallos del enlace WIFI de conexión entre el robot y el interfaz de control, distorsiones de la cámara del robot, cambios en las condiciones de iluminación, recarga de la batería del robot entre pruebas, etc... Estos problemas, si bien son intrínsecos a la aplicación de cualquier arquitectura o método en los robots reales, no resultan imprescindibles para probar el funcionamiento de la arquitectura propuesta, por lo que se ha preferido evitarlos por ahora. Por otra parte, los módulos desarrollados sobre Tekkotsu se pueden ejecutar directamente en el robot, cambiando simplemente la herramienta de compilación. Esta dualidad simulador/robot-real fue testada en las etapas iniciales de desarrollo de la arquitectura, demostrando su correcto funcionamiento, por lo cual en investigaciones futuras se tratará de aplicar la arquitectura propuesta en robots reales —*AIBO*, si continúan funcionando u otros robots distintos— para probar el impacto de estos problemas en el rendimiento de la misma.

Figura 5.8: Simulador 3D MIRAGE con motor físico Bullets en el entorno Tekkotsu

3.2.2. Entorno de pruebas

Por otra parte, la elección de la información de entrada específica para cada módulo será determinada por un experto en función del campo de dominio de la aplicación a desarrollar. Como ejemplo de aplicación se ha buscado una aplicación clásica de prueba de las capacidades de los robots, como es el campeonato de fútbol robótico RoboCup, que se lleva disputando de forma anual desde 1997. Este dominio de aplicación permite probar las técnicas y arquitecturas de *IA* aplicadas a la robótica en un marco común, y en un entorno controlado con un numero finito y bien conocido de variables. En la figura 5.9 podemos ver la estructura y elementos intervenientes en un partido de RoboCup. Al centrarse esta tesis en el desarrollo de la arquitectura para un robot autónomo

individual, utilizaremos una versión modificada en la que el número de robots que intervienen es menor. Así mismo, no haremos uso de las balizas que permiten situarse al robot en el campo, ya que nos centraremos en el aprendizaje de comportamientos reactivos puros que son relativamente independientes de la posición absoluta del robot en el campo. No obstante, sería perfectamente posible añadir un módulo de localización al robot —por ejemplo mediante un filtro de partículas apoyado en las balizas— que proporcionase una información adicional con una aproximación de la posición del robot en el campo a los comportamientos que lo demandasen para implementar su actividad. Desgraciadamente, la suspensión en la producción de la linea de robots *AIBO* en el año 2006, llevó a su eliminación como parte de las categorías de la RoboCup a partir del año 2007. Esto dificulta en cierta medida, la comparativa del funcionamiento de nuestra arquitectura con las empleadas en los tiempos actuales en este dominio de RoboSoccer.

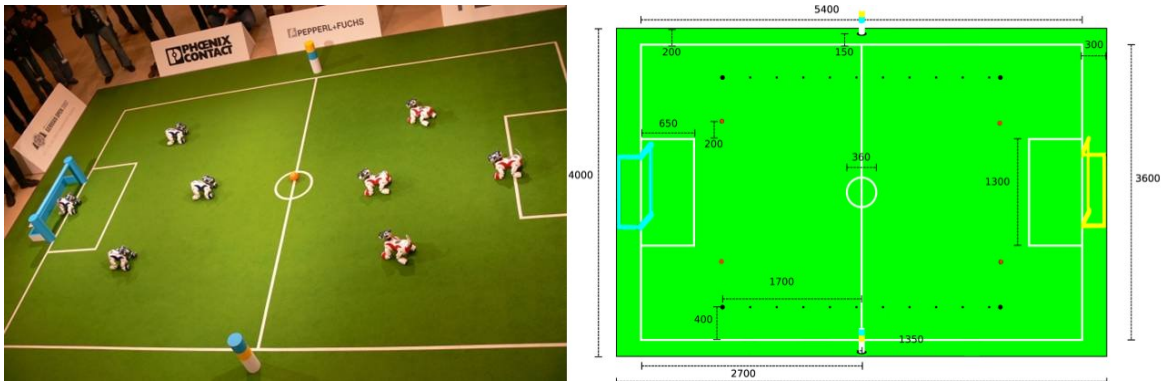


Figura 5.9: Campo de juego RoboCup, modalidad AIBO

4. Estructura de un módulo de la arquitectura en el dominio propuesto

En este apartado se describirán las características y aspectos de diseño de los módulos individuales que implementarán los comportamientos directos o de bajo nivel, que constituyen la base de nuestra arquitectura. La definición del módulo vendrá determinada por el tipo y formato de la información de entrada al mismo; por las características y operativa de la base *CBR* que integrará los conocimientos o conceptos asociados al módulo; por la representación de la información dentro de la base de conocimientos; y por la salida proporcionada por el módulo. En los siguientes apartados analizaremos cada uno de estos aspectos detallando las decisiones tomadas para el dominio de la aplicación propuesta, y justificando dichas decisiones en sus respectivos contextos. La implementación específica de ejemplos concretos se tratará en el capítulo siguiente, dedicado a la realización de experimentos y pruebas.

4.1. Información de entrada

Como hemos indicado anteriormente, la mayor parte de la información de entrada que consideraremos es de tipo visual, procedente de una imagen artificial captado por la cámara ”a bordo“ del robot. Una aproximación directa nos llevaría a alimentar

directamente la imagen, su descripción en píxeles *YUV*, como entrada de cada uno de los módulos de bajo nivel de la arquitectura. Esta aproximación, sin embargo, daría lugar a un tamaño de la información de entrada excesivamente grande —ancho_imagen x alto_imagen x 3_componentes_por_píxel—, y que además conllevaría una excesiva redundancia de información, lo cual se contradice con la visión de comportamientos simples que no contemplan el modelo completo del mundo, sino solo la información local útil para el módulo. El marco general de inteligencia propuesto, llevado al extremo, incluiría varios módulos de bajo nivel que aprenderían, a través de un sistema *CBR* y partiendo de la descripción de la imagen en píxeles, a reconocer características visuales como colores, bordes, tamaños, formas de objetos, etc, y que proporcionarían la información de estos conceptos a los módulos de nivel superior, aún reactivos, que tomarían las decisiones de comportamiento del robot en función de las instancias reconocidas. Para simplificar, y dado que la teoría algorítmica de reconocimiento de tales conceptos visuales de bajo nivel está perfectamente definida, hemos considerado utilizar módulos con esta orientación algorítmica para esta tarea, los cuales proporcionarán una descripción de los objetos de interés del experimento mediante una segmentación rápida de objeto basada en colores [264] a partir de la imagen de entrada. Por otra parte, y dado que estamos considerando un horizonte temporal reactivo en la respuesta de nuestros comportamientos, la toma de decisiones no tendrá en cuenta la evolución de las entradas, ni existirán estados internos que determinen dicha evolución: el robot tomará sus decisiones en función únicamente de la información percibida en cada instante. La información particular de entrada a cada módulo deberá ser escogida por un experto humano, en base a la funcionalidad que se pretende dotar al comportamiento directo. Para esta toma de decisiones el experto se puede ayudar de herramientas estadísticas que nos permitan analizar factores como la correlación entre los componentes del patrón de entrada, y la influencia que puedan tener sobre los aspectos que vayan a determinar la respuesta del comportamiento.

La estructura general de un patrón de entrada a un módulo de la arquitectura, en el dominio de la aplicación propuesto, vendrá dada por una descripción de los objetos localizados en cada imagen a partir de las regiones o *blobs* obtenidas tras el proceso de segmentación. La descripción típica de dichas regiones de segmentación suele realizarse a través de su *bounding box* y su área, pero con vistas a una reducción de la dimensionalidad del patrón de entrada, se ha considerado una descripción alternativa que define a cada objeto a través de su centroide, (Cx, Cy), su área y, en caso de que exista, el *clipping*, o cantidad de área que bordea con los bordes de la imagen. El *clipping* resulta imprescindible para distinguir, por ejemplo, si un objeto con poca área visible se encuentra realmente alejado, o está cercano pero solo visible parcialmente (figura 5.10). Esta transformación de información permite obtener unos descriptores más coherentes y más fáciles de tratar en operaciones adicionales como la discretización. Todos estos parámetros se normalizarán con respecto a sus valores máximos y mínimos posibles a una escala [0,1], con objeto de poder utilizarlos de forma conjunta, sin que unos pesen más que otros en función de su magnitud.

Otro aspecto clave para una buena operatividad del robot radica en disponer de información actualizada de forma constante. En la medida en que la mayoría de información procede de la visión del robot, es importante mantener enfocados los objetos de interés el mayor tiempo posible, lo cual ayudaría también a minimizar el *clipping*, consiguiendo una información más precisa de los objetos. En este mismo

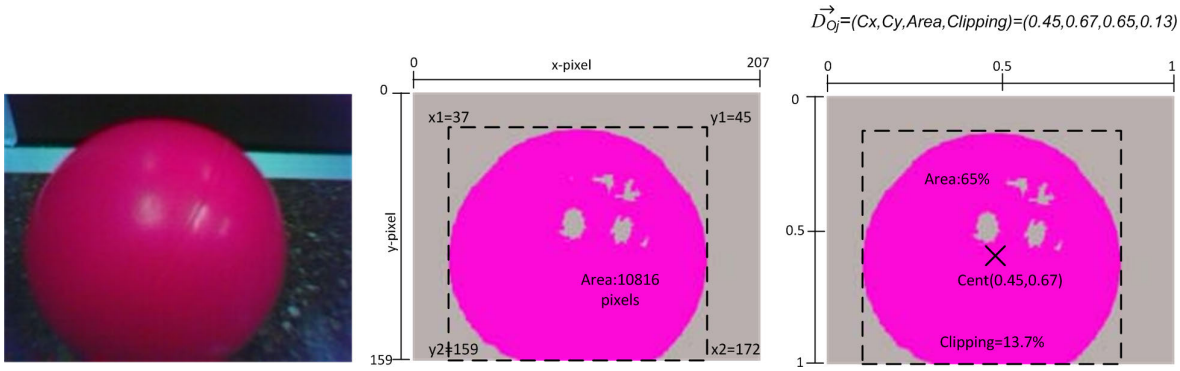


Figura 5.10: Información de entrada para descripción de un objeto

sentido, los seres humanos tenemos un mecanismo de atención, el movimiento sacádico, que permite que nuestra visión busque objetos de interés en el campo visual, permitiendo construir un mapa mental de la escena, realizar predicciones que confirmen dicho mapa, y encontrar objetos de interés que puedan motivar una actuación. A este respecto, el robot cuenta con la posibilidad de mover la cabeza en tres grados de libertad (*pan*,*nod*, y *roll*) lo cual permite apuntar la cámara hacia las zonas de interés sin tener que alinear el cuerpo hacia las mismas. Si bien se han realizado pruebas para controlar la atención visual del robot hacia los objetos de interés con vistas a aprender pautas de atención que controlen el movimiento de la cabeza [265], la dificultad para realizar dichos movimientos dentro de un horizonte reactivo, sin tener en cuenta la evolución temporal de la escena, nos ha llevado a descartar este tipo de control, al menos a este nivel. En su lugar se ha desarrollado un algoritmo de seguimiento de objetos que simplemente mueve la cabeza del robot para tratar de mantener el mayor número posible de objetos de interés en el campo visual del robot. Para simplificar este mecanismo, se ha ”anulado“ el desplazamiento de la componente *tilt* ya que, si bien permite ampliar algo más la capacidad de atención visual, sus efectos se superponen en gran medida con los de la componente *nod*.

Si bien este procedimiento reduce el *clipping* y, por tanto, permite obtener una información más fiable del área y la *Bounding Box*, dado que la información percibida depende de la orientación de la cabeza del robot, en el patrón de información de entrada se deberá proporcionar, además de la descripción de los objetos de interés, dicha posición de la cabeza del robot al tomar la imagen de referencia de los objetos (figura 5.11). Esta posición vendrá descrita por al ángulo de las componentes libres de movimiento de la cabeza del robot, normalizadas a sus valores máximos y mínimos posibles, por los motivos anteriormente comentados.

Los datos de entrada a un módulo pueden incluir información adicional de contexto. En algunos casos esta información no puede ser obtenida a través del sistema de visión, como por ejemplo las situaciones en las que el robot se encuentra en posesión de la pelota, pero necesita observar la escena para determinar un comportamiento. Para ello, hemos añadido un componente de detección de posesión de la pelota a través del sensor infrarrojo alojado en el pecho del robot.

También es posible realizar un procesamiento adicional con la información visual, para reducir la dimensionalidad de la entrada aprovechando ese conocimiento de contexto. Un ejemplo de esta situación sería la inclusión de solo una de las porterías del escenario RoboCup en la información de entrada considerando que, por las limitaciones

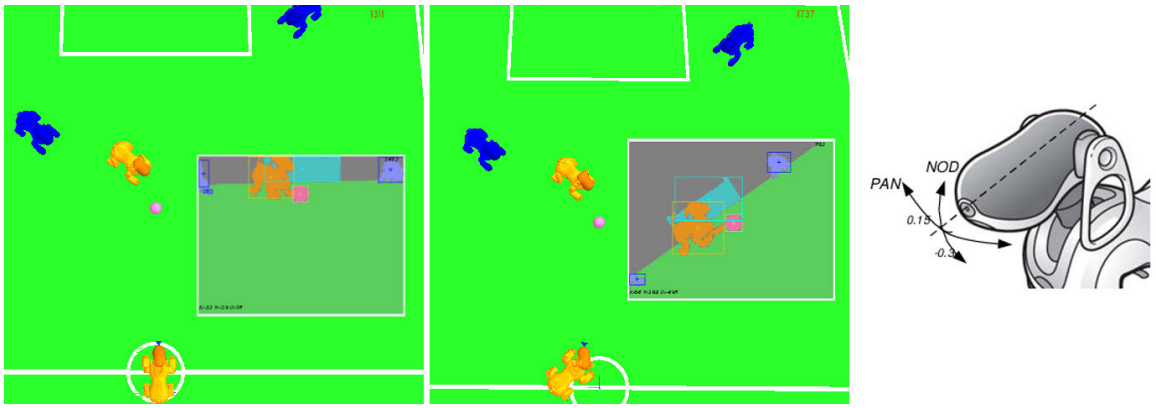


Figura 5.11: Influencia de PAN y NOD sobre la imagen de entrada

en el escenario del campo —se rodea el campo con vallas— no es posible detectar simultáneamente ambas porterías en una imagen.

De este modo, un ejemplo de un patrón de información de entrada genérico para los módulos considerados en nuestra aplicación, Mod_{input} , tendría la forma:

$$Mod_{input} = [Obj1_{desc}, Obj2_{desc}, ..., ObjN_{desc}, PAN, NOD, BallPos] \tag{5.1}$$

donde $Obji_{desc}$ representa el conjunto de descriptores de un objeto con información útil para el módulo:

$$Obji_{desc} = [Cx, Cy, Area, Clipping] \tag{5.2}$$

Los parámetros descritos constituyen toda la información obtenible de la posición, cámara y sensores de distancia de un *AIBO* en nuestro entorno de pruebas. Sin embargo, no todos los módulos necesitan el conjunto completo de estos datos. Cada módulo incluirá únicamente la información de los objetos que, a juicio del experto, influyan en su comportamiento. Los diferentes parámetros estarán normalizados a sus valores máximos y mínimo posibles, excepto el indicador de posesión de la pelota, $BallPos$, que será un indicador binario. Una alternativa sería normalizar los valores de cada uno de los parámetros con respecto a los valores máximos y mínimos observados en la información adquirida en el proceso de aprendizaje. Si bien esta alternativa permite una mayor precisión en la representación de la información útil, al no considerar los rangos extremos con una menor frecuencia de aparición, se corre el peligro de, durante la fase de operación, adquirir nueva información fuera de los rangos establecidos, lo cual obligaría a una nueva operación de normalización tras la adquisición de esta nueva información. Por este motivo, y por la orientación conceptual de la información de la que hablaremos en próximos apartados, consideraremos la normalización de los datos con respecto a sus valores máximos y mínimos posibles.

Finalmente, se ha contemplado la posibilidad de incluir también una estimación de la posición del robot en el campo, como factor de influencia adicional en su comportamiento —sistema propioceptor—, que podría modificar el mismo, en función de si , por ejemplo, el robot se encuentra en una posición de ataque o de defensa, o en la parte central o lateral del campo. Sin embargo, esto demandaría la inclusión de un módulo adicional en la arquitectura del robot que proporcionase una estimación de su posición, en función de su evolución a lo largo del tiempo, lo cual entra en conflicto con la naturaleza reactiva pura de los comportamientos que se desea probar. En cualquier

caso, la realización de dicho módulo a través de un filtro de partículas, se contempla para trabajos futuros.

4.2. Base de conocimientos: adquisición y almacenamiento de la información

La respuesta o definición de cada unos de los comportamientos directos viene determinada por la información encerrada en su base de conocimientos. Esta base es construida por aprendizaje a partir de los diferentes patrones de información que se van presentando a su entrada durante una fase inicial de entrenamiento por observación, complementada por un aprendizaje por experiencia durante la fase de operación, que permite integrar nuevos conocimientos no tenidos en cuenta inicialmente, y ajustar los posibles fallos e imprecisiones de la información almacenada (figura 5.12).

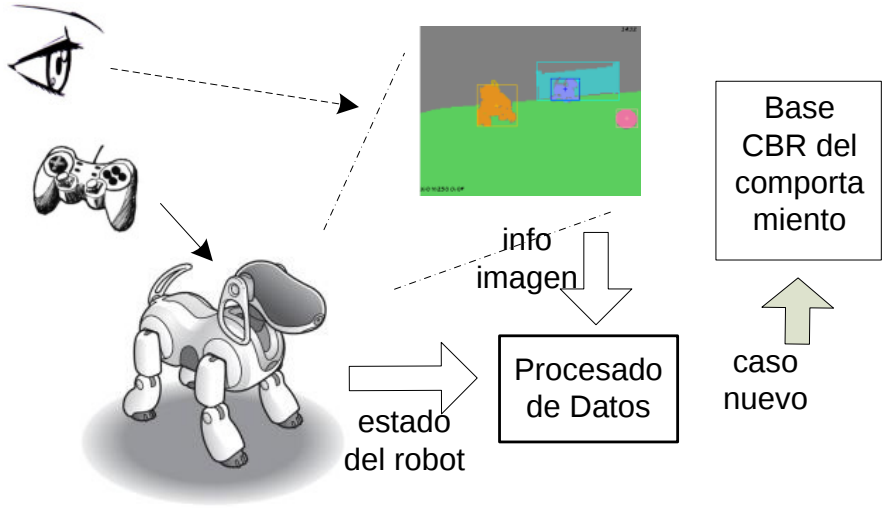


Figura 5.12: Aprendizaje por observación de un comportamiento

En la arquitectura propuesta, la base de conocimientos se implementa a partir de un sistema *CBR* ya que consideramos que, por sus características y funcionamiento, es la herramienta de *IA* más adecuada a la teoría de inteligencia en la que se basa dicha arquitectura. Como ya vimos en el capítulo 4, a la hora de diseñar un sistema *CBR* es necesario establecer unas opciones de diseño que determinarán en gran medida la eficacia en el funcionamiento del sistema, y que están relacionadas con las diferentes fases del ciclo *CBR*. Entre las decisiones a tomar tenemos: el formato de los casos que engloban la información almacenada; la estructura de la propia base de almacenamiento; las operaciones y métricas empleadas en el proceso de búsqueda de la información más similar al problema presentado; la evaluación de si la similitud del caso recuperado es suficiente para una aplicación directa de su respuesta asociada; las técnicas de adaptación del caso recuperado en caso de que no sea así; la evaluación de la eficiencia o utilidad del caso adaptado; y el mantenimiento de la información de la base de casos, para la incorporación de nuevos problemas cuya solución adaptada sea satisfactoria, y eliminación de aquellos casos que no se consideren suficientemente útiles. En los siguientes apartados indicaremos las soluciones tomadas, justificando las decisiones, teniendo en cuenta que en el capítulo 4 ya se realizó una revisión de las opciones mas relevantes de todos estos aspectos de diseño.

4.2.1. Formato del caso CBR. Discretización-conceptualización de la información de entrada

El conocimiento almacenado en el comportamiento se distribuirá en una serie de casos que representarán diversas instancias de los conceptos representados en el comportamiento. Cada caso estará constituido por una entrada, una representación de una instancia del problema a resolver, obtenida partir de la información de entrada obtenida durante el entrenamiento; y una salida que determinará la acción a tomar por el robot ante la situación representada por la entrada, y que podrá haber sido aprendida durante la fase preliminar de aprendizaje por observación; o generada en la fase de adaptación *CBR*. Como los módulos a desarrollar son comportamientos de bajo nivel, las entradas y salidas seguirán el paradigma *sense-act* propio del paradigma reactivo.

Como primera aproximación se utilizaron como entradas de cada caso, los componentes del descriptor de información de entrada indicado en la ecuación 5.1, particularizando a los objetos que el experto considera que van a influir en el comportamiento a modelar. Esta aproximación da buenos resultados, especialmente si la entrada del caso tiene pocos componentes, pero adolece de un problema de escalabilidad relacionado con la dimensionalidad de los diferentes descriptores del caso, que puede llegar a ser muy alta, o incluso infinita, para componentes continuos como el ángulo de la cabeza del robot. Además, esta visión cuantitativa de los descriptores del caso, choca con la representación conceptual de los conocimientos propugnada en el marco teórico de inteligencia bajo el que se desarrolla la arquitectura.

La reducción de la dimensionalidad se puede conseguir a través del uso de representantes de clase de caso obtenidos a través de *clustering*, si bien esta operación implica una clasificación previa de los casos, que no siempre es posible. Como alternativa se utilizará la **discretización** o **indexación** de los casos a través de una partición del espacio de cada uno de los componentes, determinada por un experto. Esta discretización, además de reducir el número máximo de instancias posibles existentes en el espacio multidimensional del caso, permite una descripción cuantitativa de la información a una descripción cualitativa o conceptual, en la que los intervalos de valores se mapearían con estados o magnitudes del concepto discretizado. Así, el área de un objeto ya no se representaría mediante el número de píxeles normalizado a su máximo valor, sino mediante descripciones de la forma ”área mínima“, ”área mediana“, ”área grande“, etc, lo cual, además de facilitar el proceso de depuración del ’razonamiento” del robot, nos permitirá construir conceptos más complejos que describen el estado del entorno y del robot respecto a la consecución de los objetivos asociados al comportamiento. De esta forma, los conceptos “área mínima” y “centroide Y bajo” obtenidos de la discretización de los componentes “área” y “centroide-Y” del caso, podrían formar un concepto más complejo, “objeto lejano” (figura 5.13). También sería posible utilizar información conceptual para una descripción a nivel general, junto con información numérica para discriminar en mayor detalle, tal como se propone por ejemplo en [266]. Este enfoque se tratará de aplicar en trabajos futuros.

La discretización de la información de entrada para conformar los casos constituye un aspecto clave en la definición de nuestra base de conocimiento, y está fuertemente influenciado por los objetivos y el diseño del comportamiento. En nuestro trabajo actual es un experto humano el que decide de forma heurística el número de intervalos de discretización y sus límites para cada componente del caso, no teniendo porque coincidir

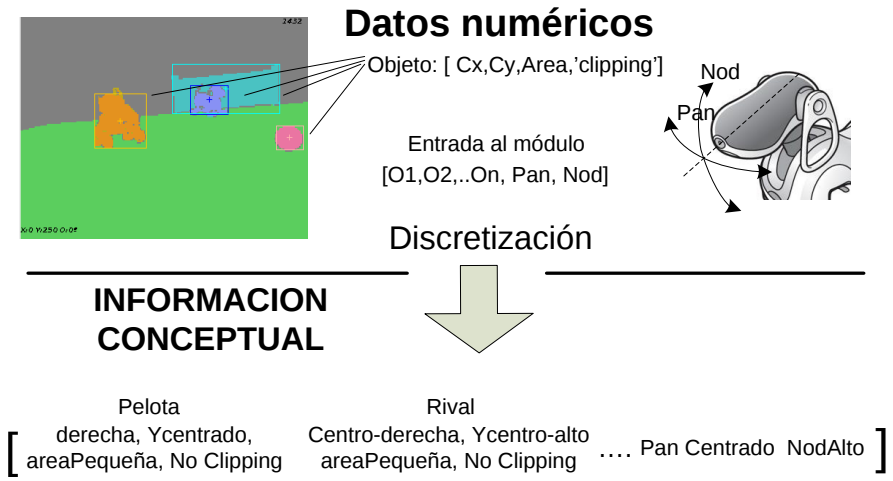


Figura 5.13: Discretización/Conceptualización de la información de entrada para conformado del caso

los valores escogidas para los diferentes componentes. No obstante, sería deseable que en un futuro dichas decisiones fuesen tomada automáticamente a partir de un análisis estadístico de los datos de entrada en su formato continuo, obtenidos en la fase de aprendizaje.

4.2.2. Preprocesado de la instancia de entrada

El proceso de discretización o conceptualización de la información de entrada indicado en el apartado anterior nos permite encajar mejor los aspectos de diseño de la arquitectura propuesta con el marco teórico de inteligencia y aprendizaje en el que se inspira la arquitectura. Sin embargo, este proceso de discretización no está exento de problemas, debido a la naturaleza cuadrúpeda del robot que estamos utilizando como demostrador. Los robots cuadrúpedos —y los que en general tiene patas—, experimentan un pronunciado balanceo durante su movimiento, el cual provoca un movimiento periódico continuo de la imagen captada. Este tipo de problemas no está presente, o al menos de forma tan acentuada en los robots rodantes. A consecuencia de este movimiento, se producen desplazamientos bruscos de los objetos visualizado por el robot, por lo que su posición, tamaño, y encuadre a lo largo del tiempo vienen determinados, no solo por la trayectoria del robot, sino por la situación del cuerpo y cabeza del robot dentro del ciclo de posiciones intermedias que permiten la locomoción. En la figura 5.14 podemos ver un ejemplo de la evolución de la coordenada x del centroide de un objeto en fotogramas consecutivos tomados desde la cámara del robot a medida que el robot se acerca al objeto en línea recta. El punto medio de la imagen está en 0.5. También se muestra en cada punto de la gráfica el porcentaje del ciclo de movimiento del robot, en el momento que fué tomada la imagen, considerando que dicho ciclo de movimiento es periódico.

En las pruebas preliminares en las que se consideraron los datos numéricos en bruto para la descripción de los casos, estos efectos eran automáticamente absorbidos por el sistema *CBR*, compensando los posibles errores a través de la alta reactividad del sistema. Sin embargo, tras llevar a cabo el proceso de discretización/conceptualización descrito, el sistema sufrió una bajada en su rendimiento, produciéndose situaciones de identificación incorrecta de las situaciones presentadas al módulo. Hay que tener en

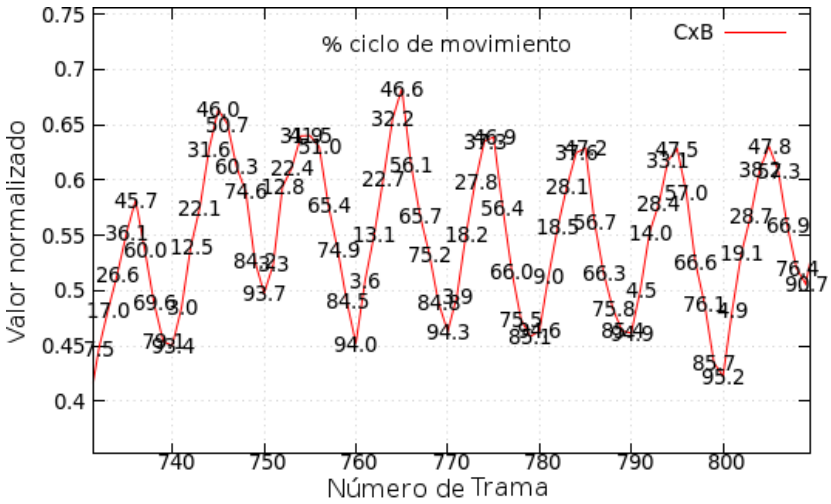


Figura 5.14: Evolucion Cx en movimiento recto debido al balanceo

cuenta que la velocidad de desplazamiento del robot utilizado no es muy alta, de manera que la distancia de desplazamiento de los objetos en el campo visual es a menudo inferior a la obtenido como consecuencia del balanceo del robot. Esta da lugar a que, tras la discretización, algunas componentes cambien de rango de forma contradictoria con el desplazamiento del robot, provocando una mala respuesta del sistema.

Este tipo de problemas ha sido afrontado tradicionalmente mediante técnicas de procesamiento visual de secuencias de imágenes, como la substracción de fondo [267]; o una caracterización cinemática del desplazamiento de la cámara durante la secuencia de movimiento, que permitiese compensar el desplazamiento de la imagen de forma algorítmica [268]. Sin embargo estas técnicas requieren a menudo de un complejidad computacional elevada, lo que las hace difíciles de utilizar en entornos muy reactivos. Para mejorar la robustez de nuestro sistema, hemos propuesto una técnica sencilla y rápida de filtrado, que evalúa la evolución de los parámetros de los objetos a lo largo del **ciclo de movimiento** del robot, devolviendo un promediado de la posición y área de los objetos a lo largo de dicho ciclo. Este ciclo de movimiento puede ser conocido en todo momento, como un parámetro interno del estado del robot. La estimación obtenida es suficientemente precisa como para permitir una importante mejora del comportamiento del sistema, compensándose en todo caso las posibles imprecisiones, a través de la naturaleza reactiva de los comportamientos. En la figura 5.15 podemos ver la posición del centroide estimada para un objeto durante un ciclo de movimiento del robot, comparada con la posición real del objeto.

Otro problema adicional surge cuando en el ciclo de movimiento estimado el objeto aparece y desaparece en las diferentes imágenes, al encontrarse bordeando los límites del campo visual (figura 5.16). La técnica tradicional para tratar este problema pasa por reducir el campo útil de la imagen del robot, rodeándola de un "marco", de forma que solo se consideren los objetos que permanezcan en el campo visual imagen-marco durante toda la secuencia observada. Esta solución fue descartada, dada la escasa resolución de la cámara de nuestra robot.

En su lugar se consideró que los objetos observados en el ciclo de movimiento serían tenidos en cuenta solo si permanecían visibles durante al menos 2/3 de las imágenes de la secuencia. Los descriptores de posición del objeto se modificarían "compensando" su

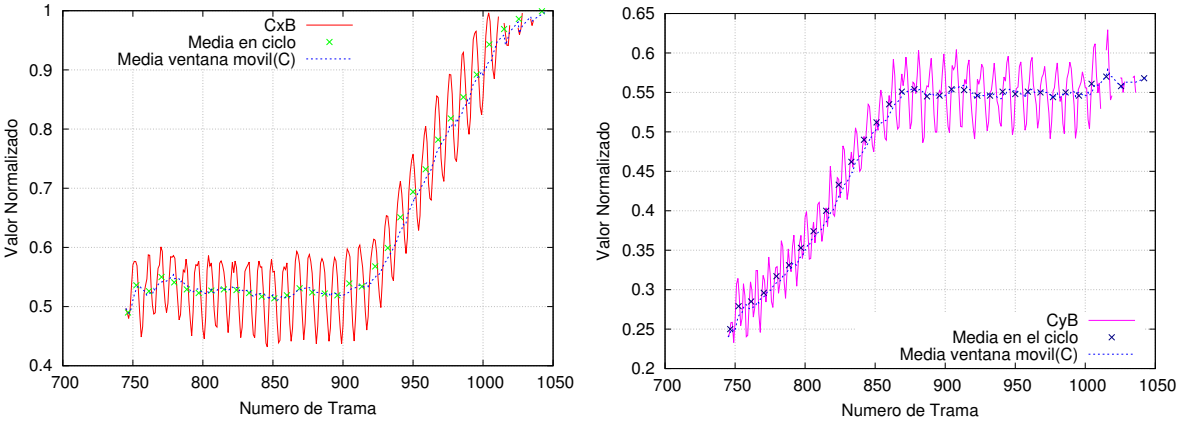


Figura 5.15: Filtrado de datos de posición (Cx,Cy) para compensar el balanceo

cercanía al borde de la imagen por la que desaparecen; y se considerarían los valores máximos de área y *clipping* observados. Pese a que los valores obtenidos no se van a corresponder con los que se promediarían en caso de un mayor campo de imagen, hay que tener en cuenta que estamos trabajando con información que finalmente será conceptual, por lo que esta combinación de valores puede perfectamente representar la descripción de un objeto incluyendo el hecho de que dicho objeto está en el borde de la imagen y apareciendo y desapareciendo durante el movimiento.

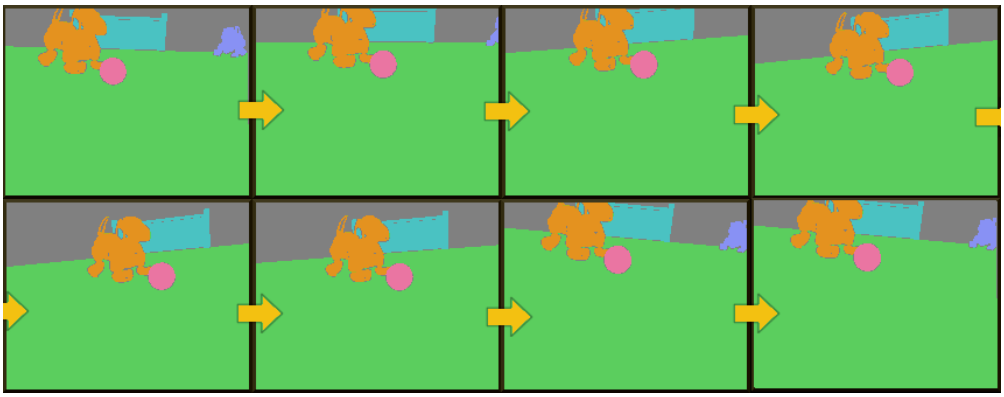


Figura 5.16: Aparicion y desaparicion de objetos en un ciclo de movimiento

4.2.3. Salida del caso CBR: Respuesta de comportamientos

Cada caso, a través de la información conceptual encerrada en sus componentes, representa una descripción de una situación del escenario en el que se desenvuelve el robot por medio de sus comportamientos. Aunque la realización de acciones no es imprescindible para la inteligencia, si que suele ser una demostración de la misma, especialmente en el caso de los comportamientos, que se definen dentro de un marco "sensación-acción". Por ello, los distintos comportamientos que diseñaremos en nuestra arquitectura tendrán asociadas respuestas de tipo motor y, dado que el sistema está continuamente interpretando la situación del entorno y realizando predicciones acerca de la misma, esas acciones motores tendrán un alcance temporal a corto plazo.

En el diseño de los comportamientos propuestos consideraremos que el robot está continuamente moviéndose o realizando acciones, ya que un estado de pausa o

parada como respuesta ante una situación implicaría un ciclo estático indefinido si consideramos únicamente como entrada la información procedente de la última imagen. Si en el futuro aumentamos el alcance temporal de la respuesta del robot más allá de un horizonte puramente reactivo, podremos considerar estados de parada momentánea del robot —por ejemplo, esperando a que un compañero de equipo se sitúe en una determinada posición antes de realizar una acción—. No obstante, en ese caso habría que aplicar también alguna componente temporal de tiempo máximo de espera ante situaciones en las que no hay variación de las condiciones del entorno, para evitar que el robot entre en un ciclo estático de forma permanente. Teniendo en cuenta que la navegación del robot considerado se realiza indicando un vector de movimiento en tres componentes —*forward*, *strafe*, y *rotate*—, hemos considerado que la salida de cada caso tendrá esta forma vectorial, añadiendo la posibilidad de desencadenar acciones automáticas pregrabadas —como chutar, o lanzarse hacia delante— en determinadas situaciones (figura 5.17).

Este vector de movimiento se mapearía con cada caso correspondiente a través de una fase de aprendizaje por observación, en el que se capturarían las situaciones experimentadas por el robot y los vectores de movimientos indicados por el entrenador humano a través de un joystick ante dichas situaciones, mientras ejecuta la pauta de conducta a aprender por el comportamiento en cuestión [196].

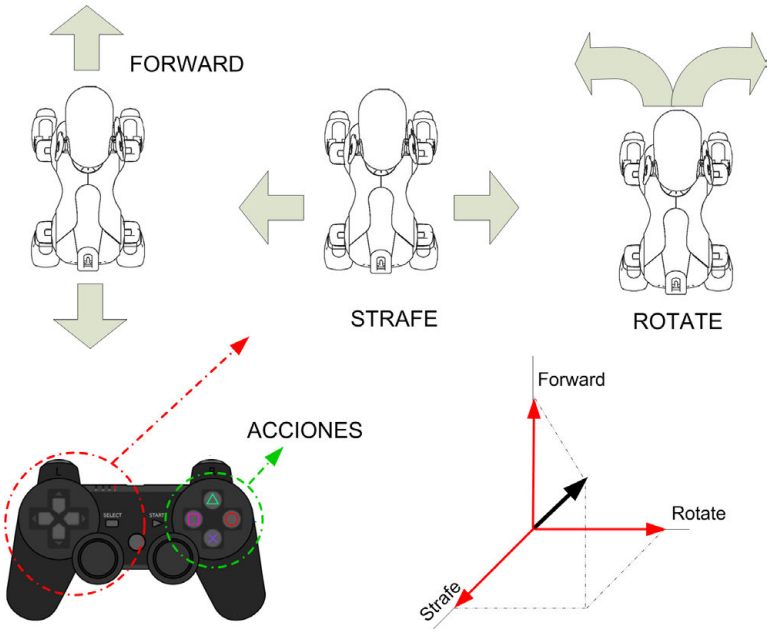


Figura 5.17: Control de movimiento del robot - Salida del caso

Sin embargo, el uso de componentes vectoriales unitarios en las 3 dimensiones indicadas presenta el problema de dificultar en exceso la combinación de las salidas de los distintos comportamientos para lograr construir un comportamiento complejo emergente a partir de varios comportamientos básicos. Más allá de la conmutación entre la salida de uno u otro módulo, cualquier tipo de combinación aditiva ponderada de las salidas de los módulos componentes podría dar lugar fácilmente a un vector nulo que provocase la parada del robot y la entrada en el ciclo estático comentado en el párrafo anterior. Para reducir este problema se ha optado por aumentar la frecuencia de observación del sistema durante la fase de aprendizaje, y asociar a cada caso no

un vector individual, sino una cadena de vectores que represente una trayectoria del robot asociada a la instancia de entrada (figura 5.18). A mayor longitud de la cadena de salida, más sencilla será la construcción del vector de salida correspondiente al comportamiento emergente (figura 5.19) ; pero también se reducirá la reactividad de dicho comportamiento, ya que se tardará mas tiempo —el empleado en aplicar cada uno de los elementos de la cadena de vectores— en realizar una nueva consulta a la base de casos y poder con ello rectificar posibles errores en la actuación del robot.

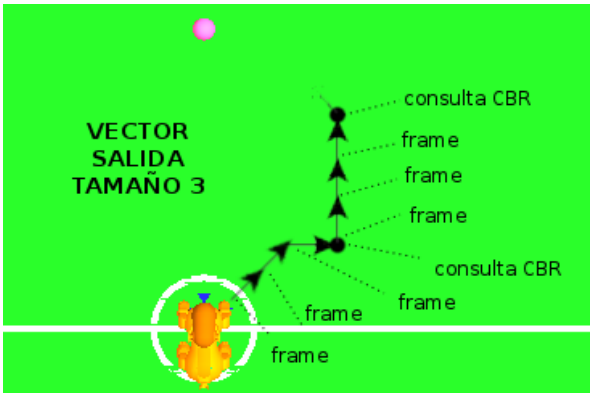


Figura 5.18: Ejemplo de acción de salida de un comportamiento

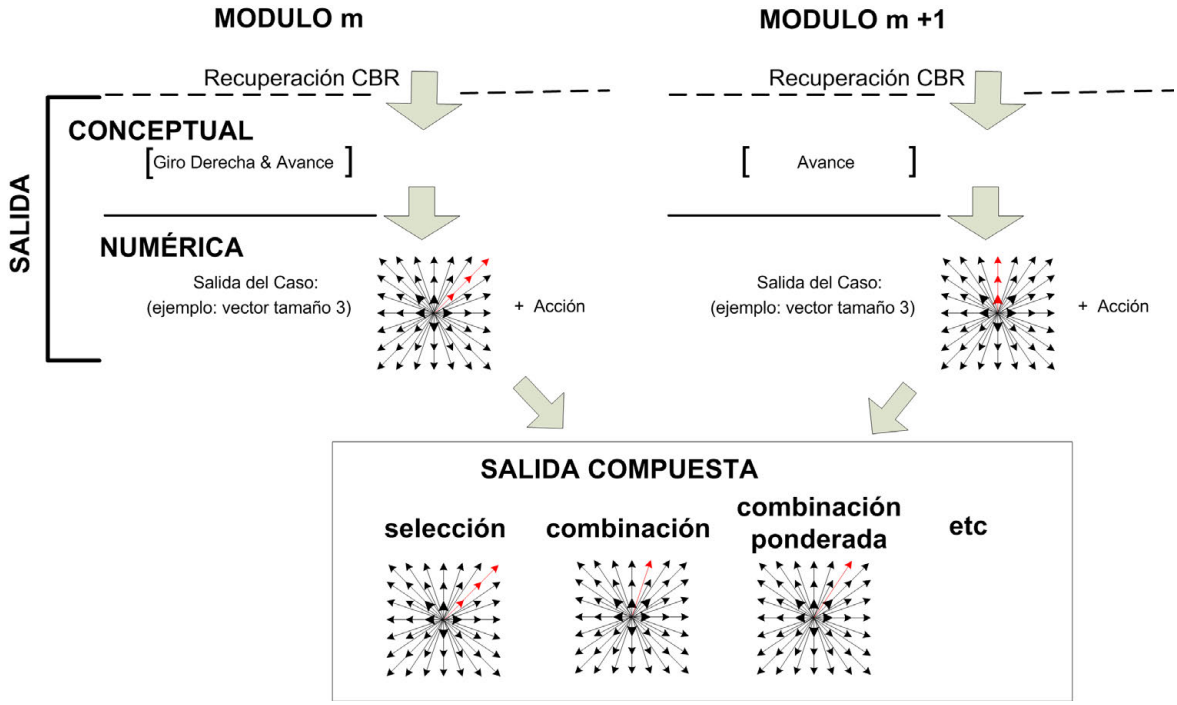


Figura 5.19: Composición de salidas de comportamientos

4.2.4. Adquisición inicial de conocimientos. Aprendizaje por observación

Para conseguir que un comportamiento ofrezca un mínimo de operatividad, es necesario proporcionarle una base de casos que integre los conocimientos suficientes para representar la diversidad de situaciones que va a experimentar el módulo

durante la fase de ejecución. Esta información se adquiere en una fase previa de aprendizaje supervisado a través de una serie de pruebas en las que el experto opera el funcionamiento del robot ante distintos escenarios, de manera que el robot almacena en la base de casos la representación de las situaciones experimentadas junto con la respuesta establecida por el operador ante dichas situaciones. Es necesario destacar que el comportamiento aprendido no tiene que ser forzosamente el más eficiente para lograr su supuesto objetivo: en teoría será una síntesis de lo que haya realizado el operador, incluyendo posibles errores o malas decisiones de éste. No obstante, si suponemos que el entrenador es suficientemente hábil en la operación del robot, el conocimiento adquirido por el robot será el adecuado. Ineficiencias puntuales, como pequeños errores en trayectorias, o dudas en la actuación serán absorbidos por el sistema y compensados por la reactividad de la arquitectura. Es importante que el entrenador base sus respuestas en la información que percibe el robot en cada momento, y no tenga en cuenta aspectos globales y temporales que no van a estar incluidos en la instancia de entrada. Por ello este proceso no resulta siempre sencillo, especialmente en lo relativo a la característica temporal del comportamiento que se está entrenando.

La adquisición de conocimientos se puede realizar mediante una prueba "no controlada" de cierta duración, en la que el experto trata de realizar la tarea asociada al comportamiento durante un tiempo suficiente —por ejemplo, jugar un partido de fútbol completo— ; o a través de pequeñas pruebas correspondientes a situaciones controladas y decididas por el experto, cada una de las cuales integraría una parte de la base de conocimientos. Esta segunda opción tiene la ventaja de que es más fácil de controlar y depurar, pero a cambio exige un procedimiento más disciplinado con una planificación de los escenarios necesarios. También es posible añadir manualmente información que no se obtenga a través del aprendizaje por observación, por ejemplo para situaciones que rara vez se van a experimentar pero para las cuales se tiene claro cual debe ser la respuesta. Cuanta mayor sea la información obtenida en esta fase de entrenamiento previo, mejor será la respuesta general del comportamiento, toda vez que el formato de los casos —elección de información relevante, rangos de discretización, etc...— sea el adecuado.

En cuanto a las situaciones que no se incorporen en esta fase, bien por omitir las pruebas correspondientes, o porque las instancias concretas no sean adquiridas, existirá la posibilidad de incorporarlas posteriormente a la base de conocimientos del módulo, a través de una fase de adaptación *CBR* realizada durante la fase operativa del comportamiento.

4.3. Organización y recuperación de la información: estructura de la base CBR

Si bien el caso representa la unidad básica de información de los módulos que implementan cada uno de los comportamientos básicos, tan importante como el diseño adecuado del caso, en cuanto a la elección de los componentes de información que se proporcionarán al módulo así como la conceptualización o discretización que se aplicará a los mismos, es la estructura en que se organizarán todos los casos dentro de la base *CBR*. Esta estructura tiene una importancia capital en la capacidad de predicción del robot y especialmente en la velocidad a la que es capaz de realizar nuevas predicciones. Todo sistema *CBR* debe ser capaz de explorar la información almacenada en su anterior

para, ante un problema que se presente, predecir cual es la respuesta más adecuada en función de las experiencias ya vividas ante esa misma situación u otra parecida.

La fase de recuperación (*retrieve*) de un sistema *CBR* conlleva codificar el problema a resolver en el mismo "lenguaje" de representación de la información en que están codificados los casos, y buscar el caso más parecido expresado en este "lenguaje". La codificación del problema se realiza de la misma manera que se indicó en el apartado 4.2.1, incluyendo la conceptualización según el mismo orden de conceptos que se utilizó para los casos.

Con respecto a la recuperación de la experiencia más cercana al problema, depende de la estructura de organización del caso (apartado 4.2) en la base *CBR* , en particular en el tiempo necesario para completar el ciclo. Las estructuras jerárquicas/en-árbol son las que permiten unos tiempos de recuperación más rápidos, ya que el avance por las diferentes ramas se realiza mediante comparaciones de componentes individuales. Sin embargo, estas estructuras plantean problemas en la descomposición del caso entre los diferentes niveles, ya que la asignación de cada componente de información debería estar relacionada con su importancia o incluir al resto de componentes como sub-categorías, algo que no siempre es posible. Una mala distribución de la información conllevaría el riesgo de "viajar" por la rama equivocada del árbol y no recuperar el caso más adecuado, especialmente cuando ninguno de los casos almacenados coincide exactamente con el problema planteada. Además, la inclusión de nuevos casos y, en general, el mantenimiento de la base presenta mayores dificultades que otras estructuras.

Una alternativa pasaría por una transformación de la información hacia una estructura en categorías u ontologías [269], pero esto exige, de nuevo, un conocimiento detallado del problema y de las relaciones entre los elementos de información del comportamiento, y parece más apropiado para niveles más abstractos del esquema de inteligencia que el nivel reactivo que estamos tratando.

Como alternativa se planteó inicialmente un almacenamiento de la información en una base plana, de forma que la recuperación se basa en la comparación uno-a-uno del problema con todos los casos almacenados en la base. En este esquema la información del caso se considera como un todo, aunque es importante asignar un peso adecuado a cada componente según su influencia en el comportamiento. De esta manera se evitan los problemas de selección propios de la estructura jerárquica, y además la inclusión de nuevos casos a la base se convierte en algo trivial. Ante un nuevo problema se calcula la semejanza entre el mismo y todos los casos de la base mediante una función de similitud, (apartado 3.2.2), que nos permite establecer una ordenación de parecido de todos los casos almacenados en la base del comportamiento con respecto al problema. A partir de esta ordenación existen diferentes estrategias, como escoger los n primeros casos mas similares y establecer una respuesta a partir de los mismos, o simplemente elegir el caso mas similar, que es la estrategia que hemos decidido seguir en nuestro trabajo (figura 5.20). En relación a la elección de la función de similitud, éste es un aspecto muy dependiente del problema y para el cual no existe una traslación sencilla entre dominios. Una hipótesis muy influyente defiende que la distancia Euclidea es más adecuada cuando las dimensiones de los estímulos están integradas perceptualmente, como por ejemplo la saturación y el brillo de un color; mientras que la distancia Manhattan o *city-block* es apropiada cuando las dimensiones de los estímulos son perceptualmente separables, como el color y la forma de un objeto [270]. Por este

motivo hemos elegido como función de similitud para nuestros experimentos de prueba, la distancia Manhattan discreta [249], ya que consideramos que la mayoría de los componentes de información pertenecen al segundo tipo. En el futuro se planteará la posibilidad de utilizar distancias mixtas en las que cada tipo de componentes sea considerado, a nivel de similitud, con una u otra distancia, según sus características de percepción. En los experimentos realizado no haremos distinción en los pesos a aplicar a cada uno de los componentes del caso, si bien este sería un apartado a estudiar en trabajos futuros.

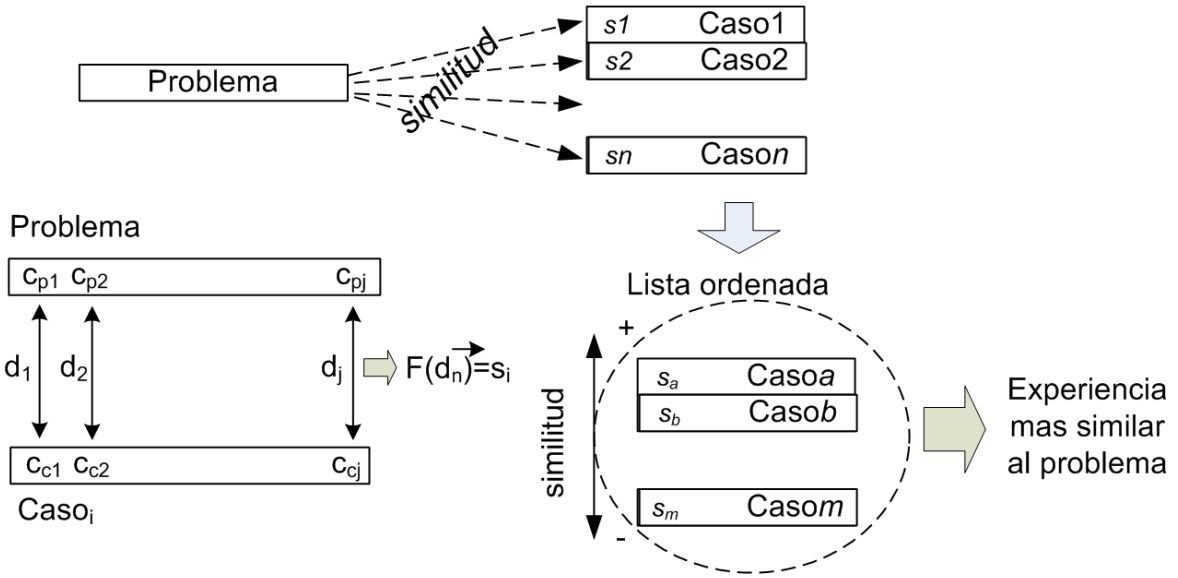


Figura 5.20: Recuperación por similitud del caso mas cercano al problema

4.3.1. Diseño de comportamientos complejos: problemas de escalabilidad

Los seres humanos realizan decenas de predicciones a cada segundo relacionados con ámbitos muy distintos del entorno en el que se desenvuelven, a veces de forma activa y otras de forma subconsciente. De la misma forma, un robot que intentase replicar el esquema de inteligencia humano basado en predicciones a partir de experiencias, necesita también ser capaz de realizar un gran número de predicciones por segundo, más aún si el comportamiento sobre el que se va a aplicar la predicción tiene una fuerte componente reactiva. Si el robot no es capaz de realizar todo el proceso de recuperación y adaptación de experiencias a tiempo, no podrá aplicar correctamente su respuesta y fracasará en su operación. Si los módulos a diseñar son sencillos, esto es, la cantidad de información asociada es pequeña, en cuanto a número de componentes del caso, y variedad de estos, no suele existir problema en la recuperación de experiencias de los mismos. En [63] se presentaron diversos experimentos en los que cada módulo o comportamiento a diseñar giraba en torno a la influencia aislada de un elemento del escenario en el comportamiento. Así, los casos que integran el conocimiento de cada módulo tienen un número pequeño de componentes y, tras una discretización, la variedad de posibles casos es menor: en estos experimentos bastaban unas pocas decenas de casos para adquirir la información necesaria para definir el comportamiento. Sin embargo, con esta filosofía de diseño, la integración de módulos de bajo nivel para obtener comportamientos emergentes se complicaba en demasía: tal como indica la

teoría de la información en *CBR* (ver apartado 3.3), ésta se tiene que repartir de una u otra forma entre los elementos que integran el sistema de conocimiento, de manera que, la que no forme parte de cada base *CBR*, se debe incluir posteriormente, en este caso en la fase de composición emergente de los módulos básicos. Y es esta inclusión la que resulta muy difícil, ya que implica un conocimiento adicional de la interrelación entre las respuestas individuales a cada uno de los objetos. En este sentido, se desaprovechan las capacidades de *CBR* para absorber automáticamente las experiencias de las cuales se nutre el aprendizaje. Por tanto, si se desea diseñar comportamientos más complejos que impliquen varios elementos de influencia, parece más adecuado considerar todos esos elementos como información de entrada de un módulo *CBR*, de manera que a través del aprendizaje se adquiera el conocimientos de la respuesta ante cada elemento individual, pero también ante ante la interacción o presencia simultánea de los mismos. Esta inclusión se ve limitada por aspectos de escalabilidad relacionados con la dimensionalidad del espacio de información de los casos, tal como se vio en el apartado 4.2.1, por lo que es indispensable escoger adecuadamente los elementos de información de entrada al módulo para evitar redundancias pero también falta de información. Aún así, la información almacenada finalmente puede ser tan amplia como para que la fase de recuperación *CBR* comprometa la reactividad en la respuesta del comportamiento. Por este motivo, se proponen alguna modificaciones a la estructura "tradicional" plana de la base de casos *CBR*, así como a la fase de recuperación, modificaciones que permitirán acelerar el proceso de recuperación *CBR* y mejorar la reactividad del sistema.

4.3.2. Estructura mixta de la base CBR y recuperación multietapa de la información

Tras descartar las estructuras jerárquica y plana para la base *CBR*, por los motivos ya expuestos, se planteó finalmente la utilización de una estructura "mixta" que se podría encuadrar dentro de la categoría de representación generalizada del caso (figura 5.21). Para ello, se distinguen ciertos componentes del caso que representan información de "contexto", con generalmente pocas alternativas que son excluyentes entre sí. Estos componentes de "contexto" implementan la parte jerárquica del esquema, estableciendo un árbol con parte de la información, y que tras atravesarse, desemboca en una u otra sub-base *CBR* con estructura plana, situada en las hojas del árbol. De esta forma se subdivide la información en tantos "subconjuntos" como hojas tengo el árbol —lo cual viene determinado por el número de componentes de contexto identificados—, y la búsqueda por similitud se produce en un subconjunto reducido de la información acelerando el tiempo de respuesta al operar sobre un número menor de casos. Por supuesto, el éxito de esta estrategia de diseño está relacionada con la identificación correcta de elementos de información de contexto, lo cual no siempre es sencillo para cualquier dominio del problema. En el dominio de aplicación que se está considerando en este trabajo, un ejemplo de información de contexto podría ser la posesión o no de la pelota; o la observación de la portería propia, la rival, o ninguna portería. Para acelerar aún mas la fase de recuperación *CBR*, se ha modificado la recuperación clásica por similitud con todos los casos, añadiendo una etapa previa de *recuperación por índice de objetos en escena*. Dicha etapa se basa en la asociación a cada caso de una información adicional que representa la existencia o no, en cada caso, de los objetos de interés en el comportamiento. Esta información toma la forma de una cadena binaria, donde cada objeto ocupa una determinada posición en la cadena, de forma que se

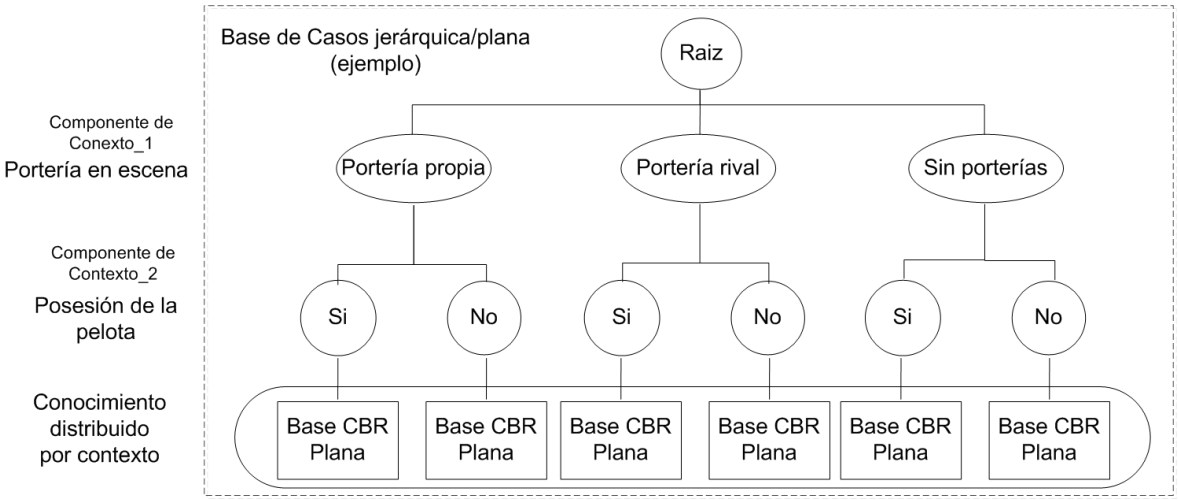


Figura 5.21: Estructura jerárquica-plana de la base de casos

puede representar su presencia ("1") o ausencia ("0") en la escena relacionada con el problema (figura 5.22). Tras la adquisición de conocimientos por aprendizaje cada caso incluiría, además de su información conceptualizada, una cadena de representación de objetos. Ante un nuevo problema, se calcularía su cadena binaria y se compararía con las cadenas de todos los casos almacenados, seleccionado solo un subconjunto de los mismos. La comparación puede ser "estricta" con una función *OR* que solo seleccionaría casos representando situaciones con exactamente los mismos objetos que el problema; o se puede utilizar una función Jaccard, que admitiría los casos pertenecientes a la clase más cercana al problema en cuanto coincidencia de objetos. En cualquier caso, el coste computacional de estas operaciones es mucho menor que el empleado en las funciones de similitud más comunes, de forma que se consiguen importantes mejoras en los tiempos de respuesta, mayores cuantas mas objetos hay presentes en la definición del comportamiento. Únicamente en situaciones en que los casos incluyan siempre los mismos objetos se produce un empeoramiento del tiempo de respuesta. En el capítulo de Experimentos se presentarán algunos resultados de este esquema de organización y recuperación de la base con respecto al esquema tradicional de base plana.

En [271] se realizó un estudio para determinar la ganancia obtenida al aplicar las variaciones propuestas sobre la fase de recuperación CBR. En dicho estudio se constató el efecto beneficioso de los cambios propuestos en la reducción del tiempo de recuperación del caso *CBR* más cercano (figura 5.23). Asimismo se observó como este efecto era más acentuado conforme aumentaba la complejidad del comportamiento a adquirir, ya que dicha complejidad se asocia a la dimensionalidad de los casos y, por tanto, a la cantidad de casos necesarios para aprender dicho comportamiento.

4.4. Fase operativa: aprendizaje por experiencia

Tras la fase de aprendizaje por observación, y si el diseño de los módulos, de la representación de los casos, y de la conceptualización de la información ha sido suficientemente correcto, el robot estará preparado para ejecutar el comportamiento aprendido. No obstante, puede que la respuesta autónoma no sea siempre la adecuada ante cualquier situación que se presente, debido principalmente a que no hay garantías de que el entrenamiento por observación haya sido exhaustivo, cubriendo todas las

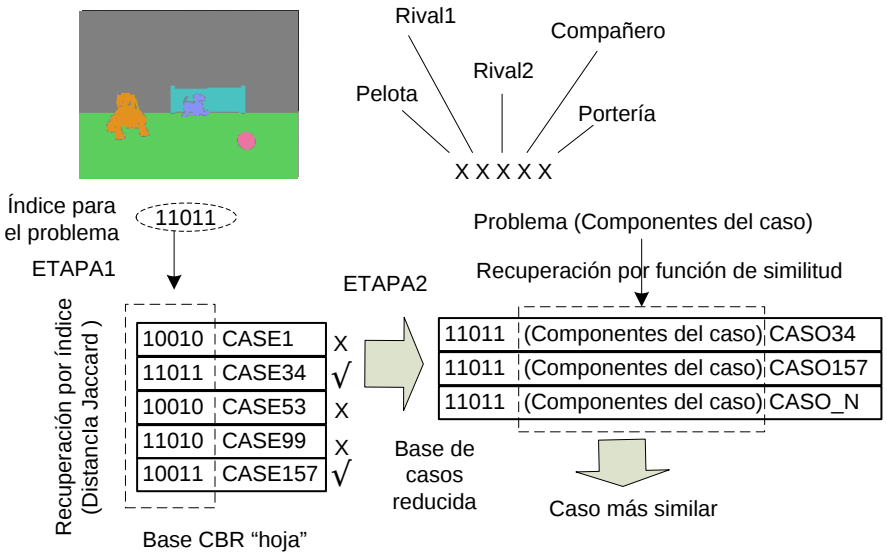


Figura 5.22: Recuperación del caso con etapa de indexación de objetos

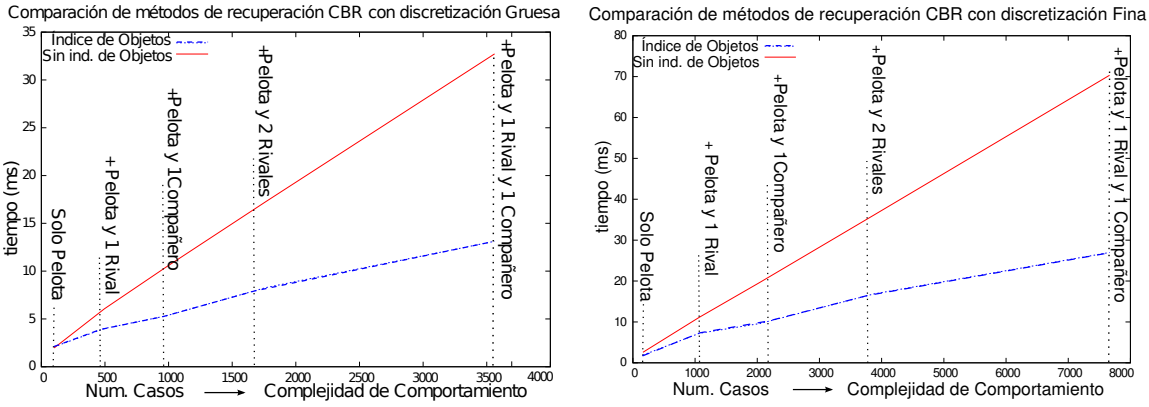


Figura 5.23: Mejoras en la fase de recuperación CBR usando una etapa previa de indexación de objetos

posibles situaciones posibles —de hecho no es suele ser deseable tal medida—. También en el aprendizaje de los seres humanos el aprendizaje por observación/tutorización solo introduce los fundamentos básicos y generales de la conducta a aprender. Es necesario que el usuario refuerce este aprendizaje, lo adapte a sus características particulares, y resuelva los "huecos" de conocimiento existentes a través un aprendizaje por experiencia. Este tipo de aprendizaje se asocia a las fases de reutilización (*reuse*), revisión (*revise*), y mantenimiento (*retain*) del ciclo *CBR*, e implica la definición de estrategias y políticas adicionales, como veremos a continuación.

4.4.1. Reutilización del caso y adaptación de la respuesta

Ya vimos en el apartado anterior como la recuperación del caso mas similar de entre los almacenados en la base *CBR* propia del comportamiento suponía, en teoría, la predicción de una respuesta adecuada —la asociada al caso recuperado— según las experiencias previas del robot. Si el problema propuesto coincide completamente con alguno de los casos almacenados, eso significa que se está volviendo a repetir una situación ya experimentada anteriormente y que, por tanto, es adecuado responder de la misma forma en que se hizo en esa ocasión previa[1]. Una posible optimización pasaría por evaluar los casos adquiridos y adaptarlos con vistas a mejorar el comportamiento aprendido. Aún así, suele ser frecuente que el problema no encuentre un caso exactamente igual en la base del comportamiento; en ese caso se realiza una predicción basada en la experiencia almacenada más similar, mediante la extracción del caso que presente mayor similitud. Este mismo proceso es el que realizamos los seres humanos cuando establecemos analogías, o encontramos situaciones completamente nuevas y exploramos nuestra memoria para encontrar alguna situación que se asemeje. Sin embargo, el caso más similar no tiene porque ser necesariamente un caso adecuado al nuevo problema; si la distancia entre el problema y el caso recuperado es muy grande significa que, si bien la situación expresada por el caso es "la más parecida entre las experiencias almacenadas", no es lo suficientemente parecida para ser utilizada directamente. En este caso es necesario plantear una nueva respuesta que puede ser derivada, en parte, de la respuesta asociada al caso recuperado.

La adaptación de un caso recuperado supone "probar algo nuevo" distinto de lo ya conocido y/o experimentado. Los seres humanos, cuando no somos capaces de encontrar una analogía para un nuevo problema, también "improvisamos" y aplicamos conocimientos que puedan ser, en principio, alejados del que estamos considerando en un momento dado. La adaptación del caso puede realizarse de muy diversas formas, desde utilizar una respuesta completamente aleatoria, hasta acudir a otros algoritmos o conocimientos de otros módulos. Lo que si parece razonable es que la nueva respuesta esté basado o incluya elementos de la respuesta asociada al caso recuperado, que tendrán un peso mayor en tanto la similitud entre el problema y dicho caso sea también mayor.

En nuestra arquitectura de comportamientos se planteó en primer lugar cual debía ser el criterio que determinase si el casos recuperado era "suficientemente parecido" al problema, como para aplicar directamente la respuesta almacenada en dicho caso. Inicialmente se consideró la posibilidad de establecer un umbral mínimo de similitud que, una vez traspasado, determinaría la adaptación de la respuesta incluida en el

[1]Suponemos que el entrenamiento por observación ha sido adecuado y sin errores

caso recuperado. Para ello se ejecutaron varias pruebas en las que se analizó qué valor mínimo de similitud, normalizada en $[0, 1]$, permitía aún un comportamiento correcto al aplicar la respuesta del caso recuperado, pero no se obtuvieron resultados concluyentes. Consideramos que esto se debe a la conceptualización de la información de entrada para definir el problema y los casos, que implica diferente número de categorías para cada componente, y en muchos casos, no equiespaciada. Además, cuando una situación o escenario está descrito por muchos componentes, pequeñas diferencias en varios de ellos se consideran tolerables desde el punto de vista de la similitud de las situaciones, pero una gran diferencia en aunque sea solo uno de ellos, no. Este último caso podría llegar a pasar desapercibido en el cálculo de la función de similitud, haciendo pasar por similar una situación descrita en un caso que sea completamente distinta a la del problema. Un ejemplo claro lo podemos ver en la posición de la cabeza del robot: un mismo escenario de objetos presentes en la imagen con una disposición del cuello muy distinta representará, por lo general, situaciones muy diferentes para el comportamiento y que de ser tomadas como similares provocará, generalmente, un comportamiento erróneo del robot (figura 5.24).

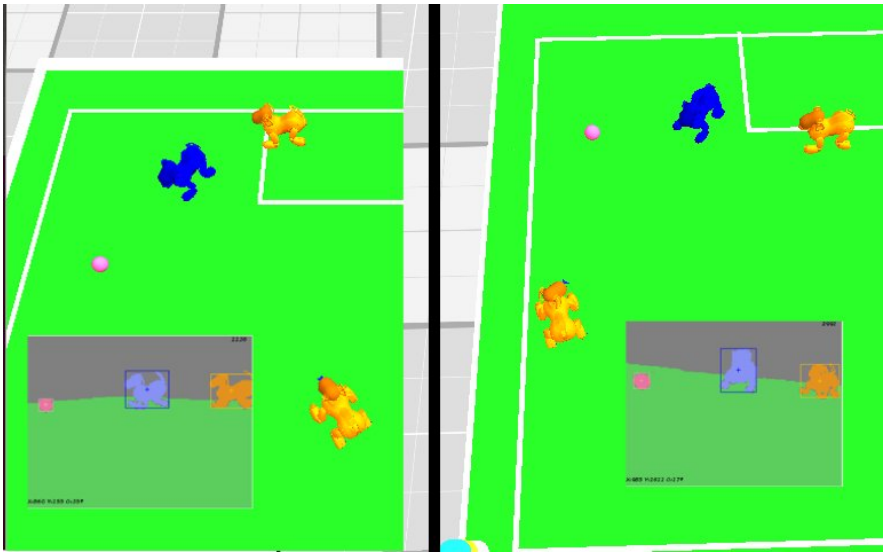


Figura 5.24: Identificación errónea de escenarios por posición de cámara

Por este motivo, se decidió utilizar un criterio distinto para decidir la adaptación o no del caso: la situación descrita por el problema sería suficientemente similar al caso recuperado como para aplicar su respuesta, si el número de "saltos" en los rangos o categorías conceptuales no superaba un máximo para cada componente individual, S_i; para el conjunto de componente que describen cada objeto; y para el conjunto de todos los componentes que describen el caso (Ecuación 5.3):

$$\text{Adaptación Caso si} \begin{cases} \exists i, & S_i > umbral_{CmpIndiv} & \text{, o} \\ \exists Obj, & \sum_{i \in ObjComp} S_i > umbral_{CmpObj} & \text{, o} \\ & \sum_{i \in CasoCmp} S_i > umbral_{CmpCaso} & \end{cases} \tag{5.3}$$

donde $umbral_{CmpIndiv}$, $umbral_{CmpObj}$, y $umbral_{CmpCaso}$, son los umbrales que determinan respectivamente si la similitud es suficiente o no, a nivel de componente

individual, de objeto, y de caso al completo. Actualmente, estos umbrales se calculan de forma heurística, a través de la observación del comportamiento del robot en situaciones en las que hay poco entrenamiento previo por observación. Hay que tener en cuenta que su valor está relacionado con el número rangos o categorías de los componentes en el proceso de conceptualización de los mismos, por lo que en el futuro se estudiarán alternativas para poder generalizar un método de cálculo.

En cuanto al algoritmo explícito para generar una respuesta adaptada, en caso de que el criterio de la ecuación 5.3 así lo determine, se plantearon diversas alternativas, desde un vector de salida completamente aleatorios, hasta la utilización de algoritmos genéticos, para la modificación de la respuesta original del caso. Hay que tener en cuenta que esta adaptación supone la aplicación de una solución no-óptima como complemento de la solución aprendida que no se considera lo suficientemente buena y aproximada a la situación real. Se consideró la utilización de un Campo de Histogramas Vectoriales (*Vector Histogram Field (VHF)*) [272], pero esta técnica es más adecuada para una representación global espacial en 2D del entorno del robot y, en nuestro caso, estamos trabajando con una proyección 2D del campo de visión local del robot, por lo que no parecía sencillo trasladar el algoritmo a estas circunstancias. Finalmente, se decidió generar un posible vector de respuesta, obtenido mediante Campos de Potencial (*PFA*) [31] [273] de los objetos que influyen en el comportamiento, y que se combinaría con la salida del caso recuperado, con pesos determinados por la similitud entre dicho caso y el problema. A pesar de que los campos de potencial parten también de una representación global 2D del entorno en la que es necesario, en principio, conocer la distancia entre el robot y los objetos, en este caso si hemos podido representar los efectos de atracción y repulsión de los mismos basándonos en su área visible en el campo de visión, que sería proporcional a su distancia o, en caso de estar parcialmente oculto en un extremo de la pantalla, indicaría una menor influencia en el comportamiento del robot, coherente con un menor área visible. El vector de campo de potencial, $\vec{V_{PF}}$ se obtendrá como un sumatorio de los vectores generados por los distintos campos de potencial de cada uno de los objetos que afectan al comportamiento (ecuación 5.4). El cómputo de los campos de potencial para cada objeto, depende de las características del comportamiento que se está diseñando, y en particular de como afecta al mismo. Así, por ejemplo, en un comportamiento para intentar tomar posesión de la pelota, los robots rivales funcionarían como repulsores, $\vec{V_{Rep_j}}$, y la pelota como atractor, $\vec{V_{Atr_i}}$. En el capítulo de experimentos se presentarán ejemplos de definición de esta función de potencial para el diseño de algunos comportamientos.

$$\vec{V}_{PF} = \sum \vec{V}_{Atr_i} + \sum \vec{V}_{Rep_j} \tag{5.4}$$

Una vez obtenido un vector de campo de potencial, $\vec{V}_{PF}$, dicho vector se combinará, de forma ponderada, con el vector de respuesta del caso recuperado, $\vec{V}_{CBRCase}$,tal como se indica en la ecuación 5.5:

$$\vec{V}_{adapt} = \vec{V}_{PF} * p_{PF} + \vec{V}_{CBRCase} * (1 - p_{PF}) \tag{5.5}$$

El parámetro $p_{PF} \in [0,1]$ representa la preponderancia o peso que se le da a la nueva respuesta generada frente a la adquirida por experiencia correspondiente a una situación que consideramos similar pero alejada como para aplicar la misma respuesta.

La elección de p_{PF} se puede hacer por diversos criterios:

- Una posibilidad sería considerar p_{PF} como d, la distancia entre el caso recuperado y el problema. Por desgracia, con esta aproximación encontramos el problema de que a menudo estamos adaptando casos que no tienen demasiada distancia con el problema, aunque sí la suficiente a nivel de"saltos" de categoría en un parámetro, objeto, o en el caso completo. En estas condiciones la aportación del vector *PF* sería casi nula en la mayoría de ocasiones. Para evitar este problema podríamos plantear algunas variantes:

  1. Normalizar con respecto al número máximo de "saltos" considerado como umbral mínimo de adaptación. Si se alcanza el umbral mínimo para considerar necesaria la adaptación, la contribución sería del 50 % por parte del vector generado mediante *PFA*, y del 50 % del obtenido a partir del caso recuperado. Para valores que superen ese umbral se consideraría cada vez más peso para el vector generado mediante *PFA*, hasta un cierto límite a partir del cual se despreciaría completamente la aportación del caso recuperado —se juzga que dicho caso es demasiado distinto al problema— y la respuesta del robot vendría dada únicamente a partir del vector *PF*. Este límite máximo también se puede determinar heurísticamente. También es posible establecer otros rangos de contribución para el vector *PF*, además de [50 %, 100 %], por ejemplo, empezando desde una contribución menor aunque entonces es muy posible que la aportación del vector *PF* sea inapreciable en gran cantidad de casos.
  2. Añadir un "factor de innovación", , que incentive la aplicación de nuevas respuestas generadas frente a las almacenadas en la base de casos. Este factor aumentaría el peso del factor d dando una mayor fuerza a $\vec{V}_{PF}$ en la generación del vector de respuesta adaptado. Se podría establecer con un valor constante elegido de forma heurística o aumentar su valor de forma adaptativa a medida que se compruebe que las respuestas generadas no se alejan los suficiente de las almacenadas en la base *CBR*.

$$\vec{V}_{adapt} = \vec{V}_{PF} * (d + \varphi_{inn}) + \vec{V}_{CBRCase} * (1 - (d + \varphi_{inn})) \tag{5.6}$$

- Como alternativa a la distancia, se podría escoger p_{PF} en función de la utilidad o eficiencia del caso recuperado, de manera que aunque este esté alejado del problema planteado, se supone que, tras su aplicación, se seguirá manteniendo una parte suficiente de su utilidad En este caso $(1-p)$ se sustituiría por el factor de utilidad η del caso recuperado (ecuación 5.10) y, al vector *PF* se le aplicaría un peso $(1-\eta)$ —suponemos que η está normalizado en [0,1]—. Por supuesto, sería necesario definir previamente un factor de utilidad a los casos, tal como se describe en el apartado 4.4.2.

En el capítulo 6 se analizarán las diferentes propuestas para determinar cuales son las más adecuadas a la estructura y características de la presente tesis. Además, se discutirá la solución de problemas relacionados con la forma de la respuesta del caso en la aplicación específica de ejemplo, como por ejemplo que hacer ante la obtención de un vector adaptado nulo, o como actuar cuando alguno de los vectores componentes incluye acciones pregrabadas.

Tras obtener el vector adaptado, éste se discretizará en una cadena de vectores, que se aplicarán al robot en los instantes apropiados. Si algún componente de la cadena

de vectores es nulo, se adelantará la siguiente consulta *CBR* para evitar la parada del robot (en nuestro sistema reactivo no contemplamos la posibilidad de que el robot esté parado).

4.4.2. Incorporación de casos y mantenimiento de la base CBR del comportamiento

Si como resultado del ciclo *CBR* se recupera un caso suficientemente similar al problema, simplemente habrá que aplicar la respuesta asociada al caso: se estará repitiendo una situación que será al menos muy parecida a otra experimentada previamente. Si, por el contrario, el caso obtenido es poco similar al problema, habremos tenido que adaptarlo para crear una nueva solución que será la finalmente aplicada. Este segundo caso se corresponde a una situación de experiencia nueva, para la cual se ha propuesto esa nueva solución. Tras aplicar dicha solución estaremos en condiciones de determinar si la misma aporta nuevos conocimientos interesantes para la definición del comportamiento, en cuyo caso, deberemos incorporar la misma al "banco de memorias" del módulo. Este proceso es el que se correspondería a la etapa de aprendizaje por experiencia, clave para adquisición de conocimientos e inteligencia de los seres humanos, y de cualquier modelo de inteligencia basado en éste. Para decidir acerca de la incorporación de la nueva experiencia en la base *CBR* como un nuevo caso, es necesario evaluar la utilidad del nuevo conocimiento y, en particular, de la solución considerada para el mismo. Esto se consigue a través de una función de eficiencia o utilidad (ver apartado 3.2.2) que determina si, tras aplicar la respuesta sugerida, el comportamiento se encuentra más cerca de lograr sus objetivos finales o intermedios.

La definición de una función de utilidad no es trivial: por un lado, es extremadamente complejo estimar un algoritmo o función que tenga en cuenta, no solo los aspectos individuales de los componentes de información del comportamiento, sino también su interrelación en conjunto. De hecho, el modelo de aprendizaje *CBR* suple en gran medida esta carencia al incorporar automáticamente todos estos aspectos a través de los casos almacenados.

Por otra parte, las consideraciones sobre que aspectos mejoran o empeoran el desempeño del comportamiento son totalmente subjetivas por parte del diseñador del comportamiento, aunque es posible llegar a un consenso en ciertos aspectos. Este aspecto dificulta la generalización de las decisiones, no solo a otros dominios del problema, sino a diferentes diseñadores dentro del mismo campo de aplicación.

En el diseño de nuestra arquitectura, y con respecto a las funciones de utilidad de los módulos, hemos considerado incluir únicamente aspectos individuales de los elementos que influyen o describen las metas y objetivos intermedios de los comportamientos. Así, tras la aplicación de la respuesta para el caso *CBR* adaptado, se procederá a evaluar individualmente la evolución de una serie de factores asociados al rendimiento del módulo y, posteriormente, se integrarán todos ellos en una función de agrupación que nos permita decidir si el nuevo conocimiento aplicado y experimentado debe incluirse dentro de la base de conocimientos del comportamiento. En caso afirmativo, se recorrerá la estructura jerárquica de la base *CBR* según la información de contexto y finalmente se depositará en la sub-base *CBR* en la hoja adecuada del árbol, para poder ser utilizada con posterioridad, en caso de volver a encontrar una situación parecida.

Para comprobar la "bondad" de la función de utilidad propuesta, se puede evaluar los resultados obtenidos durante la fase de aprendizaje por observación en la que, en

teoría, las acciones realizadas por el operador humano deben ir encaminadas —salvo errores puntuales— a acercarse a la meta final del comportamiento. En el capítulo de experimentos veremos como esto no es siempre así, ya que a menudo resulta complicado captar las sutilezas del razonamiento humano al resolver un problema, para expresarlo de forma algorítmica.

La función de utilidad genérica que utilizaremos para nuestros comportamientos es similar a la utilizada en [11], y tendrá varios componentes, algunos de los cuales serán comunes a todos o una mayoría de los comportamientos a definir, al menos en lo referente al ámbito de aplicación RoboCup que estamos considerando como ejemplo:

- Suavidad. El comportamiento del robot resultará, por lo general, tanto más eficiente cuanto menores sean los cambios de dirección propuestos a lo largo de su ejecución. Esta reducción de cambios de dirección suele conducir a trayectorias más suaves que permiten una obtención más rápida de los objetivos del comportamiento. Además, los cambios bruscos son especialmente dañinos en robots con patas ya que da lugar a efectos de deslizamiento (*slippage*) con los consecuentes errores en el sistema odométrico del robot. El factor de suavidad, F_{suav}, podría estimarse a través de la siguiente expresión:

$$F_{suav} = e^{-C_{suav} \cdot |\Delta_{dir}|} \tag{5.7}$$

  donde C_{suav} es un parámetro heurístico que permite ajustar la expresión, y Δ_{dir} es la variación del vector de avance del robot. La expresión anterior toma su máximo valor en 1 cuando la dirección de avance no varía respecto a la anterior; su valor mínimo está acotado por 0, y vendrá dado por la máxima variación posible del vector de avance —normalizado— y por el parámetro C_{suav}.

- Seguridad. En la mayoría de los comportamientos el robot debe evitar colisionar con los objetos de su entorno. Esta componente determina la penalización por dirigirse hacia los objetos considerados como obstáculos, que será tanto mayor en cuanto el ángulo de avance sea más directo con respecto al obstáculo, pero también cuando la distancia al obstáculo disminuya. Por tanto, una posible expresión para el factor de seguridad, F_{seg}, sería:

$$F_{seg_a} = 1 - e^{-C_{seg} \cdot (1 - tam_{obst}) \cdot |\ \theta_{rob-obst}|} \tag{5.8}$$

  donde C_{seg} es un parámetro heurístico que permite ajustar la expresión, tam_{obst} es el tamaño del obstáculo más cercano, y $\theta_{rob-obst}$ el ángulo de dicho obstáculo con respecto al robot. Nótese que el factor de seguridad tendrá una aportación nula cuando el obstáculo ocupa todo el area de la imagen; o cuando el robot se dirige directamente hacia él, aunque se encuentre lejano. Este segundo criterio se podría relajar usando una variante de la expresión que solo tuviese en cuenta el ángulo con respecto el obstáculo a partir de un determinado tamaño de área. La seguridad aumentaría al alejarse de los obstáculos y guardar un ángulo mayor. Otra alternativa que probaremos es la dada por:

$$F_{seg_b} = 1 - e^{-C_{seg} \cdot \frac{|\theta_{rob-obst}|}{tam_{obst}}} \tag{5.9}$$

  En la que, a medida que aumente el tamaño del obstáculo se le tiende a dar mas importancia al ángulo. Para un obstáculo muy alejado, con área cercana a

0, la seguridad tiende a 1, independientemente del ángulo. Situaciones del tipo $\theta_{rob-obst} = tam_{obst} = 1$ no pueden llegar a darse nunca. Nótese que no todos los objetos van a ser considerados obligatoriamente como obstáculos. Así, la pelota no contaría como tal en un comportamiento de acercamiento a la misma; lo mismo ocurriría con un robot enemigo al que se desea interceptar. Sea cual sea la expresión utilizada finalmente, en ausencia de obstáculos se le dará a la situación un factor de seguridad de valor 1

- Objetivo del comportamiento. El tercer factor, F_{obj} dependería de la evolución del comportamiento en el cumplimiento de sus objetivos marcados, y por tanto sería distinta según el comportamiento en cuestión. En el capítulo de Pruebas particularizaremos este factor para los comportamientos que se implementen como ejemplos.

La función de utilidad/eficiencia de cada caso, η, vendrá dada por una combinación de los factores anteriores de la forma:

$$\eta = \frac{K_{suav} \cdot F_{suav} + K_{seg} \cdot F_{seg} + K_{obj} \cdot F_{obj}}{K_{suav} + K_{seg} + K_{obj}} \tag{5.10}$$

donde K_{suav}, K_{seg} y K_{obj} representarían unos coeficientes de ponderación que definen la importancia relativa que se pretende dar a cada uno de los factores respecto a los demás. Si estos coeficientes toman valores en $[0, 1]$ siendo su sumatorio igual a 1, podríamos prescindir del denominador de la expresión. La función de utilidad toma un valor en [0,1], y nos permite medir y adaptar la eficiencia del comportamiento siguiendo diferentes criterios, según la importancia otorgada a cada uno de los factores mencionados. Una vez calculada la utilidad del nuevo caso generado, se debe establecer un criterio que determine su rechazo o inclusión en la base de casos que define al comportamiento, de manera que pase a formar parte del conocimiento asociado al mismo. Entre los criterios podríamos considerar establecer un umbral mínimo de utilidad que sea necesario superar para considerar el nuevo caso como válido; sin embargo no resulta sencillo establecer el valor de este umbral, ya que dependería mucho del conocimiento actual de la base *CBR* y, además, se impediría la adquisición de casos de baja utilidad pero que son la única solución existente para determinadas situaciones. En [11] se propone un criterio un poco más complejo en el que todos los casos son aceptados a priori, almacenando también su eficiencia asociada; posteriormente, en la fase de mantenimiento de la base *CBR*, se utiliza un mecanismo de optimización que favorece los casos más eficientes sin eliminar aquellos menos eficientes correspondientes a situaciones menos frecuentes o conocidas. En nuestra trabajo, y considerando la naturaleza eminentemente reactiva de la arquitectura empleada, proponemos una alternativa en la que se compara la utilidad obtenida al aplicar el nuevo caso, con la utilidad en la situación inmediatamente anterior: si hay una mejora en la utilidad, se considera el caso como aceptable y pasa a incorporarse en la base *CBR*. En trabajos futuros se explorará un poco mas este esquema mediante, por ejemplo, la introducción de "umbrales de mejora" y la inclusión de ciertos factores de aleatoriedad/ruido en la generación del nuevo caso.

Por otra parte el mantenimiento de la base *CBR* también podría incluir aspectos de uso de casos mas frecuentes, que verían reforzada su importancia dentro de la jerarquía de conocimientos —de la misma forma que los seres humanos reforzamos nuestras

habilidades con la práctica—, de manera que dichos aspectos también influyesen en la etapa de predicción/recuperación de casos; así como posibilidad de olvido o eliminación de casos utilizados con poca frecuencia o que demuestren tener un mal comportamiento, de manera similar a como nos ocurre a los seres humanos. Estos aspectos no han sido considerados en la presente tesis, aunque quedan pendientes para su estudio en en el futuro.

5. Composición de comportamientos reactivos emergentes

En el apartado 4.5 del capítulo 2, se presentó un estado del arte con algunas de las técnicas más empleadas para la organización de comportamientos y su composición con vistas a la obtención de comportamientos emergentes de nivel superior. En la arquitectura presentada en esta tesis, se propone emplear tanto la selección exclusiva de comportamientos como la combinación de los mismos, dependiendo de las circunstancias y de las características de los comportamientos implicados. Para ello se añadirá a nuestra arquitectura un modulo de nivel superior que se encargará de decidir el peso en la respuesta de cada comportamiento "básico" en la respuesta final del robot. Si este peso es nulo para todos los comportamientos menos uno, se estará procediendo a una selección; en caso contrario, se realizará una combinación de todos aquellos comportamientos con peso superior a cero. En cuanto a la forma en que dicho módulo se ocuparía de asignar los pesos correspondientes a cada uno de los comportamientos a combinar, existen diversas alternativas. En este trabajo se presentará una solución basada en reglas de decisión que, según determinados criterios relacionados con las características del escenario y el estado del robot, asignará dichos pesos. Esta solución exige el conocimiento de un experto que decide cual es la asignación de pesos más adecuada. En trabajos futuros se plantea que el módulo de composición de comportamientos reactivos utilice el mismo esquema de aprendizaje basado en *CBR* que los módulos básicos componentes que implementan los comportamientos reactivos. Para ello se activarían simultáneamente los módulos *CBR* correspondientes a los comportamientos individuales durante la ejecución de las secuencias de entrenamiento, pudiéndose así obtener los casos devueltos por cada uno de dichos módulos para las situaciones experimentadas. Como dichos casos tienen ya una respuesta individual asociada a una acción o movimiento, se trataría de construir la respuesta generada por el operador ante una determinada situación a través de la ponderación de las respuestas de los comportamientos individuales. De esta forma, el nuevo módulo de composición contendría una base *CBR* en la que los casos tendrían características similares a las de los comportamientos básicos, pero en los que la salida asociada a los casos sería un vector de los pesos a aplicar ante cada situación. Habría que resolver algunas incógnitas, como la posible alta dimensionalidad de los atributos de entrada para los casos, teniendo en cuenta que se trata de representar una información más general que la información local aplicada en los comportamientos básicos. De cualquier forma, estos aspectos serán tratados en futuras investigaciones. En el capítulo 6 se presentará un ejemplo concreto de composición de comportamientos a través de un módulo que emplea reglas de decisión.

6

Pruebas experimentales

Las herejías son experimentos en la búsqueda insatisfecha de la verdad
H.G. Wells (1866-1946) Escritor

1. Introducción

En este capítulo se presentarán los resultados de las pruebas y experimentos realizadas para verificar las hipótesis y teoría que constituyen el cuerpo de esta tesis, en especial lo indicado en el capítulo 5. Las pruebas se presentarán por orden cronológico, indicando los diferentes aspectos que se pretende probar en las mismas, así como los inconvenientes encontrados, que nos han llevado a incorporar nuevos elementos que han acabado formando parte de la arquitectura propuesta. El punto de partida para la realización de nuestra investigación son los trabajos realizados en [11] y [12], en los que los autores incorporaron un sistema *CBR* para la solución de diversos aspectos relacionados con el control de la navegación de un robot Pioneer a través del aprendizaje. En este caso, la única fuente de información que debía manejar el robot era una estimación de la distancia entre el robot y diferentes objetos, medida a través de un anillo sonar. Nuestro objetivo es generalizar el uso de *CBR* como paradigma de aprendizaje de comportamientos robóticos, comprobando su aplicación en robots más complejos como el *AIBO* ERS-7, un robot que, además de ser cuadrúpedo, maneja una mayor cantidad y diversidad de fuentes de información acerca del entorno. Esto hace posible el manejo de conceptos más complejos como base de conocimiento del robot, así como el diseño de conductas más avanzadas para el mismo, y estudio de situaciones de "caso peor", derivadas de los inconvenientes asociados a este modelo de robot, tales como imprecisiones en el desplazamiento, poco calidad de las imágenes de cámara, e inestabilidad de la conexión inalámbrica que permite el intercambio de información entre el interfaz y el robot. Todas estas modificaciones implican la adaptación de ciertos aspectos de la arquitectura *CBR* propuesta en los trabajos mencionados, para hacer posible un rendimiento correcto.

Dentro de la propuesta general de una arquitectura de inteligencia/control robótico basada en *CBR* nos centraremos únicamente en esta tesis en los niveles más bajos, correspondientes al diseño de comportamientos reactivos sencillos que permitan al robot

actuar en respuesta a los estímulos recibidos más actuales. Esta capa de la arquitectura constituye la base para la implementación de comportamientos más complejos que manejen conceptos abstractos de alto nivel, correspondientes a las capas deliberativas, por lo que su estudio es crucial para intentar diseñar con éxito una arquitectura completa basada en aprendizaje *CBR*. Como apartado final, se propondrá un método básico de combinación de estos comportamientos reactivos de manera que sea posible obtener, mediante emergencia, comportamientos más complejos, aún dentro de los niveles bajos de la arquitectura.

Para su funcionamiento, el robot *AIBO* integra un entorno de operación Tekkotsu, implementado sobre el sistema operativo de Sony, AperiOS, que proporciona una serie de funcionalidades básicas para la operación del robot, como un control cinemático de movimiento, un sistema de visión con localización rápida de objetos mediante segmentación, ejecución de secuencias pre-grabadas, etc. De hecho se pretende que, en un futuro, parte del trabajo de desarrollo de esta tesis se pueda incorporar a Tekkotsu, como interfaz de diseño de comportamientos por aprendizaje basado en CBR.

El control y recepción de información remota del robot se puede realizar, en principio, desde un PC vía interfaz inalámbrico, si bien gran parte de su operación será finalmente autónoma. Para la comunicación robot-PC se implementó un interfaz gráfico usuario en JAVA, partiendo del existente en el entorno Tekkotsu. Sobre este mismo entorno, inicialmente se integró la librería JAVA *AITools* proporcionada por el grupo de investigación *Gestión del Conocimiento y Aprendizaje Automático*(*Knowledge Engineering and Machine Learning (KEMLG)*) de la Universidad Politécnica de Catalunya. Esta librería nos permite añadir una base *CBR* plana al sistema, proporcionándonos las operaciones básicas de creación y gestión de la misma. En pruebas posteriores se ha diseñado y utilizado una librería C++ propia, *libCbr*, por la necesidad de una mayor capacidad de personalización en las características de los sistemas *CBR* empleados como base de nuestra arquitectura. Se ha integrado un control por joystick al módulo software para poder controlar al robot de forma remota, algo necesario para la adquisición de patrones *CBR* en la etapa de aprendizaje tutelado. Del mismo modo, se va a emplear la librería *CBR* para poder controlar el movimiento del robot a través de consultas de la información almacenada en el mismo. En el ciclo de aprendizaje *CBR*, se controla el robot con el joystick para aprender diferentes conductas relativas a la pelota, como “dirigirse a ella para chutar”, “evitarla”, “rodearla, etc”...almacenando en un fichero información significativa de las diferentes situaciones experimentadas a lo largo de la prueba, así como la respuesta indicado por el operador humano. En el comportamiento diseñado, la posición de la cabeza y cámara del robot se ajusta automáticamente para intentar mantener los objetos de interés dentro del campo de visión. Tras finalizar el entrenamiento, el fichero de información se procesa para adaptarlo al formato del módulo CBR —*AITools* en las pruebas iniciales y *libCbr* en las posteriores— , y generar la base de casos *CBR* correspondiente.

2. Primera etapa: Implementación de comportamientos simples

2.1. Escenario de pruebas

Para la realización de las pruebas iniciales se ha planteado un escenario de aplicación muy sencillo, en el que un robot *AIBO* debe ser capaz de localizar una pelota, y ejecutar diversos comportamientos entrenados relacionados con la situación de la misma; todo ello mientras el robot se mantiene dentro de los márgenes del campo, delimitados por unas líneas.

2.2. Módulo de comportamiento determinado por la pelota

Para el diseño de un primer comportamiento *CBR* básico aprendido se ha implementado, sobre la arquitectura software de Tekkotsu, un comportamiento genérico de recepción de información visual segmentada de la pelota, procedente de la cámara de a bordo, así como de información del estado de movimiento del robot [196]. En este primer conjunto de pruebas, la representación de la información —en este caso para un único objeto "pelota"— es ligeramente distinta a la que se utiliza en pruebas posteriores, indicada en el apartado 4.1 del capítulo 5, ya que el factor de *clipping* se considera aquí de forma binaria y únicamente para representar la existencia de "recorte" del objeto por alguno de los límites de la imagen, pero sin especificar su magnitud. Podemos ver la estructura de la instancia de entrada al comportamiento en la figura 6.1:

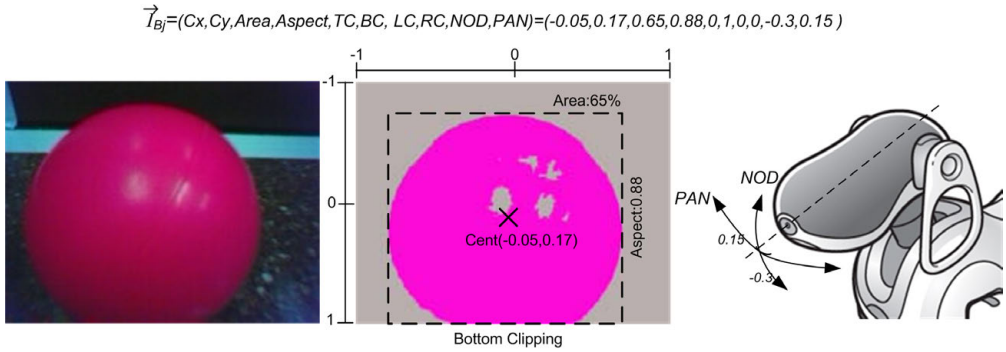


Figura 6.1: Instancia de entrada al módulo CBR de aproximación a la pelota

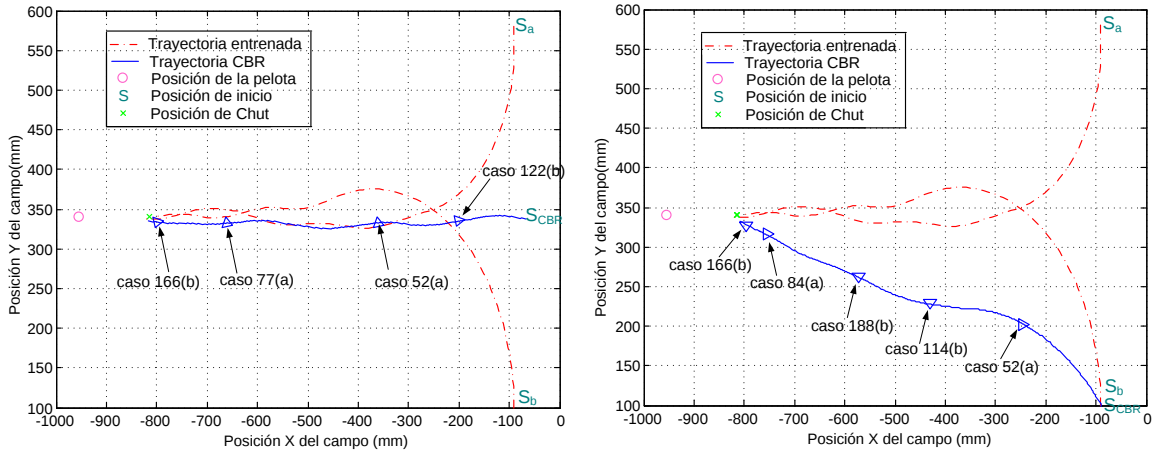
La respuesta asociada a cada caso es un vector unitario de movimiento-acción como el indicado en la figura 5.17.

2.2.1. Acercamiento a la pelota

En primer lugar se entrenó un comportamiento en el que el robot trataba de acercarse a la pelota de la forma más rápida posible y chutar tras alcanzarla, sin tener en cuenta la orientación del robot en el campo. Tras varias ejecuciones de entrenamiento realizadas desde diversas posiciones, el conocimiento del comportamiento se consiguió con un pequeño número de casos, del orden de 200, tamaño suficientemente pequeño como para permitir el uso de una base de casos con estructura plana. Para las situaciones en las que la pelota se pierde de vista, se ha generado un caso artificial

que establece un movimiento de giro continuo del robot hasta localizar de nuevo la pelota. Para la medida de similitud entre el problema presentado y los casos de la base *CBR* se emplea una distancia Manhattan [249].

En la figura 6.2 se observa un ejemplo de comparativa de la ejecución del robot en un comportamiento aprendido de acercamiento a la pelota y chut de la misma, tanto en el caso de partir de una situación ya experimentada, como de otra desconocida, ante la que se actuará infiriendo una respuesta a partir del conocimiento almacenado.



(a) Desde una posición frontal **(b) Desde una posición 45° a la izquierda**

Figura 6.2: Navegación reactiva mediante CBR en un módulo de aproximación a la pelota

Como se puede observar, la trayectoria ejecutada en modo autónomo a partir de la información almacenada en la base *CBR* durante el aprendizaje, no coincide exactamente con la ejecutada, pero logra realizar el objetivo buscado para el comportamiento. Esto se debe a varios factores: por una parte, la adquisición de la información no es continua, temporalmente hablando, y además se realiza una discretización de los datos para reducir dimensionalidad. Por otro, las condiciones del entrenamiento y las pruebas no son exactamente iguales por lo que las situaciones experimentadas por el robot difieren ligeramente en uno y otro caso. De cualquier manera, se extrae siempre el caso *CBR* más parecido a la situación en el momento de la consulta, con lo que se consigue un comportamiento global correcto. Las pequeñas discrepancias que puedan suceder se ven automáticamente corregidas por la naturaleza reactiva del comportamiento.

Por otra parte, el funcionamiento autónomo pone de manifiesto las ventajas del diseño de comportamientos aprendidos por *CBR*: las trayectorias de entrenamiento no son "perfectas" debido a la ausencia de precisión absoluta del entrenador humano; además, el robot experimenta, durante el entrenamiento, situaciones muy parecidas en las que la respuesta es ligeramente distinta. Sin embargo, el sistema *CBR* es capaz de absorber todas estas imprecisiones e inexactitudes y ofrecer una trayectoria autónoma mas suave y rápida incluso que las entrenadas. Hay que destacar que esas imprecisiones en el entrenamiento son algo deseable, ya que añaden cierta variedad al conocimiento adquirido y facilitan una generalización del mismo.

Se han realizado diversas pruebas adicionales que permitieron comprobar que la ejecución de comportamientos en los que el robot se movía por el área del campo "visitada" durante la etapa de entrenamiento por observación no presentaban ningún

problema en cuanto a la consecución de los objetivos. Así mismo, si la posición del robot no está demasiado alejada de dicho área de entrenamiento también es posible obtener buenos resultados. Solamente situaciones excesivamente alejadas de las aprendidas por el robot pueden llevar a una operatividad con resultados impredecibles respecto a los objetivos buscados.

En las figuras 6.3 y 6.4, se observa una secuencia de imágenes con el movimiento del robot en un comportamiento aprendido de acercamiento a la pelota, en una situación en que esta se va desplazando a medida que el robot se mueve. Aunque el aprendizaje de este comportamiento se realizó con una pelota fija, esto resulta irrelevante para el robot, ya que en sus respuestas solo se tiene en cuanta la información actual en el momento de la consulta a la base *CBR*.

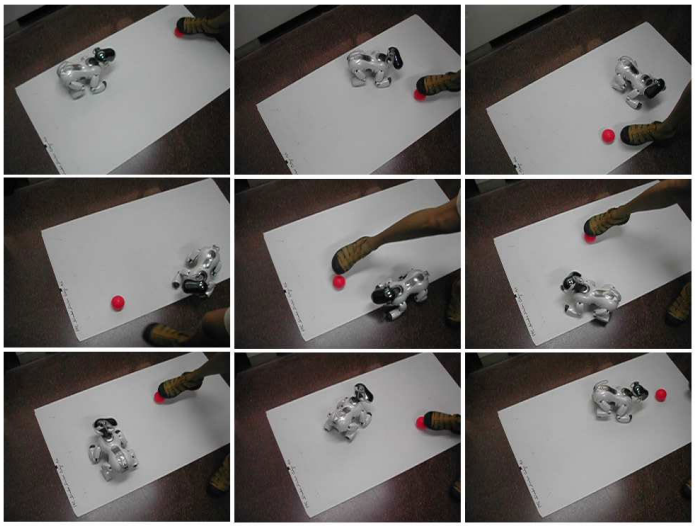


Figura 6.3: Secuencia de vídeo de aproximación a la pelota, con una pelota móvil

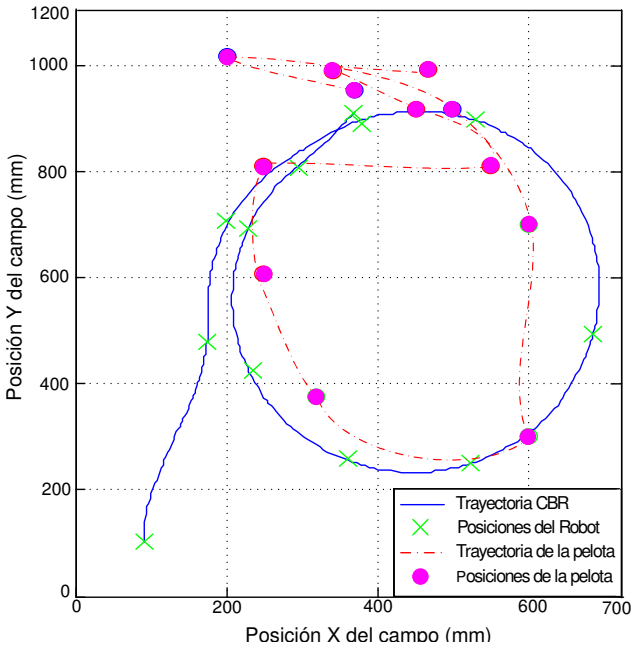


Figura 6.4: Comportamiento CBR de aproximación a la pelota, con una pelota móvil

Cabe destacar también que las oclusiones parciales del objeto director del comportamiento no afectan al buen rendimiento del comportamiento aprendido, siempre y cuando la oclusión no sea excesiva. De lo contrario, el objeto puede llegar a ser considerado en una situación radicalmente diferente para la cual se haya aprendido una respuesta distinta.

2.2.2. Otros comportamientos: Evitación de la pelota y Circunvalación de la pelota

En la figura 6.5 se muestra una secuencia de imágenes con otro ejemplo de comportamiento aprendido, en este caso con el objeto de evitar una pelota que se acerca al robot. Así mismo, en la figura 6.6 podemos ver la trayectoria realizada por el robot al ejecutar un comportamiento aprendido de rodear la pelota. La modificación de la conducta asociada al comportamiento se obtiene de forma simple, entrenando el nuevo comportamiento deseado, y generando la nueva base de casos asociada al mismo, que será la que guíe el nuevo comportamiento. La conmutación de un comportamiento a otro se consigue cargando la base *CBR* correspondiente al aprendizaje del mismo.

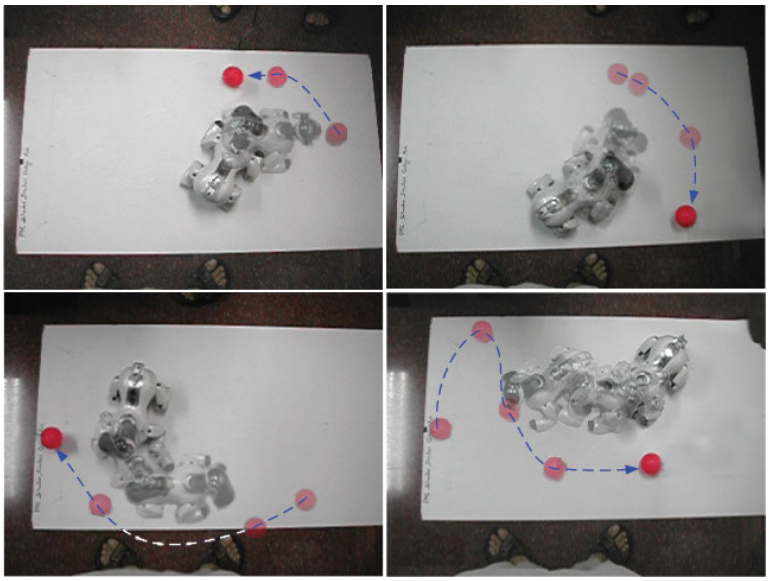


Figura 6.5: Comportamiento CBR de evitación de la pelota, con una pelota móvil

El objeto del comportamiento "rodear la pelota" es el de mantenerse aproximadamente a una misma distancia de ella, mientras se camina hacia delante utilizando, para ello, información de su posición en el campo de visión, pero también de su distancia, expresada a través de su área. En la figura 6.6 se observa como la trayectoria ejecutada por el robot mediante aprendizaje *CBR* mejora la del operador humano ya que, de todo el repertorio de casos aprendidos, el robot escoge automáticamente los mejores casos almacenados.

2.3. Módulo de posicionamiento en el campo

En el apartado anterior vimos un ejemplo de diseño de un comportamiento simple, motivado por un único estímulo aislado, como es la posición de la pelota en el campo de juego. La siguiente etapa de nuestra investigación pasa por el diseño y prueba de

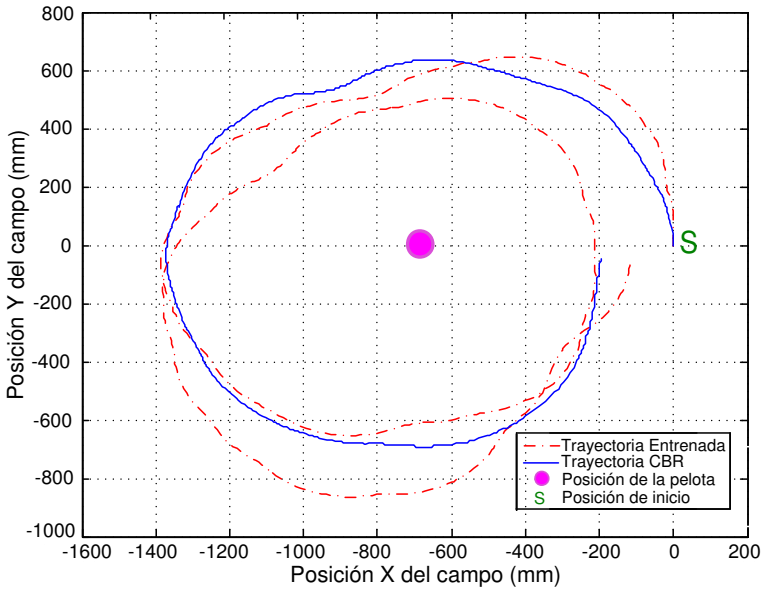


Figura 6.6: Aprendizaje CBR de comportamiento para rodear la pelota

nuevos comportamientos adicionales basados en un mayor número de estímulos y que respondan a las conductas dictadas por un entrenamiento previo por imitación. El primero de ellos es un comportamiento de búsqueda de la portería en el campo de juego: se trataría de conducir la pelota hacia el área de gol desde cualquier posición del campo, utilizando las líneas que delimitan el mismo para conocer su posición [63]. El comportamiento desarrollado sigue siendo reactivo, ya que el robot carece de un plan "global" que determine su trayectoria a lo largo de toda la prueba. En su lugar va determinando, de forma reactiva, adonde debe dirigirse en cada momento para acercarse a su objetivo (figura 6.7).

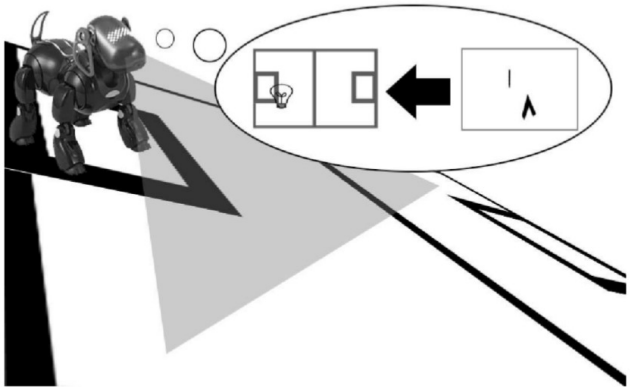


Figura 6.7: Comportamiento de localización basada en líneas

Las líneas del campo se han usado a menudo como método de posicionamiento en el entorno de aplicación Robocup, en combinación con balizas y porterías coloreadas según un determinado esquema [274] [275]. Algunos de los métodos empleados se apoyan en criterios de variación de color [276] [277] [278], para la detección de líneas mediante escaneos verticales equiespaciados de las imágenes. Esta información se combina posteriormente con las porterías y balizas detectadas mediante métodos de

localización Montecarlo, para obtener una estimación de la posición del robot. A partir de la posición y otros parámetros de estado, es posible establecer un curso de acción para el robot.

En el comportamiento basado en aprendizaje *CBR* propuesto no es necesario proporcionar un modelo explícito del entorno, y el resultado del mismo no es una posición del robot, sino un vector de movimiento hacia la portería. Las líneas del campo y del área se localizan mediante un detector de bordes *Canny* seguido de una detección de líneas por la transformada de *Hough*; para ello se ha implementado una clase openCV-C++ como clase nativa JNI. Cada línea detectada se representa en coordenadas polares mediante un par $[\rho, \theta]$, donde ρ representa la distancia entre el origen de coordenadas y el punto (x, y) de la recta más cercano al origen de coordenadas; y θ el ángulo del vector director de la recta perpendicular a la recta original y que pasa por el origen de coordenadas. Cada caso *CBR* relaciona las líneas del campo con los vectores de movimiento necesarios para alcanzar la portería desde cualquier posición del campo, de la misma forma que en los comportamientos desarrollados en el apartado 2.2 (figura 5.17). Como algunas líneas presentan errores de detección y solo es posible detectar algunas de ellas en el campo de visión se decidió considerar únicamente las 4 líneas más significativas, ordenadas de izquierda a derecha. En caso de detectarse menos líneas se utiliza un parámetro especial para representar la ausencia de las líneas que falten por completar un número de cuatro. A cada caso se le añadió, además, el alineamiento del robot y la posición de la cabeza con respecto al mismo, ya que éstos aspectos tienen una gran influencia en la representación de la imagen y, por tanto en la de las líneas detectadas (figura 6.8).

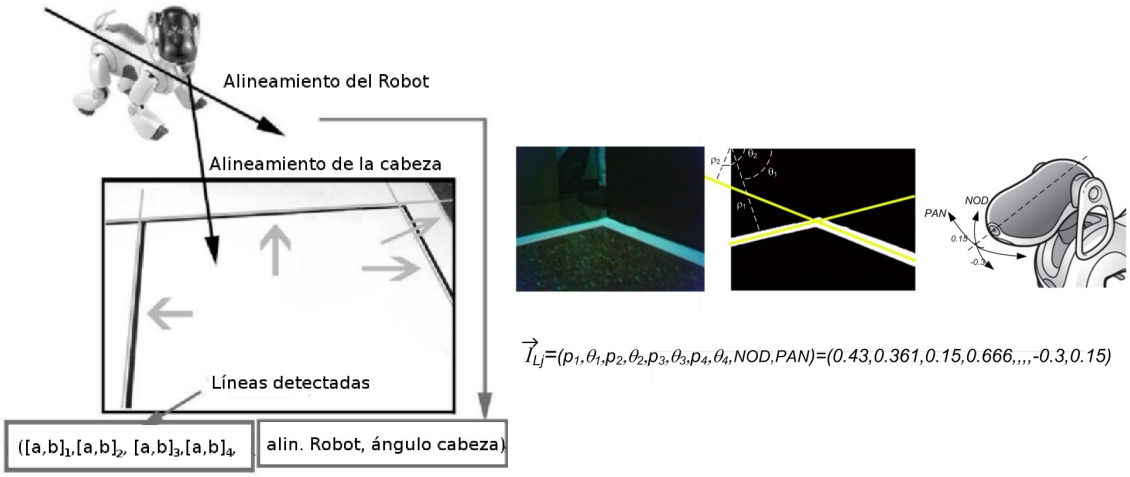


Figura 6.8: Instancia de un caso para el comportamiento de búsqueda de portería

En este comportamiento concreto es evidente la dificultad para realizar un entrenamiento exhaustivo de aprendizaje por imitación, debido a la gran variabilidad en la información de entrada al módulo. Por eso, y como elemento novedoso con respecto al diseño del comportamiento previo relacionado con la posición de la pelota, se añadió una etapa adicional de adaptación de los casos adquiridos, para situaciones en las que no se encuentre un caso adecuado dentro de la base *CBR*. En particular, la orientación relativa del robot respecto a las líneas detectadas resulta más relevante para el comportamiento que su distancia a la portería. Por ello, la adaptación consiste

en modificar el alineamiento del robot hasta que coincida con la del caso recuperado. La orientación está reflejada en el eje θ de la transformada de Hough.

En la figura 6.9 se puede ver un ejemplo de como afecta un giro del robot a la posición de una línea representada en el dominio de Hough. La figura 6.9.a representa el caso en el que el robot se dirige hacia una línea situada 90° con respecto a su alineamiento, obteniéndose la representación de la figura 6.9.b, en la que el ángulo es menor. En la figura 6.9.c se observa como se representan los ángulos de las figuras anteriores en el dominio de Hough. Si el robot está orientado en la situación reflejado en la figura 6.9.b, y el caso almacenado más similar se corresponde con el de la figura 6.9.a, en el que el se indica un movimiento hacia delante, *AIBO* debería girar a la derecha para conseguir la adaptación al caso almacenado. La magnitud del giro depende de la diferencia de θ entre las representaciones de ambas líneas, como se ve en la figura 6.9.c.

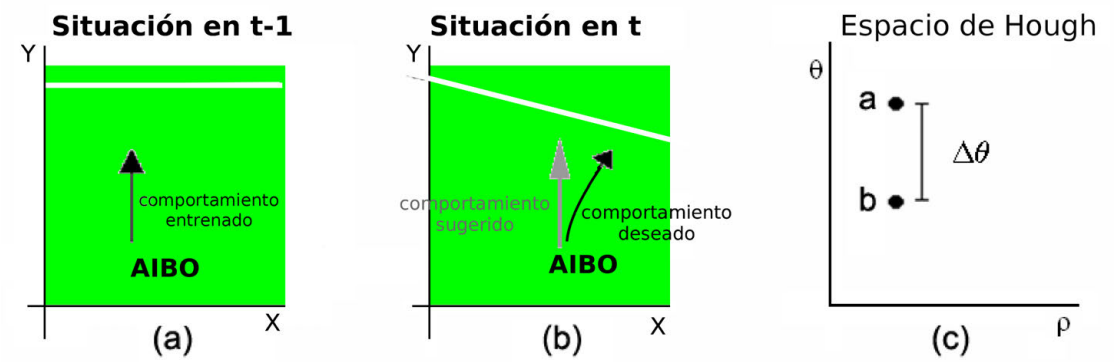


Figura 6.9: Variación de una línea respecto a la posición en el espacio de Hough

Para la situación en la que se detectan varias líneas en una misma imagen, la adaptación de los casos se realiza mediante la media de las variaciones de cada una de las líneas en θ:

$$\triangle \theta_{Prom} = \sum_{i} (|\theta_{IN}(i) - \theta_{CBR}(i)|) \tag{6.1}$$

siendo $\theta_{IN}(i)$ el valor de θ de la linea i en la situación actual; y $\theta_{CBR}(i)$ el valor de θ de la linea i en el caso recuperado.

Si θ_{Prom} supera un umbral de semejanza, U_{adap_θ}, el caso recuperado se adapta añadiendo un factor F_{adapt} a la rotación propuesta por el caso *CBR* a la posición actual del robot de la forma:

$$F_{adapt} = \frac{k_{adapt} * \theta_{Prom}}{U_{adapt_\theta}} \tag{6.2}$$

siendo k_{adapt} una constante que controla el nivel de cambio del caso. Un valor de k_{adapt} bajo da lugar a una adaptación lenta, pero con un movimiento más fluido y suave; un valor alto de k_{adapt} nos permite una adaptación más rápida, pero con movimientos más bruscos.

En la figura 6.10, se puede ver un ejemplo real de adaptación de un caso.

Se parte de una situación en la que el robot ha aprendido como moverse cuando las líneas media y final del campo, las únicas detectadas en la imagen, forman un ángulo de 90° con respecto al alineamiento del robot. No obstante, en la situación actual, el robot decide girar a la izquierda de forma que dichas líneas ya no forman los ángulos

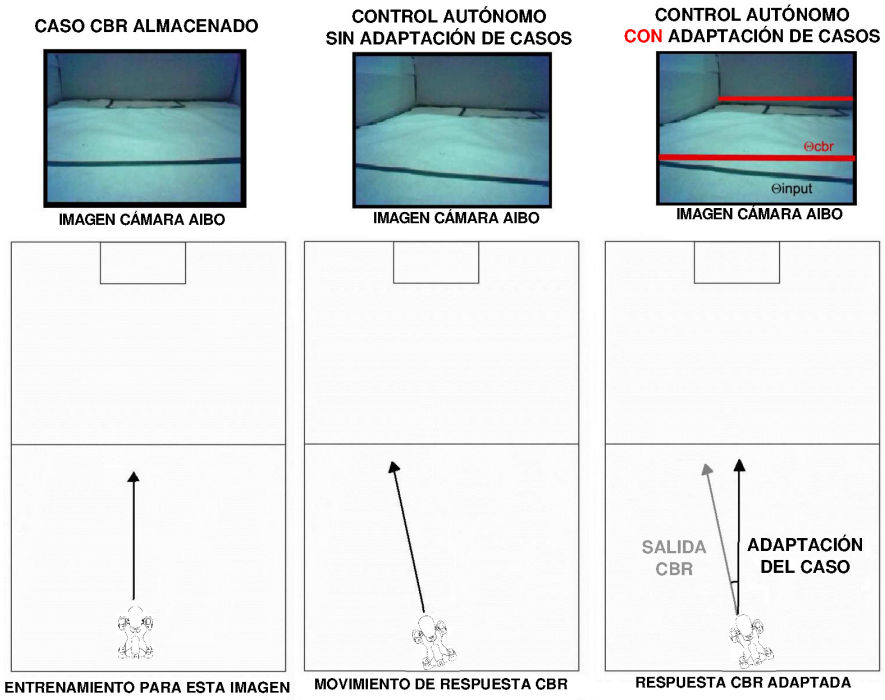


Figura 6.10: Ejemplo real de adaptación de caso a la línea

mencionados. En el caso más similar recuperado se sugiere un movimiento recto del robot, que lo llevaría hacia un extremo del campo, en vez de hacia la portería. En caso de usar adaptación, la diferencia entre los ángulos de las dos líneas detectadas en la imagen de entrada y los indicados en el caso recuperado nos permite estimar cuanto debe girar el robot. Se debe hacer notar que las líneas más lejanas en el campo de visión tienen menos peso en el cómputo de la adaptación, ya que las variaciones de ángulo suelen ser menores. Tras la aplicación de la adaptación se consigue que el robot se mueva hacia la portería, cumpliendo los objetivos del comportamiento.

Para la etapa inicial de aprendizaje por imitación, se ejecutaron las secuencias de entrenamiento mostradas en la figura 6.11, en las que *AIBO* comenzaba cada prueba encarado hacia la portería, pero desde diferentes posiciones. Estas pruebas permitieron al robot recolectar la suficiente información como para ser capaz de dirigirse hacia la portería tratando, al mismo tiempo, de mantenerse en la franja central del campo de juego. Además de casos con diferentes situaciones de líneas observadas durante el entrenamiento, se generó artificialmente un caso correspondiente a la situación en que el robot no detectase líneas en el campo, asociándole una respuesta de detención; esto se hizo así por motivos de seguridad, ya que dicha situación solía corresponderse con una aproximación muy cercana del robot a una pared. Tras un entrenamiento de 30 segundos se generó una base *CBR* de conocimiento con un total de 120 casos.

Con este aprendizaje el robot fue capaz de responder ante situaciones que no habían sido entrenadas específicamente, y cumplir el objetivo del comportamiento de alcanzar la portería (figura 6.12).

En otras situaciones, no obstante, el *AIBO* falló en el cumplimiento de los objetivos, debido a que las mismas eran demasiado distintas de las incorporadas a la base de conocimiento *CBR*. En la figura 6.13 podemos ver algunos ejemplos de esta ejecución errónea del comportamiento.

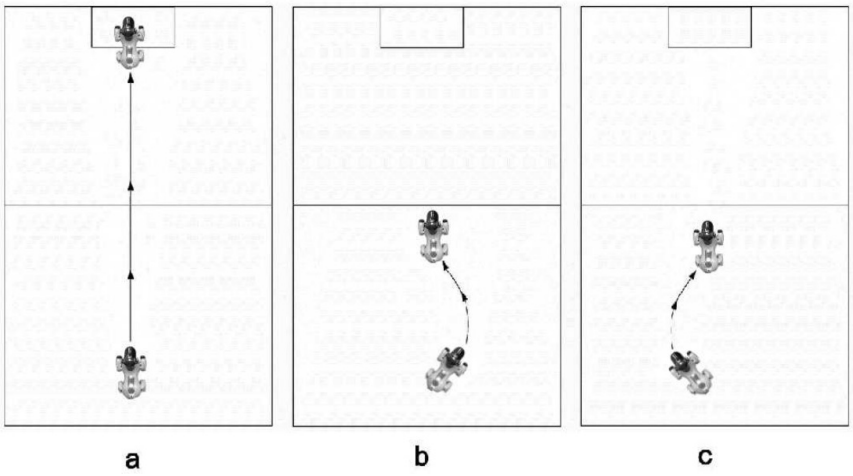


Figura 6.11: Entrenamiento para el comportamiento de búsqueda de portería

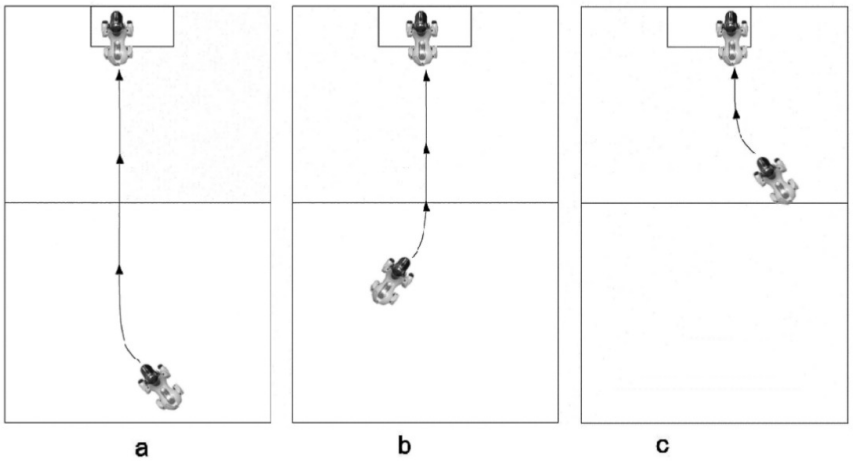


Figura 6.12: Navegación reactiva para búsqueda de portería basada en CBR

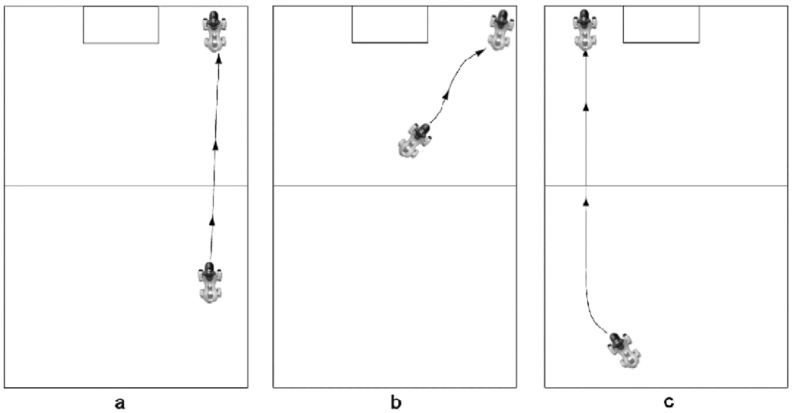


Figura 6.13: Ejecuciones incorrectas del comportamiento de búsqueda de portería para el caso de un entrenamiento CBR simple

Para estas situaciones se aplicó la etapa de adaptación comentada en el párrafo anterior lo cual permitió mejorar, en gran medida, el comportamiento del robot ante

situaciones no aprendidas (figura 6.14), si bien no siempre es posible encontrar una solución satisfactoria, como se puede apreciar en la figura 6.14.b.

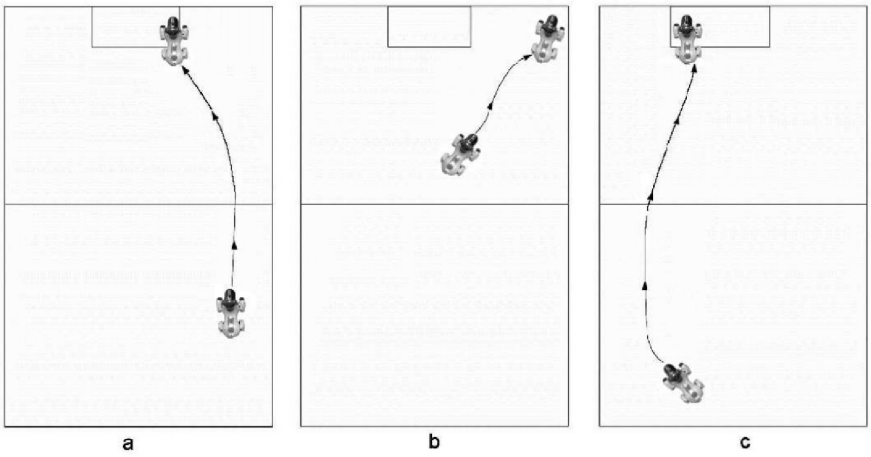


Figura 6.14: Comportamiento de búsqueda de portería para un entrenamiento CBR con adaptación

Por este motivo, al comportamiento de búsqueda de portería se le añadió un entrenamiento adicional para situaciones en las que el robot encuentra las esquinas del campo, a través de dos nuevas secuencias mostradas en la figura 6.15. Este conocimiento adicional permitió, no solo añadir, experiencia para afrontar una mayor variedad de situaciones sino que también corrigió alguno de los comportamientos erróneos previos (figura 6.16). Por lo general, cuanto más información variada exista en la base de casos, mayor versatilidad tendrá el robot para afrontar nuevos problema, si bien una cantidad excesiva de información puede traer otro tipo de problemas, como indicaremos más adelante.

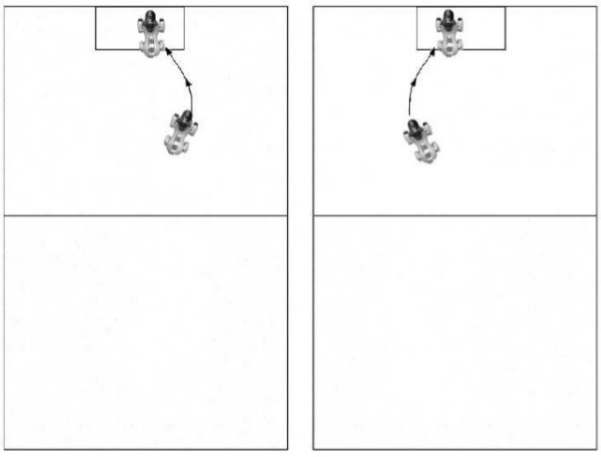


Figura 6.15: Secuencias de entrenamiento añadidas al aprendizaje previo

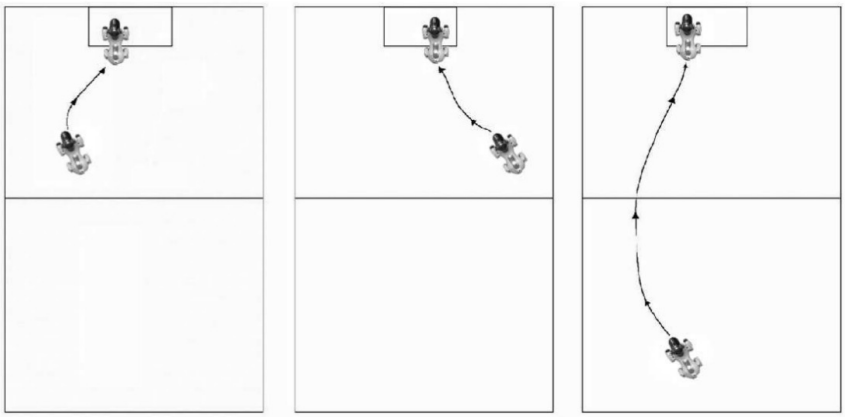


Figura 6.16: Navegación reactiva para búsqueda de portería basada en CBR y adaptación, con entrenamiento adicional

2.4. Otros ejemplos de comportamientos básicos basados en visión

Siguiendo la línea marcada por la investigación de esta tesis, se diseñaron nuevos comportamientos que permitieron observar diversas variantes en cuanto a las características de la información que constituye el conocimiento aprendido por el comportamiento y que demostraron la posible aplicación general del marco teórico propuesto.

Así, en [279] y siguiendo las mismas líneas maestras indicadas en los apartados previos, se desarrolló un comportamiento para la auto-localización basada en la identificación de rasgos en marcas distribuidas por el escenario en el que se desenvuelve el robot. A través de la librería *ARToolkit* se implementaron algoritmos de procesado de imagen que permitían calcular, en tiempo real, la posición y orientación relativa de la cámara de a bordo del robot con respecto a unas marcas artificiales con unas características definidas. Mediante aprendizaje por imitación se entrenó al robot para realizar determinados comportamientos relacionando vectores de movimiento y acción con la información del tamaño y orientación de la marca observada. En cada consulta al sistema *CBR* incorporado al comportamiento, se obtiene la información de posición y orientación en el eje de coordenadas de la marca, y se traslada al sistema de coordenadas de la cámara mediante un algoritmo de transformación matricial proporcionado por la librería, quedando finalmente descrita la marca en coordenadas polares mediante un ángulo y una distancia (figura 6.17). En el ciclo *CBR* se recupera el caso que describía a una marca más semejante entre los almacenados incluyendo, además de información descriptiva de la marca, el ángulo de la cabeza del robot respecto a su dirección de avance. La respuesta asociada a dicho caso se aplica directamente o realizando previamente una fase de adaptación muy similar a la descrita para el comportamiento diseñado en el apartado 2.3.

En la figura 6.18 podemos ver una secuencia de imágenes con la ejecución del comportamiento aprendido, en el que el robot se desplaza en el campo según la posición estimada a través de la visualización de la marca.

De manera similar, en [280] y [281], se utilizó idéntica metodología para el diseño de comportamientos CBR básicos que pudiesen ser ejecutados por robots individuales para

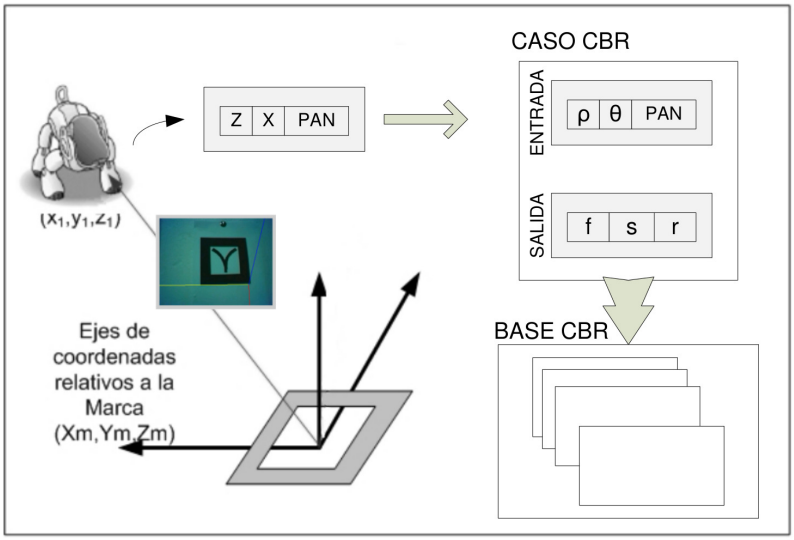


Figura 6.17: Comportamiento de auto-localización mediante marcas

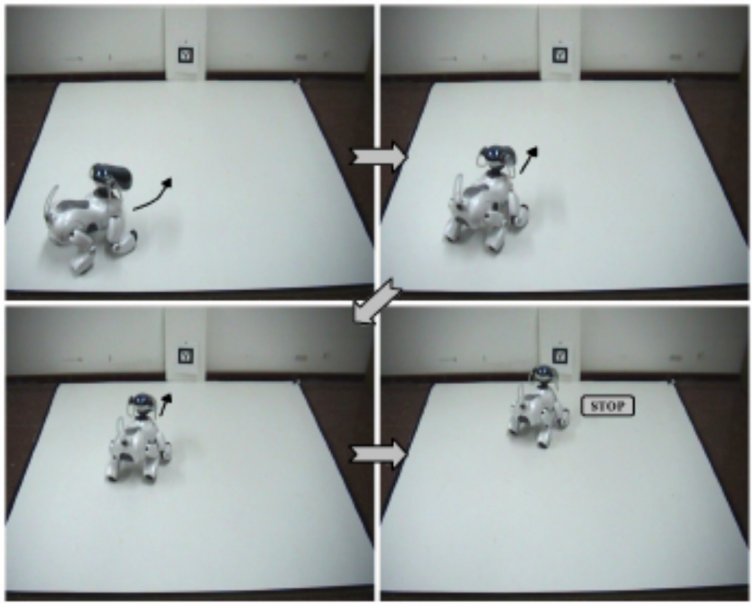


Figura 6.18: Navegación mediante reconocimiento de marcas ARToolkit

realizar acciones de equipo coordinadas. Este mismo planteamiento ha sido también empleado por otros autores en el mismo ámbito de aplicación [282].

2.5. Combinación de módulos simples: emergencia de comportamientos

2.5.1. Conmutación de comportamientos básicos

Incluso trabajando a bajo nivel, el aprendizaje de pautas aisladas que definen un comportamiento dirigido hacia un objetivo individual y/o determinado por un único elemento no suele ser suficiente para demostrar comportamientos "verdaderamente" inteligentes. Los comportamientos individuales de "alcanzar la pelota", "dirigirse a

la portería", o "localizarse en el campo" no permiten, por sí solos, que el robot sea capaz de ejecutar un comportamiento de tipo "jugar al fútbol", ni siquiera a un nivel reactivo que tenga en cuenta únicamente las percepciones actuales que está captando el robot. Es necesario **combinar** de alguna manera los comportamientos de bajo nivel para conseguir obtener un **comportamiento emergente** de nivel superior, pero aún reactivo, que implemente esa respuesta más compleja.

En [63] se propuso un esquema de obtención de un comportamiento emergente basado en los comportamientos básicos de "alcanzar la pelota" y "dirigirse a la portería". Este comportamiento debe permitir el acercamiento del robot a la portería con la pelota controlada. La combinación de los comportamientos se puede hacer a través de la combinación de los conceptos que representan los casos recuperados de las bases *CBR* de los respectivos comportamientos o, de forma más simple, mediante la combinación de las respuestas asociadas a dichos conceptos y que han sido aprendidas como las respuestas adecuadas ante la aparición de los mismos. Por su mayor simplicidad elegimos esta segunda aproximación como nuestro punto de partida para el estudio de la emergencia de comportamientos. En cualquier caso, es necesario añadir en nuestra arquitectura un nuevo componente o capa que se ocupe de esta combinación de conceptos y elaboración del correspondiente comportamiento emergente. Idealmente este módulo también tendría un componente de entrenamiento que le llevaría a aprender como la combinación de conceptos o respuestas obtenidas desde los niveles mas bajos se puede combinar de una determinada manera para obtener el comportamiento de más alto nivel deseado.

Como ya se vio en el capítulo 2, la combinación de comportamientos se suele realizar por arbitración/selección; por cooperación de comportamientos; o mediante técnicas mixtas, como la modulación entre comportamientos. En [63] optamos por un esquema de emergencia simple en el que se conmutaba entre los comportamientos "alcanzar la pelota" y "acercarse a la portería", según se diesen las condiciones adecuadas, en sintonía con el concepto de liberadores o *releasers* de la teoría clásica de comportamientos. En nuestro esquema, el robot responde inicialmente al comportamiento de alcanzar la pelota para, una vez se ha tomado posesión de esta, activar el otro comportamiento que permite dirigirse a la portería con la pelota controlada. Si en el curso de esta segunda acción se perdiese el control de la pelota, de nuevo se conmutaría al primer comportamiento hasta alcanzar la pelota otra vez. En este caso el *releaser* que permite la conmutación a uno u otro comportamiento, sería la pérdida o recuperación de la pelota, que puede ser detectada mediante el sensor alojado en el pecho del robot.

En la figura 6.19 se puede observar una secuencia de imágenes que muestra una ejecución del comportamiento emergente obtenido. Inicialmente, el robot determina que no está en posesión de la pelota, pasando a ejecutar el comportamiento de "alcanzar la pelota". Como ésta no se encuentra en el campo de visión, se recupera un caso asociado al comportamiento que lleva al robot a girar sobre sí mismo con la cabeza erguida hasta el momento en que se detecta la pelota en la imagen, y empieza la aproximación hacia la misma.

Cuando la pelota es alcanzada, se conmuta al segundo comportamiento de acercamiento a la portería, lo cual permite llevar la pelota hacia la línea de gol. Este comportamiento ha sido entrenado para que, cuando se determine que las líneas de la portería están suficientemente cerca, se chute para marcar un gol. De esta forma

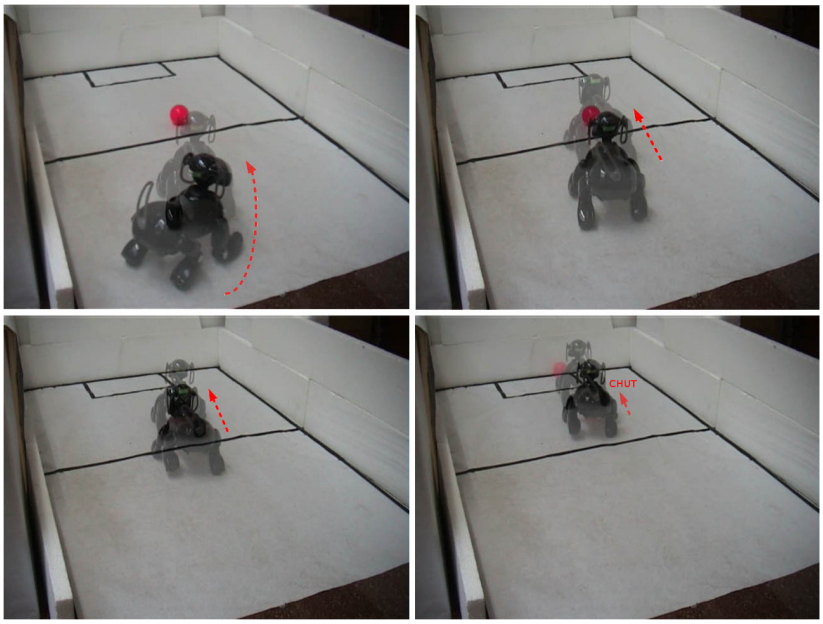


Figura 6.19: Comportamiento emergente obtenido de dos comportamientos simples mediante secuencia temporal (vista posterior)

podemos comprobar como dos comportamientos simples se pueden combinar para implementar otros más complejo de una manera sencilla.

En la figura 6.20 se observa otro experimento, representado de forma cenital en este caso. Podemos ver como, de nuevo, el robot ejecuta el comportamiento emergente de forma correcta, ya que en primer lugar es capaz de alcanzar la pelota para, a continuación, empujarla hacia la portería y chutar cuando llega a las líneas del área.

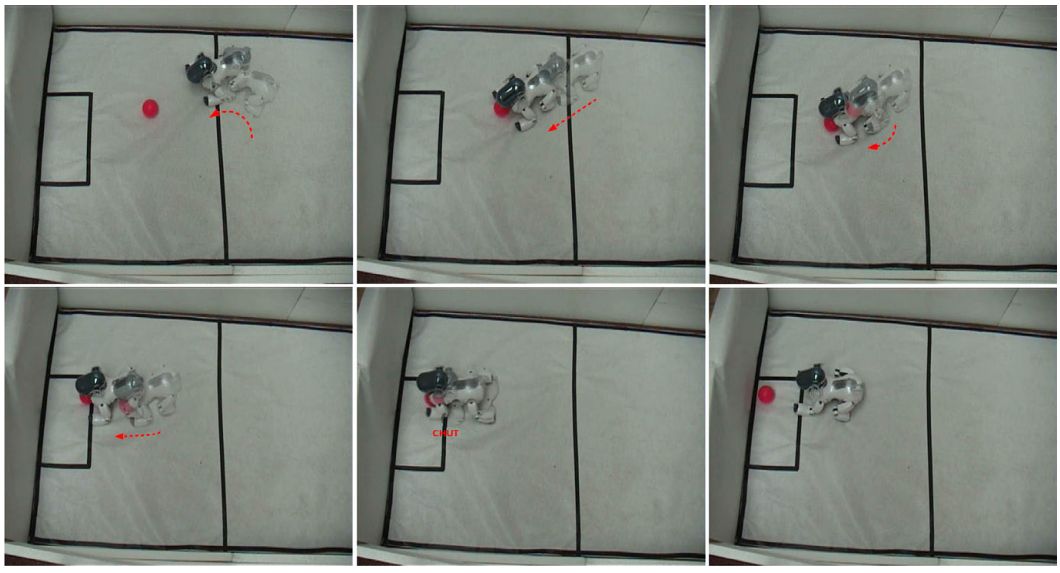


Figura 6.20: Comportamiento emergente obtenido de dos comportamientos simples secuencia temporal (vista cenital)

2.5.2. Combinación de comportamientos básicos con ponderación aprendida por CBR

En [283] se planteó la posibilidad de obtener un comportamiento emergente a partir de varios módulos básicos, mediante una combinación ponderada de las respuestas de cada uno de los módulos. Uno de los principales problemas de la obtención de comportamientos emergentes mediante la combinación de las diferentes respuestas reactivas de los módulos componentes radica en cómo establecer el peso de cada una de esas respuestas individuales en la respuesta final conjunta para obtener las acciones más adecuadas a los objetivos del comportamiento compuesto. La mayoría de los sistemas de diseño ascendente se apoyan en una capa adicional o de '*middleware*' para estimar la adecuación de los componentes, considerados de forma independiente, ante una situación. No obstante, si los comportamientos básicos están bien diseñados, solo deben incluir información local específica a su propio cometido. En consecuencia, resulta muy difícil diseñar una estrategia global a alto nivel empleando únicamente las instancias individuales obtenidas desde el bajo nivel. Además, en la mayoría de los casos, la información global no suele obtenerse como una combinación lineal de las instancias locales, de forma que se debería utilizar otro criterio para asignar pesos a la aportación de los módulos componentes. Sin embargo, cuando se identifica y obtiene la instancia adecuada que relaciona entradas y salidas, es posible aprender por imitación u observación lo que no resulta sencillo de expresar a través de ecuaciones. Así, se podría usar un módulo *CBR* para relacionar una instancia que identifique una situación con un vector de pesos que indiquen la influencia de los comportamientos individuales en la respuesta conjunta (figura 6.21).

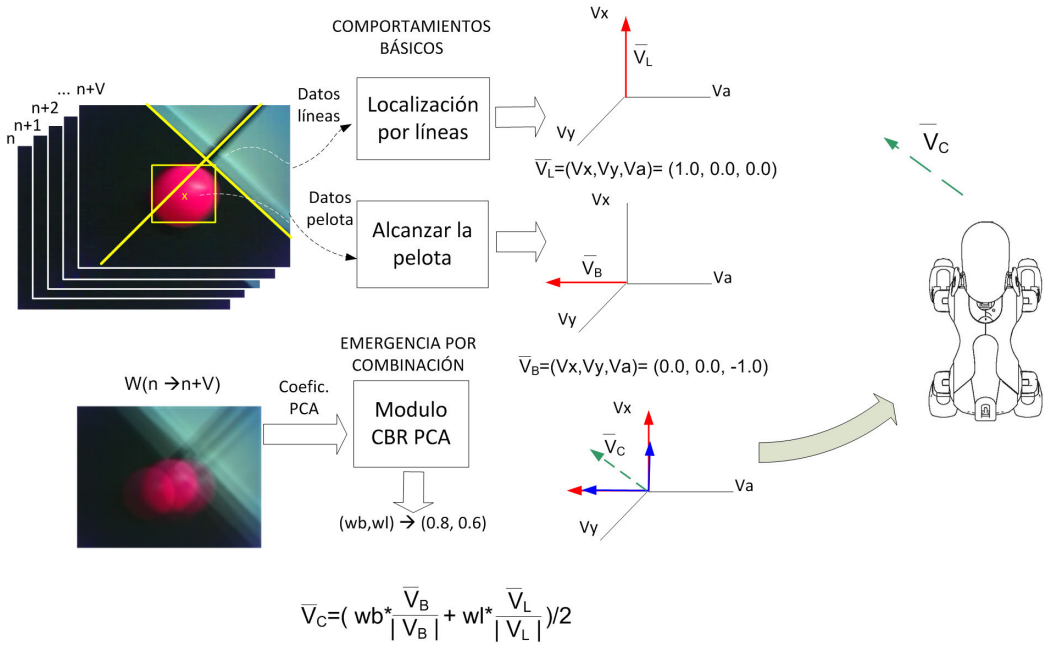


Figura 6.21: Módulo CBR de ponderación del comportamiento emergente

Uno de los principales problemas de este concepto es la representación de la situación expresada por la imagen captada por el robot. A diferencia de los módulos básicos, en este caso el módulo *CBR* trabajaría con información global, lo cual dificulta mucho el diseño de la base de casos. Se descartó rápidamente la representación de

la situación a través de los píxeles de la imagen ya que la cantidad de información de entrada sería excesiva a poco que la imagen tuviese una buena resolución. En su lugar se desarrolló una clase *JNI* en *openCV* para obtener los componentes principales de la imagen (*PCA*) [284]. *PCA* es una técnica ampliamente utilizada que permite reducir el volumen de datos necesario para la representación de una imagen dada. La independencia estadística de los componentes principales explica su adecuación para propósitos de representación y diferenciación de imágenes [285].

Dado un escenario en el que se va a desenvolver el comportamiento emergente, y que se quiere codificar de alguna forma, dicho escenario está representado por un conjunto de imágenes I observadas durante la ejecución del comportamiento. Estas imágenes proporcionan un volumen de información V_I, igual al número de píxeles en una imagen, multiplicado por el número de imágenes consideradas en el conjunto y por las componentes de color de cada píxel. Esta información define, por tanto, un espacio de dimensión N, en el que cualquier imagen del conjunto puede ser representada a partir de elementos de ese espacio. Además, y debido a la naturaleza redundante de las imágenes, se puede considerar que éstas también conforman un subespacio dentro de dicho espacio, que tendrá una dimensión P, menor que N. Si consideramos un conjunto de m capturas arbitrariamente escogidas y pertenecientes al espacio P-dimensional, se podría calcular una base de este subespacio que permitiese representar a cualquier imagen usando únicamente p componentes. La calidad de la base para representar al conjunto de m imágenes se calcularía a través del error cuadrático medio, ε^2, entre las m imágenes originales y su proyección en la base del subespacio propuesta. Este error será mínimo cuando se consigan extraer p vectores ortogonales de entre los m autovectores de la matriz de autocorrelación del conjunto de m imágenes [286]. Estos p vectores serían los componentes principales asociados a los p autovalores de de mayor valor de su matriz de autocorrelación. La base ortogonal constituida por estos p componentes, $\left\{\vec{\phi}_k\right\}_{k=1}^{p}$, nos permite obtener los vectores de características de las imágenes. Dada una nueva imagen, su vector de características, $\vec{Y}$, se obtendría proyectándola sobre la base ortogonal propuesta. $\vec{Y}$ tendría solo p componentes, y sería tan resistente al ruido o a transformaciones como la propia imagen original.

Si asumimos que el conjunto de m imágenes escogidas para el cálculo de la base es grande, y que sus proyecciones son suficientemente distintas, la base P-dimensional del subespacio podría permitirnos representar también imágenes no pertenecientes al subconjunto de m imágenes originales, en la medida en que se corresponderían con situaciones similares. La calidad en la aproximación de un vector genérico, $\vec{Y}$ con,

$$\vec{Y} = \sum_{i=0}^{N-1} y_i \vec{\delta}_i, \tag{6.3}$$

se evaluaría a través del error cuadrático medio asociado a su proyección en la base P-dimensional del subespacio:

$$\varepsilon^2 = \left| \vec{Y} - \sum_{j=1}^{P} \varphi_j \vec{\phi}_j \right|^2 \tag{6.4}$$

siendo φ_j la proyección de $\vec{Y}$ en el autovector $\vec{\phi}_j$. y_i sería el coeficiente i del vector, y $\vec{\delta}_i$ una delta de Dirac centrada en el punto i del eje de abscisas.

A partir de un conjunto de imágenes representativas de la situación que se desea aprender, se construiría una base de representación del conjunto de imágenes, de forma que cualquier imagen del mismo se pueda representar a través de unos pocos valores proyectados en la base. La clave del buen funcionamiento del método radica en no considerar todos los componentes *PCA* de las imágenes sino solo los más significativos, que permiten mantener la "esencia" del escenario al que representan. Sin embargo, la eliminación de los componentes "no esenciales" tiene también ciertos efectos negativos al dificultar la diferenciación entre distintas situaciones pertenecientes al mismo escenario, como se verá más adelante.

Por otra parte, aunque el comportamiento emergente obtenido debe tener también un carácter predominantemente reactivo, el control tiene también una cierta componente deliberativa ya que el peso de cada comportamiento básico podría cambiar dependiendo de, por ejemplo, la trayectoria de la pelota. Para introducir la componente temporal en el sistema de una forma directa y sencilla se consideró trabajar, no con imágenes individuales, sino con secuencias de imágenes obtenidas mediante una ventana temporal móvil, que se combinarían en una imagen o trama simple mediante olvido exponencial [287]. Como queremos que se mantenga el carácter reactivo del comportamiento emergente, la ventana considerada en el experimento tiene un tamaño de solo 5 imágenes.

En una primera etapa se entrenan los comportamientos básicos de forma independiente, de manera que cada uno de ellos adquiera el conocimiento para responder ante su cometido, de forma aislada o independiente. El entrenamiento se realiza por imitación del mismo modo que se explicó en los apartados 2.2 y 2.3 (figura 6.22).

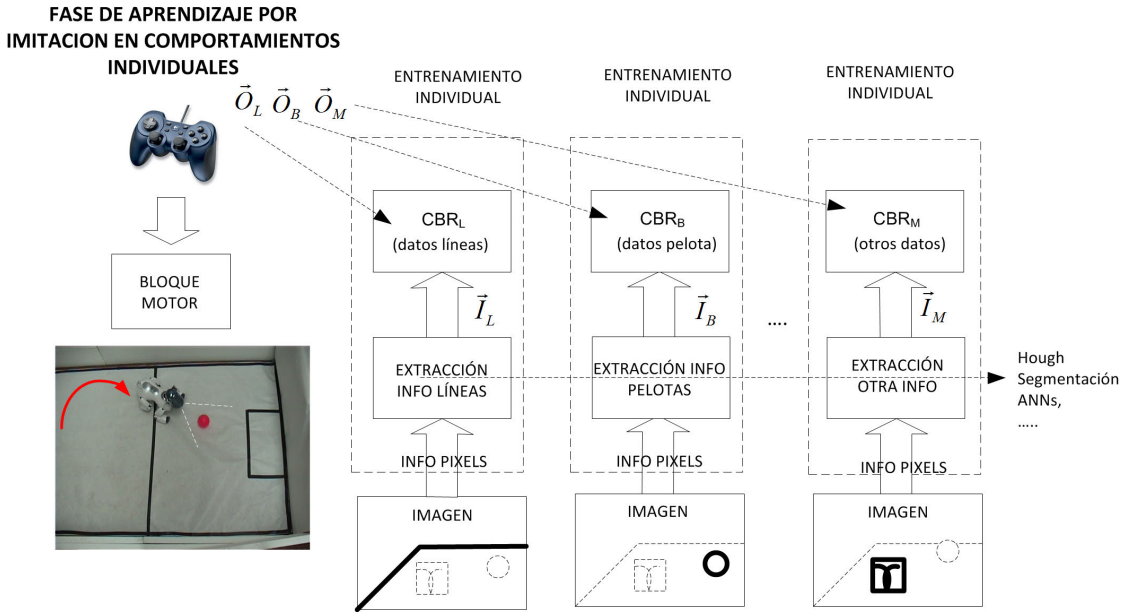


Figura 6.22: Entrenamiento de comportamientos básicos individuales

A continuación se procede a una segunda etapa de entrenamiento por imitación en la que se pretende determinar los pesos que se deben aplicar a las respuesta obtenidas desde cada uno de los comportamientos básicos. Para ello, y ejecutando el entrenador la conducta correspondiente al comportamiento emergente que se desea

obtener, se recuperan los casos individuales de los módulos participantes, y se combinan sus respuestas de forma que se obtenga el vector de movimiento ejecutado por el operador. Esto permite generar una base de casos que asocia la situación general experimentada por el robot con un vector de pesos que son los que se deberían aplicar a los módulos individuales ante dicha situación. Visto de otra forma, el robot aprende el grado de influencia de cada comportamiento básico ante determinadas situaciones experimentadas y almacena ese conocimiento a través de la relación entre los componentes *PCA* que representan las situaciones, y los pesos a aplicar a cada uno de los comportamientos de bajo nivel (figura 6.23).

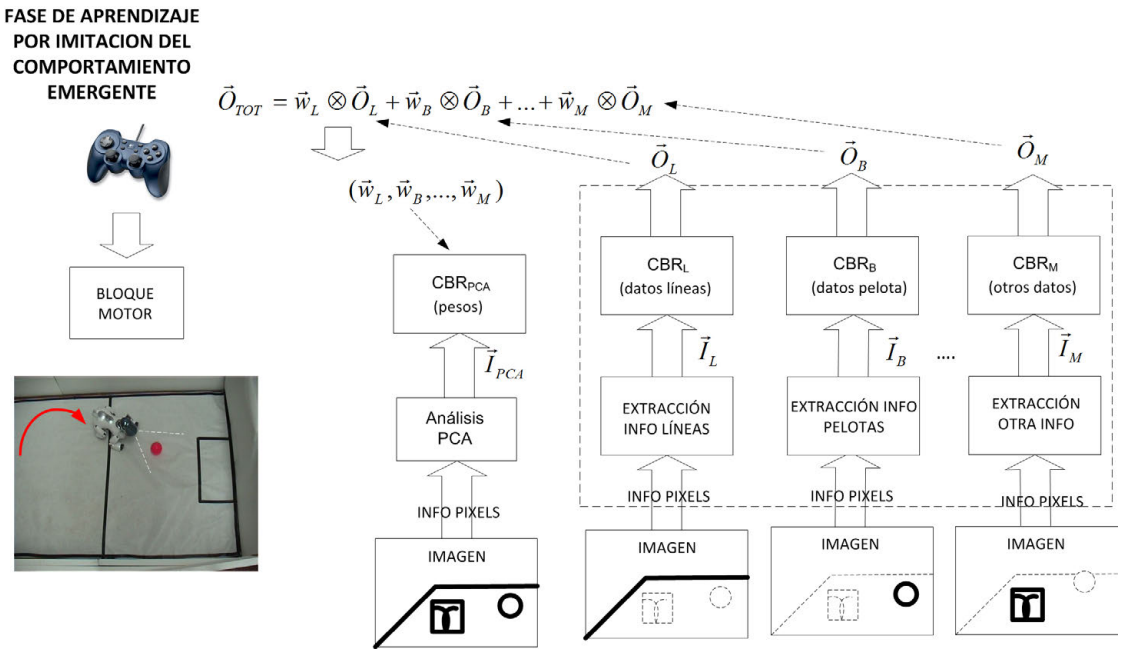


Figura 6.23: Entrenamiento de ponderación de comportamientos básicos en el comportamiento emergente

En la fase de operación se realizan simultáneamente consultas *CBR* a cada uno de los comportamientos de bajo nivel, así como al módulo de combinación emergente, y se combinan las respuestas obtenidas de los primeros según los pesos recuperados del último (figura 6.24).

Para tratar de probar esta aproximación a la construcción de comportamientos emergentes, se partió de los comportamientos básicos descritos en los apartados 2.2 y 2.3 del presente capítulo, y que permiten respectivamente alcanzar la pelota y rarla hacia la portería; y localizarse dentro de los límites del campo de juego para poder dirigirse a la portería, identificada mediante una línea vertical. En el primer intento de combinación en un comportamiento emergente, los comportamientos simplemente se alternaban, dependiendo de las condiciones presentes en el escenario, en particular la visualización y/o posesión de la pelota. Siguiendo este método de combinación, *AIBO* (casi) siempre alcanza la pelota y la dispara, pero no siempre hacia la dirección correcta (figura 6.25).

La capa de aprendizaje *CBR* basada en *PCA*, combina la respuesta de este módulo con el de localización en el campo, dando pesos a cada uno de ellos según lo aprendido a partir de las características generales de las imágenes quedando definan las situaciones. Los pesos de las respuestas de los módulos individuales se asocian, a través de casos

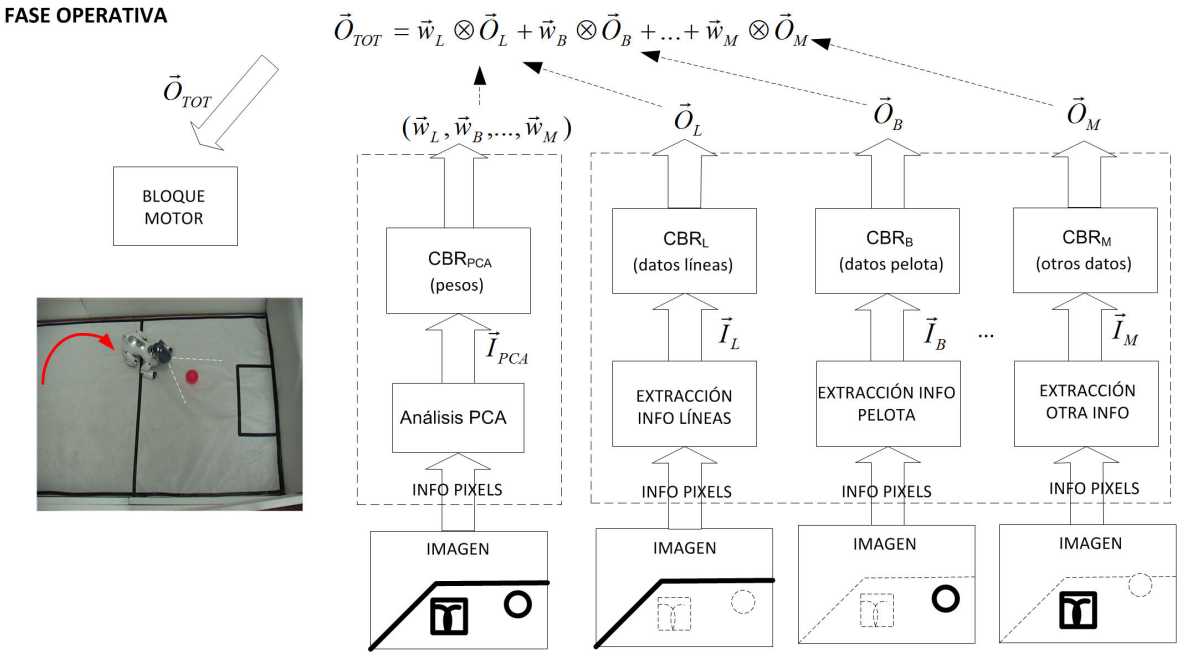


Figura 6.24: Obtención del comportamiento emergente en el modo de operación del robot

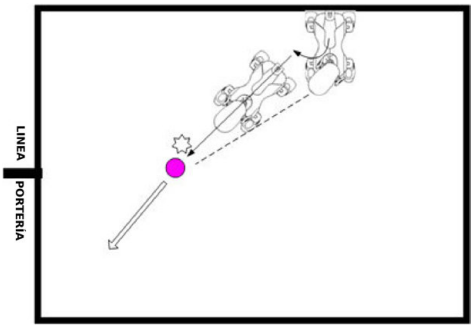


Figura 6.25: Disparo en dirección incorrecta en el módulo de acercamiento a la pelota

CBR, con las imágenes de entrada, de forma que, para cada situación experimentada, el módulo *PCA* proporciona una "intuición" al robot acerca de la importancia de la respuesta de cada uno de los módulos individuales, en esa situación en particular.

En una primera prueba se trató de resolver el problema antes mencionado del "disparo en dirección incorrecta". Para ello se trabajó en un escenario controlado y limitado, en el que el robot ya está viendo desde el principio tanto las líneas del campo y portería, como la pelota. No consideramos un escenario más amplio, en el que el robot pudiese partir de cualquier posición del campo, y con la pelota en cualquier otra posición, ya que el número de secuencias de imágenes necesarias para caracterizar dicho escenario sería excesivamente grande.

En esta prueba inicial, el *AIBO* está situado en el extremo más alejado del campo, en un ángulo de 90° con respecto a la línea de gol de forma que, para realizar una acción correcta, debe rotar hacia ella, orientarse en la misma dirección y acercarse hasta una distancia de disparo. Como hemos visto antes, el comportamiento de "alcance de la pelota", simplemente se encara hacia la pelota, se mueve hacia ella, y dispara al alcanzarla, enviando la pelota hacia un lateral del campo(figura 6.25). El

comportamiento de localización en el campo busca alguna línea de delimitación del campo, gira hasta observar la línea de gol, y se acerca hacia ella.

Para obtener un comportamiento emergente que permita disparar la pelota en la dirección correcta, tendríamos que girar el robot en dirección a la línea de gol, acercándose simultáneamente a la pelota, y disparando cuando se alcance la distancia adecuada. En el test se consideró una ventana de 5 imágenes para la representación de las secuencias, y un parámetro de olvido exponencial, λ_{olvido}, de valor 0.3. Un λ_{olvido} mayor daría mas dominancia a la última imagen de la secuencia respecto a las anteriores. Para la construcción de la base *PCA* se consideraron únicamente los 20 coeficientes más significativos, correspondientes a los autovalores mayores.

Como se puede ver en la figura 6.26, el robot gira hacia la pelota acercándose de forma orientada a la línea de gol, y dispara cuando alcanza esta, pero esta vez en dirección a la portería. Las líneas de córner y la línea de gol determinan un comportamiento inicial de a aproximación hacia la portería; y la visualización de la pelota impele a acercarse a ella y chutar. El módulo *PCA* inhibe la respuesta del comportamiento de la pelota en las etapas iniciales, otorgando un menor peso

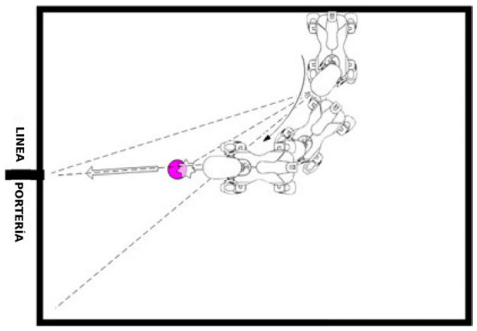


Figura 6.26: Disparo en dirección correcta con las respuestas de los comportamientos de "pelota" y "líneas" ponderados mediante el módulo PCA

En una segunda prueba, se situó la pelota en el extremo opuesto del campo, de forma que estuviera alejada de la portería, con el robot encarado hacia ella pero con la línea de gol detrás del robot. El comportamiento basado en la pelota hace que el robot atraviese todo el campo hasta alcanzar la pelota y chutar en dirección opuesta a la línea de gol rival —¡hacia nuestra propia portería!—, como se muestra en la figura 6.27.

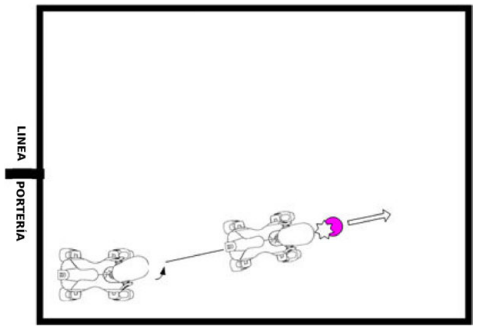


Figura 6.27: Aproximación errónea determinada por el comportamiento de acercamiento a la pelota

Esta respuesta es "normal", ya que no se incluye ningún dato de orientación

del robot en la información que define al comportamiento. El comportamiento de localización mediante líneas, busca una línea que delimite el campo y hace girar al robot sobre sí mismo, hasta localizar la línea de gol.

Se entrenó el módulo *PCA* para "flanquear" la pelota y tratar de situarse tras ella, encarando la línea de gol enemiga. En el proceso de aprendizaje se generaron 173 secuencias enventanadas, usando las mismas condiciones del experimento anterior —ventana de 5 imágenes, y factor de olvido exponencial de valor 0.3—. Con esta información se construyó una base *PCA* considerando de nuevo solamente los 20 coeficientes mas significativos, y se obtuvieron los pesos a aplicar a las respuestas de los módulos básicos antes las diferentes situaciones. Como se esperaba, el robot ya no va directamente a por la pelota, si no que es capaz de flanquearla y situarse detrás de ella para aproximarse desde la dirección correcta (figura 6.28).

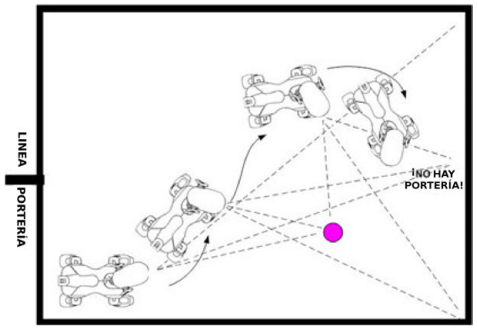


Figura 6.28: Aproximación correcta a la pelota del comportamiento emergente

Hay que destacar que en este segundo test el robot caía a menudo en trampas locales en las que respuestas compuestas consecutivas eran contradictorias provocando que, finalmente, el robot permaneciese en su posición sin moverse. Estas situaciones se podían resolver, por lo general, deteniendo y reiniciando el control *CBR* del robot, ya que las respuestas iniciales, en las que se construyen las primeras secuencias enventanadas, son simplemente un promedio de las respuestas de los comportamientos básicos. En este sentido, sería deseable incluir algún mecanismo en la capa "*middleware*" de alto nivel, que detectase y evitase estos mínimos locales.

A pesar de los buenos resultados en situaciones simples, este método presenta problemas cuando es necesario entrenar secuencias más largas que implican un mayor número de casos en la base *CBR* de composición de comportamientos emergentes. El problema está relacionado con la inclusión de información global en un único módulo lo cual entra en contradicción con la teoría de los comportamientos, según la cual estos deben tratar únicamente con información local. Por otra parte el uso de *PCA* como técnica de reducción del volumen de información no parece del todo adecuada. Si bien. el uso de componentes principales permite —en teoría— comprimir la información en un menor número de componentes esenciales comunes al comportamiento, no queda claro que sean precisamente esos componentes no comunes a todas las situaciones/secuencias las que permitan distinguir unas experiencias de otras dentro del comportamiento complejo que se está aprendiendo. Finalmente, la inclusión de componentes deliberativos a través del uso de secuencias temporales en vez de imágenes fijas parece una buena idea, pero aplicable en el desarrollo de comportamientos de más alto nivel y que incluyan dicha componente temporal a largo plazo. Por contra, en esta tesis estamos trabajando más en la línea del desarrollo de comportamientos complejos

pero a nivel reactivo. Por este motivo, se ha "aparcado" temporalmente esta línea de trabajo aunque no se descarta retomarla en el futuro, cuando se trate de incorporar a la arquitectura robótica comportamientos que incluyan componentes temporales o deliberativos en cierta medida.

2.6. Conclusiones de las pruebas de la primera etapa

Este primer conjunto de pruebas nos permitió comprobar que nuestro planteamiento de diseñar comportamientos robóticos de bajo nivel a través de un aprendizaje basado en *CBR* era acertado y podría dar lugar a arquitecturas prometedoras. Con respecto al trabajo previo relacionado, nos permitió extender y generalizar los fundamentos de la arquitectura para cualquier tipo de robot, no solo los tradicionales robots rodantes en el ámbito de la navegación. Además, incluimos en el problema y como factor de decisión de los comportamientos, información procedente de la visión artificial del robot, en vez de usar únicamente navegación basada en sonar; esto se corresponde bastante con la forma de operar del comportamiento humano, en el cual un porcentaje muy importante de la información procede de la visión. *CBR* se muestra como una herramienta muy adecuada para un diseño de comportamientos sencillo e intuitivo, basado en aprendizaje por imitación, que permite prescindir del tratamiento estadístico y algorítmico de la información y absorber automáticamente los patrones de conducta ejecutados por el operador.

Sin embargo, algunos inconvenientes empezaron a hacerse patentes. En primer lugar, los comportamientos diseñados no tenía un excesivo grado de complejidad, tanto en la variedad de respuestas, como en la información que determinaba cada una de ellas, en su representación a través de casos. Los comportamientos reactivos evitan la necesidad de un modelo global del sistema, manejando únicamente información local pero, a medida que aumenta el número de variables que influyen en el comportamiento, también lo hace el espacio multidimensional que representa el conocimiento del comportamiento y, por ello, el número de casos de la base. Como el tiempo de recuperación de la respuesta ante un problema depende del número de casos en la base *CBR*, la reactividad del sistema puede llegar a verse comprometida. Más adelante indicaremos algunas alternativas propuestas para paliar en parte este problema. Por otra parte la combinación de comportamientos simples para la emergencia de otros más complejos no siempre es tan evidente como en los caso indicados en los experimento. Cuanto más simples, locales —a nivel de tratamiento de información—, e influidos por menos elementos del entorno, sean los comportamientos básicos diseñados, mayor será la dificultad para combinarlos adecuadamente en comportamientos emergentes de alto nivel. Este hecho está en consonancia con la teoría del trasvase de conocimiento entre recipientes referida en el apartado 3.3 del capítulo 4: los comportamientos complejos implican unas respuestas determinadas por la interrelación entre los componentes que dirigen la actuación de los comportamientos básicos por lo que, si dicha interrelación no está incorporada en el conocimiento de los módulos básicos, deberá ser añadida posteriormente, de alguna forma, en la fase de combinación para la emergencia del comportamiento de nivel superior. Esto lleva a que, o bien se determina un modelo o método que permita una combinación genérica de módulos básicos, o bien se parte de módulos un poco más complejos, cuyo entrenamiento incluya componentes conductuales del comportamiento de alto nivel que se pretende diseñar,

más que acciones específicas relacionadas con elementos aislados del entorno en el que se desenvuelve el robot. En los siguientes apartados estudiaremos esta nueva aproximación, y analizaremos su rendimiento, así como las ventajas e inconvenientes que conlleva.

3. Comportamientos reactivos complejos

Para esta segunda etapa de nuestro trabajo, se estableció un nuevo planteamiento para el diseño de los comportamientos reactivos básicos. Si en el planteamiento inicial, se empezaba diseñando los comportamientos básicos en torno a objetos individuales aislados que guiaban las características de la conducta asociada, y a continuación se trataba de construir los comportamientos emergentes a partir de los básicos, ahora se partirá de una descomposición del comportamiento complejo que se desea aprender —por ejemplo, "jugar al fútbol"— en "sub-conductas" que representen aspectos de la misma, relacionados con subconjuntos de situaciones que asociaremos a dichas "sub-conductas". (figura 5.5, imagen central). Este nuevo planteamiento conlleva la realización de algunos cambios en la arquitectura y el modelo de diseño, si bien el sustrato básico de aprendizaje a través de *CBR* permanece intacto.

3.1. Modificaciones SW y de escenario de pruebas

Desde un punto de vista más técnico también se llevaron a cabo ciertos cambios en cuanto a la estructura HW/SW del sistema, así como el escenario de realización de los experimentos. Hasta ahora, la implementación de los módulos *CBR* se estaba realizando mediante la librería JAVA *AITools* con buenos resultados y satisfacción por su sencillez de uso. Sin embargo, dicha librería carecía de algunas facilidades relacionadas sobre todo con el mantenimiento y estructura de la base de casos, y además estaba muy acoplada a la aplicación para la que fue diseñada. Por eso, encontramos ciertas dificultades para realizar modificaciones en, por ejemplo, la estructura de los casos, o el proceso de recuperación de las mismos, algo que necesitábamos en el esquema propuesto. Esto nos llevó a descartar dicha librería en favor de la librería C++ *libCbr* implementada por un miembro del grupo de investigación *ISIS*, sobre la cual hemos trabajado para realizar las modificaciones necesarias para adaptarla a nuestra arquitectura. La integración entre el interfaz JAVA de Tekkotsu y la aplicación C++ se realizó a través de clases nativas *JNI*, similares a las empleadas para integrar el tratamiento *PCA* realizado con la librería *openCV*, y que vimos en el apartado 2.5.2 del presente capítulo.

Otro cambio significativo está relacionado con el escenario de las pruebas. El nuevo planteamiento implica la necesidad de contar con un campo de pruebas de mayor tamaño y que incluya varios objetos a la vez. Inicialmente se instaló en un laboratorio una moqueta verde, que permitiese construir al menos la mitad de un terreno con las medidas oficiales de Robosoccer y se añadieron algunos elementos (balizas, porterías,..) que representasen los objetos del escenario. Algunos de los robots empezaron a fallar debido a su uso intensivo (articulaciones con temblores, problemas de baterías, etc..) y además, en 2007, Sony decidió dejar de fabricar el robot, con lo que dejó de ser posible enviar a reparar los mismos. Ante esa situación se decidió empezar a realizar el grueso de las pruebas en un simulador, en particular el simulador de *AIBO* incorporado en

Tekkotsu con integración en el simulador de entornos 3D *Mirage*(figura 6.29), disponible a partir de su versión 4.01. El simulador de *AIBO* tiene la ventaja de que reproduce fielmente el funcionamiento del software en el robot; permite la posibilidad de enviar datos simulados de las lecturas de los sensores del robot, a través de ficheros de texto, así como imágenes para representar las que se obtendrían desde la cámara del robot. Así mismo, el simulador de mundos virtuales permite construir escenarios sobre los que se pueden realizar pruebas bastante fieles a las que se harían en el "mundo real". Basado en la librería de software libre *Ogre3D*, el simulador incluye el motor de simulación de fenómenos físicos, *Bullet*, con lo cual las pruebas realizadas nos van a proporcionar resultados muy similares a los que obtendríamos con el robot real.

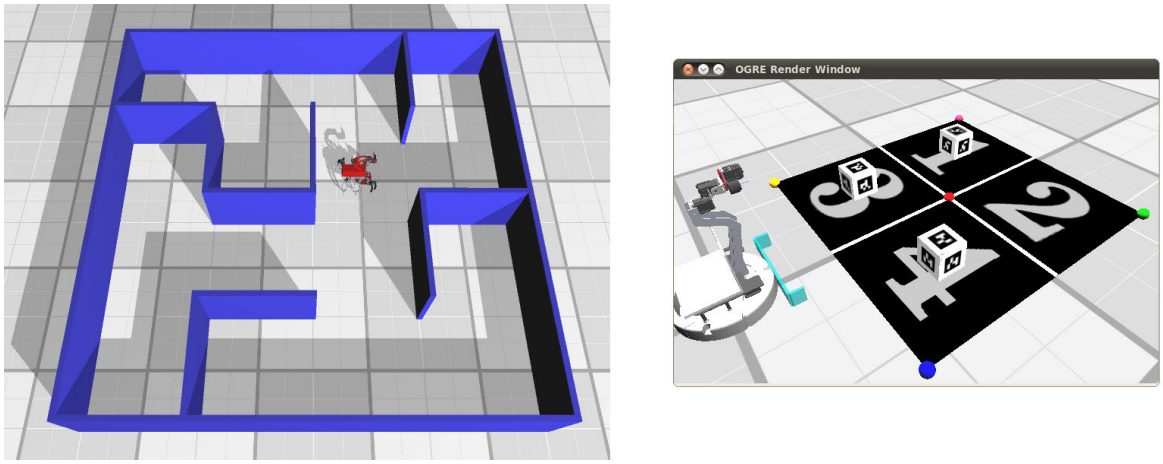


Figura 6.29: Ejemplos de experimentos con robots en un entorno Tekkotsu-Mirage

Por otra parte, el uso de un simulador presenta evidentes ventajas en el proceso de validación de los aspectos directamente relacionados con la investigación de esta tesis. La puesta en marcha de un robot real implica la solución de multitud de problemas que no están directamente relacionados con las hipótesis cuya validez se quiere demostrar, problemas que que conllevan un trabajo adicional intenso. Por ejemplo, en las pruebas iniciales con el robot real, se detectó la presencia de artefactos en la lente de la cámara del robot que oscurecían las esquinas del mismo, degradando notablemente la segmentación de las imágenes cuando los objetos caían cerca de estas esquinas. Pese a que se trabajó en una aproximación para solucionar este problema [288], todos estos aspectos accesorios ralentizan y distraen de los elementos principales de la investigación en que se centra esta tesis.

El uso de simuladores permite eludir estos problemas accesorios eliminando a través de los parámetros de configuración los aspectos no deseados. En particular, se fijaron unas condiciones estables de iluminación y se usaron objetos virtuales con colores bien definidos y poco nivel de variación por reflejos y sombras, para favorecer la segmentación. Además, al estar conectados el simulador del robot y el del entorno virtual a través de canales internos al PC de simulación, fue posible utilizar un intercambio de imágenes a través de un enlace *wireless* simulado de tipo *Transfer Control Protocol (TCP)*, con una alta tasa y de mayor resolución. Esto sería imposible en el caso del robot real, debido a la baja tasa y condiciones del enlace inalámbrico real. Por otra parte, la baja capacidad de cómputo del procesador integrado en el robot real impide llevar a cabo una segmentación de mas de 1 ó 2 tipos de objetos en

tiempo real a una tasa de imágenes adecuada para un comportamiento reactivo. Este problema ya estaba en vías de solución al trasladarse la etapa de segmentación desde el robot hacia el interfaz del PC, pero con el uso del simulador ya no supondría ningún inconveniente debido a las mayores capacidades de procesamiento de los ordenadores actuales. Finalmente, la existencia de otros modelos de robots de varios tipos (rodantes, hexápodos, manipuladores), en el entorno de simulación, hará posible, más adelante, probar la arquitectura con otros robots y otro tipo de tareas diferentes de la navegación.

No obstante, hay que resaltar que la validación con robots reales es siempre necesaria, si bien en nuestro trabajo, al haberse validado todo el conjunto de pruebas iniciales en el AIBO real y haberse comprobado el funcionamiento correspondiente en el simulador, se puede considerar este segundo conjunto de pruebas como una extensión de las anteriores. De cualquier forma, en el futuro se tratará de comprobar el trabajo desarrollado en el simulador en el robot real, o al menos medir la degradación que suponen las problemas antes mencionados en la arquitectura propuesta.

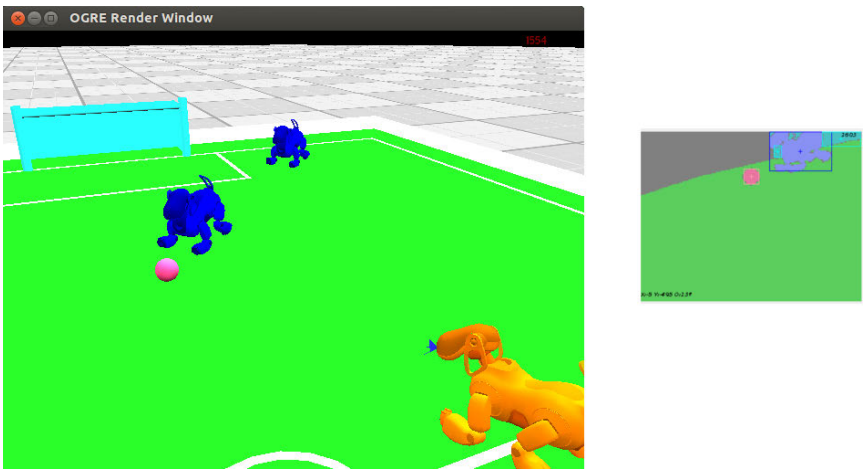


Figura 6.30: Un experimento con la arquitectura propuesta en el entorno de simulación

3.2. Diseño de un comportamiento complejo de acercamiento a la pelota

El primer comportamiento diseñado según el nuevo enfoque tiene por objeto aprender a alcanzar la pelota pero considerando la posición del resto de elementos del escenario, de forma que al alcanzarla el robot tuviese campo abierto para salir con la pelota controlada, chutar hacia campo abierto o moverla "protegido" por un jugador del mismo equipo. Este comportamiento resulta bastante más complejo de programar de forma algorítmica que los originales basados en objetos individuales, ya que la casuística es mucho mayor al considerar las combinaciones de estados de los diferentes elementos que influyen en el comportamiento. En un sistema experto al uso resulta complicado definir a priori cuantas situaciones se pueden dar por la gran cantidad de variables implicadas. No es lo mismo enseñar a través de como ejecuta el ser humano un comportamiento, que explicar porqué dicho humano toma esas decisiones. Por otra parte, un comportamiento programado algorítmicamente carece de la flexibilidad necesaria para permitir cambios de estrategia ante variantes del problema como, por ejemplo, enfrentarse a diferentes contrincantes con distintas estrategias. Finalmente, la

programación de este comportamiento algorítmico exigiría un conocimiento bastante profundo, no solo del problema a resolver, sino también de como influyen los elementos componentes del problema en el mismo, tanto individualmente, como en conjunto. En contraste, el aprendizaje *CBR* permite la absorción automática de las estrategias y tácticas seguidas por el entrenador humano considerando todos los factores que afectan al problema a resolver; y permite también modificar la estrategia sin mas que realizar el aprendizaje de una forma distinta de afrontar el problema.

Tal como indicamos en el apartado 4.1 del capítulo 5, la información de entrada al comportamiento, que proporciona el criterio para realizar una u otra acción, está constituida por la descripción del estado de varios objetos que se considera que van a influir en el comportamiento, mas la posición de la cabeza del robot, que influye en la forma en que se visualizan los objetos en la imagen. Para este comportamiento se determinó de forma heurística que los objetos a incluir en el caso serían la pelota, y los posibles robots rivales y compañeros que pudiesen aparecer en el campo de visión. Las porterías se descartaron, ya que en este caso solo se trata de llegar a la pelota hasta controlarla, sin considerar la orientación del robot en el campo al obtener su posesión. Esto es coherente con nuestra filosofía de descomponer tareas complejas en comportamientos nucleares mas sencillos, pero difíciles de precisar analíticamente. Posteriormente se diseñarán otros comportamientos que tengan en cuenta la situación de la portería para realizar acciones encaminadas a conseguir un gol propio o evitar un gol del rival. La posición del robot en el campo no se considera relevante para este comportamiento, por lo que tampoco se incluirá.

Cada caso incluye, por tanto, un máximo de 4 objetos —pelota; hasta dos robots rivales; y un posible robot compañero—, por lo que queda constituido por un total de 18 componentes, si incluimos la posición de la cabeza del robot, descrita a través de sus características de PAN y NOD (figura 6.31). Para el caso de objetos pertenecientes a una misma clase —por ejemplo, los robots rivales— estos se ordenarán de izquierda a derecha según la posición de su centroide en la imagen. Por otra parte, el sensor del pecho del robot podrá determinar si se ha alcanzado la posesión de la pelota, lo cual indicaría la finalización del comportamiento. Para la discretización/conceptualización de la información se probarán dos conjuntos de umbrales distintos, como se indicará más adelante.

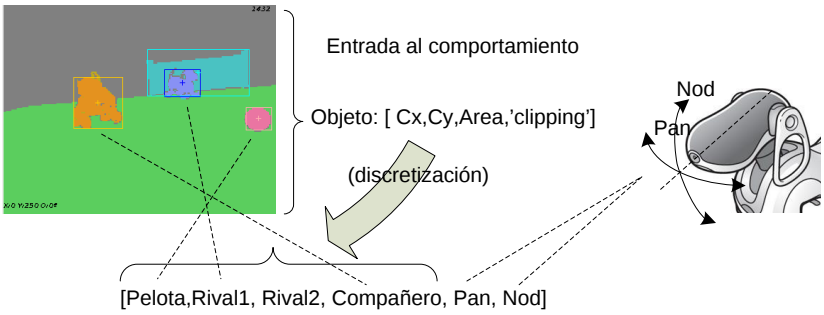


Figura 6.31: Información de entrada para los casos del comportamiento complejo de acercamiento a la pelota

Por otro lado, este comportamiento no aprovecha, en principio, la componente jerárquica en la estructura de distribución de la información mixta de la base *CBR* (apartado 4.3.2 del capítulo 5), ya que al no contar con la portería como elemento

discriminador, y considerar la posesión de la pelota como una indicación del final del comportamiento, no se realiza una distribución efectiva del conocimiento necesario para el comportamiento a lo largo de varias bases *CBR*-hoja, utilizándose finalmente solo una.

Como salida de cada caso se utiliza un vector de desplazamiento con 3 componentes, que se considera los suficientemente pequeño como para conseguir una respuesta reactiva en el comportamiento, y los suficientemente grande como para permitir una combinación con otros vectores similares procedentes de otros comportamientos, con objeto de obtener un comportamiento emergente de nivel superior, como veremos en un apartado posterior.

Para la obtención de los casos iniciales mediante el aprendizaje por experiencia, se entrenaron diversos escenarios que tratan de representar una variedad suficiente de situaciones que caracterizase —al menos inicialmente— el comportamiento del robot a diseñar. Se entrenó en primer lugar un conjunto de situaciones de acercamiento del robot a la pelota en ausencia de cualquier otro robot (figura 6.32). Las posiciones iniciales de las trayectorias estaban en las coordenadas $[0, 0]$ (S_0), $[500, 0]$ (S_1), $[-500, 0]$ (S_2), $[850, 400]$ (S_3), $[-850, 400, 0]$ (S_4), $[800, 100]$ (S_5), y $[-800, 100]$ (S_6). La posición de la pelota en el campo es indiferente, ya que lo que determinará el comportamiento del robot respecto a la misma es su posición relativa. Además, y dada la naturaleza reactiva del comportamiento, no es necesario mover la pelota durante la prueba, ya que el robot solamente tendrá en cuenta su situación en el momento de realizar la consulta *CBR*, sin considerar aspectos temporales de evolución previa. Las diferentes trayectorias fueron incluidas a medida que se observaban fallos en el aprendizaje debido a la ausencia de casos adecuados a alguno de los problemas presentados. En todas ellas el robot está visualizando la pelota al inicio de la prueba. Este entrenamiento es el equivalente al realizado para el diseño del comportamiento descrito en el apartado 2.2 del presente capítulo, con la diferencia de que los casos incluyen más componentes, si bien en este conjunto inicial estos componentes tienen un valor nulo o ausente.

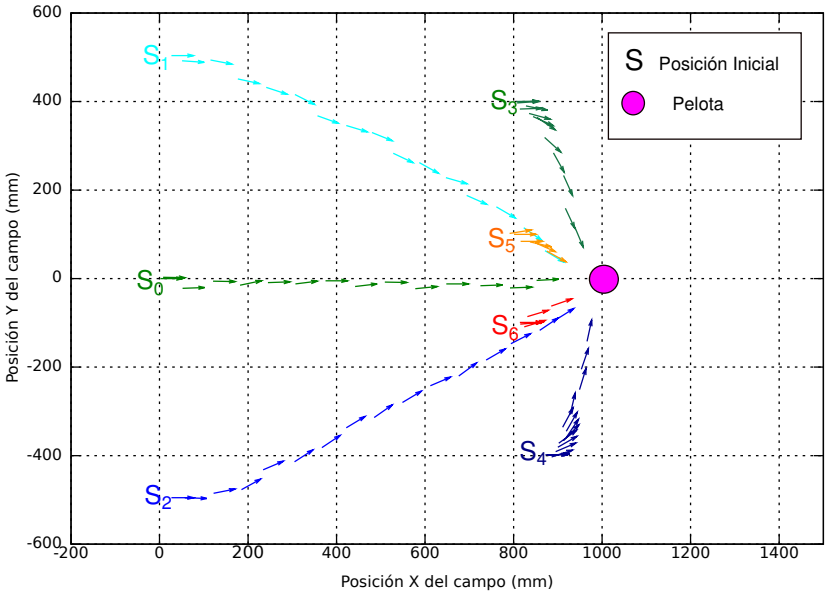


Figura 6.32: Secuencias iniciales de entrenamiento para el comportamiento "Pelota"

Como vimos en el apartado 4.3.2 del capítulo 5, la recuperación del caso se hace en

dos etapas: la primera de ellas, seleccionando únicamente los casos con coincidencia en cuanto a los objetos presentes en el problema; y recuperando, en la segunda, el caso más parecido al problema según la distancia de similitud seleccionada, una Manhattan normalizada. A destacar que se ha incluido también en la arquitectura la posibilidad de utilizar funciones de distancia compuestas en las que los distintos componentes del caso se comparen según funciones de similitud diferentes, para combinarse todas ellas globalmente de forma ponderada. Esta opción no se ha empleado en los experimentos presentados en esta tesis, aunque se analizará su uso en futuras investigaciones. Si se diese la circunstancia de que tras la primera etapa de recuperación no obtuviésemos ningún caso con el mismo conjunto de objetos, se considerarían todos los casos de la base *CBR* correspondiente a la hoja seleccionada en la estructura jerárquica. En esta situación se consideraría siempre al problema como un caso nuevo a adquirir y se aplicaría la etapa de adaptación que comentaremos más adelante.

En cuanto a la discretización/conceptualización, con objeto de reducir el espacio de casos del comportamiento, inicialmente se consideraron 5 intervalos equiespaciados para todos los elementos, aunque para el área se consideraron finalmente 7 intervalos de distancia creciente en forma cuadrática. El PAN de la cámara del robot también se modificó finalmente a 7 intervalos con tamaños crecientes de intervalos en los extremos, y con el 0 como punto central del intervalo medio. Estas dos últimas decisiones se tomaron tras observar la evolución de los valores de los componentes a lo largo de varios ensayos.

En la tabla 6.1, se indican los intervalos correspondientes a cada componente, en este conjunto de pruebas, así como su significado conceptual.

Cx		**Cy**		**Area**	
[0,0.2]	extremo-izquierda	[0,0.2]	extremo-superior	[0,0.2]	clipping-casi-nulo
[0.2,0.4]	izquierda	[0.2,0.4]	arriba	[0.2,0.4]	clipping-muy-pequeño
[0.4,0.6]	centroX	[0.4,0.6]	centroY	[0.4,0.6]	clipping-pequeño
[0.6,0.8]	derecha	[0.6,0.8]	abajo	[0.6,0.8]	clipping-medio
[0.8,1]	extremo-derecha	[0.8,1]	extremo-inferior	[0.8,1]	clipping-grande
Clipping		**Pan**		**Nod**	
[0,0.02]	area-minima	[0,0.2]	Pan-extremo-Der	[0,0.2]	Nod-Chut
[0.02,0.04]	area-muy-pequeña	[0.2,0.35]	Pan-derecha	[0.2,0.4]	Nod-muy-bajo
[0.04,0.08]	area-pequeña	[0.35,0.45]	PanC-der	[0.4,0.6]	Nod-bajo
[0.08,0.16]	area-media	[0.45,0.55]	Pan-centrado	[0.6,0.8]	Nod-medio
[0.16,0.33]	area-media-alta	[0.55,0.65]	PanC-izq	[0.8,1]	Nod-alto
[0.33,0.66]	area-grande	[0.65,0.8]	Pan-izquierda		
[0.66,1]	area-enorme	[0.8,1]	Pan-extremo-izquierda		

Tabla 6.1: Conceptualización de componentes del caso (set de pruebas 1)

Inicialmente se probó a entrenar el robot únicamente con las trayectorias correspondientes a las posiciones S_0,S_1,y S_2, que dieron lugar a una base *CBR* con un total de 52 casos tras la discretización/conceptualización y el filtrado de casos redundantes o conflictivos. Se obtuvieron unos resultados correctos en las mismas posiciones de partida, así como desde posiciones cercanas a las trayectorias empleadas. Sin embargo, posiciones muy distintas en cuanto a orientación o posición inicial con respecto a la pelota, no siempre daban los resultados correctos, observándose en el análisis de las pruebas que en dichas situaciones la similitud entre el problema describiendo la situación y el caso recuperado era muy baja.

Un aspecto adicional de las pruebas es el criterio de evaluación que determina si el robot está ejecutando correctamente el comportamiento aprendido, y en que medida.

Aunque frecuentemente se acude a criterios subjetivos para determinar la bondad del aprendizaje, suele ser más adecuado establecer un criterio mas formal y medible. Para este comportamiento, hemos determinado como criterio la situación de control o no de la pelota transcurrido un tiempo prudencial. Así, si el robot finaliza con la pelota debajo del pecho, algo detectable mediante el sensor alojado en el mismo, se dará una puntuación de 1 a la prueba; si el robot alcanza la pelota pero no logra controlarla completamente —por ejemplo, llega a la pelota pero la aleja con una pata—, se dará una puntuación de 0.5; finalmente, si el robot no logra alcanzar la pelota —por ejemplo, acaba pasando al lado de ésta—, se dará una puntuación de 0. Hay que destacar que, pese a utilizar un simulador en estas pruebas, el motor físico del mismo determina que la ejecución repetida de una prueba no es totalmente idéntica —tal como ocurriría en la vida real—. Por este motivo, se realizarán también varias pruebas partiendo desde una misma posición del campo.

Tras añadir las posiciones S_3 y S_4 a las trayectorias anteriores, para obtener una base *CBR* con 77 casos, las puntuaciones obtenidas en diez pruebas de comportamiento autónomo *CBR*, desde las posiciones entrenadas y varias posiciones adicionales fueron las indicadas en la tabla 6.2:

	Posiciones de partida					
		(0,0)	(500,0)	(-500,0)	(850,400)	(-850,400)
Puntos		90 %	85 %	90 %	90 %	70 %
	Otras posiciones					
	(-340,100)	(-200,-150)	(-400,-300)	(400,300)	(-700,-700)	(-700,-500)
Puntos	90 %	85 %	95 %	80 %	90 %	85 %

Tabla 6.2: Puntuaciones desde distintas posiciones (base CBR pruebas S0 a S4)

Las puntuaciones están expresadas en porcentaje respecto a la máxima puntuación posible. Cabe destacar que la puntuación 0 apareció únicamente en una ocasión a lo largo de todas las pruebas. Los principales errores se observaron en situaciones en las que el robot se encontraba cerca de la pelota pero en posiciones muy "escoradas" respecto a la misma. Por este motivo, se añadieron a las secuencias de entrenamiento anteriores las correspondientes a las posiciones S_5 y S_6, asociadas a giros del robot en posiciones cercanas a la pelota. Tras la discretización y filtrado, la base de casos correspondiente la base *CBR* pasó a tener 100 casos, incluyendo casos creados "artificialmente" para la detención del robot al tomar posesión de la pelota, y de giro sobre si mismo en caso de no detectar la pelota en el campo de visión. Tras repetir el mismo procedimiento de evaluación indicado anteriormente, se observó que el robot no tuvo problemas en realizar el comportamiento desde las posiciones de partida entrenadas con un alto grado de corrección, desde una gran variedad de posiciones, tanto entrenadas como no. Las puntuaciones en estas condiciones fueron las indicadas en la tabla 6.3.

También se probó una posición obtenida de forma aleatoria, tanto de la pelota $(-850, 425)$ como del robot $(700, -1500, 90°)$, con una puntuación del 95 %. Además, en todas las situaciones se alcanza al menos un 0.5 de puntuación, y en muchas de éstas no se alcanzó el 1 debido a choques de la pelota con la parte interior de las patas del robot.

Pese al buen rendimiento del comportamiento aprendido, éste sigue siendo demasiado básico para demostrar las ventajas con respecto a un comportamiento

	Posiciones de partida						
	(0,0)	(0,500)	(0,-500)	(850,400)	(850,-400)	(800,100)	(800,-100)
Puntos	95 %	90 %	90 %	100 %	100 %	100 %	95 %
	Otras posiciones						
	(-340,100)	(-200,-150)	(-400,-300)	(400,300)	(-700,-700)	(-700,-500)	
Puntos	100 %	100 %	100 %	100 %	90 %	100 %	

Tabla 6.3: Puntuaciones desde distintas posiciones (base CBR pruebas S0 a S6)

algorítmico como los que se pudieran obtener mediante *PFA* o *VHF*. Para darle interés y complejidad debemos entrenar una conducta del robot en presencia de otros elementos del escenario, que influyen en la respuesta del robot mas allá de girar hacia la pelota y dirigirse hacia ella. Además, la presencia de estos otros objetos complica la estrategia en el objetivo del comportamiento, ya que ahora no bastará con aproximarse a la pelota si no que el robot deberá intentar finalizar su acercamiento para acabar de la manera más favorable posible para sus intereses, bien encontrarse con campo abierto, bien protegido por un robot compañero de equipo. Con este objetivo se añadieron nuevas secuencias de entrenamiento que incluían un objeto adicional, bien sea un robot rival o bien un robot compañero y, a continuación nuevas secuencias con dos robots en el campo de visión del *AIBO*, tanto para el caso en que ambos fuesen rivales, como en el que uno de ellos fuese compañero de equipo. En las pruebas se consideraron distintas combinaciones de posiciones que tratasen de cubrir un espectro suficiente de situaciones suficiente para permitir una operatividad básica del comportamiento. En pruebas posteriores se añadiría una fase adicional de adquisición de conocimientos a través de la experiencia que podría suplir las posibles ausencias de algunos casos encontrados en el modo de control autónomo *CBR*.

En las figuras 6.33,6.34,6.35, 6.36 y 6.37, observamos las diferentes secuencias de entrenamiento que se realizaron para diversas situaciones de los robots, indicando la trayectoria que el operador decidió que sería mas adecuada para ser adquirida por el comportamiento.

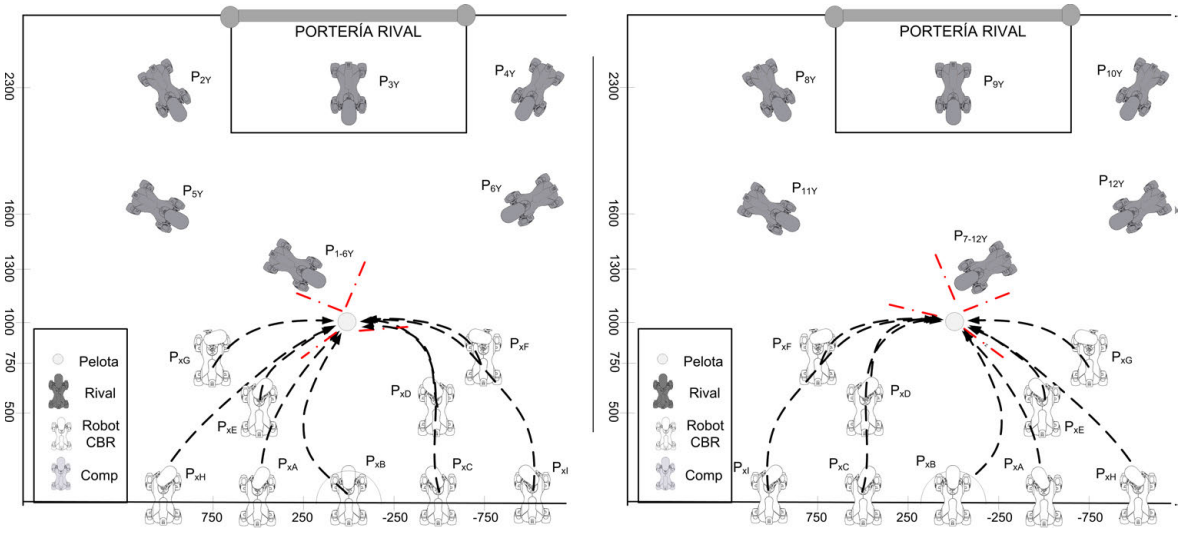


Figura 6.33: Entrenamiento con 1-2 rivales mas alejados que la pelota

En total se consideraron un conjunto de 225 situaciones de objetos en el campo, en las que básicamente se modificó el número de robots rivales y compañeros presentes,

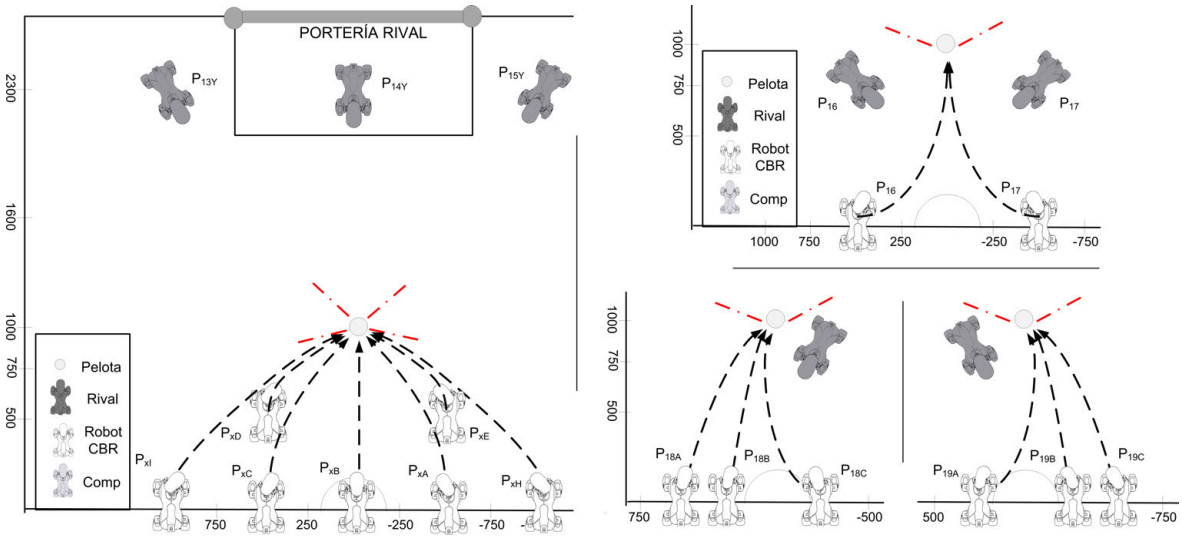


Figura 6.34: Entrenamiento con un solo rival muy cercano/muy lejano

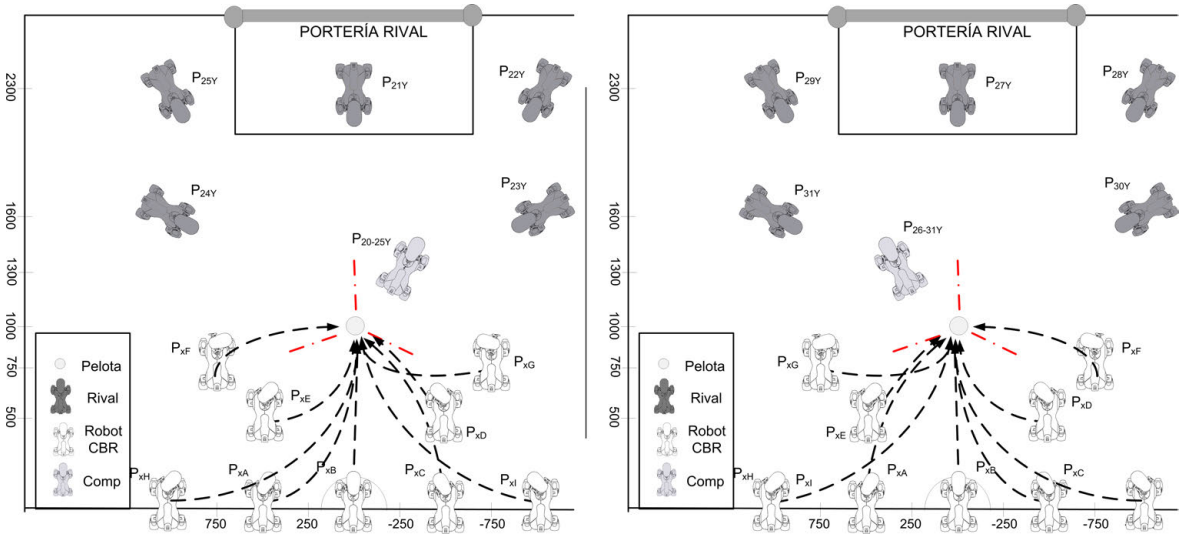


Figura 6.35: Entrenamiento con un compañero cercano y un rival

y su posición en el campo, y se entrenaron las trayectorias consideradas como más adecuadas para los objetivos del comportamiento. Las secuencias en las que aparecen simultáneamente en pantalla los tres robots posibles —dos rivales y el compañero— no se entrenaron de forma intensiva ya que, debido al reducido campo de visión de la cámara, los casos en los que aparecen y son distinguidos como objetos diferentes son muy escasos y circunscritos a posiciones muy lejanas que no afectan directamente a la conducta que se desea aprender. En todo caso, las situaciones en las que el sistema detecta mas de dos objetos identificados como "otros robots" suelen corresponderse mas bien a oclusiones y visualizaciones parciales de partes de un mismo robot, por lo que estos casos están incluidos en el conjunto de secuencias entrenadas. De cualquier forma, posibles situaciones adicionales no entrenadas en esta fase previa de aprendizaje por observación serán tratadas en la segunda fase de aprendizaje por experiencia.

Para la generación de la base de casos representativa de este comportamiento se procedió a una discretización/conceptualización con los mismos intervalos indicados en

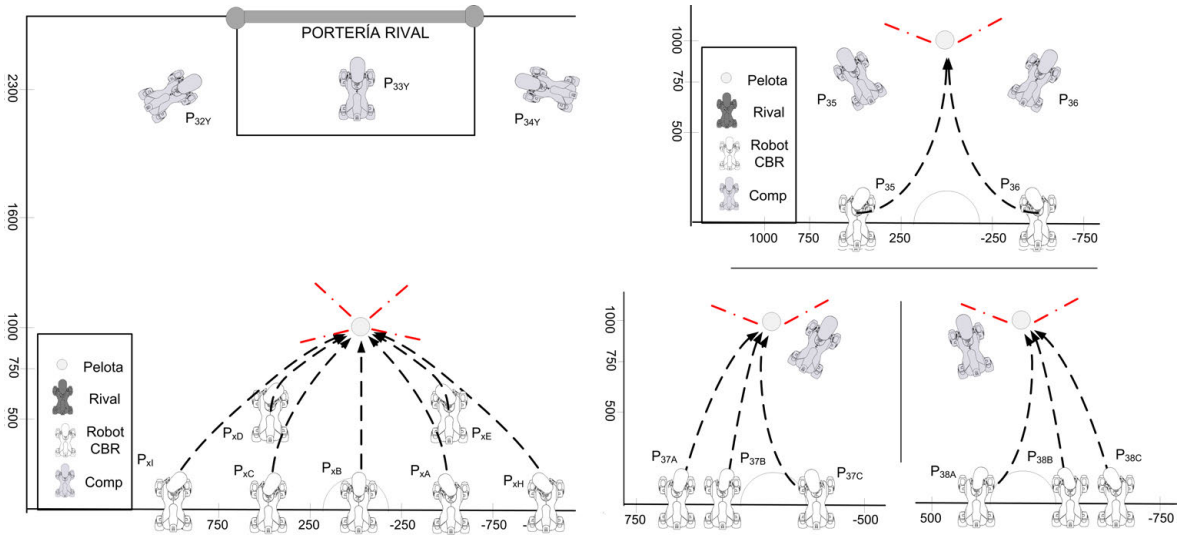


Figura 6.36: Entrenamiento con un solo compañero muy cercano/muy lejano

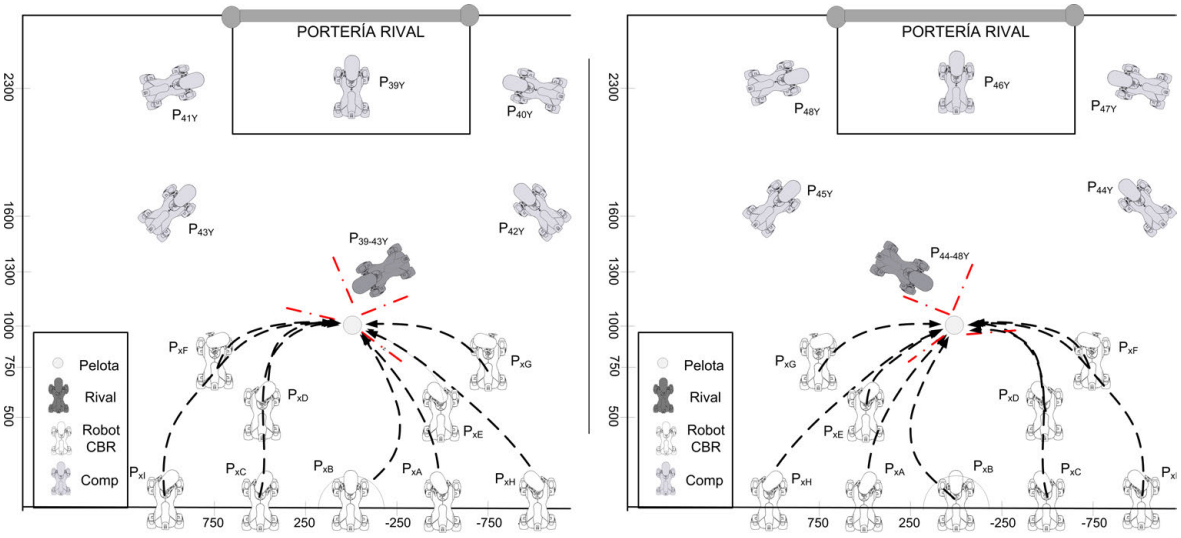


Figura 6.37: Trayectorias entrenadas para situaciones con un rival cercano y un compañero

la tabla 6.1, con lo que se obtuvo una base *CBR* inicial con aproximadamente 25000 casos.

En el transcurso de las diferentes pruebas se repitieron algunas situaciones experimentadas por el robot por lo que, tras la discretización, algunos casos almacenados presentaban la misma entrada y un pequeño número de estos mostraban una diferente salida o respuesta. Cuando esta situación ocurre se debe, bien a que la discretización no ha sido adecuada, no distinguiendo o uniendo situaciones distintas que necesitan una respuesta diferente, bien a imprecisiones o errores en el entrenamiento del robot por parte del operador humano. Tras un análisis de los casos redundantes se llegó a la conclusión de que la mayoría se correspondía con el segundo tipo, por lo que se procedió a eliminar los casos repetidos agrupándolos en uno solo cuya salida se elegiría por "voto mayoritario", es decir, utilizando la más frecuente en los casos repetidos. Un alternativa podría ser estimar y utilizar el factor de utilidad de los casos indicado en el apartado 4.4.2 del capítulo 5. Sin embargo, hemos de tener en cuenta que dicho factor

de utilidad no es sino una somera aproximación que no siempre es capaz de tener en cuenta todas las "sutilezas" de la operación realizada por el entrenador humano, por lo que nos parece más conveniente seguir el criterio de elección de la respuesta del caso por mayoría y utilizar, en todo caso, el factor de utilidad para resolver situaciones de "empate".

Tras este "filtrado", la base de casos *CBR* representativa del conocimiento asociado al comportamiento aprendido se redujo a un total de 3500 casos aproximadamente. Manualmente se añadieron unos pocos casos para el comportamiento del robot en caso de ausencia de ningún estimulo visual útil —en ese caso se procederá a girar sobre sí mismo para intentar localizar la pelota—; y se eliminaron algunos casos innecesarios captados erróneamente durante las pruebas, como detenciones momentáneas del robot o movimientos posteriores al momento de alcanzar finalmente la pelota.

Debido al elevado número de posiciones de partida empleadas en la fase de entrenamiento supervisado, para la estimación de la bondad en el comportamiento no se realizaron pruebas para todas estas posiciones, tal como se hizo en el primer conjunto (tabla 6.3). En su lugar se eligieron aleatoriamente 10 posiciones de partidas de entre todas ellas, y se aplicaron los mismos criterios de evaluación descritos anteriormente sobre 10 pruebas desde cada una de las posiciones, añadiendo la supervisión de que la trayectoria realizada por el robot fuese similar a la entrenada por el operador humano. Los resultados los podemos ver en la tabla 6.4.

	Posiciones de partida entrenadas									
	11D	18A	23I	27D	30A	32B	34B	37C	41I	42I
Puntos	100 %	100 %	100 %	95 %	100 %	95 %	90 %	85 %[1]	100 %	100 %

Tabla 6.4: Puntuaciones desde distintas posiciones entrenadas (base CBR 225 situaciones)

Del mismo modo, se eligieron aleatoriamente diversas posiciones de la pelota y uno o varios robots en el campo, en situaciones distintas a las entrenadas para analizar la respuesta del robot adaptada desde el conocimiento obtenido tras el entrenamiento. En la figuras 6.38, 6.39, 6.40, 6.41, 6.42, 6.43, 6.44, 6.45, 6.46 y 6.47, se muestran los diferentes escenarios propuestos y las trayectorias realizadas por el robot con la arquitectura de control reactivo *CBR*, para las pruebas propuestas. Los resultados "numéricos" se pueden observar en la tabla 6.5. La puntuación ya no puede tener en cuenta el "parecido" de la trayectoria realizada con la entrenada, por lo que se tomó como criterio de "éxito" que el robot llegase a obtener el control de la pelota quedando en una situación de "salida en campo abierto" o al menos con solo el robot compañero de equipo cerca. Se consideraron también erróneas las respuestas que llevaban al robot a colisionar con alguno de los otros robots del campo.

	Posiciones de partida no entrenadas									
	PNE1	PNE2	PNE3	PNE4	PNE5	PNE6	PNE7	PNE8	PNE9	PNE10
Puntos	100 %	100 %	80 %	100 %	90 %	100 %	100 %	100 %	100 %	0 %

Tabla 6.5: Puntuaciones desde posiciones aleatorias no entrenadas (base CBR 225 posiciones)

[1] La trayectoria es, en algunos casos, distinta a la entrenada

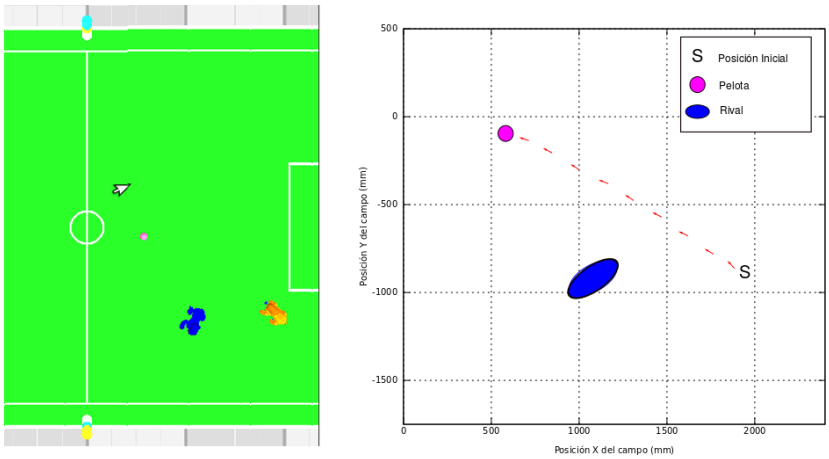


Figura 6.38: Situación Prueba-No-Entrenada1 y respuesta CBR del robot

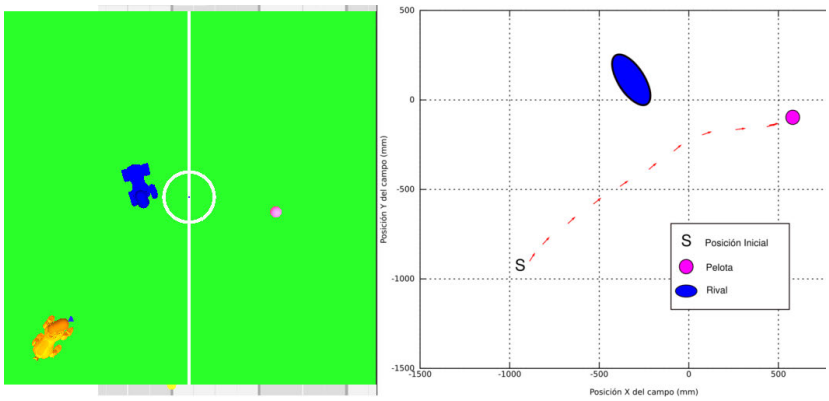


Figura 6.39: Situación Prueba-No-Entrenada2 y respuesta CBR del robot

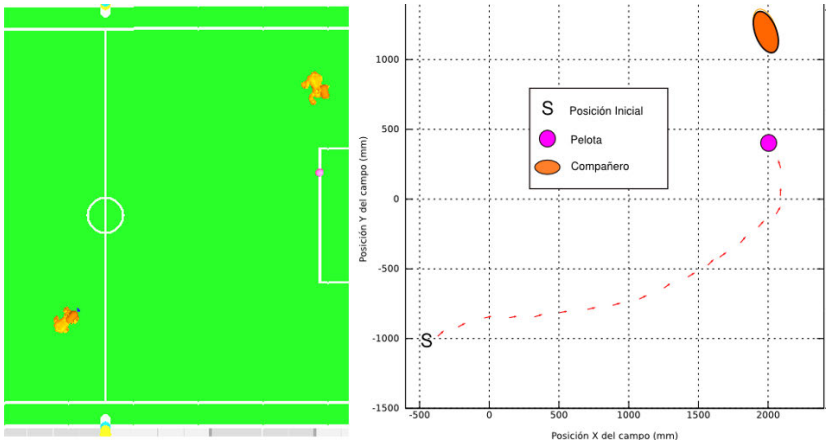


Figura 6.40: Situación Prueba-No-Entrenada3 y respuesta CBR del robot

El comportamiento fue correcto en la gran mayoría de casos considerados lográndose, en la mayoría de los casos, evitar colisiones con otros robots, y alcanzar la pelota en situaciones de "campo abierto". En alguno de los escenarios el robot realizó diversas variantes de respuesta ($PNE5$ y $PNE7$), que se consideraron correctas. Solamente en la prueba $PNE10$ se obtuvo un fracaso total ya que el robot "elegía" siempre trayectorias que le llevaban a colisionar con uno de los robots rivales. Hay que

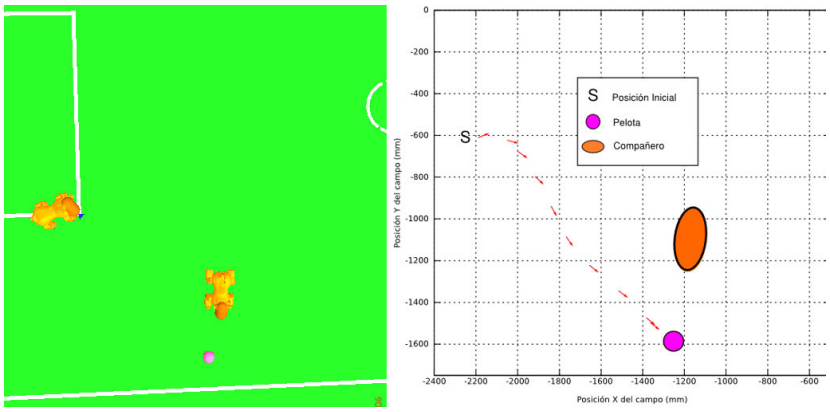


Figura 6.41: Situación Prueba-No-Entrenada4 y respuesta CBR del robot

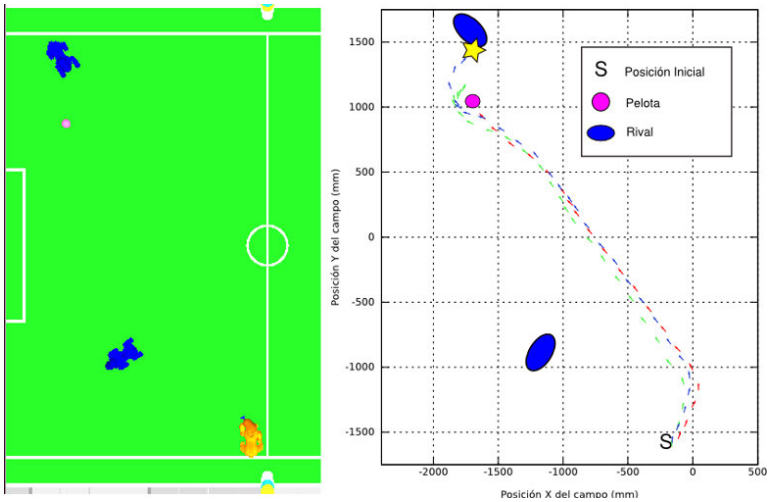


Figura 6.42: Situación Prueba-No-Entrenada5 y respuesta CBR del robot

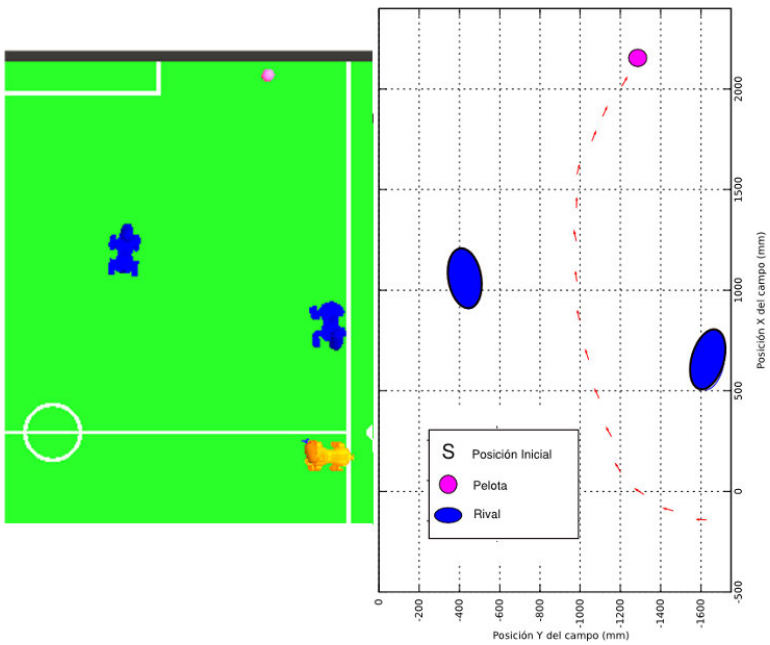


Figura 6.43: Situación Prueba-No-Entrenada6 y respuesta CBR del robot

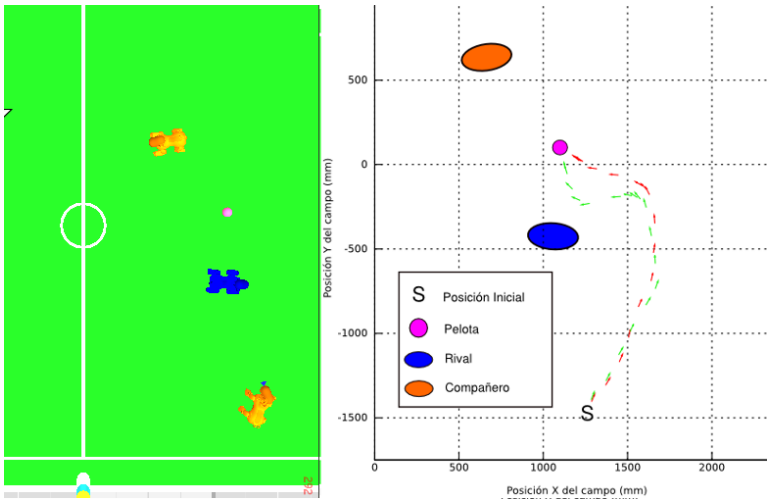


Figura 6.44: Situación Prueba-No-Entrenada7 y respuesta CBR del robot

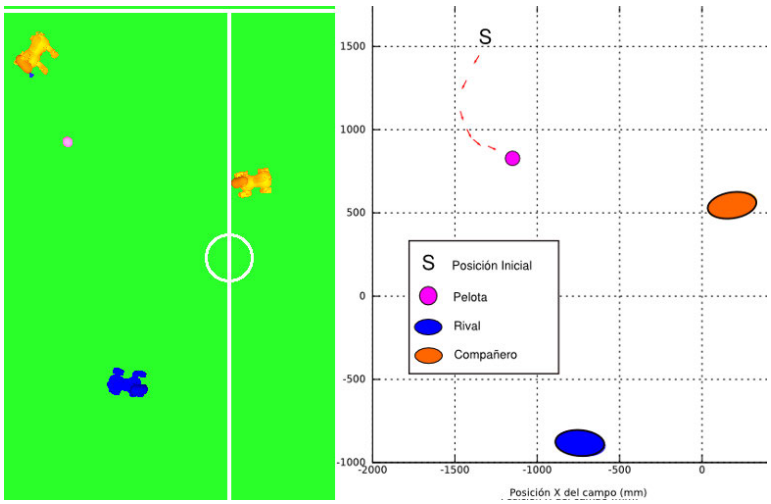


Figura 6.45: Situación Prueba-No-Entrenada8 y respuesta CBR del robot

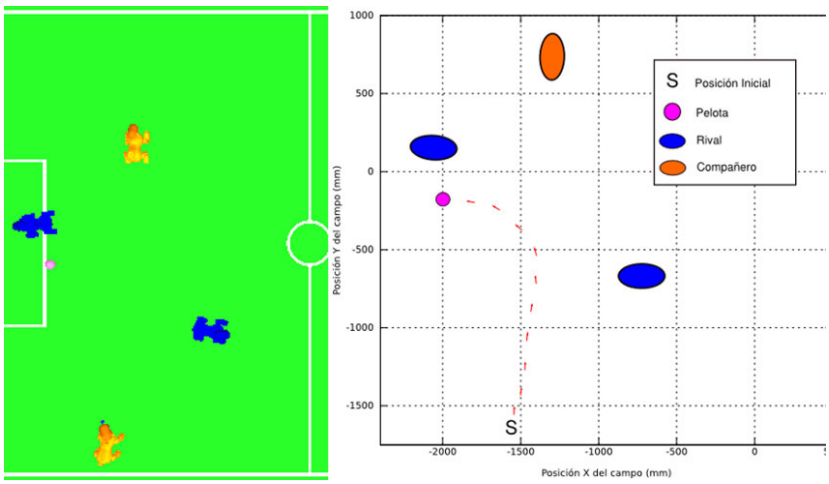


Figura 6.46: Situación Prueba-No-Entrenada9 y respuesta CBR del robot

tener en cuenta que tanto la prueba $PNE9$ como la prueba $PNE10$ se corresponden a situaciones con tres robots adicionales en el campo de juego, situaciones que no fueron

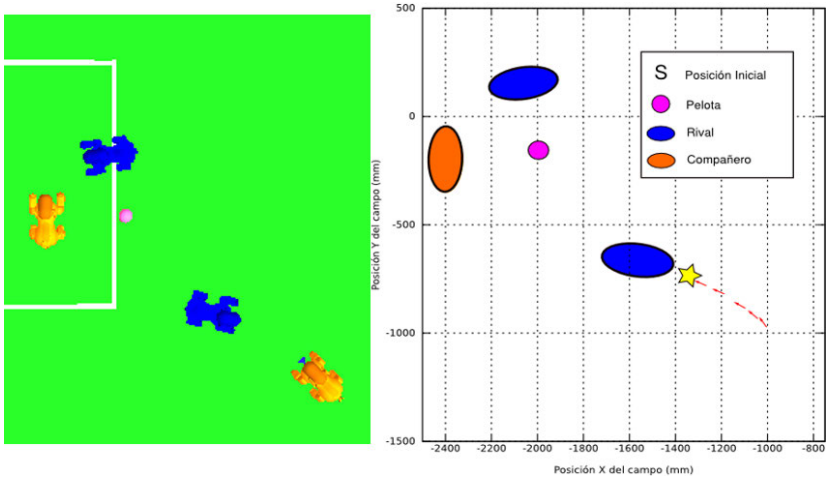


Figura 6.47: Situación Prueba-No-Entrenada10 y respuesta CBR del robot

entrenadas en absoluto en la fase de aprendizaje supervisado. Además, del análisis de la prueba $PNE10$ se determina la existencia de fenómenos visuales de "fusión" de objetos distintos en uno solo, que además llevan a situaciones "extrañas en las que uno de los objetos, parece estar incrustado en el formado por los dos robots rivales fundidos. Este tipo de fenómenos ópticos también lo sufren los seres humanos y, por lo general, suelen necesitar información complementaria de nivel más abstracto o elevado para resolver el problema. Teniendo en cuenta que estamos trabajando en un plano reactivo, lo más que se podría lograr es incorporar el conocimiento de la respuesta entrenada por un operador humano ante esa situación anómala, sin "hacerse más preguntas" acerca de su interpretación. Una opción un poco más compleja pasaría por añadir a los casos información acerca de la evolución de la posición relativa de los objetos, en cuanto a su cercanía o lejanía del robot. Este añadido supondría, una cierta pérdida del carácter reactivo puro de los comportamientos aprendidos.

En la realización de las pruebas indicadas en las tablas anteriores NO se consideró la fase de adaptación/incorporación de nuevos casos indicados en el apartado 4.4.2 del capítulo 5. En su lugar se aplicó siempre la respuesta del caso obtenido desde la base *CBR* independientemente de si la distancia del mismo con respecto al problema presentado superaba los límites. En el siguiente apartado se discutirá acerca de esta fase y se presentarán las propuestas implementadas finalmente en el ejemplo de aplicación de la arquitectura.

3.3. Incorporación de nuevos conocimientos a través de la experiencia

En el apartado 2 de este capítulo observamos como, para diseñar comportamientos sencillos en cuanto al número de parámetros o variables con influencia en el mismo, solía bastar con una fase de entrenamiento supervisado para la adquisición del conocimiento necesario para la ejecución correcta del comportamiento. La complejidad se trasladaba a la composición de dichos comportamientos sencillos para la obtención de comportamientos emergentes. Por otra parte, en el apartado 3.2 hemos comprobado como el diseño de comportamientos complejos que implican un mayor número de variables necesita de una fase de entrenamiento mas intensiva y resulta en una base

CBR con un número de casos mayor. Un entrenamiento previo que incluya todos los posibles casos contemplados con los parámetros de los casos no es deseable ni, en muchos casos, posible. Por contra, la estrategia más adecuada pasaría por entrenar únicamente, de forma supervisada, los casos más comunes o representativos, suficientes para proporcionar la respuesta del robot en la mayoría de las situaciones, ya sea por repetición o por aproximación de una situación ya entrenada. Sin embargo, siempre acabarán apareciendo situaciones no contempladas en ese aprendizaje previo, y ante las cuales la aplicación por aproximación no da buen resultado. Por este motivo es necesario, en el diseño de este tipo de comportamientos, la inclusión de una etapa de aprendizaje por experiencia en la que el robot, ante situaciones desconocidas, prueba soluciones creativas o incluso aleatorias, o combina estas con el conocimiento más parecido que encuentra entre sus experiencias previas. Los seres humanos también operamos de manera similar si bien solemos buscar analogías a un nivel más abstracto o lejano, buscando su adaptación ante el problema encontrado.

3.3.1. Elección del criterios de umbral similitud

En el apartado 4.4 del capítulo 5, se estableció un criterio para determinar cuando un caso se consideraba ”demasiado distinto“ del más cercano existente en la base *CBR*, lo cual representaría una situación suficientemente nueva como para ”probar“ una nueva solución no aprendida previamente. Ese criterio, expresado por la ecuación 5.3, tiene en cuenta la discretización/conceptualización previa de la información del caso, y viene determinado por un máximo número de saltos en las categorías conceptuales correspondientes a un parámetro del caso en particular; a un objeto completo; o todo el caso al completo. Para las pruebas experimentales presentadas, dichos umbrales se han escogido de forma heurística, eligiendo los valores de 2 ”saltos“ en el caso de las componentes individuales; 3 ”saltos“ para un objeto: y 5 ”saltos“ para el caso al completo. En la elección de estos valores se ha tenido en cuenta el número de intervalos de discretización considerados para los parámetros: por lo general, para unos mayores niveles de discretización se deberían admitir unos umbrales más altos. Por otra parte, es posible que cada parámetro individual debiese tener su propio umbral, adecuado a sus intervalos de discretización si estos son distintos de un parámetro a otro pero, por simplificar, se ha elegido considerar un umbral único para todos los parámetros. De cualquier forma este es un tema que se deberá investigar en el futuro de forma mas exhaustiva para tener un mejor criterio que determine cuando es adecuado llevar a cabo una adaptación de la solución extraída a partir de la experiencia previa. En la figura 6.48 podemos ver un ejemplo de dos situaciones que son consideradas suficientemente parecidas, según la discretización indicada en la tabla 6.1, como para poder aplicar respuesta obtenida en la situación entrenada, (izquierda) a la nueva situación encontrada durante la fase de operación (derecha). En estas situaciones se observa que la posición del conjunto en el campo es distinta, algo que no es considerado por el comportamiento al estar éste centrado en la disposición relativa de los elementos. Además, tanto el robot rival como la pelota aparecen en posiciones ligeramente más cercanas, pero que son consideradas dentro del mismo intervalo de discretización y asociadas al mismo concepto de ”área media“ y ”area pequeña“, respectivamente. Por otra parte los robots presentan orientaciones distintas, algo que el sistema actual de pre-procesamiento de información no es capaz de distinguir. En consecuencia, ambas situaciones provocan una misma respuesta de acercamiento hacia la pelota, escorando

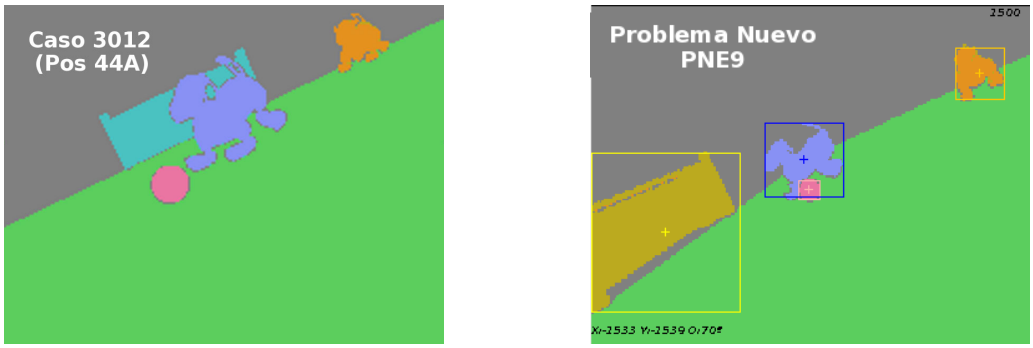


Figura 6.48: Casos/situaciones considerados como similares en el comportamiento diseñado de acercamiento a la pelota

el robot hacia la izquierda en su acción de alcanzar la pelota para no acabar apuntando hacia el robot rival, sino hacia campo abierto.

3.3.2. Generación de vector PF para adaptación de un caso no similar

Una vez que se determina que es necesario realizar una adaptación de la experiencia, es necesario establecer el mecanismo de adaptación o generación de un nuevo conocimiento. Como ya hemos comentado previamente, los seres humanos cuentan con una capacidad de analogía a un nivel de abstracción más alto, que les permite adaptar o probar experiencias procedentes de campos o ámbitos aparentemente muy alejados. En la arquitectura reactiva propuesta no contamos con esta capacidad, por lo que hemos optado por utilizar algoritmos clásicos de navegación reactiva de entre los cuales hemos elegido la generación de un vector alternativo de movimiento mediante *PFA*. Este vector se combinaría de alguna forma con el obtenido como acción de respuesta correspondiente al caso más cercano a la situación presentada, extraído de la base *CBR* del comportamiento. Para la generación del vector *PF* hemos considerado, en el comportamiento que estamos usando como ejemplo, que la pelota actúa como un atractor y los demás robots —rivales y compañeros— como repulsores:

$$\vec{V}_{PF} = \sum \vec{V}_{Atr_i} + \sum \vec{V}_{Rep_j} \tag{6.5}$$

donde los vectores de repulsión tienen sentido opuesto y los de atracción el mismo sentido del avance.

Tradicionalmente, la magnitud de cada vector suele venir dada por una función exponencial negativa de la distancia a obstáculos y repulsores:

$$V_{atr} = e^{-\alpha_A tr_i * |dAtr_i|} \quad V_{rep} = -e^{-\beta_R ep_j * |dRep_j|} \tag{6.6}$$

donde $dAtr_i$ y $dRep_j$ son las distancias a los objetos que actúan como atractores y repulsores; y α_{Atr_i} y β_{Rep_j} son parámetros heurísticos que determinan la rapidez de la caída del campo de potencial.

Sin embargo, en nuestro caso resulta complejo estimar la distancia y el ángulo a partir de únicamente la imagen de la cámara de a bordo del robot. Por ello, realizaremos una estimación aproximada del ángulo a partir de la posición de la cabeza respecto a la dirección de avance, ϕ_{head} , y del ángulo del centro de los objetos visualizados respecto del punto inferior medio de la imagen de cámara, ϕ_{img}, (figura 6.49), quedando el

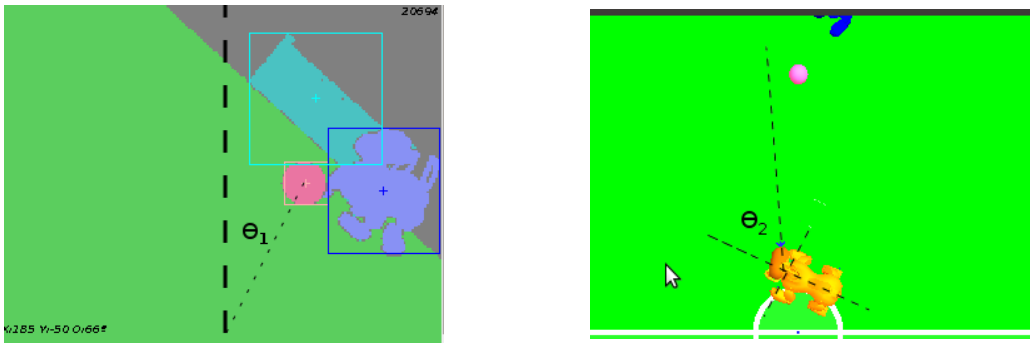


Figura 6.49: Estimación de ángulo con los objetos, a partir de la imagen de cámara

ángulo respecto del objeto como la suma de los dos anteriores, ϕ_{obj}= $\phi_{head} + \phi_{img}$. Esta estimación, aunque burda, es suficiente para nuestros objetivos.

En cuanto a la distancia, usaremos en su lugar el área del objeto visualizado como elemento determinante de la magnitud de los vectores atractores y repulsores. De esta manera, a mayor área se considerará una menor distancia al objeto.

En el caso de los atractores usaremos funciones de la forma $e^{-\alpha_{Atr_i} * |Aa_i|}$ para la cual la atracción es mayor cuanto menor es el área visible, Aa_i, con valores extremos en 1 para $Aa_i = 0$ y en $e^{-\alpha_{Atr_i}}$ para $Aa_i = 1$. Para los repulsores, hemos elegido funciones de la forma $1 - e^{-\beta_{Rep_j} * |Ar_j|}$ en las que la repulsión aumenta con el área, Ar_j , con valores extremos en $1 - e^{-\beta_{Rep_j}}$ para $Ar_j = 1$ y en 0 para $Ar_j = 0$. Finalmente, hemos de tener en cuenta que el área aumenta o disminuye de forma aproximadamente cuadrática al acercarse o alejarse de los objetos. Por ello, y para hacer más pronunciada la repulsión a distancias, emplearemos un Aa_i correspondiente a la raiz cuadrada del área visible normalizada para el caso de los objetos atractores. No obstante, usaremos un Ar_j correspondiente al área normalizada de los objetos repulsores, para hacer más pronunciado el efecto de repulsión en áreas cercanas a los objetos.

$$V_{atr} = e^{-\alpha_{Atr_i} * |Aa_i|} \quad V_{rep} = -(1 - e^{-\beta_{Rep_j} * |Ar_j|}) \tag{6.7}$$

Por tanto, la magnitud del vector PF vendrá dada por la proyección vectorial de las magnitudes indicadas en la ecuación 6.7, sobre los vectores directrices atractores y repulsores respectivamente (figura 6.50).

$$V_{PF} = \sum e^{-\alpha_{Atr_i} * |Aa_i|} \cdot \vec{V}_{Atr_i} - \sum -e^{-\beta_{Rep_j} * |1 - Ar_j|} \cdot \vec{V}_{Rep_j} \tag{6.8}$$

A menudo existe un único objeto atractor en un escenario con múltiples objetos repulsores, por lo que, para evitar la anulación de dicho atractor, se suele ponderar por un factor k_{rep}, proporcional al número de repulsores considerados:

$$V_{PF} = k_{rep} * e^{-\alpha_{Atr_i} * |Aa_i|} \cdot \vec{V}_{Atr_i} - \sum^{k} -e^{-\beta_{Rep_j} * |1 - Ar_j|} \cdot \vec{V}_{Rep_j} \tag{6.9}$$

donde k_{rep} es el número de componentes repulsores. En el escenario de pruebas escogido es poco probable que aparezcan mas de dos repulsores que influyan en el robot al mismo tiempo que el único atractor existente. Por este motivo obviaremos inicialmente este factor, dándole un valor de k_{rep}= 1.

En las pruebas realizadas hemos elegido, de forma heurística, un mismo valor de 2 para los α_{Atr_i} y β_{Rep_j}. Es necesario indicar que el vector $\vec{V_{PF}}$ finalmente obtenido

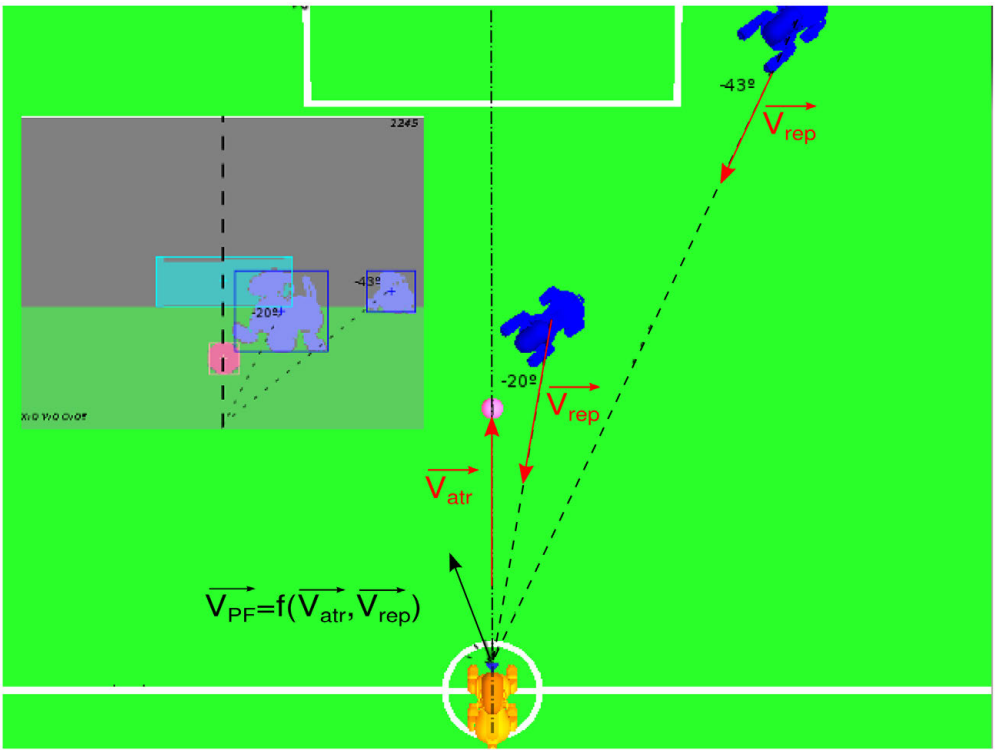


Figura 6.50: Generación vector PF para adaptación de la respuesta de un caso

es unitario, por lo que si la respuesta de los casos que determinan el comportamiento resulta en una cadena de vectores, habrá que escalar $\vec{V_{PF}}$ según el tamaño de dicha cadena.

3.3.3. Combinación entre vector PF y vector del caso recuperado

Una vez obtenido un vector $\vec{V_{PF}}$ se deberá generar la respuesta del robot al nuevo caso presentado como una combinación de dicho vector y del obtenido en el caso recuperado de la base. Recordemos que el caso recuperado ha sido considerado como poco adecuado y necesitado de una adaptación, que se va a realizar a través de $\vec{V_{PF}}$. En esa combinación, se dará mayor o menor peso al vector de respuesta del caso recuperado, $\vec{V}_{CBRCase}$, en función de "como de poco adecuado" se le considere tal como se vio en la ecuación 5.5 del apartado 4.4.1 del capítulo 5. De entre los criterios comentados se prefirió finalmente determinar la aportación de cada componente en función de la similitud o distancia entre el problema y el caso recuperado, pero considerando la variante, qué establecía un 50 % de aportación para el umbral mínimo de adaptación (6 saltos para el conjunto del caso; 4 para un objeto completo; y 3 para un componente particular del caso) y un límite de justo el doble (12, 8, y 6), a partir del cual considerar una única aportación del vector *PF*. Entre ambos valores, la variación de la aportación será de tipo lineal.

$$\vec{V}_{adapt} = \vec{V}_{PF} \cdot d_{norm} + \vec{V}_{CBRCase} \cdot (1 - (d_{norm})) \tag{6.10}$$

con

$$d_{norm} = \begin{cases} \max\limits_{i}(\dfrac{(S_i - S_{i_{min}}) \cdot 0.5}{S_{i_{max}} - S_{i_{min}}} + 0.5) & S_i < S_{i_{max}}, \forall S_i \in \{S_{caso}, S_{obj}, S_{comp}\} \\ 1 & S_i \succeq S_{i_{max}}, \exists S_i \in \{S_{caso}, S_{obj}, S_{comp}\} \end{cases} \quad (6.11)$$

donde S_i indica el número de saltos encontrados en los respectivos componentes del caso completo, S_{caso}; de un objeto, S_{obj}; y de un parámetro del caso, S_{comp}. Y $S_{i_{min}}$, $S_{i_{max}}$ son, respectivamente, los umbrales mínimo para adaptar el caso, y máximo para aplicar solamente la salida generada por *PFA*, de los correspondientes discriminantes.

Se probaron otros intervalos de aportación de $\vec{V_{PF}}$ en diferentes pruebas ($[0, 100\,\%]$,$[25\,\%, 100\,\%]$,$[75\,\%, 100\,\%]$). En los que partían de niveles inferiores, el vector *PF* no proporcionaba ninguna aportación en un porcentaje alto de casos. Por otra parte, en el rango $[75\,\%, 100\,\%]$, el control estaba demasiado influenciado por el vector *PF*, lo cual resultaba perjudicial en las condiciones en las que éste funciona peor (cercanía del objetivo, obstáculos y objetivos cercanos,...). Finalmente se decidió optar por una variación de la aportación entre $[50\,\%, 100\,\%]$.

Si el resultado de la combinación da lugar a un vector con magnitud superior a la máxima considerada para el movimiento del robot, se truncará su valor a dicho máximo. En el ejemplo que estamos considerando, en el que la respuesta del robot viene dada por un cadena de tres vectores unitarios, la máxima magnitud en cada componente del vector resultante será de 3. Como resultado de la composición de vectores también se puede obtener un vector completamente nulo. Esta respuesta no está contemplada en el diseño de comportamientos de tipo reactivo que estamos planteando, por lo que no es una respuesta aceptable. Como alternativa, consideraremos la aplicación de la última respuesta ejecutada como método para "escapar" de la posición conflictiva.

Un problema adicional podría venir de la aparición de mínimos locales en el movimiento del robot, dados por la ejecución de órdenes que temporalmente, en su conjunto, se anulan entre sí. Poco se puede hacer ante este tipo de problemas trabajando únicamente en el campo reactivo si bien, dada la naturaleza inexacta del movimiento del robot cuadrúpedo, hay una tendencia a salir de estos mínimos tarde o temprano. No obstante, en versiones más avanzadas de la arquitectura que incluyan componentes deliberativos se deberían contemplar y evitar este tipo de inconvenientes.

3.3.4. Elección de parámetros para función de utilidad/eficiencia

Como se indicó en el apartado 4.4.2 del capítulo 5, resulta importante asociar a cada caso almacenado en la base de conocimientos del comportamiento, un factor de utilidad o eficiencia que resulta de la aplicación de la acción asociada al caso y que viene dada por la evaluación de una serie de factores relacionados con la suavidad y seguridad en la realización del comportamiento, y con los objetivos del mismo. En la implementación práctica de la función de utilidad para el comportamiento que estamos diseñando como ejemplo inicialmente dimos un mismo peso, $\frac{1}{3}$, a los tres factores, K_{suav},K_{seg}, y K_{obj}. Para la elección de los parámetros que determinan las exponenciales de evolución de los tres factores, vamos a considerar una estimación de resultados "deseados" en diferentes valores para cada uno de ellos:

- Para la suavidad, el ángulo máximo de variación es de 180°, y el mínimo de 0°. Además, al trabajar con salidas compuestas por una cadena de 3 vectores, la

mínima variación —aparte de 0°— es de 15°. Teniendo en cuenta la velocidad de movimiento y giro del robot, hemos considerado una caída de la aportación del 50 % para una variación de 45°, que se consigue aproximadamente con un valor C_{suav}= 2.5. Así, la mínima variación de 15° supondrá una aportación del 81 % del máximo; y para una variación de 90° la aportación estaría por debajo del 30 %.

- Para la seguridad estableceremos, a través de pruebas, un factor de aportación del 50 % para situaciones en las que el área ocupe el tamaño suficiente de la pantalla como para que aún sea posible evitar el obstáculo en un movimiento de avance-giro simultáneo. Mediante pruebas, estimamos que este área está aproximadamente entre los valores $0.25 - 0.30$, cuando el objeto está situado justo enfrente de nuestro robot. Si queremos incluir también el ángulo del robot-obstáculo, estimamos un área de 0.22 para un ángulo aproximado de 22.5°. Para la expresión que estamos buscando el ángulo se normaliza al máximo posible: teniendo en cuenta que el PAN máximo del robot es de 60°, y que el máximo ángulo visible en la imagen está por debajo de 90°, una normalización conservadora se haría con respecto a 150°. En estas condiciones obtendríamos un valor de $C_{seg} = 0.69$ para ajustar la expresión 5.8 a los valores buscados; y un valor $C_{seg} = 1.01$ si empleamos la expresión 5.9.

- En cuanto al objetivo del comportamiento, en este caso en particular, se trata de avanzar hacia la pelota y llegar hasta ella de forma alineada pero, al mismo tiempo, en una situación ventajosa respecto al resto de robots situados en el campo. Como la inclusión de la diferente casuística es compleja, y depende de las estrategias del entrenador —por ejemplo, en unos casos se puede decidir que no es problemático alcanzar la pelota encarando a un robot compañero—, hemos centrado el factor de objetivo únicamente en la pelota. Por ello, la expresión utilizada puede tener una forma similar a la empleada para el concepto de seguridad, pero aumentando el nivel de aportación cuanto mas cerca de la pelota, y más centrado respecto a ella, se encuentre el robot. Una posible expresión sería la indicada en la ecuación 6.12.

$$F_{aprox-pel} = e^{-C_{ap} \cdot \frac{1-tam_{pel}}{tam_{pel} \cdot (1-|\ \theta_{rob-pel}|)}} \tag{6.12}$$

  En esta expresión, la aportación a la utilidad o eficiencia aumentaría conforme el tamaño de la pelota, tam_{pel}, aumenta y/o el ángulo entre ambos robot y pelota disminuye, obteniéndose una aportación máxima a la eficiencia para la pelota ocupando completamente la pantalla. Para un encaramiento directo por la pelota, el factor crecería de forma casi lineal. Valores mayores de $\theta_{rob-pel}$, provocarían que se necesitase estar cada vez más cerca del obstáculo para que se produzcan cambios apreciables de eficiencia. El parámetro C_{ap} se puede ajustar heurísticamente, estimando el valor deseado de este factor para determinadas situaciones de distancia/área de la pelota, y ángulo respecto a la misma. Así, en este ejemplo de diseño hemos considerado una aportación del 10 % para un ángulo de 60° —0.4 normalizado— y un área normalizada de 0.05 %, lo que nos permite obtener un valor C_{ap}= 0.05

Finalmente, la expresión que usaremos para estimar una utilidad/eficiencia en las acciones realizadas por la expresión vendrá dada, para el ejemplo de comportamiento que estamos diseñando, por la ecuación 6.13.

$$\begin{aligned}\eta =& \frac{1}{3} \cdot e^{-2.5 \cdot |\Delta_{dir}|} \\ &+ \frac{1}{3} \cdot (1 - e^{-0.69 \cdot (1 - tam_{obst}) \cdot |\ \theta_{rob-obst}|}) \\ &+ \frac{1}{3} \cdot e^{-0.05 \cdot \frac{1 - tam_{pel}}{tam_{pel} \cdot (1 - |\ \theta_{rob-pel}|)}}\end{aligned} \tag{6.13}$$

donde todas las variables estarán normalizadas en $[0, 1]$, estando Δ_{dir} normalizado respecto de $[0, 180°]$; y $\theta_{rob-obst}$, $\theta_{rob-pel}$ normalizados respecto de $[0, 150°]$.

En la figura 6.51 podemos ver un ejemplo de la evolución del factor de utilidad, y sus diferentes factores componentes, en la realización de la prueba mostrada en la figura 6.46, según la ecuación 6.13:

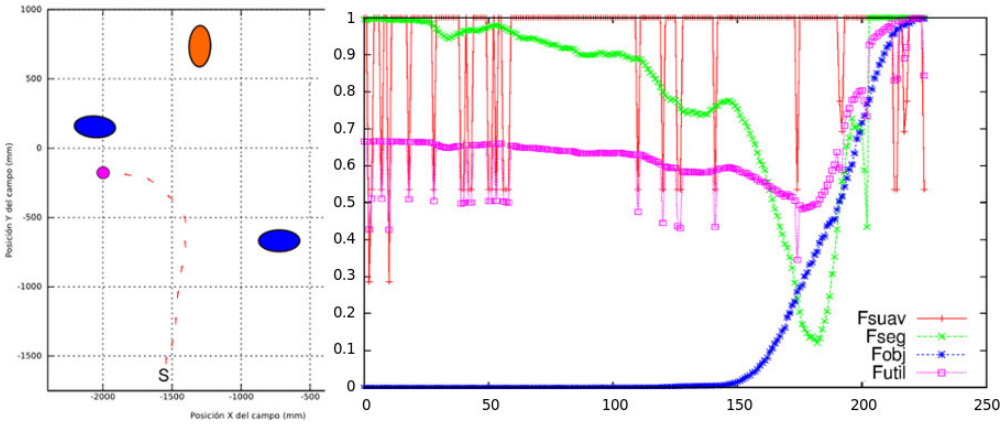


Figura 6.51: Evolución del factor de utilidad para la Prueba-No-Entrenada9

Hay que destacar que, al proceder la respuesta *CBR* del robot de diferentes casos procedentes de diversas pruebas independientes, se producen ciertas oscilaciones importantes en el factor de suavidad. La evolución de los factores de objetivo y seguridad parece bastante razonable y acorde al comportamiento del robot durante la prueba. En el caso del factor de seguridad, los picos de bajada de este factor se corresponden a situaciones de encaramiento directo con un robot-obstáculo cuando hay cierta cercanía al mismo. En esta prueba no fue necesario acudir a la adaptación de la respuesta de casos recuperados, con lo cual no hubo incorporación de nueva información procedente de la experiencia.

3.3.5. Pruebas de adaptación-incorporación de casos

Para comprobar la validez del esquema de adaptación e incorporación de nuevos casos, de acuerdo a lo expuesto en los párrafos anteriores, se realizaron diversas pruebas en el escenario de trabajo.

Base de conocimientos casi vacía

Para el primer conjunto de pruebas, se partió de una base *CBR* de conocimientos casi vacía, incluyendo únicamente tres casos correspondientes a un estado de parada del robot al alcanzar la pelota, un giro sobre sí mismo en caso de no visualizar ningún objeto, y una situación de avance recto ante una visualización de la pelota a distancia

media. El robot, en las diferentes pruebas, deberá construir su base de conocimientos a partir de la combinación de esa poca información existente junto con las respuestas de adaptación generadas a través de *PFA*, siempre y cuando dichas respuestas se consideren con una utilidad suficiente. En las figuras 6.52, 6.53, 6.54 y 6.55, podemos ver las trayectorias realizadas en estas condiciones para diferentes escenarios con diversos grados de complejidad.

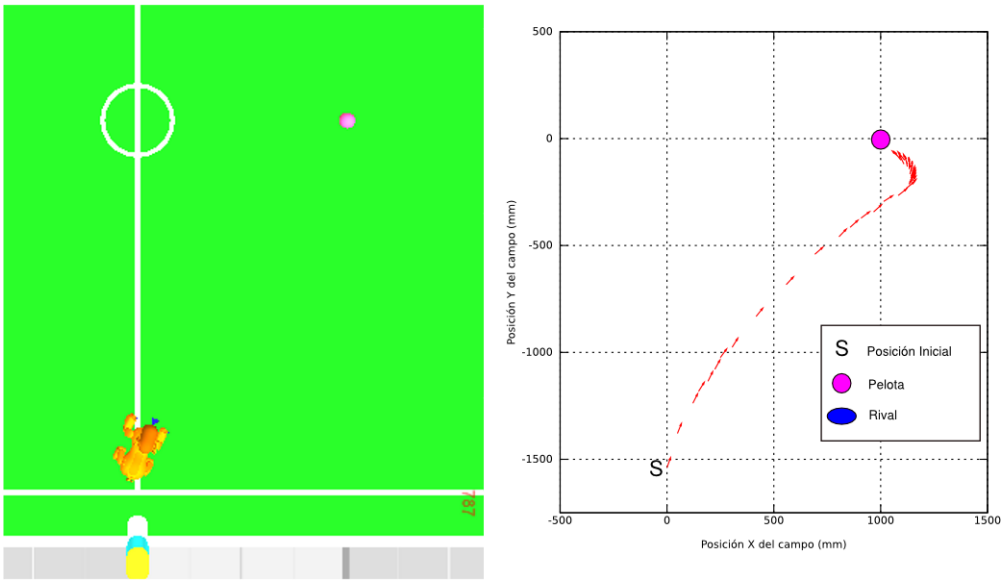


Figura 6.52: Trayectoria para una base CBR inicial mínima (CBRMin1)

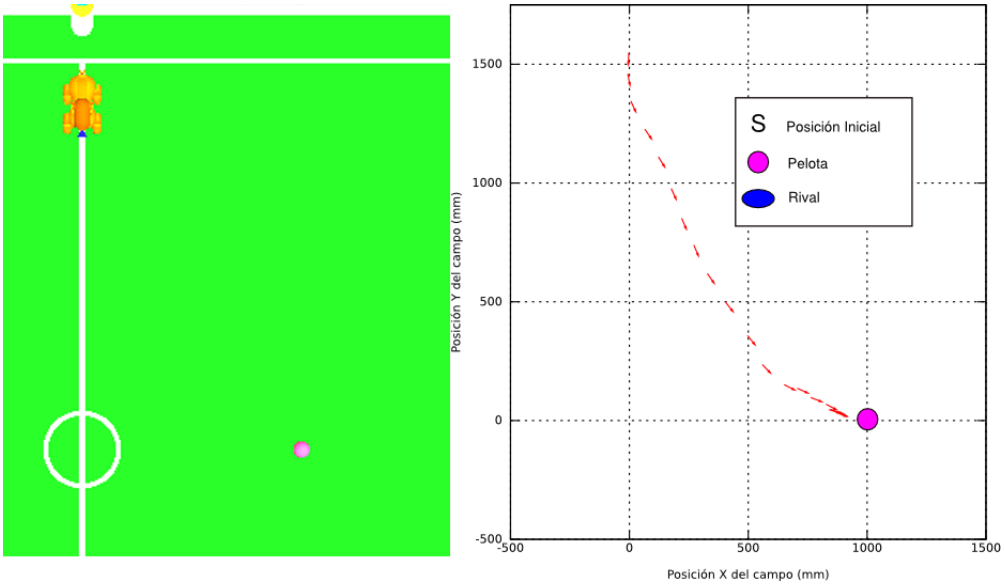


Figura 6.53: Trayectoria para una base CBR inicial mínima (CBRMin2)

El número de casos adquiridos varía ligeramente en repeticiones de las mismas pruebas, pero suele oscilar entre los 8 y 15 casos. Algo parecido ocurre con el rechazo de incorporación de nuevos casos por no presentar mejoras en el factor de utilidad.

Hay que tener en cuenta que la casi total ausencia de conocimiento previo lleva a una operación muy basada en campos de potencial, pero con la posibilidad de

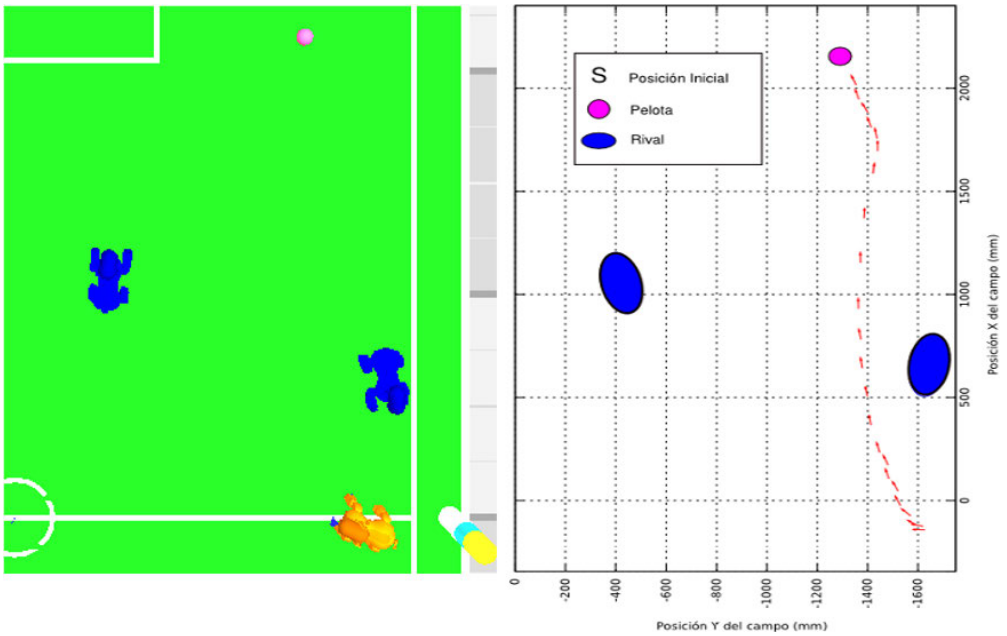


Figura 6.54: Trayectoria para una base CBR inicial mínima (CBRMin3)

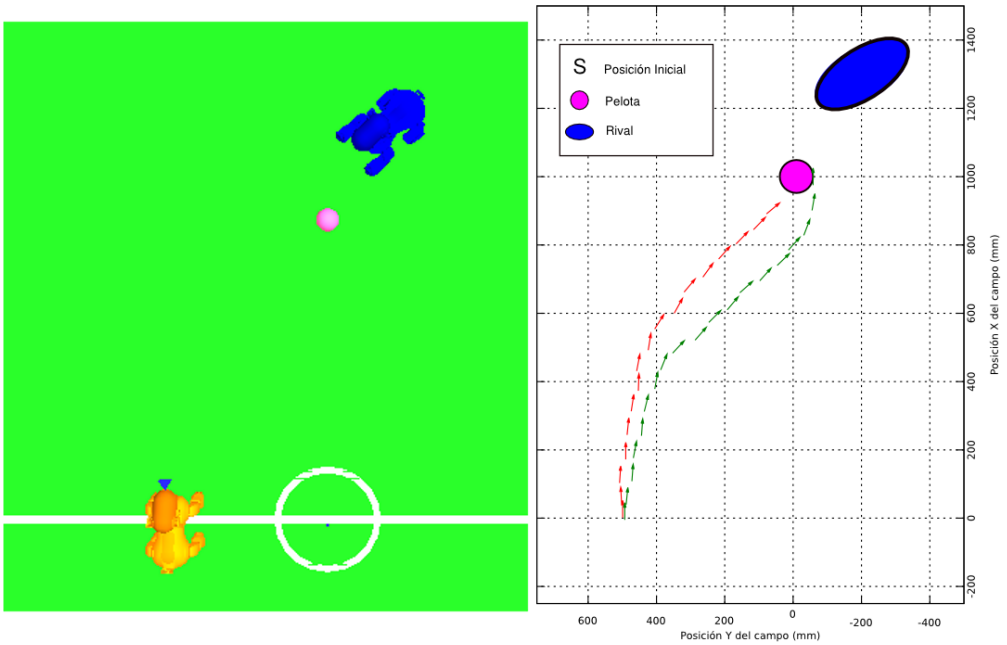


Figura 6.55: Trayectoria para una base CBR inicial mínima (CBRMin4)

ir incorporando, paulatinamente, aquellas situaciones cuya aportación se considere beneficiosa. Los comportamientos demostrados en las diferentes pruebas suelen ser bastante correctos en cuanto a la evitación de obstáculos individuales y acercamiento al objetivo. Sin embargo, se detectan algunos problemas en las situaciones cercanas al objetivo final, derivadas de la reducción del potencial de atracción y el centrado del campo de visión en únicamente dicho objetivo. Así, en algunas de las pruebas, el robot acaba realizando un acercamiento tipo "espiral" en zonas muy cercanas a la pelota. Además, existen situaciones conflictivas muy complejas de resolver mediante un sistema *PFA*, como aquellas en las que obstáculo y objetivo ocupan posiciones

muy cercanas, que pueden llevar no solo a comportamientos de respuesta incorrectos, sino a otros errores como repeticiones cíclicas de movimiento. Por todo ello, se hace patente la necesidad de la fase previa de entrenamiento por observación que hemos propuesto para nuestro sistema *CBR*. Cuanto más completa sea esta etapa, respecto a las situaciones que se espera que el robot experimente durante su funcionamiento, mejor estará caracterizado el comportamiento, y también se favorecerá la fase de adaptación e incorporación de nuevos conocimientos.

Escenario simple: acercamiento a la pelota sin otros robots

Como siguiente experimento para determinar la eficacia del esquema de adaptación propuesto, volvemos al escenario sencillo mostrado en la figura 6.32, en la que el robot se entrena para ser capaz de aproximarse a la pelota en ausencia de rivales en el campo. Sobre la base de conocimiento formada se realizó, en primer lugar, un diezmado aleatorio de un 50 % de los casos de la base, de forma que ésta pasó de unos 100 casos a 50. En estas condiciones, se volvieron a repetir las pruebas desde las posiciones de partida del entrenamiento, para obtener la tabla 6.6, similar a la 6.3. Se observó un cierto empeoramiento en los resultados, de acuerdo a los criterios de calidad ya vistos pero, aún así, el sistema demostró una gran robustez frente a una degradación distribuida de la información de la base. En todas las pruebas el robot fue capaz de llegar cerca de la pelota, fallando solo en algunos casos, en cuanto a su posicionamiento para obtener su control.

	Posiciones de partida						
	(0,0)	(0,500)	(0,-500)	(850,400)	(850,-400)	(800,100)	(800,-100)
Puntos	85 %	90 %	65 %	90 %	85 %	100 %	70 %
	Otras posiciones						
	(-340,100)	(-200,-150)	(-400,-300)	(400,300)	(-700,-700)	(-700,-500)	
Puntos	70 %	85 %	90 %	85 %	80 %	80 %	

Tabla 6.6: Puntuaciones desde distintas posiciones (base CBR pruebas S0 a S6) con diezmado aleatorio del 50 %, y sin adaptación por experiencia

Se repitieron las pruebas anteriores activando el mecanismo de adquisición de nuevos casos a través de la experiencia, con la obtención de los resultados de la tabla 6.7.

	Posiciones de partida						
	(0,0)	(0,500)	(0,-500)	(850,400)	(850,-400)	(800,100)	(800,-100)
Puntos	90 %	100 %	100 %	95 %	85 %	100 %	85 %
Casos Adq.	**0** - 1 %	**1** - 2	**1** - 2	1	0 - **1** - 2	0 - **1** - 2	0 - **1** - 2
	Otras posiciones						
	(-340,100)	(-200,-150)	(-400,-300)	(400,300)	(-700,-700)	(-700,-500)	
Puntos	85 %	85 %	95 %	100 %	90 %	90 %	
Casos Adq.	0 - **1**	0 - **1**	0 - **1** - 2	0 - **1** - 2	0 - **1**	0 - **1** - 2	

Tabla 6.7: Puntuaciones desde distintas posiciones (base CBR pruebas S0 a S6) con diezmado aleatorio del 50 %, y con adaptación por experiencia

El comportamiento, sin llegar a los niveles obtenidos con la base completa, mejora al funcionamiento sin algoritmo de adaptación por experiencia. Los casos adicionales añadidos tras la adaptación son escasos, algo lógico teniendo en cuenta la poca duración de las secuencias de prueba, si bien se realizaron algunas adaptaciones

adicionales, que no fueron incorporadas al no suponer mejoras en el factor de utilidad del comportamiento. Los casos adquiridos son nuevamente distintos, adaptados a la información que se detecta como ausente en la prueba correspondiente. Destacamos que muchas de las situaciones de éxito parcial —alcanzar la pelota sin control de la misma— se correspondieron a pruebas en las que no se adaptaron nuevos casos.

Dentro de este mismo escenario de pruebas se planteó un nuevo tipo de diezmado de la base eliminando, ésta vez, secuencias completas de entrenamiento. De esta forma, la ausencia de conocimiento no es tan aleatoria ni distribuida, sino que se concentra en situaciones que representan situaciones novedosas no de forma puntual, que se pueden solucionar merced a la capacidad reactiva del sistema, sino a más largo plazo. Para ello, se eliminaron sucesivamente las secuencias de entrenamiento S2, S4, y S6, correspondientes al acercamiento del robot desde la zona derecha, a distintas distancias. En las figuras 6.56, 6.57, y 6.58, podemos observar la trayectoria ejecutada por el robot tanto en el caso de incorporación de nuevos conocimientos a través de la experiencia, como en el de simple aplicación de la respuesta del caso más parecido, independientemente del nivel de similitud del mismo.

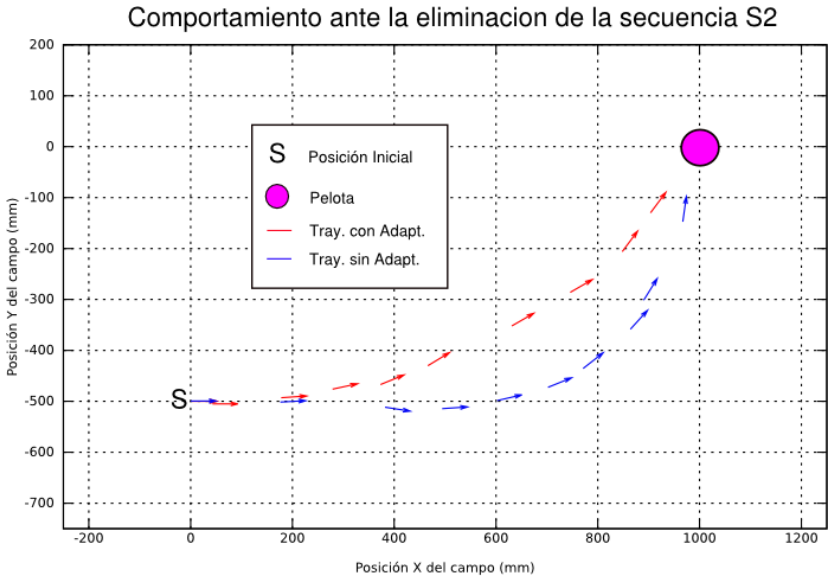


Figura 6.56: Trayectoria desde la posición S2 (base CBR pruebas S0-S1,S3-S6) con y sin adaptación por experiencia

Tras la ejecución de las pruebas, y como consecuencia de la adaptación, se produjo un crecimiento de las bases CBR de un 3 %, 7 %, y 13 %, en las pruebas mostradas en las figuras 6.56, 6.57, y 6.58, respectivamente. Se observa como la incorporación de casos a través de la experiencia, compensa la carencia de información en la base *CBR*, permitiendo un mejor comportamiento de la versión de *CBR* con adaptación. La figura 6.58 se corresponde a una base en la que el robot no tiene ningún conocimiento acerca de que debe hacer cuando está situado a la derecha de la pelota, por lo que en el caso sin adaptación acaba realizando una trayectoria que lo aleja completamente de la misma. En el caso de CBR con adaptación, si bien el robot no es capaz de controlar completamente la pelota —debido a los problemas de *PFA* en situaciones cercanas al objetivo—, al menos es capaz de mantener su trayectoria hacia la misma, gracias a la adaptación. Con el añadido de cierto conocimiento correspondiente a posiciones

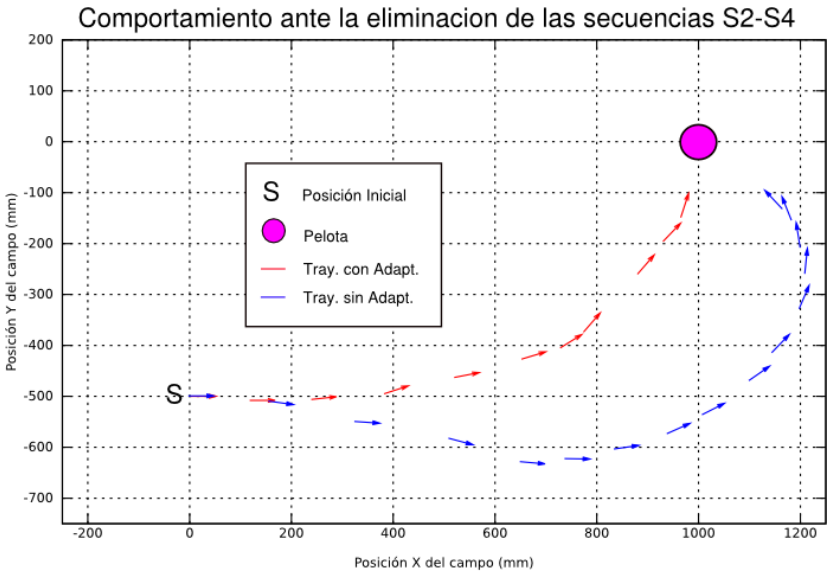


Figura 6.57: Trayectoria desde la posición S2 (base CBR pruebas S0-S1,S3,S5,S6) con y sin adaptación por experiencia

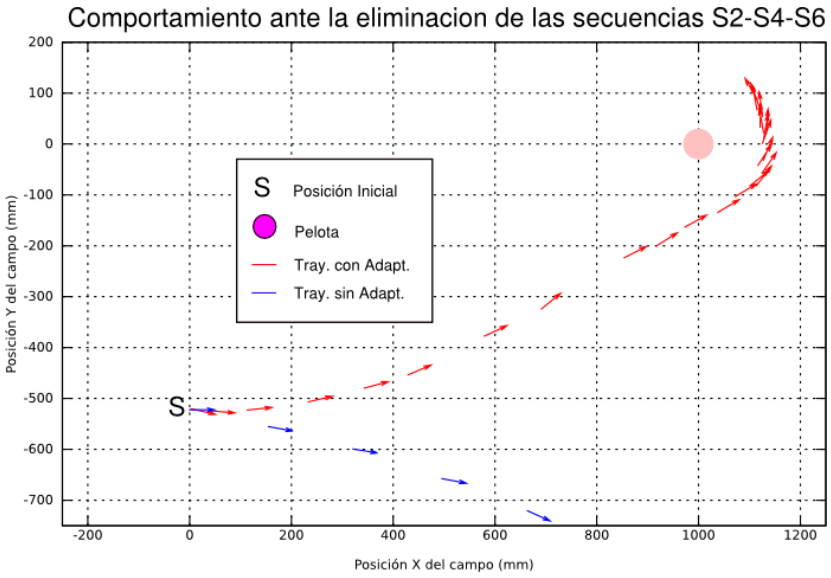


Figura 6.58: Trayectoria desde la posición S2 (base CBR pruebas S0-S1,S3,S5) con y sin adaptación por experiencia

cercanas, sería capaz de actuar perfectamente, sin necesidad de un entrenamiento intenso previo de las secuencias $S2$, $S4$, y $S6$.

Escenario complejo: acercamiento a la pelota con varios robots

Finalmente, se realizaron varias pruebas para comprobar el sistema de adaptación *CBR* en un escenario más complejo en el que se contempla la existencia de todos los posibles agentes presentes en el comportamiento: la pelota, los dos robots rivales, y el robot compañero. Para ello se procedió a repetir alguna de las pruebas correspondientes a la base *CBR* completa desde posiciones no entrenadas (pruebas PNEx), pero realizando un diezmado de la base *CBR* de forma aleatoria y distribuida. Se probaron

diezmados del 50 % y 75 % de la base —1800 y 950 casos— En las pruebas $PNE1$ a $PNE9$ se comprobó que, a pesar de que el comportamiento no era tan perfecto como con la base completa, el robot acaba encontrando una forma —mas o menos directa— de alcanzar la pelota, esquivando a los robots rivales, y finalizando con control de la misma —si bien no siempre en situación favorable de campo abierto—. El comportamiento con *CBR* adaptado se mostró ligeramente mejor que el de aplicación *CBR* directa, pero sin importantes diferencias. En la figura 6.59 podemos ver un ejemplo de una de estas pruebas.

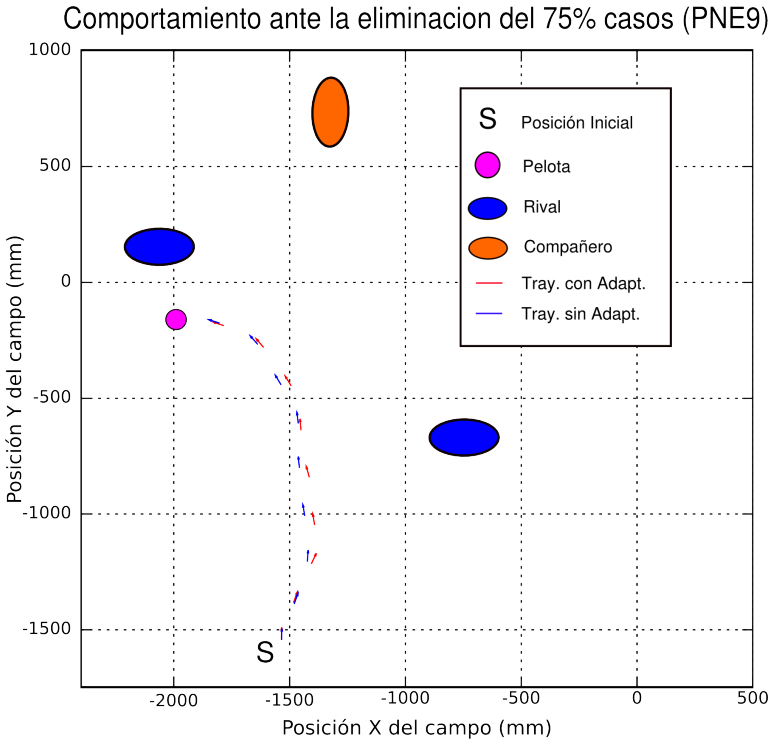


Figura 6.59: Trayectoria prueba PNE9 (diezmado 75 % base) con y sin adaptación por experiencia

De nuevo se hace patente la robustez del sistema de control reactivo basado en *CBR* ante la eliminación de información, siempre y cuando esta no se concentre en una trayectoria o sub-escenario particular. La reactividad del sistema suple la falta momentánea de información. Para hacer patente la influencia positiva de la adaptación, debemos acudir a escenarios no o poco entrenados: un ejemplo es la prueba $PNE10$ en la que, como ya vimos en el apartado anterior, la trayectoria de respuesta del robot es errónea al encontrarse con situaciones no entrenadas en las que aparecen simultáneamente los tres robots en escena. La aplicación de un diezmado sobre la base *CBR* no tiene ningún efecto positivo en esta prueba $PNE10$ en el caso en que no activemos el mecanismo de adaptación de *CBR*, como cabía esperar. Sin embargo, la utilización del mecanismo de adaptación e incorporación de nuevos casos si demuestra un efecto beneficioso ante las situaciones desconocidas que llevaban a un mal comportamiento previo, como se puede ver en la figura 6.60.

Las pruebas anteriores dejan patente la importancia de añadir a la fase de entrenamiento por observación de nuestro sistema *CBR*, una segunda fase de aprendizaje por experiencia, que permita subsanar, en cierta medida, los posibles

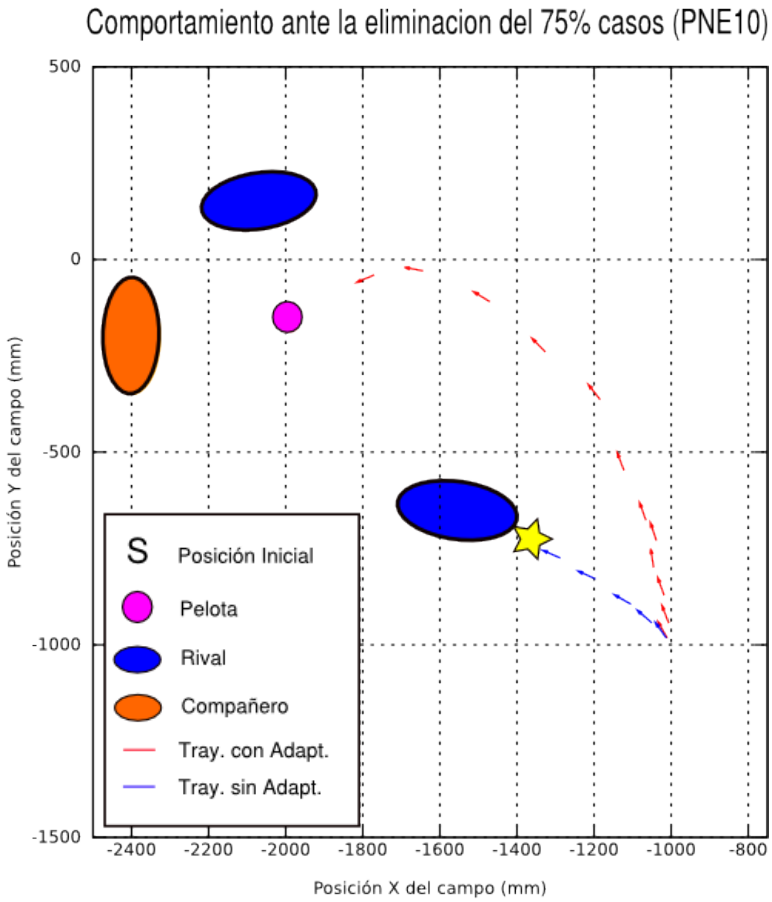


Figura 6.60: Trayectoria para la prueba PNE10 (diezmado 75 % base) con y sin adaptación por experiencia

"agujeros" de información, en el conocimiento obtenido en la primera fase. Es necesario destacar, no obstante, la importancia de contar con una base de conocimientos lo mas completa posible, ya que el suplemento aportado por adaptación de la experiencia es tan eficaz como lo sea el algoritmo empleado para dicha adaptación. En un sistema que contase con una arquitectura completa, incluyendo niveles mas abstractos/deliberativos, la adaptación podría aprovechar mejor los conocimientos almacenadas a altos niveles de abstracción a través de analogías entre conceptos, pudiendo simplificarse en cierta medida el componente de adaptación que hemos implementado, en el caso de nuestra arquitectura reactiva, a través del mecanismo de campos de potencial.

3.4. Diseño de un comportamientos complejos adicionales

Los siguientes sub-apartados tratarán, de forma concisa, aspectos concretos de diseño de otros dos comportamientos básicos de tipo complejo que hemos desarrollado para nuestro robot, con objeto de acometer el análisis de la obtención de comportamientos emergentes a través de la combinación de dichos comportamientos básicos.

3.4.1. Diseño de un comportamiento de evitación de obstáculos

Este comportamiento no resulta, en realidad mucho mas complejo que los ya probados en el apartado 2, pero ha sido incluido para permitir contar con un repertorio mayor de comportamientos a combinar para la obtención de comportamientos emergentes. Básicamente, el comportamiento deberá permitir al robot evitar el choque con otros robots a medida que se desplaza por el terreno de juego. Para ello, el mecanismo de atención a objetos del robot pasará a centrarse también en los obstáculos cuando se considere que estos están suficientemente cerca de nuestro robot para suponer un peligro de colisión. En cada caso se considerarán únicamente los objetos correspondientes a los robots, tanto rivales como compañeros. El mecanismo de atención hacia estos objetos se activará únicamente cuando se determine que estos pueden estar lo suficientemente cerca, algo que se identificará o bien a través de un tamaño de área suficientemente grande, o bien por la existencia de *clipping* con unas determinadas características —por ejemplo, un *clipping* que ocupe todo el área lateral de la imagen—. A efectos de respuesta del comportamiento se ignoraran aquellos objetos que se consideren lejanos, lo cual redunda en una reducción de la base de conocimientos necesaria para el aprendizaje del comportamiento. De todas formas, la cercanía de objetos supone una amplia ocupación del espacio visual del robot, por lo que la influencia de mas de dos objetos cercanos simultáneamente está prácticamente descartada. En la fase de entrenamiento por imitación se consideraron situaciones en las que solo aparecía un robot, tanto rival como compañero, en el campo de visión, así como situaciones en las que aparecía un rival y un compañero a la vez. No se entrenaron situaciones con dos rivales ya que, debido a la condición de que el comportamiento se active cuando el rival está muy cerca, un segundo rival se superponía, a efectos de detección de objetos, con el primero. El número de pruebas necesario fue bastante bajo —diez en total—, la base de casos generada contenía finalmente un total de 160 casos. Para este comportamiento no se diseñó una funcionalidad de aprendizaje por experiencia, debido a su sencillez.

En la figura 6.61 podemos ver un ejemplo ilustrativo de este comportamiento.

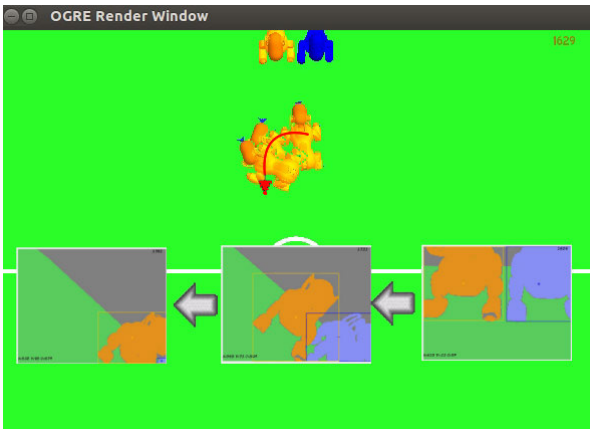


Figura 6.61: Ejemplo de ejecución del comportamiento de evitación de obstáculos

3.4.2. Diseño de un comportamiento complejo de posicionamiento para pase o chut

El tercer comportamiento que se ha tratado de diseñar implementa el posicionamiento que debe realizar el robot una vez ha tomado el control de la pelota, para decidir entre chutar la pelota hacia la portería, o intentar un pase a un compañero que no se encuentre obstaculizado por un robot rival. Este comportamiento implica un mayor número de elementos que afectan a las decisiones a tomar: además de de otros robots, compañeros o rivales, nuestros robot debe tener en cuenta la presencia de las porterías, tanto la propia como la rival. No se considerará la presencia de la pelota como elemento adicional, ya que el comportamiento debe tener efecto, en principio, cuando se esté en posesión de la misma según el sensor del pecho del robot, situación que implica una posición alta de la cabeza del robot con objeto de visualizar adecuadamente los objetos de interés, robots y portería. De hecho, con objeto de facilitar la fase de entrenamiento del robot, prescindiremos de la pelota en el campo, "trucando" el sensor de detección para considerar que ésta siempre está presente, y poder concentrar en entrenamiento en las acciones de posicionamiento ante la presencia de obstáculos y porterías. La respuesta ante cada caso es un vector de movimiento similar al del comportamiento de aproximación a la pelota pero, en este caso, solo se contemplarán desplazamientos de tipo rotación y lateral (*strafe*), además de la posibilidad de realizar una acción de chut mediante una secuencia pre-grabada, en caso de que se juzgue conveniente. Es evidente que, en pruebas reales en las cuales se incluya la pelota, es posible que se pierda el control de ésta en ocasiones; en ese caso, deberá ser un mecanismo de nivel superior el que lo detecte y conmute o combine los comportamientos básicos para tomar las acciones adecuadas.

De nuevo se realizó una fase previa de entrenamiento por observación/imitación con diferentes secuencias y situaciones, algunas de las cuales podemos ver en la figura 6.62. El diseño de este comportamiento puso de manifiesto algunas de las debilidades del esquema que habíamos planeado. Así, la influencia de algunos de los elementos —como la presencia de un robot enemigo— en la respuesta del robot, depende de cercanía o lejanía tanto nuestro robot, como a la portería o el robot compañero posible receptor del pase. En el esquema programado en nuestra herramienta la discretización de los objetos se hace por igual para todos ellos, considerando los intervalos de área observada para representar conceptos de tamaño o distancia, tal como vimos en el apartado 4.2.1 del capítulo 5. Sin embargo, habría sido más adecuado permitir diferentes intervalos de discretización para los distintos objetos: de esta forma, en vez de conceptos de tipo "objeto a media distancia" u "objeto a gran distancia", se podrían incluir conceptos más elaborados de la forma "jugador a distancia suficiente para no influir en la decisión del pase" o "portería demasiado lejana para intentar un chut". Además, esto redundaría en una disminución del número de pruebas con las que es necesario entrenar al robot para que pueda reconocer los diferentes escenarios en los que se pueda encontrar durante el experimento. Estas modificaciones no se han realizado en el presente trabajo, pero se implementarán en etapas futuras de nuestra investigación.

Como resultado de este aspecto de discretización, la diversidad de situaciones a considerar para un entrenamiento completo del robot es demasiado elevada. Por este motivo hemos considerado únicamente las correspondientes a un posicionamiento del robot encarado en su zona de ataque, y con la posibilidad de encontrar un robot compañero al que pasar, y un enemigo que pueda obstaculizar las acciones, además de

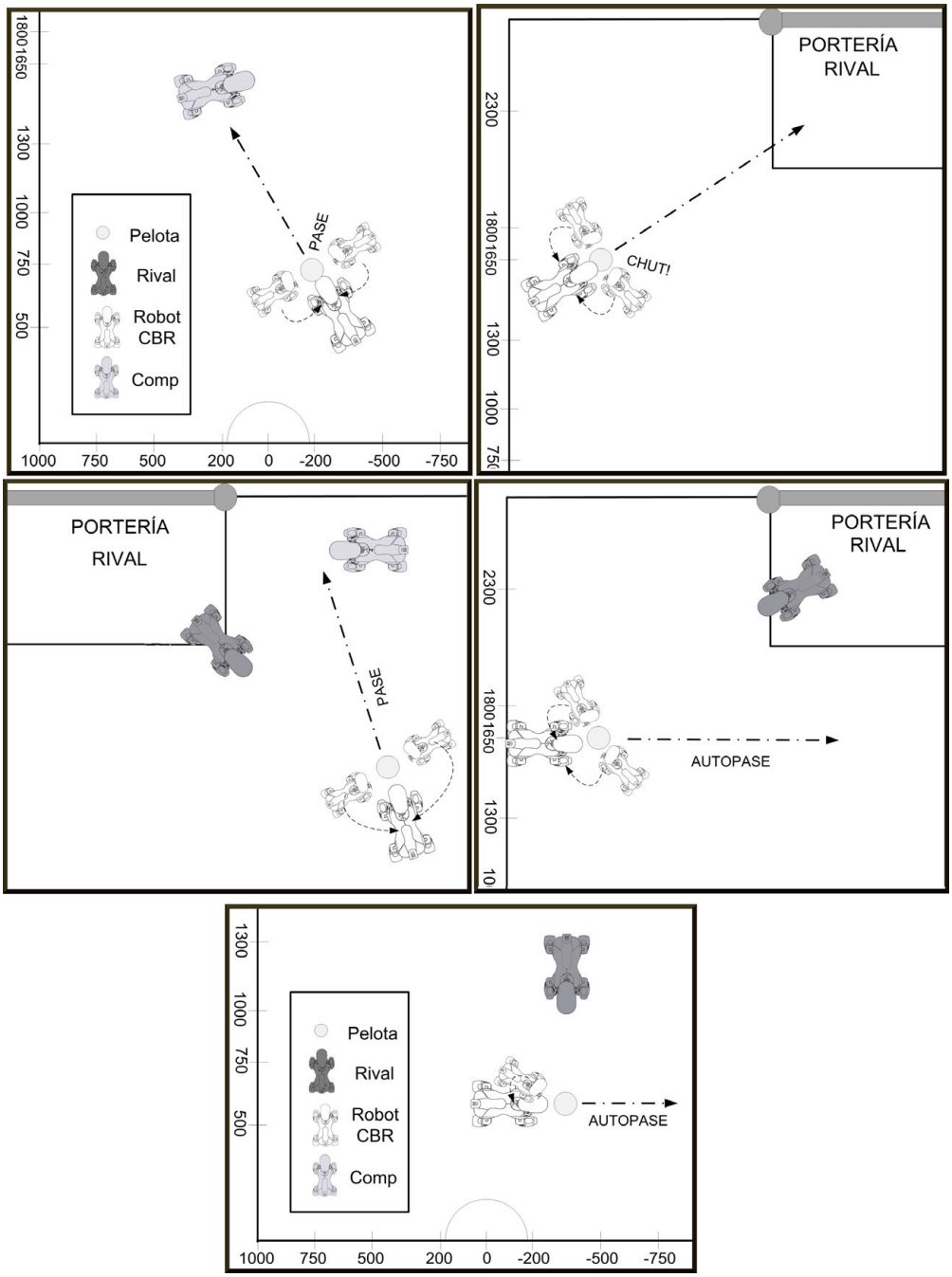


Figura 6.62: Ejemplos de secuencias de entrenamiento para el comportamiento de pase o chut

la propia portería rival. Aún así, el número de secuencias entrenadas ronda las 500, aunque de poca duración en su mayoría. En estas condiciones, se entrenó al robot para decidir entre un intento de pase a un compañero, de disparo hacia portería, o de autopase, siempre intentando dejar al robot en la mejor situación posible respecto al resto de elementos, tras realizar la acción.

Hay que destacar que el entrenamiento completo de este comportamiento resulta bastante complejo desde un punto de vista de programación algorítmica, existiendo multitud de casos "especiales" y de decisiones que dependen, no solo de la mayor o menor certeza de acabar la acción elegida con éxito, sino de aspectos subjetivos del entrenador humano, como por ejemplo que prefiera no arriesgar el control de la pelota, o hacerlo. Por otra parte, con este entrenamiento no tiene porqué existir simetría en el comportamiento entrenado respecto a las posiciones en lado derecho e izquierdo. En el entrenamiento se ha procurado responder a las "sensaciones de lo que se debe hacer en cada momento" más que a una evaluación pausada del mejor curso de acción.

Tras el entrenamiento de las secuencias de aprendizaje no se logró un comportamiento tan correcto como en el caso del comportamiento de acercamiento a la pelota. Si bien en los casos mas sencillos, que implican un único objeto, el robot fue capaz de aprender el comportamiento deseado de forma correcta, en escenarios que implican un mayor número de objetos no siempre se consiguió este objetivo. En ocasiones el robot entra en un comportamiento cíclico de recuperación de casos que le llevan a realizar secuencialmente acciones opuestas, con lo que no se logra el objetivo final de alcanzar una situación óptima para el intento de pase, autopase o chut. Tras analizar el problema, llegamos a la conclusión de que se debía a la existencia de situaciones simétricas en cuanto a lo que percibe el robot de los objetos implicados en la escena, pero que exigen acciones distintas dependientes de otros factores, como la posición del robot en el campo. Y es que, durante el entrenamiento, el operador humano no suele actuar de forma totalmente reactiva teniendo en cuenta únicamente la información del momento actual. Por el contrario, recuerda la coyuntura que ha experimentado en los instantes previos, y planea de antemano posibles acciones futuras, por lo que sus acciones no son siempre "lógicas" desde un punto de vista puramente reactivo. Para paliar este problema tenemos intención de añadir, en pruebas futuras, información adicional al caso, como pueda ser la posición del robot o la línea del horizonte. Adicionalmente, se podría incluir en el caso información adicional con componente temporal, como por ejemplo la evolución en tamaño o posición de los objetos, o las acciones más cercanas realizadas por el robot hasta llegar a la situación actual. Como aspecto negativo, la inclusión de esta información, redundará en un aumento de la dimensión del espacio del caso, lo cual se traduciría en un mayor número de casos necesarios para almacenar el conocimiento asociado al comportamiento.

Para las pruebas de combinación de comportamientos básicos para la obtención de comportamientos emergentes, de este comportamiento solamente se aprenderán las situaciones adecuadas a los escenarios que vayamos a utilizar en las pruebas, evitando, de esta forma, los inconvenientes ya mencionados. Por este mismo motivo, para este comportamiento no se ha incorporado un esquema de adaptación como el del apartado 3.3, si bien las consideraciones a realizar serían muy similares, siguiendo las indicaciones ya tratadas en el apartado 4.4.2 del capítulo 5.

3.5. Emergencia de comportamientos basada en reglas de decisión

Para finalizar la parte experimental de esta tesis, se realizará un ejemplo básico de obtención de un comportamiento emergente a partir de los comportamientos básicos diseñados, que se han descrito en los apartados anteriores. Tal como vimos en el apartado 4.5 del capítulo 2, la emergencia de comportamientos en las arquitecturas *BBR* se consigue a través de una combinación dinámica de un repertorio básico de comportamientos elementales. En la arquitectura de inteligencia propuesta sería deseable que esta emergencia de comportamientos se materializase a varios niveles de la jerarquía, pasando a incorporarse los comportamientos generados al repertorio de los correspondientes niveles superiores. A medida que se asciende por la jerarquía, los comportamientos del repertorio, tanto generados por aprendizaje directo como por combinación emergente, representarían conceptos cada vez más abstractos y generales, y con una mayor vigencia temporal. No obstante, dado que nuestro trabajo práctico se centra principalmente en los niveles reactivos inferiores, el ejemplo desarrollado tendrá asimismo un ámbito reactivo. El objetivo será, por tanto, desarrollar una "prueba de concepto" (*proof of concept*) que ejemplifique la filosofía de desarrollo propuesta.

Por otra parte, y atendiendo a estructura de "nido" de la arquitectura de inteligencia propuesta (apartado 3.2), resultaría bastante coherente que los módulos que se encargasen de realizar la combinación de módulos o comportamientos de nivel inferior para generar comportamientos emergentes, tuviesen las mismas características y mecanismos de funcionamiento que los módulos que combinan, es decir, sistemas CBR en los que los factores de selección y combinación de los módulos inferiores son también aprendidos en una fase previa de entrenamiento y/o a través de la experiencia del robot. El principal inconveniente para la implementación de estos módulos de combinación/arbitraje siguiendo esta filosofía CBR radica en la elección de la información de entrada que define las "situaciones" antes las cuales el módulo aprende a dar un peso u otro a los comportamientos componentes. Esta información de entrada debería estar relacionada con conceptos que englobasen estados de los comportamientos de bajo nivel. Por ejemplo, en un módulo de combinación de comportamientos emergentes para "aprender a jugar al fútbol en la Robocup", un módulo o comportamiento de nivel inferior dedicado a la aproximación a la pelota podría representar un concepto de "avance por banda para buscar controlar la pelota", o "pelota bajo control", mientras que otro comportamiento inferior dedicado a evitar colisiones podría estar representando un concepto de "evitando colisión de objeto muy cercano por la izquierda" o de "vía libre sin obstáculos". Idealmente, estos conceptos formarían un espacio de posibles situaciones que supondría la información de entrada en el modulo de combinación emergente, y que permitiría aprender una respuesta adecuada en forma de selección o combinación ponderada de los módulos inferiores. Sin embargo, esta representación de las ejecuciones de los comportamientos como conceptos de alto nivel no ha sido desarrollada todavía en nuestro trabajo, por lo que no puede considerarse aún el diseño de un módulo de combinación que atienda a estas características. Opciones alternativas como i) la inclusión de todos los objetos presentes en el escenario o ii) el uso de *PCA* para la representación "completa" de las situaciones o escenarios que experimenta el robot en un momento dado, también fueron descartadas al considerarse inadecuadas:

- La inclusión de todos los objetos haría innecesaria la existencia de comportamientos de bajo nivel con información local, ya que un único comportamiento con inclusión de toda la información posible podría, en teoría, aprender las respuestas del robot ante cualquier circunstancia (figura 5.5.der). No obstante, el comportamiento resultante sería finalmente inmanejable ya que, al trabajar con la información global, necesitaría de casos con un elevadísimo número de componentes. Esto redundaría en una dimensionalidad excesiva de la información necesaria para caracterizar y entrenar el comportamiento, con un número excepcional de casos en la base *CBR* y los problemas asociados de memoria necesaria y tiempos de recuperación de los casos. Finalmente, esta aproximación prescindiría totalmente de la estructura jerárquica del modelo de arquitectura de inteligencia propuesto, estructura que aparece como imprescindible para un diseño adecuado.

- La compresión de las imágenes que representan los escenarios a través de sus componentes principales parece, en principio, una opción razonable para una reducción de la información necesaria para la representación de las situaciones que experimenta el robot. El tamaño del caso podría controlarse mediante la elección del número de componentes deseados, siempre que se compruebe que dichos componentes son capaces de discriminar situaciones que, para el robot, han de ser consideradas como diferentes. Y aquí es donde surge el posible problema de esta aproximación: es posible que, precisamente los componentes de menor peso eliminados en la representación del conjunto de escenarios sean los que contengan las pequeñas diferencias o "sutilezas" que permiten distinguir situaciones que, aunque parecidas, exigen respuestas distintas.

Por todos estos motivos y, en espera de explorar en investigaciones futuras la combinación basada en aprendizaje *CBR* a partir de representaciones conceptuales de los comportamientos inferiores, en este trabajo hemos desarrollado, como demostración experimental, un módulo de combinación emergente basado en reglas de decisión establecidas por un experto. Este módulo no es demasiado complejo, dado el pequeño número de comportamientos componentes, aunque en él trataremos de utilizar las dos técnicas básicas de organización de comportamientos, como son la selección temporal, y la combinación directa y simultanea de los comportamientos que procedan.

Para ello, se ha modificado nuestro sistema para que, en todo momento, los tres módulos o comportamientos básicos desarrollados devuelvan un caso *CBR* acorde a la información local correspondiente de la situación experimentada por el robot. Las respuestas de los casos recuperados de cada módulo *CBR* se proporcionan al módulo de combinación, el cual recibe también información global del escenario, que se formatea de forma adecuada para la ejecución de las reglas. Dadas las características de los comportamientos básicos de nuestro sistema (aproximación a la pelot, evitación de obstáculos, y selección de pase o chut) el sistema de reglas propuesto es bastante sencillo:

- Regla 1: Cuando el robot tome posesión de la pelota se **selecciona** el comportamiento de selección de pase o chut.

- Regla 2: Cuando el robot no tenga posesión de la pelota, se **combinan** las salidas de los casos correspondientes a los comportamientos de aproximación a la pelota

y evitación de obstáculos. En esta combinación se aplica un peso mayor a la salida del comportamientos de evitación de obstáculos cuando se detecte un obstáculo cercano, mientras que se dará más importancia a la salida del comportamiento de aproximación a la pelota si no se detectan obstáculos cercanos. Esta combinación se puede expresar, para el caso particular de ejemplo, con la ecuación 6.14:

$$\vec{V}_{Comb} = \vec{V}_{Obst} \cdot p_{RD}(t) + \vec{V}_{AprBall} \cdot (1 - p_{RD}(t)) \tag{6.14}$$

donde $p_{RD}(t)$ representa el peso otorgado al comportamiento de evitación de obstáculos por el sistema reglas de decisión del módulo, en un instante t. Este peso se calcula de forma empírica en función del tamaño estimado del objeto más cercano a nuestro robot.

3.5.1. Resultados experimentales

Tal como indicamos en el apartado 3.4.2, no fue posible entrenar el comportamiento de pase o chut de forma completa ante cualquier situación que se pudiese dar en el escenario de pruebas propuesto. Por este motivo, las pruebas de combinación de comportamientos emergentes se circunscriben a situaciones específicas que sí han sido entrenadas para dicho comportamiento, incluyendo también las situaciones completas más sencillas, como las correspondientes a la presencia de un único robot aislado, tanto amigo, como enemigo, y de la portería sin otros robots presentes. Es bastante posible que en el proceso de posicionamiento del robot para realizar el pase o chut, se puedan producir desplazamientos de la pelota que lleven a perder su posesión, lo cual dispararía la activación del comportamiento de aproximación a la pelota hasta llegar de nuevo a su control. Si en esta aproximación, el robot alcanza una posición muy diferente a la entrenada para el comportamiento de pase o chut, no hay garantías de que realice una respuesta coherente.

En primer lugar se probó la combinación de los comportamientos de aproximación (módulo B) y de pase-chut(modulo C), a través de la selección de uno u otro según la posesión de la pelota. Para ello se manejaron tres escenarios entrenados para el módulo de pase o chut, añadiendo el entrenamiento correspondiente a la actuación ante elementos aislados, como robots rivales y compañero, y portería sin más robots presentes.

En las figuras 6.63, 6.64, y 6.65, podemos ver algunos ejemplos de ejecución del comportamiento emergente en cada uno de los escenarios propuestos.

Se observa como, en el acercamiento a la pelota, el robot evita trayectorias directas que le puedan llevar a situaciones finales de control en las que existan únicamente robots rivales en alineación frontal (prueba J1A de la figura 6.63), mientras que la aproximación es mas directa si hay también robots compañeros a la vista (prueba J1B de la figura 6.63, J2B de la figura 6.64, y J3A de la 6.65). En la prueba J2A de la figura 6.64, nuestro robot nunca llega a ver al robot compañero, por lo que actúa de forma similar a la prueba J1A. En las aproximaciones más oblicuas, el robot debe situarse en una posición adecuada por acción del comportamiento C, al alcanzar la pelota. Se puede ver que este giro evita situaciones de riesgo cuando hay solo un robot rival cerca de la portería, decidiendo finalmente situaciones de autopase (pruebas J1A y J1C de la figura 6.63; y J2A y J2C de la figura 6.64), mientras que la presencia de un robot compañero en situación ventajosa provoca una acción de pase hacia el mismo (pruebas

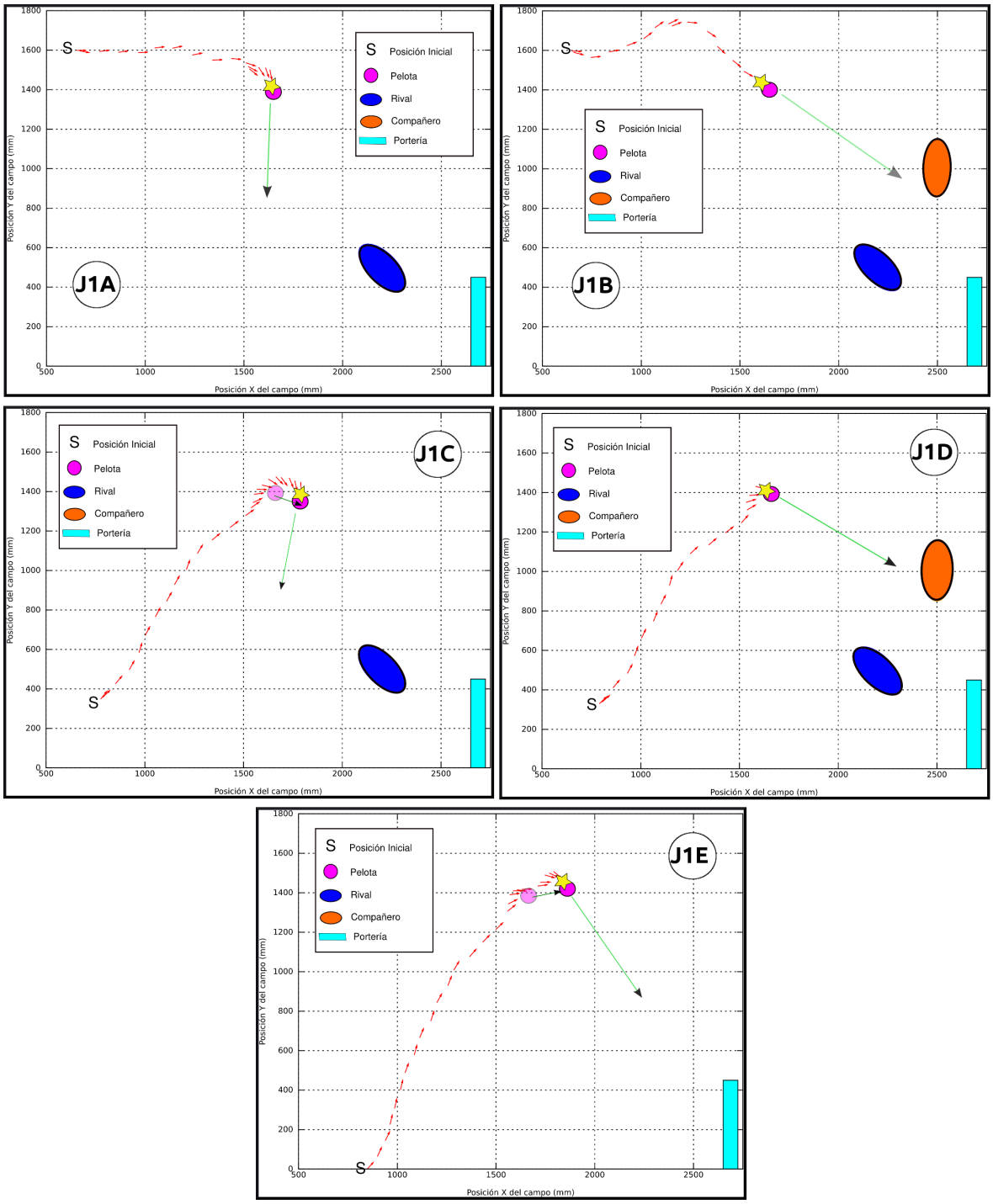


Figura 6.63: Ejemplo de comportamiento emergente por selección de los módulos B y C en el escenario 1

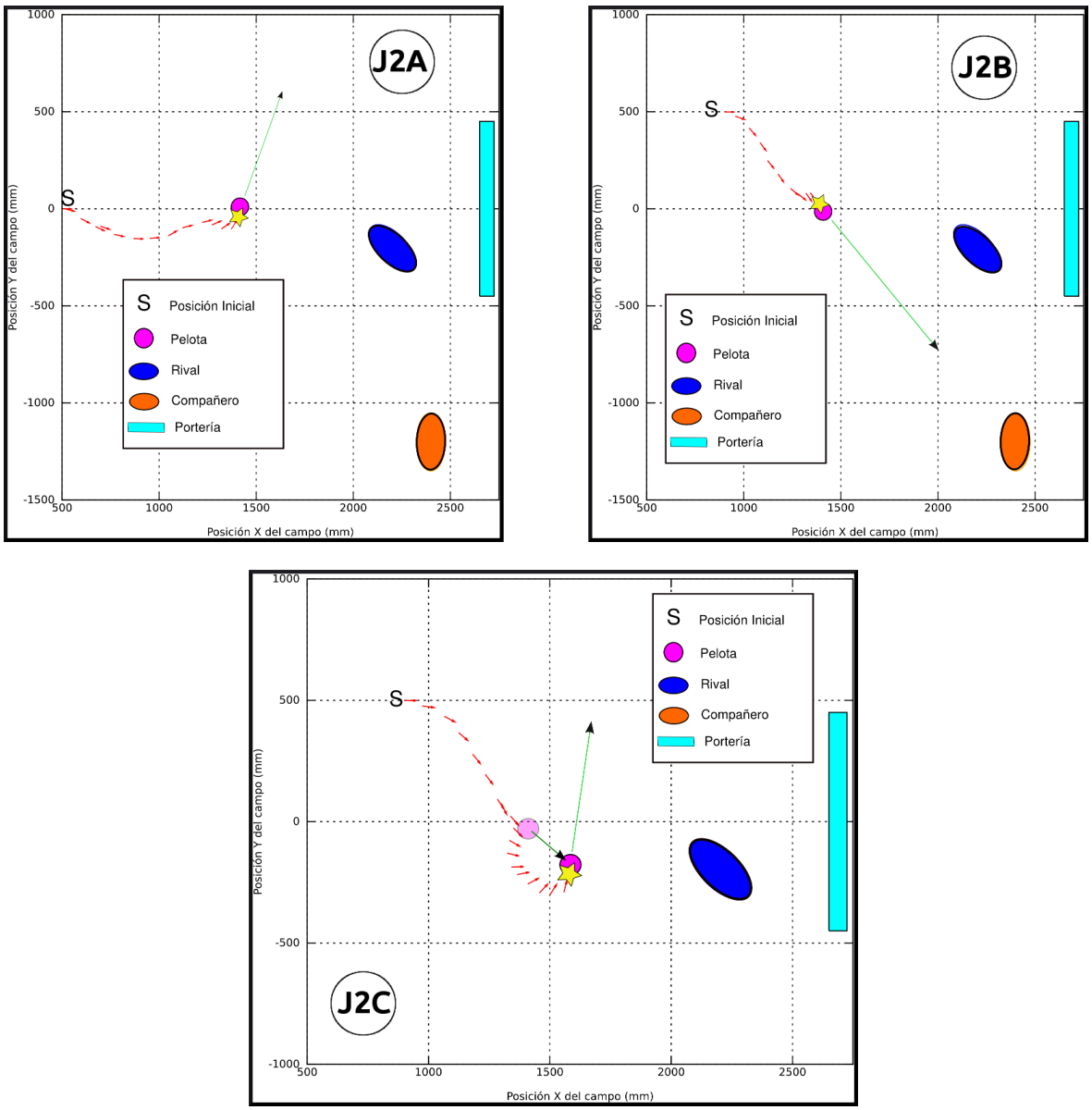


Figura 6.64: Ejemplo de comportamiento emergente por selección de los módulos B y C en el escenario 2

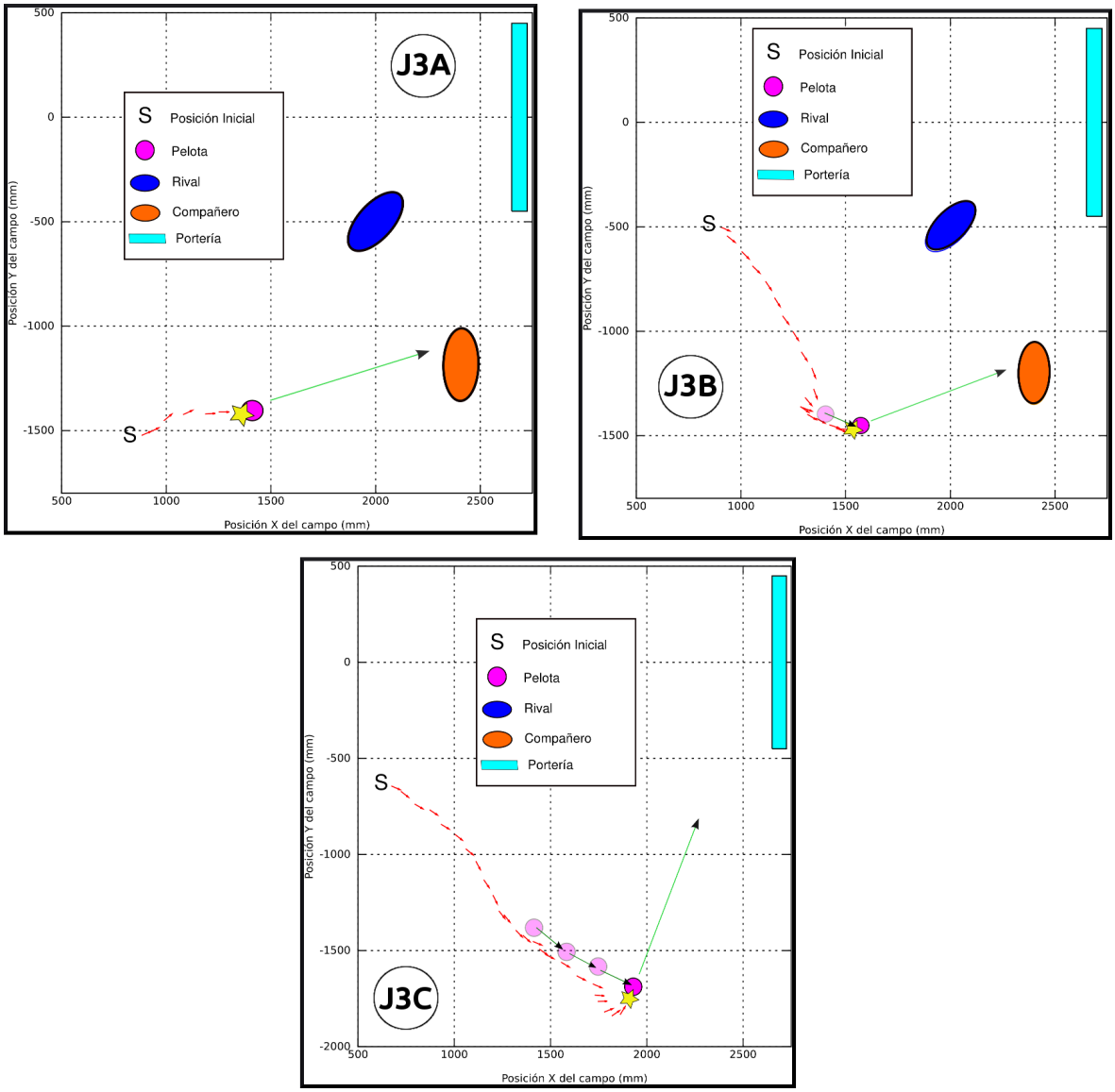


Figura 6.65: Ejemplo de comportamiento emergente por selección de los módulos B y C en el escenario 3

J1B y J1D de la figura 6.63; J2B de la figura 6.64; y J3A y J3B de la figura 6.65). La prueba J2B es la única que presenta una decisión un poco arriesgada, ya que un chut poco preciso podría dejar la pelota en las cercanías del jugador rival. Finalmente, si al alcanzar la pelota no se visualizan otros robots cercanos a la portería —especialmente robots rivales— nuestro robot gira hasta situarse de forma alineada con la portería e intentar un chut hacia la misma (prueba J1E de la figura 6.63; y J3C de la figura 6.65).

La conmutación de uno a otro comportamiento se produce correctamente, por lo que ambos son capaces de desempeñar su funcionalidad de forma adecuada en las condiciones establecidas. Incluso en ocasiones, una vez que se ha seleccionado el comportamiento C, de pase o chut, se pierde el control de la pelota, volviéndose a elegir de nuevo el comportamiento de búsqueda de pelota hasta alcanzarle y conmutar de nuevo a la selección de pase o chut. Solamente en las situaciones en las que este proceso de pérdida y recuperación de control de pelota lleva al robot a situaciones muy alejadas de las entrenadas en el módulo C, la respuesta no es totalmente adecuada.

Para la siguiente fase de este conjunto de pruebas se ha añadido un tercer comportamiento de evitación de obstáculos (módulo A), y se ha programado la lógica de combinación indicada en el párrafo anterior, teniendo en cuenta la distancia a los obstáculos más cercanos. El módulo C actúa en paralelo con el módulo B, y se inhabilita cuando el robot toma posesión de la pelota según el esquema de selección ya probado. Para la elección del parámetro $p_{RD}(t)$, se realizaron diversas pruebas y, finalmente, se eligió de forma heurística una función de la forma:

$$f(p_{RD})(t) = \frac{(\operatorname{sgn}(x(t) - \mu_{umbral}) \cdot \sqrt[3]{|\gamma \cdot (x(t) - \mu_{umbral})|}) + 1}{2} \tag{6.15}$$

donde μ_{umbral} es un umbral normalizado en $[0, 1]$, a partir del cual se quiere un aumento rápido del peso, y γ determina la rápidez en la transición de la función. También es posible usar un exponente mayor de 3 para la función raíz, si se desea una transición más rápida con respecto al umbral de la función de peso. La función resultante se acotará en $[0, 1]$, por lo que si los resultados obtenidos están por encima y debajo de los límites superior e inferior respectivamente, se aproximarán a dichos límites.

En nuestro sistema hemos considerado que un robot, tanto amigo como enemigo, supone un obstáculo significativo para la navegación de nuestro robot cuando su area ocupa al menos un 40 % de la imagen, o su *clipping* por los bordes laterales supera un 35 % del perímetro total de la imagen. Estos valores se han calculado de forma heurística a través de varias pruebas y teniendo en cuenta la velocidad de marcha del robot, por lo que otras velocidades distintas podrían necesitar de otros valores para asegurar que el robot tiene tiempo de esquivar el obstáculo sin detener la marcha de forma muy apreciable. En valores inferiores a los indicados, el efecto del comportamiento de "esquivar obstáculos" debe ser inapreciable sobre el resto de comportamientos en funcionamiento; por otra parte, nos interesa que, una vez superado ese valor, el incremento de la influencia de este módulo debe ser rápido, so pena de no tener tiempo para evitar la colisión. Finalmente, la función de peso considerada en nuestros experimentos tendrá la forma de la expresión 6.16, en la que se tienen en cuenta los dos posibles componentes indicadores de obstáculo próximo, tanto el área, como el *clipping* lateral, considerando el peso mayor de entre los dos. Dado que el mecanismo de atención a obstáculos tiende a enfocar lo más posible al robot, es bastante probable

que el peso máximo venga dado solo por la componente de *clipping* únicamente en los instantes iniciales y finales de la atención al obstáculo.

$$p_{RD}(t) = \max_i \left(\frac{(\text{sgn}\,(area_{porc}(t) - 0.4) \cdot \sqrt[3]{|(area_{porc}(t) - 0.4)|}) + 1}{2}, \frac{(\text{sgn}\,(clipLat_{porc}(t) - 0.35) \cdot \sqrt[3]{|(clipLat_{porc}(t) - 0.35)|}) + 1}{2} \right) \tag{6.16}$$

En la figura 6.66, se presentan los resultados del esquema de combinación/selección de los tres comportamientos básicos desarrollados para dar lugar a un comportamiento emergente de "acercamiento y chut, esquivando jugadores". Para ello se han modificado ligeramente los escenarios utilizados en las pruebas anteriores, incluyendo un nuevo robot rival que tiene por objeto obstaculizar a nuestro propio robot. En los escenarios presentados en esta tanda de pruebas, nuestro robot aparece inicialmente muy cerca de un robot rival, para forzar que el mayor peso del comportamiento de esquivar obstáculos sea más visible ya que, de lo contrario, nuestro robot evitaría aproximarse demasiado a los rivales en su trayectoria de acercamiento a la pelota debido a la ligera influencia del comportamiento de esquivar obstáculos en el comportamiento específico de búsqueda de pelota.

También se realizaron pruebas adicionales en las que el robot rival era manejado por un operador humano para interceptar la trayectoria de acercamiento a la pelota de nuestro robot, de manera que se modificase el peso del los módulos B (aproximación a la pelota) y A (esquivar obstáculos) en la respuesta final del robot. En estas pruebas se observó que el comportamiento era el esperado, en particular con respecto a la combinación de los módulos A y B, y su alternancia/selección con el módulo C cuando se tomaba o perdía el control de la pelota. Recordemos que el módulo C (posicionamiento para chutar o pasar) no se entrenó completamente debido a los problemas ya mencionados por lo que solamente cuando la situación del robot era muy diferente a la entrenada en el momento de alcanzar la pelota, se pueden producir fallos en la respuesta del robot durante la ejecución de dicho módulo C. Estas pruebas están disponibles en formato de vídeo [289].

3.6. Conclusiones de las pruebas de la segunda etapa

Tras la realización de las pruebas de esta segunda etapa se demuestra la viabilidad de la arquitectura de comportamientos propuesta así como la posibilidad de obtener comportamientos emergentes de nivel superior a partir de los comportamientos entrenados. No obstante es necesario destacar las fortalezas y debilidades de la aproximación propuesta:

- La transición desde comportamientos muy simples basados en un solo elemento (pelota, líneas del campo, marcas,...) hacia comportamientos más complejos que consideren la influencia simultánea de varias elementos, permite una mayor versatilidad en los comportamientos finales obtenidos, trasladando la complejidad de la combinación de los comportamientos componentes al interior de los mismos. Esto ofrece la ventaja de que esta complejidad queda absorbida en gran parte en el entrenamiento de los mismos, por lo que un módulo de organización/emergencia

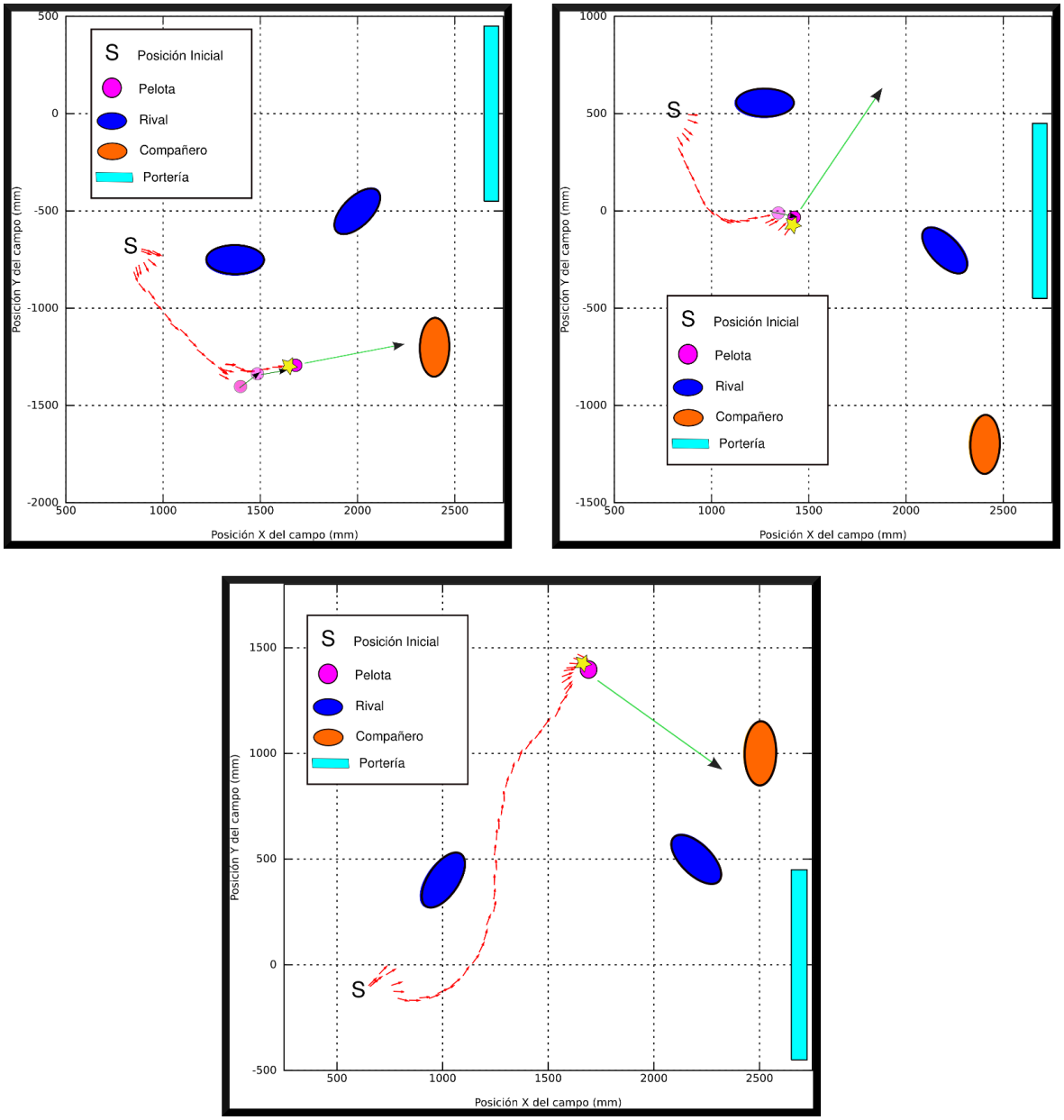


Figura 6.66: Ejemplos de comportamiento emergente por selección y combinación de los módulos A, B y C

sencillo como el basado en reglas de decisión permite obtener muy buenos resultados.

- Si bien el procesamiento de la información se realiza de forma numérica, parece bastante deseable utilizar representaciones conceptuales de la misma, como forma de construir conceptos cada vez más generales o abstractos. En nuestra propuesta se ha realizado una discretización de los principales parámetros para, por ejemplo, establecer distintas posiciones conceptuales de los objetos en el campo de visión del robot, o distintos tamaños como expresión de su distancia. Sin embargo, y tal como hemos visto a través del diseño no completamente exitoso del módulo C, estas representaciones conceptuales no deberían ser absolutas sino estar ligadas a su significado con respecto al comportamiento al que van a definir.

- Las pruebas realizadas han puesto también de manifiesto las limitaciones para diseñar comportamientos puramente reactivos en caso que la respuesta de los mismos necesite de una componente temporal. Si la respuesta de los comportamientos depende únicamente de los estímulos presentados en un instante dado, se corre el riesgo de entrar en bucles cíclicos en los que se repiten las acciones y la presentación de unos mismos estímulos de forma alternada. Así, se pone de manifiesto que los comportamientos reactivos puros no pueden formar por sí solos una arquitectura completa, excepto para casos sencillos —como pudiese ser la arquitectura de comportamientos de un robot insecto—. Es necesaria la existencia de niveles superiores de tipo deliberativo que incluyan la componente temporal, y permiten una generalización de bajo nivel. No obstante, la capa reactiva cumple un papel fundamental en la arquitectura, y su diseño a partir del aprendizaje es plausible y ofrece ventajas respecto al diseño basado en algoritmia pura.

- Se hace patente la necesidad de un experto que seleccione y organice la información necesaria para cada módulo componente y que, asimismo, establezca las reglas de decisión del módulo de organización de comportamientos —al menos en la aproximación propuesta—. En un futuro sería deseable reducir esta dependencia del experto a través de técnicas estadísticas que analizasen la influencia de la información en los comportamientos que se desea aprender y permitiesen seleccionar tanto los datos fundamentales como sus relaciones.

- A medida que aumenta el número de parámetros que influyen en el comportamiento, empezamos a encontrar problemas computacionales en el proceso de ejecución del ciclo *CBR*, por el manejo de una cantidad excesiva de información; por ello , es necesario buscar estructuras de la información que permitan acelera el proceso de recuperación de la información relevante desde la base *CBR*. En este sentido, hemos propuesto una organización de la información en una estructura mixta de árbol, en la cual cada hoja es una base plana, y que se puede recorrer a través de una información de contexto inequívoca. Además, hemos modificado el proceso de recuperación de la información del ciclo *CBR* para realizar una fase previa de "selección de casos por objetos en escena" antes de la recuperación clásica de casos más cercanos mediante funciones de similitud. Estas modificaciones permiten acelerar el proceso de recuperación de

información, especialmente en situaciones en las que existe un gran número de ramas en el árbol jerárquico construido.

- El aprendizaje por experiencia resulta básico para completar el conocimiento adquirido durante el entrenamiento para el diseño de un comportamiento dada la dificultad, en muchos casos, de entrenar completamente y de forma previa todas las situaciones que se deban afrontar en dicho comportamiento. Por otro lado, es necesario un mínimo entrenamiento previo que haga posible una ejecución básica del comportamiento, a partir de la cual se puedan completar el conocimiento ausente a través de la experiencia.

7 Conclusiones y líneas futuras

"En la vida, cada final es sólo el principio de otra historia"
Julian Barnes (1946-) Novelista

1. Conclusiones

En esta tesis se ha estudiado el diseño y organización de comportamientos reactivos de bajo nivel basados en aprendizaje, como base de una arquitectura de inteligencia robótica a nivel reactivo. Dichos comportamientos se fundamentan en *CBR*, una herramienta de Inteligencia Artificial especialmente adecuada a las características de la arquitectura propuesta. Así mismo, esta arquitectura formaría parte, como nivel inferior, de una propuesta de arquitectura completa de inteligencia robótica basada en la obra de Jeff Hawkins, *On Intelligence*, en la que se postula que el aprendizaje de información, y la capacidad de predecir respuestas a partir de dicha información a diferentes niveles conceptuales son la base de la inteligencia humana y pueden ser un buen punto de partida para la implementación de una inteligencia artificial.

La mayoría de las soluciones para el diseño de comportamientos reactivos, tanto en el caso de arquitecturas reactivas puras como en el de capas inferiores de arquitecturas híbridas completas, se basan en soluciones algorítmicas pre-programadas que determinan un funcionamiento inmutable del comportamiento y tienen poca capacidad de adaptación ante entornos dinámicos con estructura no completamente definida. Siendo esta adaptabilidad una característica fundamental en los robots autónomos modernos, en la presente Tesis hemos analizado la posibilidad de que estos comportamientos reactivos pudiesen ser aprendidos en vez de programados, siendo este aprendizaje continuado durante toda ejecución del comportamiento [62] [196] [3]. Una ventaja adicional a esta mayor capacidad de adaptación, radica en que estos comportamientos reactivos puedan ser definidos de forma más sencilla que en el caso algorítmico, al no necesitarse un nivel de comprensión tan detallado de sus mecanismos. Efectivamente, en los comportamientos aprendidos se debe identificar cuales son los estímulos que van a influir en el comportamientos, pero su interrelación puede ser absorbida de forma intrínseca a través del entrenamiento realizado por un entrenador humano que ejecuta las acciones asociadas al comportamiento en presencia

de dichos estímulos. Esto incluiría también los posibles errores e imprecisiones en el funcionamiento del robot y en la identificación del entorno, tan difíciles de incluir por la vía algorítmica.

Un aspecto clave para este diseño de comportamientos por aprendizaje radica en la identificación de estos estímulos o información local al comportamiento, y la organización de esta información de una forma que sea representativa y fácil de utilizar en una posterior identificación de situaciones para la realización de la respuesta adecuada. Se ha realizado un estudio detallado de los diferentes aspectos que puede tomar esta información —vocabulario, función de similitud, base de casos, y función de adaptación— y se han tomado una serie de decisiones en cuanto a que versión utilizar de las mismas en el dominio en el que se han realizado los experimentos, la aplicación Robocup con robots tipo AIBO. Se ha constatado la importancia de conciliar una representación numérica de la información con otra conceptual algo que, en nuestras pruebas, se ha tratado de realizar mediante una discretización. Esta discretización debe ser adaptada a los aspectos conceptuales que se trata de representar en cada uno de los comportamientos diseñado.

También se ha abordado la problemática relacionada con la escalabilidad de la arquitectura a medida que aumenta la información necesaria para la definición de los comportamientos. Este aumento provoca el incremento proporcional en el tiempo de respuesta del modulo *CBR* para la predicción de la respuesta almacenada ante el escenario planteado, algo especialmente crítico en comportamientos de tipo reactivo. Para solucionar estos problemas se han propuesto modificaciones en la estructura de la base *CBR* que almacena el conocimiento, así como en el proceso de recuperación de dicha información. Estas modificaciones han demostrado un comportamiento satisfactorio [271].

Para las demostraciones experimentales se realizaron dos etapas de pruebas. En el primer conjunto de pruebas, y con objeto de constatar la viabilidad inicial de nuestra propuesta, se implementaron y probaron varios comportamientos "simples", en el sentido de que su funcionamiento estaba ligado a un único objeto o estímulo como fuente de información local [196] [63] [279]. Pese al buen desempeño individual de estos comportamientos, se encontraron ciertas dificultades para combinarlos en la obtención de comportamientos emergentes de nivel superior, algo primordial en el modelo de arquitectura jerárquica de *IA* propuesta [281] [283].

Ante estos problemas encontrados, se procedió a un cambio de filosofía de diseño de comportamientos en lo relativo a su ámbito de acción. Si hasta ahora los comportamientos propuestos expresaban un curso de acción en relación a un elemento u objeto, en diferentes situaciones del mismo, los nuevos comportamientos complejos a diseñar expresan ahora diversas sub-conductas o realizaciones parciales de la conducta que se desea aprender en el comportamiento general del robot. Esta segunda realización tiene la ventaja de que el propio entrenamiento absorbe también los aspectos de interrelación entre los componentes de información asociados al comportamiento, interrelaciones que antes había que diseñar a través del módulo de arbitración. De los tres comportamientos diseñados como ejemplo, dos obtuvieron resultados satisfactorios ante una variedad de situaciones y condiciones; el tercero presentó algunos problemas relacionados con la codificación conceptual de la información y la propia naturaleza reactiva del comportamiento a diseñar, problemas que se solucionarán en trabajos futuros con las modificaciones adecuadas de las herramientas de prueba. De cualquier

forma, este tercer comportamiento se pudo diseñar con éxito a través del aprendizaje, para sub-escenarios específicos, en los que demostró también un desempeño correcto. En cuanto a la combinación de estos comportamientos básicos en un comportamiento emergente, se consiguió con éxito a través de un módulo adicional de organización basado en reglas de decisión que permiten bien la selección del comportamiento adecuado en situaciones excluyentes, bien una combinación ponderada de aquellos comportamientos que pueden funcionar en paralelo.

Finalmente podemos concluir que el diseño de comportamientos de tipo reactivo a través del aprendizaje es una estrategia prometedora para una propuesta de arquitectura de *IA* de bajo nivel, o como capa inferior de una arquitectura más completa que incluya etapas de alto nivel de tipo deliberativo. Además, *CBR* se postula como una herramienta muy adecuada para la implementación de la arquitectura propuesta, ya que responde a los principios fundamentales del modelo de inteligencia humana basada en aprendizaje y predicción [1], modelo que consideramos muy adecuado para un intento de diseño de una arquitectura general de inteligencia robótica.

2. Líneas Futuras

Como primer trabajo de continuación de la investigación de esta tesis tenemos prevista la revisión del proceso de conceptualización de la información local a los comportamientos, de forma que podamos asegurar un funcionamiento correcto —dentro de las limitaciones del horizonte reactivo— de todos los comportamientos diseñados. Asimismo, se procederá a diseñar un mayor número de comportamientos básicos que implementen otras funcionalidades relacionadas con la aplicación Robocup, lo cual nos permitirá probar de forma más general la bondad de la arquitectura propuesta. A continuación, se procederá a probar los comportamientos de la segunda fase de pruebas en un grupo de robots reales, y analizar como afecta al buen funcionamiento de la arquitectura propuesta esa traslación de la arquitectura al mundo real, con sus imprecisiones e incertidumbres añadidas.

En cuanto a otros aspectos del diseño de los comportamientos, se pretende analizar de forma más exhaustiva algunas particularidades del proceso de almacenamiento y recuperación de información propios del ciclo *CBR*. Queremos prestar especial atención a la conceptualización del conocimiento asociado a los comportamientos, ya que nos parece un aspecto clave, no solo en mejorar la escalabilidad de la arquitectura, sino en una posterior integración de conocimientos/comportamientos hacia niveles superiores más abstractos. Una línea de trabajo muy prometedora es la propuesta en [266], en la que se presenta una representación dual, numérica y conceptual, de la información que podría ser útil para combinar los aspectos de recuperación de información dentro del módulo CBR de un comportamiento, y de construcción de comportamientos más complejos a través de la emergencia. En este sentido habría que estudiar como agrupar las representaciones conceptuales de nivel inferior para convertirlas en información conceptual básica de un nivel superior. También se tiene previsto un análisis mas exhaustivo de las métricas de distancias a considerar en la recuperación de casos más similares de la base CBR, al menos en el dominio del problema tratado, analizando cuales son las más adecuadas a las características de los atributos del caso. Adicionalmente, se pretende analizar la idoneidad de aplicar diferentes distancias a grupos de atributos de los casos, según las conclusiones obtenida

en la etapa previa; estas distancia "sub-locales" se combinarían finalmente en una distancia global de forma ponderada a la importancia del atributo o grupo de atributos en el comportamiento y la situación actual experimentada.

Posteriormente, se tratará de mejorar la etapa de mantenimiento del sistema CBR asociado a cada comportamiento considerando, además de la adquisición de nuevos conocimientos por experiencia, la re-evaluación de la utilidad de la información ya aprendida, utilidad que podría ser un factor más en el proceso de recuperación *CBR*, y que podría llevar a incluir un aspecto de "olvido" de información que se considere poco útil para los objetivos del comportamiento.

Finalmente, y en lo relativo a la emergencia de comportamientos, se estudiará la transformación del módulo de organización emergente de comportamientos básicos, desde un sistema basado en reglas de decisión, a otro basado en aprendizaje *CBR*. Este nuevo módulo aprendería el peso o influencia que se tiene que aplicar a los comportamientos a combinar, o la selección de un determinado comportamiento particular, a partir de un entrenamiento por observación/imitación, similar al empleado para el diseño de los módulos individuales. De esta forma se ajustaría mejor el modelo de arquitectura a la estructura de "nido" propuesta en el modelo de inteligencia humana de [1], en la cual todos los componentes tienen una misma estructura y funcionamiento y solo se diferencia en el nivel de abstracción y alcance temporal de los conceptos manejados. A este respecto, será necesario determinar la forma de representar las unidades de información que se obtienen como salida de los comportamientos inferiores, en forma de entradas o atributos de casos *CBR* en los niveles superiores. Pensamos que, de nuevo, la conciliación entre la representación numérica y conceptual jugará un papel clave en este proceso.

Listado de publicaciones

Se enumera a continuación la producción científica relacionada con el trabajo desarrollado en esta Tesis:

- I. Herrero, C. Urdiales, J.M. Peula, I. Sanchez-Tato, y F.Sandoval. A guided learning strategy for vision based navegation of 4-legged robots. *AI Communications*, 19(2):127 – 136, 2006
- M.I. Sánchez Tato, J.M. Peula, C. Urdiales, I. Herrero, y F. Sandoval. Navegación por aprendizaje mediante marcas de visión. En *XXI Simposium Unión Ciéntifica Internacional de Radio, URSI2006*, 2006
- M.I. Sánchez Tato, J.M. Peula, C. Urdiales, I. Herrero, y F. Sandoval. Coordinación por aprendizaje mediante localización basada en marcas de visión. En *XXII Simposium Unión Ciéntifica Internacional de Radio, URSI2007*, 2007
- M. García Beltrán. Corrección de vignetting en el sistema de visión de un AIBO-ERS7. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, 2009
- J.M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato, y 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, June 2009
- J.M. Peula, C. Urdiales, I. Herrero, y F. Sandoval. Implicit robot coordination using Case-Based Reasoning behaviors. En *International Conference on Intelligent Robots and Systems, IROS13*, pags. 5929–5934. IEEE/RSJ, 2013
- S. Figueroa. Control de la cabeza de un robot AIBO mediante gafas de Realidad Virtual. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, 2014
- I. Herrero, C. Urdiales, J.M. Peula, y F. Sandoval. A bottom-up robot architecture based on learnt behaviors driven design. En Ignacio Rojas, Gonzalo Joya, y Andreu Catala, editors, *Advances in Computational Intelligence*, vol. 9094 of *Lecture Notes in Computer Science*, pags. 159–170. Springer International Publishing, 2015

Bibliografía

[1] J.Hawkins y S. Blakeslee. *On Intelligence*. Owl Books, 2004.

[2] R.R. Murphy. *Introduction to AI Robotics*. MIT Press, Cambridge, MA, USA, 1st edition, 2000.

[3] A. Poncela, C. Urdiales, y F. Sandoval. A CBR approach to behaviour-based navigation for an autonomous mobile robot. En *2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007.

[4] M. Wang y J.N.K. Liu. Fuzzy logic-based real-time robot navigation in unknown environment with dead ends. *Robotics and Autonomous System*, 56(7):625–643, July 2008.

[5] J.C. Murray, H.R. Erwin, y S. Wermter. Robotic sound-source localisation architecture using cross-correlation and recurrent neural networks. *Neural Networks*, 22(2):173–189, 2009.

[6] R.A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[7] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. En *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2, pags. 500–505, Marzo 1985.

[8] E. Aguirre y A. González. Fuzzy behaviors for mobile robot navigation: design, coordination and fusion. *International Journal of Approximate Reasoning*, 25(3):255 – 289, 2000.

[9] A. Zhu y S.X. Yang. Neurofuzzy-based approach to mobile robot navigation in unknown environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(4):610–621, July 2007.

[10] K.H. Low, W.K. Leow, y M.H. Ang Jr. A hybrid mobile robot architecture with integrated planning and control. En *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, AAMAS '02, pags. 219–226, New York, NY, USA, 2002. ACM.

[11] E.J. Pérez Rodríguez. *Arquitectura de Navegación Distribuida para Agentes Robóticos*. Tesis Doctoral, Departamento de Tecnología Electrónica, Universidad de Málaga, 2006.

[12] A. Poncela González. *Exploración completa basada en comportamientos cooperativos para un agente autónomo móvil*. Tesis Doctoral, Departamento de Tecnología Electrónica, Universidad de Málaga, 2008.

[13] R.C. Arkin. *Behaviour-Based Robotics*. MIT Press, 1998.

[14] J.S. Albus. An introduction to intelligent and autonomous control. Capítulo: A Reference Model Architecture for Intelligent Systems Design, pags. 27–56. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[15] H. Hu y M. Brady. A parallel processing architecture for sensor based control of intelligent mobile robots. *Robotics and Autonomous systems*, 17:235 – 257, 1996.

[16] E. Coste-Maniere y R. Simmons. Architecture, the backbone of robotic systems. En *Proceedings IEEE International Conference on Robotics and Automation, 2000 (ICRA '00)*, vol. 1, pags. 67–72, 2000.

[17] R.P. Bonasso, D. Kortenkamp, D.P. Miller, y M.Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:237–256, 1995.

[18] R. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1), February 1994.

[19] R. Alami, R. Chatila, S. Fleury, M. Ghallab, y F. Ingrand. An architecture for autonomy. *International Journal of Robotics Research*, 17:315–337, 1998.

[20] M. Lindstrom, A. Oreback, y H.I. Christensen. BERRA: a research architecture for service robots. En *Proceedings IEEE International Conference on Robotics and Automation, 2000 (ICRA '00)*, vol. 4, pags. 3278–3283, 2000.

[21] O. Fuentes, R.P.N. Rao, y M. Van Wie. Hierarchical learning of reactive behaviors in an autonomous mobile robot. En *IEEE International Conference on Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century*, vol. 5, pags. 4691–4695, Oct 1995.

[22] M. Carreras, P. Ridao, R. Garcia, y Z. Ursulovici. Learning reactive robot behaviors with a Neural-Q learning approach, 2002.

[23] M.J. Mataric. *Interaction and Intelligent Behavior*. Tesis Doctoral, Department of Electronic Engineering and Computer Sciencie, 1994.

[24] T.L. Dean y M.P. Wellman. *Planning and Control*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[25] A. Meystel. An Introduction to Intelligent and Autonomous Control. Capítulo: Nested Hierarchical Control, pags. 129–161. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[26] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23, 1986.

[27] R.A. Brooks. Intelligence without reason. En *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pags. 569–595, Sydney, Australia, 1991. Morgan Kaufmann.

[28] R.C. Arkin. Motor schema–based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112, August 1989.

[29] M.J. Mataric. *The Biology and Technology of Intelligent Autonomous Agents*, Capítulo: From Local Interactions to Collective Intelligence, pags. 275–295. Nº 144 en NATO ASI Series F. Springer-Verlag, 1995.

[30] M. A. Goodrich. Potential Fields Tutorial. `http://borg.cc.gatech.edu/ipr/files/goodrich_Potential_fields.pdf`, 2002. Online; recuperado en Octubre de 2015.

[31] Y. Koren y J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. En *Proceedings 1991 IEEE International Conference on Robotics and Automation*, vol. 2, pags. 1398–1404, Abr. 1991.

[32] R.C. Arkin y T. Balch. AuRA: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:175–189, 1997.

[33] R.R. Murphy. Biological and cognitive foundations of intelligent sensor fusion. *IEEE Transactions on Systems, Man, and Cybernetics*, 26:42–51, 1996.

[34] R.J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. Tesis Doctoral, Yale University, New Haven, CT, USA, 1989. AAI9010653.

[35] R.J. Firby. Task networks for controlling continuous processes. En *Proceedings of the Second International Conference on AI Planning Systems*, pags. 49–54, 1994.

[36] K. Konolige, K. Myers, E. Ruspini, y A. Saffiotti. The Saphira architecture: A design for autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:215–235, 1997.

[37] J. Hawkins y D. George. Hierarchical Temporal Memory. `http://numenta.com/learn/hierarchical-temporal-memory-white-paper.html`, 2006. [Online; recuperado en Julio de 2015].

[38] M.O. Franz y H.A. Mallot. Biomimetic robot navigation. *Robotics and utonomous Systems*, 30:133–153, 2000.

[39] R.W. Burkhardt. *Patterns of Behavior: Konrad Lorenz, Niko Tinbergen, and the Founding of Ethology*. University of Chicago Press, Chicago, 2005.

[40] K. Lorenz y N. Tinbergen. *Taxis und Instinkthandlung in der Eirollbewegung der Graugans*. Parey, 1938.

[41] K. Lorenz. The comparative method in studying innate behavior patterns. En *Symposium of the Society of Experimental Biology*, pags. 221– 268, 1950.

[42] J. J. Gibson. *Perceiving, Acting, and Knowing*, Capítulo: The Theory of Affordances, pags. 274 – 288. Lawrence Erlbaum Associates, 1977.

[43] U. Neisser. Direct perception and recognition as distinct perceptual systems. Cognitive Sciencie Society, 1991.

[44] M.J. Mataric. Behavior-based control: Main properties and implications. En *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, pags. 46–54, 1992.

[45] R. Clarke. Asimov's Laws of Robotics: Implications for Information Technology. *Computer*, 27(1):57–66, January 1994.

[46] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. Bradford Books. MIT Press, 1986.

[47] Y. Faihe y J.-P. Müller. Analysis and design of robots behavior: Towards a methodology. En Andreas Birk y John Demiris, editors, *Learning Robots*, vol. 1545 of *Lecture Notes in Computer Science*, pags. 46–61. Springer Berlin Heidelberg, 1998.

[48] A. Bonarini. The Body, the Mind or the Eye, first? En Manuela Veloso, Enrico Pagello, y Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, vol. 1856 of *Lecture Notes in Computer Science*, pags. 210–221. Springer Berlin Heidelberg, 2000.

[49] Y. Bestaoui Sebbane. Deterministic decision making. En *Planning and Decision Making for Aerial Robots*, vol. 71 of *Intelligent Systems, Control and Automation: Science and Engineering*, pags. 171–244. Springer International Publishing, 2014.

[50] K. Rosenblatt y D. Payton. A fine-grained alternative to the Subsumption Architecture for mobile robot control. En *Proceedings of the AAAI Symposium on Robot Navigation*, pags. 317–324, 1989.

[51] M.J. Mataric y R.A. Brooks. Learning a distributed map representation based on navigation behaviors. En *Proceedings of 1990 USA Japan Symposium on Flexible Automation*, pags. 499–506, 1990.

[52] I. Harvey, P. Husbands, D. Cliff, A. Thompson, y N. Jakobi. Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.

[53] K. Mülling, J. Kober, O. Kroemer, y J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[54] J. Kober, B. Mohler, y J. Peters. Learning perceptual coupling for motor primitives. En *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008 (IROS 2008)*, pags. 834–839. IEEE, 2008.

[55] J. Kober, J. Andrew Bagnell, y J. Peters. Reinforcement learning in Robotics: A survey. *The International Journal of Robotics Research*, 2013.

[56] C. Lazarus y H. Hu. Using genetic programming to evolve robot behaviours. En *Proceedings of the 3rd British Conference on Autonomous Mobile Robotics and Autonomous Systems*, 2001.

[57] D. Bajaj y M.H. Ang Jr. An incremental approach in evolving robot behavior. En *Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision*, 2000.

[58] V.R. Cruz-Álvarez, F. Montes-Gonzalez, E. Mezura-Montes, y J. Santos. Robotic behavior implementation using two different differential evolution variants. En Ildar Batyrshin y Miguel González Mendoza, editors, *Advances in Artificial Intelligence*, vol. 7629 of *Lecture Notes in Computer Science*, pags. 216–226. Springer Berlin Heidelberg, 2013.

[59] T. Jin, Y. Son, y H. Hashimoto. Mobile robot control using fuzzy-neural-network for learning human behavior. En Irwin King, Jun Wang, Lai-Wan Chan, y DeLiang Wang, editors, *Neural Information Processing*, vol. 4234 of *Lecture Notes in Computer Science*, pags. 874–883. Springer Berlin Heidelberg, 2006.

[60] L.I. Helgadottir, J. Haenicke, T. Landgraf, R. Rojas, y M.P. Nawrot. Conditioned behavior in a robot controlled by a spiking neural network. En *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pags. 891–894, Nov 2013.

[61] B. Johnen, C. Scheele, y B. Kuhlenkotter. Learning robot behavior with artificial neural networks and a coordinate measuring machine. En *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pags. 208–213, Dec 2011.

[62] E.J. Pérez, C. Urdiales, F. Sandoval, y J. Vazquez-Salceda. A CBR-strategy for autonomous reactive navigation learning. En *Proceedings of the International Symposium on Robotics and Applications*, pags. 179 – 186, Seville, Spain, 2004.

[63] J.M. Peula, C. Urdiales, I. Herrero, I. Sánchez-Tato, y 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, June 2009.

[64] A. Marczyk. Algoritmos genéticos y computación evolutiva. `http://the-geek.org/docs/algen/`, 2004. [Online; recuperado en Octubre de 2015].

[65] J.D. Olden y D.A. Jackson. Illuminating the black box: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(1-2):135 – 150, 2002.

[66] F. Michaud. Selecting behaviors using fuzzy logic. En *Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on*, vol. 1, pags. 585–592. IEEE, 1997.

[67] A.D. Mali. On the evaluation of agent behaviors. *Artificial Intelligence*, 143(1):1 – 17, 2003.

[68] C. Breazeal, N. DePalma, J. Orkin, S. Chernova, y M. Jung. Crowdsourcing Human-Robot interaction: New methods and system evaluation in a public environment. *Journal of Human-Robot Interaction*, 2(1):82–111, 2013.

[69] M.J. Mataric. Integration of representation into goal-driven behavior-based robots. *Robotics and Automation, IEEE Transactions on*, 8(3):304–312, Jun 1992.

[70] L.E. Parker. Heterogeneous multi-robot cooperation. Informe técnico, 1994.

[71] D.W. Payton, D. Keirsey, D.M. Kimble, J. Krozel, y J.K. Rosenblatt. Do whatever works: A robust approach to fault-tolerant autonomous control. *Applied Intelligence*, 2(3):225–250, 1992.

[72] C. Ferrell. Robust agent control of an autonomous robot with many sensors and actuators. Informe técnico, Many Sensors and Actuators, Master's thesis, MIT, Department of EECS, 1993.

[73] T. Fukai y S. Tanaka. A simple neural network exhibiting selective activation of neuronal ensembles: from winner-take-all to winners-share-all. *Neural computation*, 9(1):77–97, 1997.

[74] S. Snaith y O. Holland. *From Animals to Animats*. MIT Press, Cambridge, 1991.

[75] P. Maes y R.A. Brooks. Learning to coordinate behaviors. En *Proceedings of AAAI-90*, vol. 2, pags. 796–802, Boston, MA, jul 1990. AAAI Press/The MIT Press.

[76] F. Michaud y M.J. Mataric. Behavior evaluation and learning from an internal point of view. En *Proceedings of FLAIRS-97, Daytona*, pags. 11–14, 1997.

[77] P. Redgrave, T.J. Prescott, y K. Gurney. The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience*, 89(4):1009 – 1023, 1999.

[78] M. Wahde. The Utility Function method for behaviour selection in autonomous robots. En David Bradley y David W. Russell, editors, *Mechatronics in Action*, pags. 135–156. Springer London, 2010.

[79] J. Von Neumann y O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[80] J. Pettersson. *Generating motor behaviors for bipedal robots using biologically inspired computation methods*. Institutionen för maskinoch fordonssystem, Chalmers tekniska högskola, 2003.

[81] J. Pettersson y M. Wahde. Application of the Utility Function method for behavioral organization in a locomotion task. *Transactions on Evolutionary Computation*, 9(5):506–521, October 2005.

[82] M. Botros. Evolving complex robotic behaviors using genetic programming. En Nadia Nedjah, LuizadeMacedo Mourelle, y Ajith Abraham, editors, *Genetic Systems Programming*, vol. 13 of *Studies in Computational Intelligence*, pags. 173–191. Springer Berlin Heidelberg, 2006.

[83] M.M. Joshi y M.A. Zaveri. Reactive navigation of autonomous mobile robot using neuro-fuzzy system. *International Journal of Robotics and Automation (IJRA)*, 2(3):128, 2011.

[84] M. Likhachev, M. Kaess, y R.C. Arkin. Learning behavioral parameterization using spatio-temporal case-based reasoning. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)*, pags. 1282–1289, 2002.

[85] R.C. Arkin y D. MacKenzie. Temporal coordination of perceptual algorithms for mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 3(10):276–286, 1994.

[86] M.J. Matarić. Reinforcement learning in the multi-robot domain. En *Robot colonies*, pags. 73–83. Springer, 1997.

[87] T. Balch. Integrating RL and behavior-based control for soccer. En *RoboCup-97: Proceedings of the First Robot World Cup Soccer Games and Conferences*. Citeseer, 1997.

[88] G. D. Konidaris y G. M. Hayes. An architecture for behavior-based reinforcement learning. *Adaptive Behavior*, 13(1):5–32, 2005.

[89] B.M. El-Bagoury, A.-B. Salem, y H.-D. Burkhard. Hierarchical Case-Based Reasoning behavior control for humanoid robot. *Annals of the University of Craiova-Mathematics and Computer Science Series*, 36(2):131–140, 2009.

[90] M.W. Floyd, M. Drinkwater, y D.W. Aha. Case-Based behavior adaptation using an inverse trust metric. Informe técnico, DTIC Document, 2014.

[91] S. Nagata, M. Sekiguchi, y K. Asakawa. Mobile robot control by a structured hierarchical neural network. *Control Systems Magazine, IEEE*, 10(3):69–76, 1990.

[92] Y.-K. Na y S.-Y. Oh. Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification. *Autonomous Robots*, 15(2):193–206, 2003.

[93] P. Maes. How to do the right thing? *Connection Sciencie Journal. Special Issue on Hybrid sistems*, 3(1):291 – 323, 1989.

[94] J. Rosenblatt. *DAMN: a distributed architecture for mobile navigation*. Tesis Doctoral, Robotics Institute, Carnegie Mellon University, 1997.

[95] M.J. Degnan. Aristotle's logic. *Philosophical Books*, 35(2):81–89, 1994.

[96] D.K. Bloch. *Aristotle on Memory and Recollection: Text, Translation, Interpretation, and Reception in Western Scholasticism*, vol. 1. Brill, 2007.

[97] H.C. Warren. *A History of the Association Psychology*. Charles Scribner's Sons, 1921.

[98] A. Bonner. *The Art and Logic of Ramon Llull: A User's Guide*. Studien Und Texte Zur Geistesgeschichte Des Mittelalters. Brill, 2007.

[99] R. Llull. Raimundus Lullus Ars Magna Tree. https://commons.wikimedia.org/wiki/File:Ramon_Llull_-_Ars_Magna_Tree_and_Fig_1.png, 1305. [Imagen Online; recuperado en Octubre de 2015].

[100] E.J. Aiton. *Leibniz A Biography*. Taylor & Francis, 1985.

[101] G.W. Leibniz. *Sämtliche Schriften und Briefe*. O. Reichl, Darmstadt, 1923.

[102] D. Berlinski. *The Advent of the Algorithm: The 300-Year Journey from an Idea to the Computer*. Harcourt, 2001.

[103] A.M. Turing. Computability and λ-definability. *The Journal of Symbolic Logic*, 2(04):153–163, 1937.

[104] A. Church. A set of postulates for the foundation of Logic. *Annals of mathematics*, 33(2):346–366, 1932.

[105] Bilby. Turing test. https://commons.wikimedia.org/wiki/File:Turing_Test_version_3.png, 2008. [Imagen Online; recuperado en Octubre de 2015].

[106] A.M. Turing. Computing machinery and intelligence. *Mind*, pags. 433–460, 1950.

[107] S.J. Russell y P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[108] J. Levin. *Stanford Encyclopedia of Philosophy*, Capítulo: Functionalism. Stanford University, 2004.

[109] E.B. Baum. *What Is Thought?* MIT Press, 2004.

[110] P. Dayan y L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005.

[111] A. Zell. *Simulation neuronaler Netze*. Oldenbourg, 1997.

[112] J.R. Searle. Minds, brains and programs. *Behavioral and Brain Sciences*, 3:417 – 424, 1980.

[113] N. Block. *Perception and Cognition: Issues in the Foundations of Psychology*, Capítulo: Troubles with Functionalism. Minneapolis: University of Minnesota Press, 1978.

[114] G. Graham. *Stanford Encyclopedia of Philosophy*, Capítulo: Behaviorism. Stanford University, 2005.

[115] L.R. Medsker. *Hybrid Intelligent Systems*. Kluwer Academic Publishers, 1995.

[116] J. Garner. *Stanford Encyclopedia of Philosophy*, Capítulo: Connectionism. Stanford University, 2004.

[117] S. Haykin. *Neural Networks: A Comprehensive Foundation (2ed)*. Prentice Hall, 1999.

[118] K. M. Tadiou. Basic structure of an artificial neural network (ANN). `http://futurehumanevolution.com/artificial-intelligence-future-human-evolution/artificial-neural-networks`, n.d. [Imagen Online; recuperado en Julio de 2015].

[119] D. E. Rumelhart, G. E. Hinton, y R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. Capítulo: Learning Internal Representations by Error Propagation, pags. 318–362. MIT Press, Cambridge, MA, USA, 1986.

[120] D.J. Felleman y D.C. Van Essen. Distributed hyerarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1 – 47, Jan./Feb. 1991.

[121] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[122] R.K. Lindsay, B.G. Buchanan, y E.A. Feigenbaum. DENDRAL: A case study of the first expert system for scientific hypothesis formation. *Artif. Intell.*, 61(2):209–261, June 1993.

[123] E.H. Shortliffe. *Mycin: A Rule-based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection.* Tesis Doctoral, Stanford, CA, USA, 1975. AAI7513600.

[124] P.E. Hart, R.O. Duda, y Stanford Research Institute. Artificial Intelligence Group. *Prospector: A Computer-based Consultation System for Mineral Exploration.* Technical note. Stanford Research Inst., 1977.

[125] L. Medsker. Design and development of hybrid neural network and expert systems. En *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 3, pags. 1470–1474, Jun 1994.

[126] C.W. Omlin y C.L. Giles. Rule revision with recurrent neural networks. *Knowledge and Data Engineering, IEEE Transactions on*, 8(1):183–188, Feb 1996.

[127] A. Newell y H.A. Simon. *Human problem solving.* Prentice-Hall, 1972.

[128] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, New York, NY, USA, 2000.

[129] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[130] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[131] M. Minsky. Minsky's frame system theory. En *TINLAP '75: Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pags. 104–116, Morristown, NJ, USA, 1975. Association for Computational Linguistics.

[132] M. Negnevitsky. *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2001.

[133] J.H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.

[134] I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Library Translation.*, 1965.

[135] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[136] A. Aamodt y E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–52., 1994.

[137] E. Black. Behaviorism as a learning theory. Informe Técnico inst1229 CL, UH. EDU, 1995.

[138] R.S. Sutton y A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[139] Learning-theories.com. Cognitivism. `http://www.learning-theories.com/cognitivism.html`, 2007. [Online; recuperado en Diciembre de 2014].

[140] J. Bruner. *The Culture of Education*. Harvard University Press, 1996.

[141] Y.X. Zhong. Structuralism? Functionalism? Behaviorism? or Mechanism? - Looking backward and forward on better approach to AI -. En *Granular Computing, 2006 IEEE International Conference on*, pags. 32– 37, 2006.

[142] K. Murphy. A brief introduction to graphical models and Bayesian networks. `http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`, 1998. Online; recuperado en Julio de 2015.

[143] G. Roth y U. Dicke. Evolution of the brain and intelligence. *Trends in cognitive sciences*, 9(5):250–257, May 2005.

[144] Power Analytics. Self-learning power analytics system approved in factory acceptance test for offshore oil platform. `http://www.poweranalytics.com/pa_articles/self_learning.php`, 2010. [Online; recuperado 13-Marzo-2015].

[145] Lockheed Martin. Lockheed martin brain-inspired computation: research overview. `www.atl.external.lmco.com/papers/1597.pdf`, 2008. [Online; recuperado 13-Marzo-2015].

[146] Venture Beat. Palm founders are back with grok, a neuroscience-inspired big data engine. `http://venturebeat.com/2012/12/18/numenta-grok/`, 2012. [Online; recuperado 13-Marzo-2015].

[147] U.S. Dpt. of Energy Pacific Northwest National Laboratory. An advanced reasoning system for enhanced decision making. `http://i4.pnnl.gov/focusareas/acamp.stm`, 2011. [Online; recuperado 13-Marzo-2015].

[148] L. Frank y R. Hohimer. Modeling human behavior to anticipate insider attacks. *Journal of Strategic Security*, 4(2):3, 2011.

[149] L. Skála. Hybrid decision-making system of artificial creature combining planner and neural network. Proyecto Fin de Carrera, Department of Cybernetics, Czcech Technical University in Prague, 2013.

[150] D. Kadleček. *Learning motivation driven reinforcement and hierarchies: Automatic Creation of Behaviors*. Tesis Doctoral, Department of Cybernetics, Czcech Technical University in Prague, 2008.

[151] S. Grand. Moving AI out of its infancy: Changing our preconceptions. *IEEE Intelligent Systems*, 19(6):74 – 77, 2004.

[152] P. Dayan. Palmistry. *PLoS Biol*, 11(2):e394, 2004.

[153] B. Goertzel. On biological and digital intelligence. `http://www.goertzel.org/dynapsyc/2004/OnBiologicalAndDigitalIntelligence.htm`, 2004. [Online; recuperado en Julio de 2015].

[154] M. Kruusmaa. Global navigation in dynamic environments using Case-Based Reasoning. *Autonomous Robots*, 14(1):71 – 91, 2003.

[155] H. Liu y H. Iba. *Genetic and Evolutionary Computation, GECCO 2004*, Capítulo: Humanoid Robot Programming Based on CBR Augmented GP, pags. 708–709. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004.

[156] C. Urdiales y U. Cortés. Situated robotics: from learning to teaching by imitation. *Cognitive Processing*, 6:196 – 201, 2005.

[157] C. Urdiales, E. J. Perez, J. Vázquez-Salceda, M. Sànchez-Marrè, y F. Sandoval. A purely reactive navigation scheme for dynamic environments using Case-Based Reasoning. *Auton. Robots*, 21(1):65–78, 2006.

[158] R. Ros, M. Veloso, R. López de Màntaras, C. Sierra, y J.L. Arcos. Retrieving and reusing game plays for robot soccer. En *Lecture Notes in Computer Science, Proceedings of ECCBR-2006*, Oludeniz, Turkey, 2006.

[159] R.C. Schank y R.P. Abelson. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ, 1977.

[160] R.C. Schank. *Dynamic Memory*. Cambridge University Press, New York, 1982.

[161] J.R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22:261–295, 1983.

[162] J. Kolodner. Experiential processes in natural problem solving. Informe Técnico GIT-ICS-85/23, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, 1985.

[163] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Scienie*, 7:155 – 170, 1983.

[164] J. G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. En R. S. Michalski, J. G. Carbonell, y T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach: Volume II*, pags. 371–392. Kaufmann, Los Altos, CA, 1986.

[165] B. Ross. Some psychological results on Case-Based Reasoning. En *Proceedings of the workshop on Case-Based Reasoning*, pags. 144–148, Morgan Kaufmann, 1989. DARPA, Morgan Kaufmann.

[166] L. Wittgenstein. *Philosophical investigations.* Blackwell, 1953.

[167] J. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243–280, 1983.

[168] J. Kolodner. Reconstructive memory, a computer model. *Cognitive Science*, 7:281–328, 1983.

[169] R.L. Simpson Jr. A computer model of Case-Based Reasoning in problem solving: an investigation in the domain of dispute mediation. Informe técnico, Georgia Institute of Technology, Atlanta, GA, USA, 1985.

[170] K. Sycara. Using Case-Based Reasoning for plan adaptation and repair. En DARPA, editor, *Proceedings Case-Based Reasoning Workshop*, pags. 425–434, Clearwater Beach, Florida, 1988. DARPA, Morgan Kaufmann.

[171] K.J. Hammond. *Case-Based planning: Viewing plannig as a memory task.* Academic Press., 1989.

[172] T.R. Hinrichs. *Problem solving in open worlds.* Lawrence Erlbaum Associates, 1992.

[173] R. Bareiss. *PROTOS; a unified approach to concept representation, classification and learning.* Technical report ai88-83, Dep. of Computer Sciences .University of Texas at Austin, 1988.

[174] K. Branting. Exploiting the complementarity of rules and precedents with reciprocity and fairness. En *Proceedings from the Case-Based Reasoning Workshop 1991*, pags. 39–50, Washington DC, 1991. DARPA, Morgan Kaufmann.

[175] K. Ashley. *Modeling legal arguments: Reasoning with cases and hypotheticals.* Bradford Books, Cambridge, 1991.

[176] C.B. Skalak y Rissland E. Arguments and cases: An inevitable twining. *Artificial Intelligence and Law, An International Journal*, 1:3 – 48, 1992.

[177] P. Koton. *Using experience in learning and problem solving.* Mit/lcs/tr-441, Massachusetts Institute of Technology, Laboratory of Computer Science, October 1989.

[178] S. Sharma y D. Sleeman. REFINER; a Case-Based differential diagnosis aide for knowledge acquisition and knowledge refinement. En *EWSL 88; Proceedings of the Third European Working Session on Learning*, pags. 201–210. Pitman, 1988.

[179] R. Oehlmann. Learning causal models by self-questioning and experimentation. En *AAAI-92 Workshop on Communicating Sientific and Technical Knowledge. American Association of Artificial Intelligence*, 1992.

[180] K.D. Althoff. Knowledge acquisition in the domain of cnc machine centers; the MOLTKE approach. En Jean-Gabriel Ganascia John Boose, Brian Gaines, editor, *EKAW-89; Third European Workshop on Knowledge-Based Systems*, pags. 180–195., Paris, Francia, 1989.

[181] M.M. Richter y S. Weiss. Similarity, uncertainty and Case-Based Reasoning in PATDEX. En *Proc. Automated Reasoning: Essays in honour of Woody Bledsoe*, pags. 249–265, 1991.

[182] A. Aamodt. *A Knowledge intensive approach to problem solving and sustained learning*. University microfilms pub 92-08460., Norwegian Institute of Technology, University of Trondheim, 1991.

[183] I.D. Watson y S. Abdullah. Developing Case-Based Reasoning systems: A case study in diagnosing building defects. En *Proc. IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, nº 1994/057, pags. 1/1–1/3., 1994.

[184] S. Yang y D. Robertson. A Case-Based Reasoning system for regulatory information. En *Proc. IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, nº 1994/057, pags. 3/1–3/3, 1994.

[185] C.J. Moore, M.S. Lehane, y C.J. Proce. Case-Based Reasoning for decision support in Engineering design. En *Proc. IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, nº 1994/057, pags. 4/1–4/4, 1994.

[186] Enric Plaza y R. Lopez de Mántaras. *A Case-Based apprentice that learns from fuzzy examples*, pags. 420–427. Elsevier North-Holland, Inc., New York, NY, USA, 1990.

[187] B. López y E. Plaza. Case-Base planning for medical diagnosis. methodologies for intelligent systems. En *7th. International Symposium, ISMIS-93. Lecture Notes in Artificial Intelligence*, vol. 689. Springer Verlag, 1993.

[188] R. Ros, R. López De Mántaras, J.L. Arcos, y M. Veloso. Team playing behavior in robot soccer: A case-based reasoning approach. En Springer-Verlag, editor, *Lecture Notes in Computer Science, Proceedings of the EICCBR 2007*, nº 4626, pags. 46–60, 2007.

[189] U. Cortes, C. Urdiales, R. Annicchiaricco, C. Barrue, A.B. Martinez, y C. Caltagirone. *Advanced Computational Intelligence Paradigms in Healthcare*, vol. 1, Capítulo: Assistive Wheelchair Navigation: A Cognitive View, pags. 165–183. Springer, 2007.

[190] L.K. Branting y D.W. Aha. Stratified Case-Based Reasoning: Reusing hierarchical problem solving episodes. En *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence*, Montreal, Canada, 1995.

[191] S. Fox y D.B. Leake. Combining Case-Based planning and introspective reasoning. En *Proceedings of the 6th Midwest Artificial Intelligence and Cognitive Science Society Conference*, Carbondale, IL, 1995.

[192] K.Z. Haigh y M. Veloso. Route planning by analogy. En *Proceedings of Int. Conf. on Case-Based Reasoning*, pags. 160 – 180, Berlin, 1995. Springer Verlag.

[193] M. Likhachev y R.C. Arkin. Spatio-temporal Case-Based Reasoning for behavioral selection. En *Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pags. 1627 – 1634, Seoul, Korea, 2001.

[194] A. Ram y J.C. Santamaria. A multistrategy Case-Based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. En *Proceedings of the 2nd Int. Workshop on Multistrategy Learning*, Harpers Ferry, West Virginia, 1993.

[195] J. Wendler, S. Brüggert, H.-D. Burkhard, y H. Myritz. Fault-tolerant self localization by Case-Based Reasoning. En P. Stone, T. Balch, y G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, vol. 2019 of *LNAI*, pags. 259–26. Springer, 2000.

[196] I. Herrero, C. Urdiales, J.M. Peula, I. Sanchez-Tato, y F.Sandoval. A guided learning strategy for vision based navegation of 4-legged robots. *AI Communications*, 19(2):127 – 136, 2006.

[197] J. Wendler, P. Gugenberger, y M. Lenz. CBR for dynamic situation assessment in an agent-oriented setting. En D. Aha y J. Daniels, editors, *Proc. of the AAAI-98 Workshop on Case-Based Reasoning Integrations*, Madison, USA, 1998.

[198] A. Karol, B. Nebel, C. Stanton, y M.-A. Williams. Case-Based game play in the Robocup four-legged league part I: The theoretical model. En D. Polani, B. Browning, A. Bonarini, y K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, vol. 3020 of *Lecture Notes in Computer Science*, pags. 739–747. Springer Berlin Heidelberg, 2004.

[199] M.M. Richter. Knowledge containers. En Ian Watson, editor, *Readings in Case-Based Reasoning*. Morgan Kaufmann Publishers, 2003.

[200] L. Birnbaum y G. Collings. Remindings and engineering design themes: A case study in indexing vocabulary. En *Proceedings of the Second Workshop on Base-Based Reasoning*, Pensacola Beach, FL, 1989.

[201] J. Kolodner. *Case-Based Reasoning*. Morgan-Kauffman, 1993.

[202] M. Lebowitz. Experiments with incremental concept formation: Unimem. *Machine Learning*, 2(2):103–138, 1987.

[203] E. Simoudis y P. Miller. Automated support for developing retrieve-and-propose systems. En *Proceedings of the 1993 AAAI Workshop on Case-Based Reasoning*, pags. 147–152, Washington, DC, 1993.

[204] H.-D. Burkhard y M.M. Richter. Soft computing in Case-Based Reasoning. Capítulo: On the Notion of Similarity in Case Based Reasoning and Fuzzy Theory, pags. 29–45. Springer-Verlag, London, UK, UK, 2001.

[205] A. Stahl. Learning similarity measures: A formal view based on a generalized CBR model. En Héctor Muñoz Avila y Francesco Ricci, editors, *ICCBR*, vol. 3620 of *Lecture Notes in Computer Science*, pags. 507–521. Springer, 2005.

[206] D. Wettschereck y D. W. Aha. Weighting features. En Manuela Veloso y Agnar Aamodt, editors, *Case-Based Reasoning, Research and Development, First International Conference*, pags. pages 347–358, Berlin, 1995. Springer Verlag.

[207] D. Patterson, S.S. Anand, y J. Hughes. A knowledge light approach to similarity maintenance for improving Case-Base competence. En Mirjam Minor, editor, *ECAI Workshop Notes*, pags. 65–78, Berlin, 2000. Humboldt University.

[208] S. Boriah, V. Chandola, y V. Kumar. Similarity measures for categorical data: A comparative evaluation. En *SDM*, pags. 243–254. SIAM, 2008.

[209] K. Racine y Q. Yang. Maintaining Unstructured Case Bases. En D.B. Leake y E. Plaza, editors, *Case-Based Reasoning Research and Development.*, pags. 553–564. Springer Verlag, 1997.

[210] B. Smyth y E. McKenna. Modelling the competence of Case-Based Reasoning systems. En Barry Smyth and Padraigh Cunningham, editor, *Advances in Case-Based Reasoning, Proceedings of the 4th European Workshop on Case-Based Reasoning, EWCBRY98*, pags. 208–220, Berlin, 1998. Springer-Verlag.

[211] D.B. Leake y D.C. Wilson. When experience is wrong: Examining CBR for changing tasks and environments. En Klaus-Dieter Althoff, Ralph Bergmann, y Karl Branting, editors, *ICCBR*, vol. 1650 of *Lecture Notes in Computer Science*, pags. 218–232. Springer, 1999.

[212] K. Hanney y M. Keane. Learning adaptation rules from cases. En I. Smyth y B. Faltings, editors, *Advances in Case-Based Reasoning.* Springer Verlag, 1996.

[213] K. Hanney y M. Keane. The adaptation knowledge bottleneck: How to ease it by learning from cases. En *Proceedings of the Second International Conference on Case-Based Reasoning.* Springer Verlag, 1997.

[214] W. Wilke, I. Vollrath, y R. Bergmann. Using knowledge containers to model a framework for learning adaptation knowledge. En Dietrich Wettschereck y David W. Aha, editors, *European Conference on Machine Learning. MLNet Workshop Notes - Case-Based Learning: Beyond Classification of Feature Vectors*, pags. 68–75, Naval Research Laboratory, Washington, D. C., USA, 1997. Navy Center for Applied Research in Artificial Intelligence.

[215] D.A. Sharaf-Eldeen, I.F. Moawad, K.E. Bahnasy, y M.E. Khalifa. Learning and applying range adaptation rules in Case-Based Reasoning systems. En Aboul Ella Hassanien, Abdel-Badeeh M. Salem, Rabie Ramadan, y Tai-Hoon Kim, editors, *AMLTA*, vol. 322 of *Communications in Computer and Information Science*, pags. 487–495. Springer, 2012.

[216] D.B. Leake, A. Kinley, y D. Wilson. Learning to improve case adaptation by introspective reasoning and CBR. En M. Veloso y A. Aamodt, editors, *Case-Based Reasoning Research and Development*, pags. 229–240. Springer, Berlin„ 1995.

[217] R. Bergmann y I. Vollrath. Generalized cases: Representation and steps towards efficient similarity assessment. En *KI - Kunstliche Intelligenz*, pags. 195–206, 1999.

[218] D.B. Leake, A. Kinley, y D. Wilson. Linking adaptation and similarity learning. En *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, 1996.

[219] I.D. Watson. An introduction to Case-Based Reasoning. En Ian D. Watson, editor, *Progress in Case-Based Reasoning, First United Kingdom Workshop, Salford, UK, January 12, 1995, Proceedings*, vol. 1020 of *Lecture Notes in Computer Science*, pags. 3–16. Springer, 1995.

[220] Angi Voß. Principles of case reusing systems. En *EWCBR '96: Proceedings of the Third European Workshop on Advances in Case-Based Reasoning*, pags. 428–444, London, UK, 1996. Springer-Verlag.

[221] I. Watson y S. Perera. The evaluation of a hierarchical case representation using context guided retrieval. *Lecture Notes in Computer Science*, 1266:255–??, 1997.

[222] E. Plaza. Cases as terms: A feature term approach to the structured representation of cases. En *ICCBR*, pags. 265–276, 1995.

[223] K.E. Sanders, B.P. Kettler, y J.A. Hendler. The case for graph-structured representations. En *ICCBR*, pags. 245–254, 1997.

[224] Q. Yang y J. Wu. Enhancing the effectiveness of interactive Case-Based Reasoning with clustering and decision forests. *Applied Intelligence*, 14(1):49–64, 2001.

[225] B. Smyth y M.T. Keane. Remembering to forget: A competence-preserving case deletion policy for Case-Based Reasoning systems. En *IJCAI*, pags. 377–383, 1995.

[226] D. Aha y L.A. Breslow. Learning to refine case libraries: Initial results. Informe Técnico AIC-97-005, Research Laboratory, Navy Center, 1997.

[227] Gardingen D. y Watson I. A web based CBR system for heating ventilation and air conditioning systems sales support. *Knowledge-Based Systems*, 12:207–214(8), October 1999.

[228] Cunningham P., Smyth B., y Hurley N. On the use of CBR in optimisation problems such as the TSP. En *Case-Based Reasoning Research and Development*, Lecture Notes in Artificial Intelligence, pags. 401–410. Springer Verlag, 1995.

[229] C. Riesbeck. An interface for Case-Based knowledge acquisition. En J. Kolodner, editor, *Proceedings of the DARPA Case-Based Reasoning Workshop*, pags. 312–326, San Mateo, 1988. Morgan Kaufmann.

[230] J. Lieber. A criterion of comparison between two case bases. En *EWCBR*, pags. 87–100, 1994.

[231] E. McKenna y B. Smyth. A competence model for Case-Based Reasoning. En *9th Irish Conference on Artificial Intelligence and Cognitive Science*, 1998.

[232] J. Zhu y Q. Yang. Remembering to add: Competence-preserving case-addition policies for case base maintenance. En *IJCAI*, pags. 234–241, 1999.

[233] K. Racine y Q. Yang. On the consistency management of large case bases: the case for validation. En *Proceedings of the AAAI-96 Workshop on Knowledge-base validation*, 1996.

[234] D.W. Aha. The omnipresence of Case-Based Reasoning in Science and application. *Knowl.-Based Syst.*, 11(5-6):261–273, 1998.

[235] J. Kogan, C. Nicholas, y M. Teboulle. *Grouping Multidimensional Data: Recent Advances in Clustering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[236] J. R. Quinlan. Induction over large data bases. Informe técnico, University of Stanford, Stanford, CA, USA, 1979.

[237] R. Bergmann, M.M. Richter, S. Schmitt, A. Stahl, y I. Vollrath. Utility-oriented matching: A new research direction for Case-Based Reasoning. En H.-P. Schnurr et al., editor, *Proc. of the 9 th German Workshop on Case-Based Reasoning, GWCBR'01*, Baden-Baden, Germany, 2001. Shaker Verlag.

[238] B. Smyth y E. McKenna. Footprint-based retrieval. *Lecture Notes in Computer Science*, 1650:343–??, 1999.

[239] D. McSherry. Retrieval failure and recovery in recommender systems. *Artif. Intell. Rev.*, 24(3-4):319–338, 2005.

[240] D. Bridge y A. Ferguson. An expressive query language for product recommender systems. *Artificial Intelligence Review*, 18(3-4):269–307, 2002.

[241] J.W. Schaaf. Fish and shrink. a next step towards efficient case retrieval in large-scale case bases. En *EWCBR*, pags. 362–376, 1996.

[242] S. Cost y S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[243] H. Osborne y D.G. Bridge. A case base similarity framework. En *EWCBR*, pags. 309–323, 1996.

[244] S. Santini y R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[245] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[246] D.B. Leake. Adaptive similarity assessment for Case-Based explanation. *International Journal of Expert Systems Research and Applications*, 2(8):165–194, 1995.

[247] R. Bergmann y W. Wilke. Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995.

[248] L. Wang, T. Nakamura, M. Wang, H. Seki, y Hidenori Itoh. A method of generating calligraphy of japanese character using deformable contourse. En *IJCAI (2)*, pags. 1050–1055, 1997.

[249] Greek Universities Network Portal. Distance based classifiers. `http://portal.gunet.gr/index.pl?id=3865&isa=Item&op=download`. Online; recuperado en Junio de 2014.

[250] K. Hanney, M. T. Keane, B. Smyth, y P. Cunningham. Systems, tasks and adaptation knowledge in Case-Based Reasoning. En *Proceedings of the AAAI Fall Symposium on Adaptation of Knowledge for Reuse*, nº TCD-CS-95-04, Boston, USA., 1995.

[251] R. Bergmann y W. Wilke. Towards a new formal model of transformational adaptation in Case-Based Reasoning. En Henri Prade, editor, *Proceedings of 13th European Conference on Artificial Intelligence*, pags. 53–57. John Wiley and Sons, 1998.

[252] M. Maher y A. Gomez de Silva Garza. The adaptation of structural system designs using genetic algorithms. En *Proceedings of the International Conference on Information Technology in Civil and Structural Engineering Design - Taking Stock and Future Directions*, Glasgow, Scotland, 1996.

[253] Michael van Lent y John Laird. Learning by observation in a complex domain. En *Proceedings of the 11th workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada., 1998.

[254] M.M. Veloso y J.G. Carbonell. Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Mach. Learn.*, 10(3):249–278, 1993.

[255] E. Schwitzgebel. *Stanford Encyclopedia of Philosophy*, Capítulo: Belief. Stanford University, 2006.

[256] L.I. Smith. A tutorial on Principal Components Analysis. `http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`, 2002. [Online; recuperado en Julio de 2015].

[257] L.R. Rabiner y B.-H. Juang. *Fundamentals of speech recognition*. Prentice-Hall Inc., New Jersey, 1993.

[258] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[259] J.C. Bezdek, M.R. Pal, J. Keller, y R. Krisnapuram. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer Academic Publishers, Norwell, MA, 1999.

[260] S. Lenser, J. Bruce, y M. Veloso. A modular hierarchical behavior-based architecture. En A. Birk, S. Coradeschi, y S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002.

[261] C. Jones y M.J. Mataric. *Encyclopedia of Cognitive Science*, Capítulo: Cognitive Processing Through the Interaction of Many Agents. Nature Publishing Group, 2002.

[262] E. Tira-Thompson y D.S. Touretzky. The Tekkotsu robotics development environment. En *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pags. 6084–6089, May 2011.

[263] Bullet Physics Library. `bulletphysics.org`. Online; recuperado en Septiembre de 2015.

[264] J. Bruce, T. Balch, y M. Veloso. Fast and inexpensive color image segmentation for interactive robots. En *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, pags. 2061–2066, 2000.

[265] S. Figueroa. Control de la cabeza de un robot AIBO mediante gafas de Realidad Virtual. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, 2014.

[266] Sánchez-Marrè M., R-Roda I., Comas J., Cortés U., y Poch M. L
'Eixample Distance: A new similarity measure for case retrieval. En *Proceedings of II Congrés Català dIntelligéncia Artificial (CCIA - 98)*, pag. 2, 1998.

[267] M. Piccardi. Background subtraction techniques: a review. En *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, pags. 3099–3104, Oct 2004.

[268] T. González Sánchez. Artificial vision in the Nao humanoid robot. Proyecto Fin de Carrera, Universitat Politècnica de Catalunya, 2009.

[269] J.A. Recio-García y B. Díaz-Agudo. Ontology based CBR with jCOLIBRI. En Richard Ellis, Tony Allen, y Andrew Tuson, editors, *Applications and Innovations in Intelligent Systems XIV*, pags. 149–162. Springer London, 2007.

[270] W.R. Garner. *The processing of information and structure*. The Experimental Psychology Series/ Arthur W. Melton consulting ed. L. Erlbaum Associates; distributed by Halsted Press, New York, 1974.

[271] I. Herrero, C. Urdiales, J.M. Peula, y F. Sandoval. A bottom-up robot architecture based on learnt behaviors driven design. En Ignacio Rojas, Gonzalo Joya, y Andreu Catala, editors, *Advances in Computational Intelligence*, vol. 9094 of *Lecture Notes in Computer Science*, pags. 159–170. Springer International Publishing, 2015.

[272] I. Ulrich y J. Borenstein. VFH*: local obstacle avoidance with look-ahead verification. En *Proceedings IEEE International Conference on Robotics and Automation, 2000 (ICRA '00)*, vol. 3, pags. 2505–2511, 2000.

[273] J. Meyer, R. Adolph, D. Stephan, A. Daniel, V. Seekamp, M.and Weinert, y U. Visser. Decision-making and tactical behavior with potential fields. En GalA. Kaminka, PedroU. Lima, y Ral Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI*, vol. 2752 of *Lecture Notes in Computer Science*, pags. 304–311. Springer Berlin Heidelberg, 2003.

[274] F. Martín, V. Matellán, J.M. Cañas, y P. Barrera. Visual based localization for a legged robot. En A. Bredenfeld, A. Jacoff, I. Noda, y Y. Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, vol. 4020 of *Lecture Notes in Computer Science*, pags. 708–715. Springer Berlin Heidelberg, 2006.

[275] M. Sridharan, G. Kuhlmann, y P. Stone. Practical vision-based Monte Carlo localization on a legged robot. En *IEEE International Conference on Robotics and Automation (ICRA)*, pags. 3366–3371. IEEE, 2005.

[276] T. Röfer y M. Jüngel. Vision-based fast and reactive Monte-Carlo localization. En D. Polani, A. Bonarini, B. Browning, y K. Yoshida, editors, *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pags. 856–861. IEEE, 2003.

[277] T. Röfer y M. Jüngel. Fast and robust edge-based localization in the Sony Four-Legged RobotLeague. En D. Polani, B. Browning, A. Bonarini, y K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, vol. 3020 of *Lecture Notes in Computer Science*, pags. 262–273. Springer Berlin Heidelberg, 2004.

[278] M. Jamzad, S.B. Sadjad, V.S. Mirrokni, M. Kazemi, H.R. Chitsaz, A. Heydarnoori, M.T. Hajiaghayi, y E. Chiniforooshan. A fast vision system for middle size robots in Robocup. En Andreas Birk 0002, Silvia Coradeschi, y Satoshi Tadokoro, editors, *RoboCup*, vol. 2377 of *Lecture Notes in Computer Science*, pags. 71–80. Springer, 2001.

[279] M.I. Sánchez Tato, J.M. Peula, C. Urdiales, I. Herrero, y F. Sandoval. Navegación por aprendizaje mediante marcas de visión. En *XXI Simposium Unión Ciéntifica Internacional de Radio, URSI2006*, 2006.

[280] M.I. Sánchez Tato, J.M. Peula, C. Urdiales, I. Herrero, y F. Sandoval. Coordinación por aprendizaje mediante localización basada en marcas de visión. En *XXII Simposium Unión Ciéntifica Internacional de Radio, URSI2007*, 2007.

[281] J.M. Peula, C. Urdiales, I. Herrero, y F. Sandoval. Implicit robot coordination using Case-Based Reasoning behaviors. En *International Conference on Intelligent Robots and Systems, IROS13*, pags. 5929–5934. IEEE/RSJ, 2013.

[282] R. Ros. *Action selection in cooperative robot soccer using Case-Based Reasoning*. Tesis Doctoral, Universidad Politécnica de Catalunya, 2008.

[283] I. Herrero. Combinación de módulos CBR básicos mediante aprendizaje basado en PCA de secuencias. Informe interno 05, Departamento de Tecnología Electrónica, Universidad de Málaga, 2007.

[284] J.E. Jackson. *A User's Guide to Principal Components*. Wiley, New York, 1991.

[285] H. Murase y S. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal on Computer Vision*, 14(1):5 – 24, 1995.

[286] L. Sirovich y R Everson. Analysis and management of large scientific databases. *Int. Journal of Supercomputing Applications*, 6(1):50 – 68, 1992.

[287] N. Friedman y S. Russell. Image segmentation in video sequences: A probabilistic approach. En *Annual Conference on Uncertainty in Artificial Intelligence*, pags. 175–181, 1997.

[288] M. García Beltrán. Corrección de vignetting en el sistema de visión de un AIBO-ERS7. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, 2009.

[289] I. Herrero. Vídeos de pruebas experimentales. `https://www.youtube.com/playlist?list=PLqMHBMYEJUKSHeK3Y2DxSuL7SdIribMF9`, 2015. Online; recuperado en Octubre de 2015.