UNIVERSIDAD DE MALAGA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

TESIS DOCTORAL

EXPLORACIÓN COMPLETA BASADA EN COMPORTAMIENTOS COOPERATIVOS PARA UN AGENTE AUTÓNOMO MÓVIL

AUTOR: Alberto Poncela González
Ingeniero de Telecomunicación

2008

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

CERTIFICO:

Que D. Alberto Poncela González, 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:

EXPLORACIÓN COMPLETA BASADA EN COMPORTAMIENTOS COOPERATIVOS PARA UN AGENTE AUTÓNOMO MÓVIL

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

Y para que conste a efectos de lo establecido en el Real Decreto 56/2005 regulador de los estudios de Tercer Ciclo-Doctorado, AUTORIZO la presentación de esta Tesis en la Universidad de Málaga.

Málaga, a 27 de Febrero de 2008

Fdo.: Cristina Urdiales García
Profesora Titular del Departamento de Tecnología Electrónica

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

TESIS DOCTORAL

EXPLORACIÓN COMPLETA BASADA EN COMPORTAMIENTOS COOPERATIVOS PARA UN AGENTE AUTÓNOMO MÓVIL

AUTOR: Alberto Poncela González
Ingeniero de Telecomunicación

DIRECTOR: Cristina Urdiales García
Dra. Ingeniera de Telecomunicación

Admirando su belleza en la distancia, comenzó a llorar
Unicamente lo había hecho una vez, al saberse inmortal
eXplícame por qué estás llorando, mi perfecta mujer robot
Imperfecta yo soy, ya que sus ojos nunca me pertenecerán

A Auxi,
un sueño hecho realidad

Agradecimientos

Esta tesis no habría salido adelante sin la ayuda y colaboración de muchas personas, siendo mi deseo el reflejar mi más sincero agradecimiento público a todas ellas.

En primer lugar, es una satisfacción personal darle las gracias por tantas y tantas cosas a mis padres, José Manuel y Rosa, que siempre me animaron para ser mejor. El orgullo que siento por ellos sé que es recíproco.

Es para mí también una alegría poder agradecer su apoyo y ayuda a mis hermanos Raquel y Javier, así como a Juana. Siempre me acuerdo de vosotros, así que una parte de esta Tesis os pertenece

Gracias por todo, Cristina. Para mí no eres mi directora de Tesis, sino una amiga. Tu trabajo y dirección ha sido fundamental para que esta Tesis llegara a buen puerto. En esta ocasión y más que nunca te agradezco todo el esfuerzo que has depositado en esta Tesis.

A Francisco Sandoval, por darme la oportunidad de integrarme y trabajar en un tema apasionante dentro del Departamento de Tecnología Electrónica. También me gustaría agradecer a la Comisión Interministerial de Ciencia y Tecnología (CICYT) y al Ministerio de Educación y Ciencia (MEC), cuya financiación a través de los proyectos TIC2001-1758, TIN2004-07741-C02-01 y TEC2006-11689-C01/TCM ha ejercido de mecenas para la realización de este trabajo.

A los voluntarios de los experimentos con usuarios del control asistido y a la *Fondazione IRCCS Santa Lucia* de Roma. A los primeros por prestar desinteresadamente su tiempo y esfuerzo a la ciencia. Sin todos ellos no habría sido posible la investigación. Gracias. A los segundos, por permitir el uso de los planos de su hospital.

Nunca podría olvidarme de mis amigos, los cuales confían en mí mucho más de lo que me merezco. Gracias por haber estado a mi lado en los buenos y los malos momentos. Esta Tesis también va por vosotros. Mención especial merecen Eduardo Javier Pérez y José Manuel Cano. Ellos como pocos han prestado su apoyo y han hecho que todo el esfuerzo dedicado a esta Tesis merezca la pena.

Y gracias a tí, Auxi, mi sueño hecho realidad, por haber llenado el hueco que quedaba en mi corazón y ser mi fuente de inspiración.

En fin, gracias a todos, pues un trocito de esta Tesis también es vuestro, y es un verdadero honor para mí que sea así. Nunca os olvidaré.

Resumen

La robótica se está convirtiendo con el paso de los años en un sector estratégico, puesto que está asociada con el desarrollo tecnológico. La dirección hacia la que nos lleva esta situación es la de una sociedad donde la robótica de servicio va a eclosionar. En este ámbito la exploración de un entorno total o parcialmente desconocido es vital en campos como la robótica de rescate, la robótica aplicada a la vigilancia, la exploración espacial o en entornos de riesgo como volcanes. Su objetivo es adquirir el máximo grado de conocimiento sobre el entorno de trabajo.

En esta Tesis se desarrolla un sistema eficiente de navegación basado en comportamientos para la exploración completa de entornos dinámicos de interior total o parcialmente desconocidos por parte de un agente autónomo móvil. El sistema se basa en una arquitectura híbrida que combina las ventajas de los sistemas reactivos y los deliberados, ya que actualmente es la mejor alternativa para trabajar con agentes autónomos móviles. En el sistema se distinguen las tres capas típicas de toda arquitectura híbrida: la capa reactiva encargada de implementar los comportamientos reactivos, la capa deliberada que implementa la planificación de rutas y la capa intermedia que sirve de interfaz entre los niveles reactivo y deliberado. Estas tres capas constituyen el eje principal del sistema de exploración desarrollado.

El nivel reactivo está basado en la cooperación de tres comportamientos habitualmente empleados en la exploración de entornos, Seguir Pared, Seguir Pasillo y Cruzar Puerta. En su diseño e implementación se han seguido dos estrategias. En primer lugar se ha considerado un nuevo modelo analítico basado en la suma de varias fuerzas. Debido a que este modelo presenta algunos problemas como las oscilaciones y la dificultad para navegar entre obstáculos cercanos, también se ha optado por desarrollarlos según el Razonamiento Basado en Casos, que permite aprender de la experiencia para mejorar el funcionamiento. A este respecto, se propone una nueva filosofía de diseño aplicable a cualquier comportamiento reactivo, incorporando mecanismos de razonamiento y aprendizaje que permiten la adaptación a diferentes plataformas con sólo pequeños cambios. Ambas estrategias han sido comparadas tanto en entornos simulados como reales, demostrando que el Razonamiento Basado en casos supera al modelo analítico en la operación del agente.

La cooperación entre los tres comportamientos reactivos viene determinada por el nivel intermedio del sistema. Se ha diseñado y desarrollado una técnica que calcula los factores de ponderación a aplicar a los tres comportamientos de navegación reactiva, para combinar linealmente sus comandos de movimiento. De este modo se consigue que cooperen en la exploración del entorno de forma eficiente, adaptándose a las circunstancias y particularidades del entorno en cada momento. La técnica se basa en un nuevo proceso de extracción de un vector de características de tamaño reducido a partir de un mapa probabilístico en las proximidades del robot, gracias al análisis de componentes principales. Los resultados en entornos simulados y reales, tanto cualitativos como cuantitativos, demuestran la validez de la propuesta. Para la presentación de estos resultados se han desarrollado dos nuevas representaciones, la representación de selección y la representación de fusión.

El nivel deliberado del sistema se encarga de calcular la ruta de exploración completa que

el agente debe seguir para cubrir las zonas inexploradas del entorno. El método está soportado por una estructura piramidal jerárquica que integra los paradigmas métrico y topológico de modelado de entornos. El entorno queda dividido en regiones, estando las áreas no exploradas explícitamente representadas. El mapa topológico generado permite la obtención de las regiones no exploradas, así como de la distancia entre ellas. Con estos datos, el problema de la planificación de la ruta de exploración es similar al problema del viajante (TSP). Para resolverlo se propone un algoritmo genético que ha dado unos resultados satisfactorios en las pruebas simuladas y reales. Presenta la ventaja de que escogiendo los nodos que se quiere visitar, por defecto todos los no explorados, es adaptable a muchos otros problemas de razonamiento de alto nivel.

Todo el sistema ha sido integrado tanto en una plataforma simulada como en la plataforma robótica real *Pioneer P2AT* . El robot sigue la ruta ordenada de exploración completa proporcionada por el nivel deliberado, dirigiéndose en todo momento hacia la siguiente zona no explorada propuesta. La navegación se realiza mediante los tres comportamientos reactivos, estando sus comandos de movimiento ponderados por los factores que se obtienen en la capa intermedia. El sistema ha demostrado el cumplimiento de los objetivos planteados, siendo capaz de explorar de forma eficiente entornos total o parcialmente desconocidos.

Abstract

Robotics is becoming a strategic sector with the passage of time, since it is associated with technology evolution. This situation leads to a society where service robotics will emerge, where the exploration of a totally or partially known environment is essential in fields like rescue robotics, robotics applied to vigilance, space exploration or in risk environments as volcanos. Its main objective is to acquire the maximum degree of knowledge about the working environment.

In this Thesis we develop an efficient behaviour-based navigation system to achieve a complete exploration of indoor, dynamic, total or partially unknown environments with an autonomous mobile agent. Nowadays, the best alternative to deal with autonomous mobile agents is the use of a hybrid architecture. Thus, our system is based on a hybrid architecture, which combines the advantages of reactive and deliberative approximations. We can distinguish in the exploration system the three typical layers of hybrid architectures: the reactive layer that implements reactive behaviours, the deliberative layer that implements route planning processes and the middle layer which acts as an interface between the reactive and the deliberative levels. These three layers makes up the main key of the developed system.

The reactive level is based on the cooperation of three usually used behaviours when exploring environments, Wall Following, Corridor Following and Door Crossing. Two strategies have been adopted for their design and implementation. First, a new analytical model has been considered, using the addition of several forces. Since this model presents some problems, namely oscillations and the difficulty to navigate among close obstacles, behaviours have been also developed with a Case-Based Reasoning approximation, that allows learning from experience to improve the performance. This work proposes a new philosophy of design, which is applicable to any reactive behaviour. This methodology incorporates reasoning and learning mechanisms that allows its adaptation to different platforms with minor changes. Both strategies have been compared in simulated and real environments. The Case-Based Reasoning approximation have proven its superiority when the agent operates.

Cooperation among the three reactive behaviours is determined by the middle level. A new technique has been designed and developed. It calculates the weighting factors to apply to the three reactive behaviours, to linearly combine their motion commands. Thus, they cooperate when exploring the environment, adapting to the circumstances and peculiarities of the environment. The technique is based on a new process that extracts a short size feature vector from a probabilistic map built around the robot, by means of a principal component analysis. Results in simulated and real environments, qualitative and quantitative, proves the validity and feasibility of the method. Two new representations have been developed to present those results, the selection representation and the fusion representation.

The deliberative level is concerned with the calculation of the complete exploration route the agent must follow to cover all the unexplored areas of the environment. The method is supported by a hierarchical pyramidal structure that integrates metric and topological paradigms for environment modelling. The environment is divided into regions, where unexplored areas are explicitly represented. The generated topological map permits the obtaining of unexplored

regions and the distance among them, as well. Hence, the problem of planning an exploration route is similar to the traveling salesman problem (*TSP*). A genetic algorithm is proposed to solve the *TSP*. It has given successful results, both in real and simulated experiments. It has the advantage of its adaptability to many other problem of high level reasoning, since nodes to be visited can be chosen to fit several criteria.

The whole system has been integrated in a simulated platform and in the real platform *Pioneer P2AT*. The robot follows the ordered exploration route provided by the deliberative layer, directing to the next proposed unexplored area at each time instant. Navigation is achieved by means of the three reactive behaviours, being their motion commands weighted by the factors obtained in the middle layer. The system has proven the completion of the planned objectives, being able to explore total o partially unknown environments in an efficient way.

Índice General

Índice de Figuras **v**

Índice de Tablas **xi**

Lista de Acrónimos **xiii**

Lista de Símbolos **xv**

1 Introducción **1**
1 La Robótica . . . 1
1.1 Historia . . . 1
1.1.1 Robótica Antigua . . . 2
1.1.2 Robótica Moderna . . . 3
1.2 Situación actual y tendencias . . . 8
1.2.1 Robótica industrial . . . 9
1.2.2 Robótica de servicio . . . 10
1.2.3 Robótica de seguridad y para el espacio . . . 13
2 Motivación y Justificación . . . 13
3 Objetivos . . . 16
4 Organización del texto . . . 17

2 Arquitectura del Sistema **19**
1 Introducción . . . 19
1.1 Arquitecturas deliberadas . . . 20
1.2 Arquitecturas reactivas . . . 20
1.3 Arquitecturas híbridas . . . 21
2 La arquitectura *DLA* . . . 23
2.1 El estilo de la arquitectura *DLA* . . . 24
3 Estructura de la arquitectura DLA para el sistema propuesto . . . 26
3.1 Esquema general . . . 26
3.2 La capa *InterfazRobot* . . . 30
3.3 La capa *ModeladoEntorno* . . . 30
3.3.1 El módulo *MapaMétrico* . . . 32
3.3.2 El módulo *MapaLocal* . . . 34
3.4 La capa *InterfazUsuario* . . . 35
4 Conclusiones . . . 38

3 Comportamientos para Navegación Reactiva **41**
1 Introducción . . . 41
1.1 Navegación reactiva . . . 42
2 Comportamientos analíticos . . . 44
2.1 Seguir Pared . . . 45
2.2 Seguir Pasillo . . . 50
2.3 Cruzar Puerta . . . 54
3 Comportamientos mediante CBR . . . 58
3.1 El *CBR* . . . 58
3.2 Diseño e implementación . . . 59
3.2.1 Representación y almacenamiento de los casos . . . 60
3.2.2 El ciclo *CBR* aplicado al diseño de comportamientos reactivos . 62
3.2.2.1 Recuperación del caso: Recuperar . . . 63
3.2.2.2 Adaptación del caso: Reusar . . . 66
3.2.2.3 Evaluación del caso: Revisar . . . 67
3.2.2.4 Aprendizaje del caso: Retener . . . 67
3.2.3 Clasificación de la base de casos . . . 70
3.3 Los comportamientos . . . 72
3.3.1 Seguir Pared . . . 73
3.3.2 Seguir Pasillo . . . 76
3.3.3 Cruzar Puerta . . . 82
4 Método de aprendizaje asistido . . . 86
4.1 Cooperación humano/robot . . . 86
4.2 Aproximación reactiva para el aprendizaje asistido . . . 88
5 Experimentos y resultados . . . 90
5.1 Pruebas de los comportamientos . . . 90
5.1.1 Seguir Pared . . . 91
5.1.2 Seguir Pasillo . . . 95
5.1.3 Cruzar Puerta . . . 99
5.2 Pruebas del método de aprendizaje asistido . . . 104
5.2.1 Estudio con usuarios . . . 105
5.2.2 Resultados del funcionamiento del robot . . . 107
5.2.3 Resultados del funcionamiento del control asistido . . . 110
6 Conclusiones . . . 116

4 Fusión de Comportamientos **119**
1 Introducción . . . 119
1.1 Integración de comportamientos . . . 121
1.1.1 Coordinación de comportamientos . . . 121
1.1.2 Competición de comportamientos . . . 122
2 Algoritmo de caracterización de lugares . . . 123
2.1 Cálculo del contorno del mapa local . . . 128
2.1.1 Umbralización . . . 129
2.1.2 Filtrado . . . 131
2.1.3 Obtención de la *RLOI* . . . 131
2.1.4 Obtención del contorno . . . 132
2.2 Cálculo del mapa de profundidad . . . 134
2.3 Obtención de la FFT del mapa de profundidad . . . 136
2.4 Extracción del vector de características . . . 137

2.4.1 Obtención de la base . . . 139
2.4.2 El vector de características . . . 143
2.5 Análisis del vector de características . . . 144
2.5.1 Comparación del vector de características . . . 144
2.5.2 Cálculo de los factores de ponderación . . . 153
3 Cooperación de comportamientos . . . 155
4 Prestaciones temporales . . . 156
5 Experimentos y resultados . . . 158
5.1 Entornos simulados . . . 159
5.2 Entornos reales . . . 162
6 Conclusiones . . . 166

5 Exploración Completa de Entornos 169
1 El problema de la exploración completa . . . 169
1.1 Con conocimiento *a priori* del entorno . . . 171
1.2 Sin conocimiento *a priori* del entorno . . . 171
2 Construcción del mapa topológico . . . 172
2.1 Estado del arte . . . 174
2.1.1 Paradigma topológico . . . 174
2.1.2 Integración de los paradigmas métrico y topológico . . . 175
2.2 Modelado del entorno . . . 179
2.2.1 Preprocesado del mapa métrico . . . 179
2.2.2 La estructura jerárquica . . . 181
2.2.3 El mapa topológico . . . 185
3 Algoritmo de exploración completa . . . 190
3.1 Obtención de las regiones no exploradas . . . 192
3.2 Cálculo de la matriz de distancias . . . 193
3.3 Cálculo de la ruta . . . 194
3.3.1 Algoritmos para la optimización de rutas . . . 194
3.3.1.1 La búsqueda exhaustiva . . . 195
3.3.1.2 Redes neuronales . . . 195
3.3.1.3 *Simulated annealing* . . . 196
3.3.1.4 Colonias de hormigas . . . 197
3.3.1.5 Búsqueda tabú . . . 197
3.3.1.6 Algoritmos genéticos . . . 198
3.3.2 Planificación de rutas de exploración completa . . . 199
3.4 Prestaciones temporales . . . 202
4 Experimentos y resultados . . . 204
4.1 Entornos simulados . . . 204
4.2 Entornos reales . . . 206
4.3 Un caso de aplicación: planificación en un entorno hospitalario . . . 209
5 Conclusiones . . . 212

6 Resultados 215
1 Entorno de pruebas . . . 215
2 Entornos simulados . . . 218
2.1 Entorno simulado de prueba 1 . . . 219
2.2 Entorno simulado de prueba 2 . . . 222
2.3 Entorno simulado de prueba 3 . . . 225

2.4 Entorno simulado de prueba 4 . . . 227
2.5 Entorno simulado de prueba 5 . . . 228
3 Entornos reales . . . 231
3.1 Entorno real de prueba 1 . . . 231
3.2 Entorno real de prueba 2 . . . 233
3.3 Entorno real de prueba 3 . . . 234
3.4 Entorno real de prueba 4 . . . 235
3.5 Entorno real de prueba 5 . . . 237
4 Conclusiones . . . 239

7 Conclusiones y Líneas Futuras 241
1 Conclusiones . . . 241
2 Líneas futuras . . . 247
3 Publicaciones . . . 249
3.1 Revistas . . . 249
3.2 Capítulos de libro . . . 251
3.3 Congresos . . . 252

Bibliografía 255

A Descripción del Software del Sistema 279
1 La capa *InterfazRobot* . . . 283
1.1 Módulos . . . 283
1.2 Librerías . . . 285
1.3 Utilidades . . . 285
2 La capa *Comportamientos* . . . 286
2.1 Módulos . . . 286
2.2 Librerías . . . 290
2.3 Utilidades . . . 291
3 La capa *ModeladoEntorno* . . . 293
3.1 Módulos . . . 293
3.2 Librerías . . . 294
3.3 Utilidades . . . 294
4 La capa *FusiónComportamientos* . . . 294
4.1 Módulos . . . 295
4.2 Librerías . . . 295
4.3 Utilidades . . . 298
5 La capa *Planificación* . . . 301
5.1 Módulos . . . 301
5.2 Librerías . . . 302
5.3 Utilidades . . . 305
6 La capa *InterfazUsuario* . . . 305
6.1 Módulos . . . 306
6.2 Librerías . . . 306
6.3 Utilidades . . . 306

B Cuestionarios de Usuario del Control Asistido 307

Índice de Figuras

1.1 (a) Robot de *Rossum's Universal Robots* de Capek; (b) robot industrial *Unimate*; (c) robot autónomo *SHAKEY*; (d) robot humanoide P3; (e) robot humanoide ASIMO. 6
1.2 Robots exploradores: (a) *Lunakhod 1*; (b) *Viking 1* y *2*; (c) *Dante II*; (d) *Sojourner*; (e) *Spirit* y *Opportunity*. 8
1.3 Parque mundial de robots en el año 2004, excluido Japón. 10
1.4 Juguetes-robot: (a) *Furby*; (b) *Aibo*; (c) *Robosapien*. 12

2.1 Esquema general del sistema de exploración. 27
2.2 Mapas métricos: (a) entorno simulado; (b) mapa construido durante la exploración del entorno en (a); (c) entorno real; (d) mapa construido en un instante de la exploración del entorno en (c); (e) mapa construido en otro instante de la exploración del entorno en (c). 34
2.3 Mapas locales: (a) entorno real; (b) capturado en el entorno de la Figura 2.2a; (b) capturado en la posición A del entorno en (a); (c) capturado en la posición B del entorno en (a). 36
2.4 Interfaz de usuario del módulo *Interacción*. 37

3.1 Navegación reactiva, modelos analíticos: (a) campos de potencial; (b) histograma del vector de campo; (c) bandas elásticas; (d) ventana dinámica; (e) diagrama de proximidad. 43
3.2 Fuerzas del comportamiento Seguir Pared. 46
3.3 Seguir Pared, modelo analítico: (a) entorno simulado 1; (b) entorno simulado 2; (c) entorno simulado 3; (d) mapa métrico obtenido en (a); (e) mapa métrico obtenido en (b); (f) mapa métrico obtenido en (c). 48
3.4 Seguir Pared, seguimiento de obstáculo con modelo analítico: (a) entorno simulado 4; (b) entorno simulado 5; (c) mapa métrico obtenido en (a); (d) mapa métrico obtenido en (b). 49
3.5 Fuerzas del comportamiento Seguir Pasillo. 50
3.6 Seguir Pasillo, diferentes anchuras con modelo analítico: (a) entorno simulado 1; (b) entorno simulado 2; (c) entorno simulado 3; (d) mapa métrico obtenido en (a); (e) mapa métrico obtenido en (b); (f) mapa métrico obtenido en (c). 52
3.7 Seguir Pasillo, pasillo con obstáculos, modelo analítico: (a) entorno simulado 4; (b) entorno simulado 5; (c) entorno simulado 6; (d) mapa métrico obtenido en (a); (e) mapa métrico obtenido en (b); (f) mapa métrico obtenido en (c). 53
3.8 Seguir Pasillo, pasillo en L con modelo analítico: (a) entorno simulado 7; (b) entorno simulado 8; (c) mapa métrico obtenido en (a); (d) mapa métrico obtenido en (b). 54
3.9 Fuerzas del comportamiento Cruzar Puerta. 55

3.10 Cruzar puerta, entornos simulados con modelo analítico: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7; (h) 8; (i) 9; (j) 10; (k) 11; (l) 12; (m) 13; (n) 14; (o) 15. . . . 57
3.11 Definición del caso para el *CBR*. . . . 61
3.12 El ciclo *CBR*. . . . 63
3.13 Seguir Pared con *CBR*, aprendizaje por observación. . . . 73
3.14 Seguir Pared con *CBR*, adaptación: (a) sin adaptación; (b) con adaptación; (c) con adaptación teniendo en cuenta el aprendizaje por experiencia en (b). . . . 74
3.15 Seguir Pared con *CBR*, seguimiento de obstáculo: (a) aprendizaje por observación; (b) operación en el mismo entorno que (a); (c) operación en un entorno distinto de (a). . . . 75
3.16 Seguir Pared, comparación entre el modelo analítico y el basado en el *CBR*: (a) funcionamiento con el modelo analítico; (b) funcionamiento con el *CBR*; (c) trayectoria recorrida por el robot en (b). . . . 77
3.17 Seguir Pasillo con *CBR*, aprendizaje por observación. . . . 78
3.18 Seguir Pasillo con *CBR*, diferentes anchuras: (a) entorno simulado 1; (b) entorno simulado 2. . . . 78
3.19 Seguir Pasillo con *CBR*, adaptación: (a) sin adaptación; (b) con adaptación; (c) con adaptación teniendo en cuenta el aprendizaje por experiencia en (b). . . . 79
3.20 Seguir Pasillo con *CBR*, aprendizaje por observación: (a) obstáculo en la parte superior; (b) obstáculo en la parte inferior. . . . 80
3.21 Seguir Pasillo con *CBR*, funcionamiento con obstáculos: (a) aprendizaje por observación con obstáculo en la parte superior; (b) aprendizaje por observación con obstáculo en la parte inferior; (c) aprendizaje por observación con obstáculo en la parte superior. . . . 81
3.22 Seguir Pasillo con *CBR*, entornos con un obstáculo en la parte superior y otro en la parte inferior: (a) entorno simulado 3; (b) entorno simulado 4. . . . 81
3.23 Cruzar Puerta con *CBR*, aprendizaje por observación. . . . 82
3.24 Cruzar Puerta con *CBR*, operación en entornos simulados: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7. . . . 83
3.25 Cruzar Puerta con *CBR*, operación en entornos simulados: (a) 8; (b) 9; (c) 10; (d) 11; (e) 12; (f) 13; (g) 14. . . . 84
3.26 Cruzar Puerta con *CBR*: (a) aprendizaje por observación; (b) operación en entorno simulado. . . . 85
3.27 Combinación de los comandos del robot y el humano. . . . 87
3.28 Definición de los tres factores involucrados en la eficiencia. . . . 89
3.29 Seguir Pared, entornos reales de prueba: (a) 1; (b) 2. . . . 92
3.30 Seguir Pared, entorno real de la Figura 3.29a: (a) aprendizaje por observación; (b) operación con el modelo analítico; (c) operación con el *CBR*. Entorno real de la Figura 3.29b: (d) operación con el modelo analítico; (e) operación con el *CBR*. 92
3.31 Seguir Pared, entorno real de prueba: (a) planta; (b) operación con el modelo analítico. Operación con el *CBR*: (c) sin adaptación; (d) con adaptación; (e) con adaptación teniendo en cuenta el aprendizaje por experiencia en (d). . . . 94
3.32 Seguir Pared, entorno real de la Figura 3.31a: (a) aprendizaje por observación; (b) operación con el *CBR*. . . . 95
3.33 Seguir Pasillo, entorno real de prueba: (a) planta; (b) aprendizaje por observación; (c) operación con el modelo analítico; (d) operación con el *CBR*. . . . 96
3.34 Seguir Pasillo, entorno real de prueba: (a) planta; (b) operación con el modelo analítico; (c) operación con el *CBR*. . . . 97

3.35 Seguir Pasillo, pasillo en L: (a) planta; (b) operación con el modelo analítico; (c) operación con el *CBR*, sin adaptación; (d) operación con el *CBR*, con adaptación; (e) operación con el *CBR*, teniendo en cuenta el aprendizaje por experiencia en (d). 98
3.36 Seguir Pasillo, funcionamiento con obstáculos: (a) planta 1; (b) aprendizaje por observación en (a); (c) planta 2; (d) aprendizaje por observación en (b). 99
3.37 Seguir Pasillo, funcionamiento con obstáculos. Entorno de la Figura 3.36a: (a) operación con el modelo analítico; (b) operación con el *CBR*, entrenamiento con obstáculo en la parte superior. Entorno de la Figura 3.36b: (c) operación con el modelo analítico; (d) operación con el *CBR*, entrenamiento con obstáculo en la parte superior; (e) operación con el *CBR*, entrenamiento con obstáculo en la parte inferior. 100
3.38 Cruzar Puerta, entorno real: (a) planta; (b) aprendizaje por observación. 100
3.39 Cruzar Puerta, entornos reales con el modelo analítico: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7; (h) 8. 101
3.40 Cruzar Puerta, entornos reales con el *CBR*: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7; (h) 8. 102
3.41 Cruzar Puerta, entorno real de prueba; (a) planta; (b) aprendizaje por observación; (c) operación en el entorno de la Figura 3.40b. 103
3.42 Cruzar Puerta, entorno real con puerta estrecha: (a) operación con el modelo analítico; (b) operación con el *CBR*. 104
3.43 Entornos reales: (a) Seguir Pared (FW); (b) Pasillo (NC); (c) Doble Pasillo (DNC); (d) Cruzar Puerta (PD). 106
3.44 Trayectorias llevadas a cabo por el robot: (a) entorno FW; (b) entorno NC; (c) entorno DNC; (d) entorno PD. 107
3.45 Funcionamiento del robot en los entornos de test FW y NC: (a) representación 3D de la eficiencia del robot en el test FW; (b) representación 3D de la eficiencia del robot en el test NC; (c) proyección XZ de la eficiencia del robot en (a); (d) proyección XZ de la eficiencia del robot en (b); (e) proyección XY de la eficiencia del robot en (a); (f) proyección XY de la eficiencia del robot en (b). 109
3.46 Funcionamiento del robot en los entornos de test DNC y PD: (a) representación 3D de la eficiencia del robot en el test DNC; (b) representación 3D de la eficiencia del robot en el test PD; (c) proyección XZ de la eficiencia del robot en (a); (d) proyección XZ de la eficiencia del robot en (b); (e) proyección XY de la eficiencia del robot en (a); (f) proyección XY de la eficiencia del robot en (b). 110
3.47 Usuario relacionado con la robótica: (a) representación 3D de la eficiencia del control asistido en el test DNC; (b) trayectoria. Usuario no relacionado con la robótica: (c) representación 3D de la eficiencia del control asistido en el test DNC; (d) trayectoria. 113
3.48 (a) Tasa de aprendizaje en el test FW; (b) tasa de aprendizaje en el test PD. . . 113
3.49 El usuario percibe el control de la máquina: (a) representación 3D de la eficiencia del control asistido en el test FW; (b) trayectoria. 115
3.50 Tasa de aprendizaje en el test NC. 116
3.51 Trayectoria recorrida por el robot en el test FW: (a) usuario zurdo; (b) usuario diestro. 116

4.1 Esquema general del algoritmo de extracción del vector de características. 127
4.2 Cálculo del contorno del mapa local: (a) mapa local; (b) umbralización, (c) filtrado; (d) *RLOI*; (e) contorno. 129

4.3 Cálculo del contorno del mapa local: mapa local, umbralización, filtrado, *RLOI*, contorno; (a) pared; (b) pasillo; (c) puerta. . . . 133
4.4 Mapas de profundidad del contorno de la *RLOI* de los mapas locales en la Figura 4.3: (a) pared; (b) pasillo; (c) puerta. . . . 136
4.5 $|FFT|$ de los mapas de profundidad de la Figura 4.4: (a) pared; (b) pasillo; (c) puerta. . . . 137
4.6 Mapas locales a partir de los que se calcula la base del sistema. . . . 140
4.7 (a) Porcentaje de la varianza explicado por cada componente principal; (b) porcentaje de la varianza explicado por las m primeras componentes principales. . . . 141
4.8 (a) $|FFT|$ del mapa de profundidad del contorno de la *RLOI* de los mapas locales de la base; (b) vectores de características de los mapas locales de la base. . . . 142
4.9 Vectores de características de los mapas locales de la Figura 4.3. . . . 144
4.10 Mapas locales de los patrones del sistema. Patrón 1-3: pared; Patrón 4-6: pasillo; Patrón 7-9: puerta. . . . 145
4.11 Vectores de características que conforman los patrones del sistema: (a) Patrón 1-3: pared; (b) Patrón 4-6: pasillo; (c) Patrón 7-9: puerta; (d) comparación de todos los patrones. . . . 146
4.12 Mapas locales de prueba. Mapa 1-3: pared; Mapa 4-6: pasillo; Mapa 7-9: puerta. 149
4.13 Comparación de los vectores de características de los mapas locales de prueba de la Figura 4.12: (a) Mapas 1-3: pared; (b) Mapas 4-6: pasillo; (c) Mapas 7-9: puerta; (d) todos. . . . 150
4.14 Comparación de los vectores de características de los mapas locales de prueba de la Figura 4.12 con los patrones: (a) Mapa 1; (b) Mapa 2; (c) Mapa 3; (d) Mapa 4; (e) Mapa 5; (f) Mapa 6; (g) Mapa 7; (h) Mapa 8; (i) Mapa 9. . . . 152
4.15 Entorno simulado 1: (a) planta; (b) *RS*; (c) *RF*. . . . 160
4.16 Entorno simulado 2: (a) planta; (b) *RS*; (c) *RF*. . . . 161
4.17 Entorno simulado 3: (a) planta; (b) distancia de Tanimoto entre vectores de características consecutivos de la trayectoria; (c) *RS*; (d) *RF*. . . . 162
4.18 Entorno real 1: (a) planta; (b) *RS*; (c) *RF*. . . . 163
4.19 Entorno real 2: (a) planta; (b) *RS*; (c) *RF*. . . . 163
4.20 Entorno real 3: (a) planta; (b) *RS*; (c) *RF*. . . . 165
4.21 Entorno real 4: (a) planta; (b) *RS*; (c) *RF*. . . . 166

5.1 Descomposición de un entorno en *quadtrees* propuesta por Zelinsky. . . . 176
5.2 Método de Thrun para la obtención de un mapa topológico: (a) diagrama de Voronoi; (b) líneas críticas; (c) regiones; (d) mapa topológico. . . . 177
5.3 (a) Descomposición en regiones propuesta por Kraetzschmar; (b) descomposición en regiones según el método de Fabrizi. . . . 178
5.4 Método de Arleo para calcular mapas topológicos: (a) entorno de prueba; (b) partición de resolución variable; (c) mapa topológico. . . . 178
5.5 Preprocesado de un mapa métrico de 256x256 celdas: (a) mapa métrico de partida; (b) umbralización, $U_{obst} = 60\%$ y $U_{libre} = 40\%$; (c) crecimiento de obstáculos, $Obst\_Inc = 1$. . . . 180
5.6 Estructura jerárquica: (a) *Creación e Inicialización*; (b) *Enlazado*; (c) *Clasificación*. 184
5.7 Estructura jerárquica: (a) nivel 7, 2x2 celdas; (b) nivel 6, 4x4 celdas; (c) nivel 5, 8x8 celdas; (d) nivel 4, 16x16 celdas; (e) nivel 3, 32x32 celdas; (f) nivel 2, 64x64 celdas; (g) nivel 1, 128x128 celdas; (h) nivel 0, 256x256 celdas. . . . 185
5.8 Mapa topológico: (a) mapa métrico de partida; (b) descomposición en regiones; (c) mapa topológico. . . . 187

5.9 Eliminación del crecimiento de obstáculos en la construcción del mapa topológico. 187
5.10 Mapa topológico para diferentes valores de $Dist\_Max$: (a) $Dist\_Max = 10$; (b) $Dist\_Max = 20$; (c) $Dist\_Max = 40$; (d) $Dist\_Max = 60$; (e) $Dist\_Max = 80$; (f) $Dist\_Max = 100$. 188
5.11 Mapa topológico para diferentes valores de A_{min}: (a) $A_{min} = 2$; (b) $A_{min} = 8$; (c) $A_{min} = 16$; (d) $A_{min} = 64$. 189
5.12 Representación métrico-topológico de un entorno parcialmente explorado: (a) entorno simulado; (b) entorno real. 191
5.13 Fases del algoritmo de exploración completa a partir de la representación métrico-topológica. 192
5.14 Obtención de las regiones no exploradas de un entorno. 193
5.15 Ruta de exploración completa en mapas de 256x256 celdas: (a) 6 nodos inexplorados; (b) 10 nodos inexplorados; (c) 13 nodos inexplorados. 203
5.16 Cálculo de la ruta de exploración completa en un entorno simulado: (a) planta; (b) etapa 1 de la exploración; (c) etapa 2 de la exploración; (d) etapa 3 de la exploración; (e) etapa 4 de la exploración. 205
5.17 Planta de los entornos reales de prueba: (a) entorno real 1; (b) entorno real 2. . . 207
5.18 Cálculo de la ruta de exploración completa en el entorno real 1: (a) etapa 1 de la exploración; (b) etapa 2 de la exploración; (c) etapa 3 de la exploración; (d) etapa 4 de la exploración; (e) etapa 5 de la exploración. 208
5.19 Cálculo de la ruta de exploración completa en el entorno real 2: (a) etapa 1 de la exploración; (b) etapa 2 de la exploración. 209
5.20 (a) Mapa original del hospital; (b) mapa métrico obtenido al procesar el mapa original del hospital. 210
5.21 Mapa topológico superpuesto al mapa original del hospital, junto con la situación de varias habitaciones importantes. 211

6.1 Entorno de pruebas: (a) simulador de la plataforma robótica *Nomad 200* de *Nomadics*; (b) plataforma robótica real *Pioneer P2AT*. 216
6.2 Aprendizaje por observación: (a) Seguir Pared; (b) Seguir Pasillo; (c) Cruzar Puerta. 218
6.3 Exploración de entorno simulado: (a) trayectoria; (b) RF; (c) comportamiento Seguir Pared; (d) comportamiento Seguir Pasillo. 219
6.4 (a)-(i) Mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno de la Figura 6.3a. 221
6.5 Exploración de entorno simulado: (a) trayectoria; (b) RF; (c) comportamiento Cruzar Puerta. 223
6.6 (a)-(l) Mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno de la Figura 6.5a. 224
6.7 Exploración de entorno simulado: (a) trayectoria; (b) RF. 226
6.8 Exploración de entorno simulado: (a) trayectoria; (b) RF. 227
6.9 Exploración de entorno simulado: (a) trayectoria; (b) mapa métrico cuando todo el entorno está explorado; (c) RF. 229
6.10 Exploración de entorno simulado con $Tam\_Celda = 80mm$: (a) trayectoria; (b) mapa métrico cuando todo el entorno está explorado; (c) mapa topológico sobre el mapa métrico en (b); (d) RF. 230
6.11 Aprendizaje por observación: (a) Seguir Pared; (b) Seguir Pasillo; (c) Cruzar Puerta. 231

6.12 Exploración de entorno real: (a) planta; (b) RF; (c) mapa topológico sobre el mapa métrico cuando todo el entorno está explorado; (d) comportamiento Seguir Pared; (e) comportamiento Seguir Pasillo. 232
6.13 Exploración de entorno real: (a) planta; (b)RF; (c) mapa métrico cuando todo el entorno está explorado; (d) mapa topológico sobre el mapa métrico en (c). . . 233
6.14 Exploración de entorno real: (a) planta; (b) mapa métrico cuando todo el entorno está explorado; (c) RS; (d) RF; (e) comportamiento Seguir Pared; (f)-(h) mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno en (a). 235
6.15 Exploración de entorno real: (a) planta; (b) mapa métrico cuando todo el entorno está explorado; (c) mapa topológico sobre el mapa métrico en (b); (d) RF; (e) modelo de entorno y trayectoria de exploración aleatoria. 236
6.16 Exploración de entorno real: (a) planta; (b) RS; (c) RF; (d)-(g) mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno en (a); (h) mapa topológico sobre el mapa métrico cuando todo el entorno está explorado. 238

A.1 Esquema general del sistema de exploración junto con el principal *software* desarrollado en *C*. 281

Índice de Tablas

1.1 Cronología de la Robótica Antigua: desde la época antigua hasta el siglo XIX. . 3
1.2 Cronología de la Robótica moderna: desde el siglo XX hasta nuestros días. . . . 4

3.1 Características de los usuarios. 106
3.2 Resultados estadísticos del funcionamiento del robot. 108

4.1 Distancia de Tanimoto entre los vectores de características que conforman los patrones. 148
4.2 Distancia de Tanimoto entre los vectores de características de los mapas locales de la Figura 4.12. 151
4.3 Distancia de Tanimoto de los vectores de características de los mapas locales de prueba de la Figura 4.12 a los patrones del sistema. 153
4.4 Tiempo de cálculo del contorno del mapa local. 157
4.5 Tiempo de cálculo de los factores de ponderación a aplicar para conseguir la cooperación de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta. 157

5.1 Tiempo de construcción del mapa topológico. 190
5.2 Tiempo aproximado de ejecución del algoritmo de búsqueda exhaustiva para obtener la ruta óptima de exploración completa. 199
5.3 Tiempo de ejecución del algoritmo de exploración completa. 203
5.4 Lugares visitados por el personal, así como el coste durante la situación de emergencia. 212

6.1 Parámetros empleados durante las pruebas del sistema de exploración. 217

B.1 Modelo de cuestionario de usuario. 308
B.2 Cuestionario del Sujeto Nº1. 309
B.3 Cuestionario del Sujeto Nº2. 310
B.4 Cuestionario del Sujeto Nº3. 311
B.5 Cuestionario del Sujeto Nº4. 312
B.6 Cuestionario del Sujeto Nº5. 313
B.7 Cuestionario del Sujeto Nº6. 314
B.8 Cuestionario del Sujeto Nº7. 315
B.9 Cuestionario del Sujeto Nº8. 316
B.10 Cuestionario del Sujeto Nº9. 317
B.11 Cuestionario del Sujeto Nº10. 318
B.12 Cuestionario del Sujeto Nº11. 319
B.13 Cuestionario del Sujeto Nº12. 320
B.14 Cuestionario del Sujeto Nº13. 321

Lista de Acrónimos

ACO	*Ant Colony Optimization*
CBR	*Case Based Reasoning*
DFT	*Discrete Fourier Transform*
DIF	*Decentralized Information Filter*
DLA	*Distributed and Layered Architecture*
DWA	*Dynamic Window Approach*
EB	*Elastic Bands*
ENA	*Elastic Net Algorithm*
EUROP	*EUropean RObotics Platform*
FFT	*Fast Fourier Transform*
HMM	*Hidden Markov Models*
IA	*Inteligencia Artificial*
KEMLg	*Knowledge Engineering and Machine Learning Group*
MDS	*Multi-Dimensional Scaling*
ND	*Nearness Diagram*
PCA	*Principal Component Analysis*
PFM	*Potential Fields Method*
POMDP	*Partially Observable Markov Decision Process*
RGB	*Red Green Blue*
RF	*Representación de Fusión*
RLOI	*Región Libre de Obstáculos de Interés*
RS	*Representación de Selección*
SPA	*Sense-Plan-Act*
SSH	*Spatial Semantic Hierarchy*
SOM	*Self Organizing Maps*
SVM	*Support Vector Machine*
TSP	*Traveling Salesman Problem*
VFH	*Vector Field Histogram*

Lista de Símbolos

A_{min}	Área mínima de una región
$A(x, y, l)$	Área de la celda (x, y) del nivel l de la pirámide
$A(x_1, y_1, l)$	Área de la celda (x_1, y_1) del nivel l de la pirámide
$A(x_2, y_2, l)$	Área de la celda (x_2, y_2) del nivel l de la pirámide
$\overrightarrow{\alpha}$	Vector con las orientaciones de los sensores virtuales del mapa de profundidad
$\triangle\alpha$	Diferencia angular entre dos sensores consecutivos de $\overrightarrow{\alpha}$
α_{comp}	Ángulo entre la orientación del robot y la dirección para implementar un comportamiento
α_{dif}	Ángulo entre la orientación actual del robot y el vector de comando
α_{min}	Ángulo entre la orientación actual del robot y el obstáculo más cercano
α_P	Ángulo respecto de la pared más cercana a la derecha del robot
β	Anchura del lóbulo principal del sensor sonar
$\{\beta_{ik}\}_{k=1}^{P}$	Vector de características del vector $\overrightarrow{\lvert FFT\rvert}_i$
(c_x, c_y)	Posición del centro del mapa local
$\overrightarrow{C}$	Vector con el contorno de la $RLOI$ de un mapa local
C_k	Punto k del contorno $\overrightarrow{C}$
C_{lt}	Constante que pondera el factor η_{lt}
C_{se}	Constante que pondera el factor η_{se}
C_{su}	Constante que pondera el factor η_{su}
$Cl(x, y, l)$	Clase asignada a la celda (x, y) del nivel l de la pirámide
$Cl(x_1, y_1, l)$	Clase asignada a la celda (x_1, y_1) del nivel l de la pirámide
$Cl(x_2, y_2, l)$	Clase asignada a la celda (x_2, y_2) del nivel l de la pirámide
Cl_i	Clase i en que se segmenta el mapa métrico
Cl_j	Clase j en que se segmenta el mapa métrico
$Ct(x, y, l)$	Centro de masas de la región en la base asociada a la celda (x, y, l)
$Ct(x_1, y_1, l)$	Centro de masas de la región en la base asociada a la celda (x_1, y_1, l)
$Ct(x_2, y_2, l)$	Centro de masas de la región en la base asociada a la celda (x_2, y_2, l)
$Ct(x_p, y_p, l+1)$	Centroide de la región en la base asociada a la celda $(x_p, y_p, l+1)$
$Ct(n_i)$	Centroide de la región del mapa topológico a la que representa el nodo n_i
$Ct(n_j)$	Centroide de la región del mapa topológico a la que representa el nodo n_j
$\overrightarrow{CA}$	Vector con un caso del sistema CBR
$\overrightarrow{CA}^{new}$	Vector con el caso nuevo en la posición del robot
$\overrightarrow{CA}^{recup}$	Vector con el caso recuperado de la base de casos

$\overrightarrow{CA}_D$	Vector con el caso $\overrightarrow{CA}$ discretizado
$\overrightarrow{CA}_D^{new}$	Vector con el caso $\overrightarrow{CA}^{new}$ discretizado
$\overrightarrow{CA}_D^{IN}$	Entrada del caso discretizado $\overrightarrow{CA}_D$
$\overrightarrow{CA}^{IN}$	Entrada del caso $\overrightarrow{CA}$
$\overrightarrow{CA}^{OUT}$	Salida del caso $\overrightarrow{CA}$
C_{FFT}	Matriz de covarianza de $\left\{\overrightarrow{\lvert FFT\rvert}_m\right\}_{m=1}^{M}$
$\overrightarrow{C}^N$	Subvector de $\overrightarrow{C}$ con los puntos cuya orientación más se acerque a las de $\overrightarrow{\alpha}$
C_i^N	Componente i del vector $\overrightarrow{C}^N$
CP	Comportamiento Cruzar Puerta
d	Distancia al obstáculo detectado
d_{min}	Distancia al obstáculo más próximo
d^{min}	Distancia minima a los patrones del sistema
$\overline{d_{pared}}$	Distancia media a los patrones de la categoría pared
d_{pared}^{min}	Distancia mínima a los patrones de la categoría pared
$\overline{d_{pasillo}}$	Distancia media a los patrones de la categoría pasillo
$d_{pasillo}^{min}$	Distancia mínima a los patrones de la categoría pasillo
$\overline{d_{puerta}}$	Distancia media a los patrones de la categoría puerta
d_{puerta}^{min}	Distancia mínima a los patrones de la categoría puerta
d_C	Mitad de la anchura del pasillo
d_{CPSEC}	Distancia de seguridad en el comportamiento CP
d_D	Distancia que el robot debe mantener con la pared derecha
d_F	Distancia a los obstáculos frontales del robot
d_L	Distancia a la parte izquierda del marco de la puerta
d_R	Distancia a la parte derecha del marco de la puerta
d_P	Distancia mínima actual a la pared derecha
d_{SCSEC}	Distancia de seguridad en el comportamiento SC
d_{SPSEC}	Distancia de seguridad en el comportamiento SP
$\overrightarrow{D}$	Vector de distancias de un vector de características a los patrones del sistema
$D(n_i, n_j)$	Distancia entre los nodos n_i y n_j del mapa topológico
$D(\overrightarrow{CA}_D^{new}, \overrightarrow{CA}_D^{i})$	Distancia Manhattan discretizada entre los casos discretizados $\overrightarrow{CA}_D^{new}$ y $\overrightarrow{CA}_D^{i}$
$D(\overrightarrow{VC}_i, \overrightarrow{VC}_j)$	Distancia de Tanimoto entre $\overrightarrow{VC}_i$ y $\overrightarrow{VC}_j$
$Dist\_Max$	Distancia máxima entre las regiones de un entorno
$Dist\_MaxMin$	Distancia mínima entre clases del algoritmo *MaxMin*
ε^2	Error cuadrático medio
$f_{CPavoid}$	Fuerza que evita los obstáculos en el comportamiento CP
f_{CPort}	Fuerza que mantiene el robot perpendicular al marco de la puerta en el comportamiento CP
$f_{SCavoid}$	Fuerza que evita los obstáculos en el comportamiento SC
f_{SCdist}	Fuerza que mantiene el robot en el centro del pasillo en el comportamiento SC
$f_{SCparal}$	Fuerza que mantiene el robot paralelo a la pared en el comportamiento SC
$f_{SPavoid}$	Fuerza que evita los obstáculos en el comportamiento SP
f_{SPdist}	Fuerza que mantiene la distancia a la pared en el comportamiento SP
$f_{SPparal}$	Fuerza que mantiene el robot paralelo a la pared en el comportamiento SP

$\overrightarrow{FFT}$	Vector con la FFT del mapa de profundidad del contorno de la $RLOI$ de un mapa local
FFT_n	Componente n del vector $\overrightarrow{FFT}$
$\|FFF\|$	Módulo de la FFT
$\overrightarrow{\|FFT\|}$	Vector con el módulo de cada componente de $\overrightarrow{FFT}$
$\|FFT_i\|$	Módulo de la componente i del vector $\overrightarrow{FFT}$
$\left\{\overrightarrow{\|FFT\|}_m\right\}_{m=1}^{M}$	Conjunto de M vectores $\overrightarrow{\|FFT\|}_m$
$\overrightarrow{\varphi}$	Vector de fases o direcciones posibles para $\theta(x,y)$
φ_i	Dirección genérica del vector $\overrightarrow{\varphi}$
$\overrightarrow{\Phi}_i$	Autovector i de la matriz de covarianza C_{FFT}
$\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$	Base de componentes principales
$G(x,y)$	Valor asignado a una celda (x,y) durante el primer paso de la umbralización
G_{fondo}	Nivel de gris de las zonas fuera de la región de interés
$G_{no_explore}$	Nivel de gris de las zonas no exploradas
G_{region}	Nivel de gris de la región de interés
$H(x,y,l)$	Homogeneidad de la celda (x,y) del nivel l de la pirámide
$H(x_1,y_1,l)$	Homogeneidad de la celda (x_1,y_1) del nivel l de la pirámide
$H(x_2,y_2,l)$	Homogeneidad de la celda (x_2,y_2) del nivel l de la pirámide
$H(x_p,y_p,l+1)$	Homogeneidad de la celda $(x_p,y_p,l+1)$
$Hijo_k$	Primer individuo que se genera a partir de $Padre_i$ y $Padre_j$
$Hijo_l$	Segundo individuo que se genera a partir de $Padre_i$ y $Padre_j$
η_{lt}	Factor de longitud de la trayectoria de la eficiencia η
η_{se}	Factor de seguridad de la eficiencia η
η_{su}	Factor de suavidad de la eficiencia η
η_C	Eficiencia del comando de movimiento del control asistido
η_H	Eficiencia del comando de movimiento del humano
η_R	Eficiencia del comando de movimiento del robot
θ	Ángulo de una celda respecto de la orientación del robot
$\theta(x,y)$	Fase de una celda (x,y) con respecto del centro (c_x,c_y)
$k_{CPavoid}$	Constante de ponderación de la fuerza $f_{CPavoid}$
k_{CPort}	Constante de ponderación de la fuerza f_{CPort}
$k_{SCavoid}$	Constante de ponderación de la fuerza $f_{SCavoid}$
k_{SCdist}	Constante de ponderación de la fuerza f_{SCdist}
$k_{SCparal}$	Constante de ponderación de la fuerza $f_{SCparal}$
$k_{SPavoid}$	Constante de ponderación de la fuerza $f_{SPavoid}$
k_{SPdist}	Constante de ponderación de la fuerza f_{SPdist}
$k_{SPparal}$	Constante de ponderación de la fuerza $f_{SPparal}$
K	Número de puntos del contorno $\overrightarrow{C}$
$L_C(n_i,n_j)$	Lista ordenada de nodos del camino para ir del nodo n_i al nodo n_j
$L_C(n_i,n_j,k)$	Nodo almacenado en la posición k de la lista $L_C(n_i,n_j)$
L_N	Lista de nodos no explorados del mapa topológico
L_{N_a}	Lista de nodos no explorados del mapa topológico accesibles desde la posición del robot
λ_i	Autovalor i de la matriz de covarianza C_{FFT}
$\{\lambda_k\}_{k=1}^{P}$	P primeros autovalores de C_{FFT} ordenados de mayor a menor valor
M_C	Matriz de conexión entre los nodos del mapa topológico
$M_C(n_i,n_j)$	Elemento (n_i,n_j) de la matriz de conexión M_C

M_D	Matriz de distancias entre los nodos del mapa topológico
$M_D(n_i, n_j)$	Elemento (n_i, n_j) de la matriz de distancias M_D
$\overrightarrow{MP}$	Vector con el mapa de profundidad del contorno $\overrightarrow{C}$
MP_i	Componente i del mapa de profundidad
n_i	Nodo i del mapa topológico
n_j	Nodo j del mapa topológico
ns_i	Número de casos pertenecientes a la clase S_i
N	Tamaño del mapa de profundidad y de su $\|FFT\|$
N_a	Número de nodos inexplorados del mapa topológico accesibles desde la posición del robot
$N_{med}xM_{med}$	Tamaño de la máscara del filtro de la mediana
$\overrightarrow{NR}$	Vector con el número de rangos de cada entrada del caso $\overrightarrow{CA}$
$Obst\_Inc$	Número de celdas en que se crecen los obstáculos
OL_{N_a}	Lista ordenada de nodos no explorados accesibles del mapa topológico con la ruta de exploración completa
$\overrightarrow{O_\varphi}$	Vector de distancias a los obstáculos más cercanos
O_{φ_i}	Distancia al obstáculo más cercano en la dirección φ_i
$pos_i(1)$	Primera posición aleatoria en $Padre_i$
$pos_i(2)$	Segunda posición aleatoria en $Padre_i$
$pos_j(1)$	Primera posición aleatoria en $Padre_j$
$pos_j(2)$	Segunda posición aleatoria en $Padre_j$
P	Tamaño de la base de componentes principales
$P(x, y, l)$	Probabilidad de ocupación de la celda (x, y) del nivel l de la pirámide
$P(x_1, y_1, l)$	Probabilidad de ocupación de la celda (x_1, y_1) del nivel l de la pirámide
$P(x_2, y_2, l)$	Probabilidad de ocupación de la celda (x_2, y_2) del nivel l de la pirámide
$P(x_p, y_p, l+1)$	Probabilidad de ocupación de la celda $(x_p, y_p, l+1)$
P_{inex}	Probabilidad de ocupación de las celdas no exploradas
P_{libre}	Probabilidad de ocupación de las celdas libres
$P_{espacio}$	Decremento de la probabilidad de ocupación de las celdas libres
P_{n_i}	Probabilidad de ocupación del nodo n_i del mapa topológico
P_{obst}	Probabilidad de ocupación de las celdas obstáculo
$P^t_{ocup}(\theta, \rho)$	Probabilidad de ocupación de una celda en el instante t
$P^{t-1}_{ocup}(\theta, \rho)$	Probabilidad de ocupación de una celda en el instante $t-1$
$P_{ocupado}$	Incremento de la probabilidad de ocupación de las celdas ocupadas
$Padre_i$	Padre que se cruza con $Padre_j$ para generar los hijos de la siguiente generación en el algoritmo genético
$Padre_j$	Padre que se cruza con $Padre_i$ para generar los hijos de la siguiente generación en el algoritmo genético
PC_i	Componente principal i
PC_{jk}	Proyección de $\overrightarrow{\|FFT\|}_j$ sobre el vector $\overrightarrow{\Phi}_k$ de la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$
$\{PC_{jk}\}_{k=1}^{P}$	P primeras componentes principales que constituyen el vector de características $\overrightarrow{VC}_j$
$\overrightarrow{PT}_i$	Prototipo de la clase S_i
r_i	Número de rango al que pertenece la lectura del sonar i
R	Número de regiones no conectadas tras aplicar el filtro de la mediana
ρ	Distancia de una celda hasta el robot
$\rho(x, y)$	Módulo de una celda (x, y) con respecto del centro (c_x, c_y)

s_i	Lectura del sonar i
S	Número de clases en que se clasifica la base de casos
S_i	Clase i resultante de la clasificación de la base de casos
SC	Comportamiento Seguir Pasillo
SP	Comportamiento Seguir Pared
Σ	Suma de las componentes del vector $\overrightarrow{W}_C$
TA	Tasa de aprendizaje
Tam_Celda	Tamaño de la celda de los mapas probabilísticos
$Tam\_MapaMetrico$	Número de celdas del mapa métrico
$Tam\_MapaLocal$	Número de celdas del mapa local
U_{libre}	Umbral de probabilidad para las celdas libres
$U_{metrico}$	Mínimo número de celdas que cambian para planificar
$U_{no\_explore}$	Umbral de probabilidad para las celdas libres
$U_{ocupada}$	Umbral de probabilidad para las zonas ocupadas
U_{obst}	Umbral de probabilidad para las celdas ocupadas
U_{pesos}	Umbral de activación de campo de potencial
U_{reusar}	Umbral de adaptación de casos en el ciclo CBR
v_r	Velocidad de rotación que el robot debe aplicar
v_{rC}	Velocidad de rotación del comando de movimiento del control asistido
v_{rH}	Velocidad de rotación del comando de movimiento del humano
v_{rMAX}	Máxima velocidad de rotación programada en el robot
v_{rR}	Velocidad de rotación del comando de movimiento del robot
v_t	Velocidad de traslación que el robot debe aplicar
v_{tC}	Velocidad de traslación del comando de movimiento del control asistido
v_{rH}	Velocidad de traslación del comando de movimiento del humano
v_{tMAX}	Máxima velocidad de traslación programada en el robot
v_{tR}	Velocidad de traslación del comando de movimiento del robot
v_{CPr}	Velocidad de rotación del comportamiento CP
v_{CPt}	Velocidad de traslación del comportamiento CP
v_{SCr}	Velocidad de rotación del comportamiento SC
v_{SCt}	Velocidad de traslación del comportamiento SC
v_{SPr}	Velocidad de rotación del comportamiento SP
v_{SPt}	Velocidad de traslación del comportamiento SP
$Var(PC_i)$	Porcentaje de la varianza explicado por la componente principal PC_i
$\sum_{k=1}^{m} Var(PC_k)$	Porcentaje de la varianza explicado por las m primeras componentes principales
$\overrightarrow{VC}$	Vector de características genérico
$\overrightarrow{VC}_i$	Vector de características de $\overrightarrow{\lvert FFT \rvert}_i$
$\overrightarrow{VC}_j$	Vector de características de $\overrightarrow{\lvert FFT \rvert}_j$
w_i	Peso que pondera la lectura discretizada del sonar i
w_{pared}	Peso de los comandos de movimiento del comportamiento Seguir Pared
$w_{pasillo}$	Peso de los comandos de movimiento del comportamiento Seguir Pasillo
w_{puerta}	Peso de los comandos de movimiento del comportamiento Cruzar Puerta
$\overrightarrow{W}$	Vector de pesos que pondera las entradas de los casos

$\overrightarrow{W}_C$	Vector con los factores de ponderación para los comportamientos SP, SC y CP
(x, y)	Celda genérica del mapa local
(x, y, l)	Celda en la posición (x, y) del nivel l de la pirámide
(x_1, y_1, l)	Celda (x_1, y_1) del nivel l de la pirámide
(x_2, y_2, l)	Celda (x_2, y_2) del nivel l de la pirámide
$(x_p, y_p, l + 1)$	Padre de alguna celda (x, y, l)
$(X, Y)_{(x,y,l)}$	Padre de la celda (x, y) del nivel l de la pirámide
$(X, Y)_{(x_1,y_1,l)}$	Padre de la celda (x_1, y_1) del nivel l de la pirámide
$(X, Y)_{(x_2,y_2,l)}$	Padre de la celda (x_2, y_2) del nivel l de la pirámide

Capítulo 1

Introducción

En este capítulo se describen los objetivos, la motivación y la justificación de la presente Tesis, de cara a la implementación de un sistema eficiente de navegación basado en comportamientos para la exploración completa de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil.

El capítulo comienza con una introducción al tema de la Robótica en la Sección 1. Este análisis se hace desde el punto de vista del origen de la Robótica actual, la cual ha desembocado en un estado de la disciplina que abarca un sinfín de aplicaciones. Así mismo, se muestran también las tendencias más significativas en nuestro futuro más inmediato. En la Sección 2 se enmarca la presente Tesis dentro del campo de la Robótica, justificando la motivación a partir de la cual se ha desarrollado. A continuación, en la Sección 3 se detallan los objetivos perseguidos. Finalmente, en la Sección 4 se presenta la estructura y división en capítulos de la Tesis.

1 La Robótica

Al hacer referencia al término Robótica existe una percepción bastante difundida de estar tratando sobre un tema de ciencia ficción. Nada más lejos de la realidad, puesto que la Robótica es una ciencia que abarca multitud de aplicaciones. Hoy en día, hablar de Robótica es hacerlo de progreso y desarrollo tecnológico [GTR07]. Sin embargo, todavía queda un largo camino por recorrer. En esta sección se pretende hacer una revisión de la Robótica desde su origen hasta nuestros días, lo cual determina lo que será la nueva Robótica del siglo XXI. En primer lugar se presenta una cronología de los principales hechos acontecidos relacionados con la Robótica. A continuación se describe la situación actual de la disciplina, así como sus tendencias, todo ello desde el punto de vista de las distintas aplicaciones hacia las que va destinada esta ciencia.

1.1 Historia

La Robótica es una disciplina de la cual se empezó a hablar como tal a principios del siglo XX, cuando se acuñó el término robot [Cap20]. Por ello, su historia la podemos dividir en dos

etapas diferenciadas. La primera de ellas abarca desde los tiempos anteriores a nuestra era hasta finales del siglo XIX, lo que se conoce como la Robótica Antigua. La segunda comprende desde el comienzo del siglo XX hasta nuestros días, constituyendo la Robótica Moderna. A continuación se realiza un recorrido cronológico de estas dos etapas a través de los personajes e hitos más importantes.

1.1.1 Robótica Antigua

En la Tabla 1.1 se muestra un resumen de los principales personajes que impulsaron la Robótica a través de sus investigaciones en la época anterior al siglo XX.

En la civilización egipcia encontramos muestras de los intentos del hombre por automatizar diferentes procesos. Así, hacia el año 1300 a.C. Amenhotep, hijo de Hapu, hace construir una estatua de Memom, que emitía sonidos al ser iluminada por los rayos del sol al amanecer. Ya en el siglo VI a.C. King-su Tse inventa una urraca voladora de madera y bambú y un caballo capaz de dar saltos. Posteriormente, Arquitas de Tarento construyó hacia el año 350 a.C. una paloma voladora de madera. Este filósofo y matemático griego es considerado como el padre de la ingeniería mecánica y el precursor occidental de la Robótica. En otro orden de cosas, el filósofo Aristóteles nos dejó sobre la misma época una visión futurista, hecha realidad hoy en día, en cuanto a la Robótica. Un siglo más tarde, hacia el 200 a.C., Cresibio diseña un reloj de agua con figuras móviles. Su discípulo Filón de Bizancio inventa un autómata acuático y una catapulta automática. Unos siglos después, en el comienzo de nuestra era, Herón de Alejandría muestra en su libro *Automata* (año 62) juguetes capaces de moverse por sí solos de forma repetida.

En la Edad Media también hubo personajes históricos relacionados con la Robótica, a pesar de la oscuridad y el decaimiento de la humanidad en la época. A destacar son el hombre de hierro de Alberto Magno y la cabeza parlante de Roger Bacon, los autómatas más famosos del medievo, ambos en el siglo XIII. En el Renacimiento surge, al amparo del resurgimiento de la ciencia después de unos siglos turbios, Leonardo da Vinci. Este genio multidisciplinar diseñó en 1495 uno de los primeros autómatas humanoides del mundo occidental, un caballero con armadura capaz de incorporarse, agitar los brazos y mover la cabeza [Ros06]. Esto nos hace reflexionar acerca de lo que la investigación en la robótica humanoide, tan de moda hoy en día, podría evolucionar si Leonardo estuviera entre nosotros. Nunca lo sabremos, aunque a él siempre lo podremos considerar como un precursor y visionario respecto a los robots humanoides.

También en el Renacimiento, Johann Müller construyó en lo que hoy es Alemania varios pájaros de metal y madera capaces de volar (1533). En España también hubo avances en la Robótica Antigua. Por ejemplo, Juanelo Turriano, relojero del emperador Carlos V, construye hacia 1550 el Hombre de Palo, un monje autómata capaz de andar y mover la cabeza. Unos años después, sobre 1600, Salomón de Camus construyó un coche en miniatura con caballos, lacayos y una dama en su interior, los cuales se movían de forma armónica.

En el siglo XVIII hubo grandes avances. Uno de los más famosos investigadores es Jacques de Vaucanson, quien montó en 1738 un autómata flautista que podía ejecutar melodías barrocas.

Fecha	Personaje	Hecho
1300 a.C.	Amenhotep	Estatua de Memon
500 a.C.	Kin-su Tse	Caballo de madera capaz de dar saltos
350 a.C.	Arquitas de Tarento	Paloma voladora de madera
200 a.C.	Cresibio	Reloj de agua con figuras móviles
200 a.C.	Filón de Bizancio	Catapulta automática
62	Herón de Alejandría	Juguetes autónomos
1250	Alberto Magno	Hombre de hierro
1250	Roger Bacon	Cabeza parlante
1495	Leonardo Da Vinci	Caballero con armadura
1533	Johann Müller	Pájaros voladores de metal y madera
1550	Juanelo Turriano	Hombre de Palo
1600	Salomón de Camus	Coche en miniatura con caballos
1738	Jacques de Vaucanson	Autómata flautista y pato mecánico
1769	Johann von Kempelen	Máquina para jugar al ajedrez
1770	Pierre Jaquet-Droz	Maquetas con paisajes animados
1785	David Rontgen	Autómata pianista
1801	Joseph Jacquard	Sistema de funcionamiento automático de los telares
1898	Nicola Tesla	Barco teledirigido

Tabla 1.1: Cronología de la Robótica Antigua: desde la época antigua hasta el siglo XIX.

Así mismo, alcanzó uno de los hitos determinantes en la Robótica Antigua, al construir un pato mecánico capaz de graznar, comer y completar la digestión. 30 años más tarde, en 1769, Johann von Kempelen construye una máquina casi infalible para jugar al ajedrez, un verdadero robot de servicio para entretenimiento, aunque en realidad quien jugaba al ajedrez era el campeón de la época Johann Allgaier [SMS+07]. Al año siguiente (1770), Pierre Jaquet-Droz y su hijo desarrollaron diversos muñecos capaces de escribir, dibujar y tocar melodías. Finalmente, en 1785, David Rontgen, en colaboración con Pierre Kintzing, creó un autómata pianista para la corte de María Antonieta.

El comienzo del siglo XIX vio el nacimiento de una aportación clave, la de Joseph Jacquard en 1801. Su contribución a la Robótica consiste en un sistema de funcionamiento automático de los telares, basado en cartones multiperforados. Este sistema fue incorporado en el siglo XX por los primeros ordenadores. El final del siglo XIX nos trajo, por otro lado, el que se considera primer robot de la historia. Fue inventado por Nicola Tesla en 1898. Consistía en un barco teledirigido que fue presentado en Nueva York. Con este hito se da paso a la Robótica Moderna.

1.1.2 Robótica Moderna

El avance de la Robótica durante el siglo XX y el comienzo del XXI ha sido imparable. Todo ello nos hace afirmar que, así como la sociedad de finales del siglo XX y comienzos del XXI se ha visto invadida por los ordenadores, la sociedad del siglo XXI se basará en una masiva intervención de robots en todos los aspectos de la vida cotidiana y productiva [GTR07]. Debido a que los hitos relacionados con la disciplina acontecidos durante los últimos 100 años son muy

Fecha	Personaje	Hecho
1920	Karel Capek	Obra *Rossum's Universal Robots*
1942	Isaac Asimov	Las Tres Leyes de la Robótica
1946	George Devol	Dispositivo de propósito general para controlar máquinas
1953	Grey Walter	*ELSIE*, primer robot autónomo móvil de la historia
1954	George Devol	Primer robot programable de la historia
1959	John McCarthy Marvin Minsky	Laboratorio de Inteligencia Artificial en el *MIT*
1962	Unimation	*Unimate*, primer robot industrial de la historia
1968	SRI	*SHAKEY*, robot móvil con capacidad visual
1968	Stanley Kubrick	*2001: Una Odisea del Espacio*
1970	Victor Scheinman	*Stanford Arm*, brazo articulado
1977	George Lucas	*La Guerra de las Galaxias*
1986	Honda	Robótica humanoide
1994	*NASA* Carnegie Mellon University	Dante II, explorador de volcanes
1996	Honda	*P3*, robot humanoide
1997	*NASA*	*Sojourner*, robot explorador en Marte
2000	Honda	*Asimo*, robot humanoide
2004	*NASA*	*Spirit* y *Opportunity*, robots exploradores en Marte

Tabla 1.2: Cronología de la Robótica moderna: desde el siglo XX hasta nuestros días.

numerosos, únicamente se presentan en la Tabla 1.2 los que se consideran más significativos. Los hechos que se muestran abarcan aplicaciones muy diversas de la Robótica, las cuales serán tratadas con más detalle en la Sección 1.2.

Karel Capek usó por primera vez el término robot en su obra de teatro *RUR* en 1920 [Cap20]. Esta palabra deriva del vocablo checo *robota*, que significa trabajo, en el sentido de la obligatoriedad, entendido como servidumbre o trabajo forzado. La obra aborda el concepto de la fabricación en línea ejecutada por robots humanoides, siendo éstos una aspiración del ser humano desde las culturas antiguas. El aspecto de un robot de la obra se muestra en la Figura 1.1a. Seis años más tarde, en 1926, Fritz Lang llevó esta obra a la gran pantalla en su película *Metrópolis*.

Otro de los precursores de la Robótica es Isaac Asimov, quien utilizó por primera vez el término Robótica con el sentido de disciplina encargada de construir y programar robots. Otra de sus aportaciones fundamentales consiste en Las Tres Leyes de la Robótica, definidas en su relato *Runaround* [Asi42].

A mediados de los años 40, George Devol estaba especialmente interesado en el diseño de una máquina que fuera de fácil manejo, adaptable a su entorno y que funcionara de forma automática. Este objetivo lo lograría en 1954, tras patentar un manipulador programable, auténtico embrión del robot industrial, y acuñar el término automatización. En esta época comienza un interés inusitado por la robótica industrial, para intentar conseguir que las máquinas sustituyan a los humanos en los procesos de fabricación. Unos años antes, en 1946, George Devol ya había patentado un dispositivo de propósito general para controlar máquinas. Gracias a todos sus

esfuerzos, Devol fundó junto con Joseph Engelberger la primera compañía de robots, *Unimation*. Esta empresa suministró a *General Motors* en 1962 el primer robot industrial empleado de forma práctica en la producción, *Unimate* (Figura 1.1b). Este robot consistía en un brazo robótico diseñado para para levantar y apilar grandes piezas de metal caliente de hasta 225 kilos de una troqueladora de fundición por inyección. En el mismo ámbito de la robótica industrial, Victor Scheinman crea en 1970 el brazo robótico *Stanford Arm*. Sus principios cinemáticos de diseño se convierten en un estándar, cuya influencia permanece aún en nuestros días.

Coetáneo del pionero Devol nos encontramos con Grey Walter. Este famoso investigador tiene en su poder el honroso mérito de haber sido el primero en construir un robot autónomo móvil, *ELSIE* (*Electro-Light-Sensitive Internal-External*) en 1953. Se limitaba a seguir una fuente de luz utilizando un sistema mecánico realimentado sin incorporar inteligencia adicional. Posteriormente, en 1968, apareció *SHAKEY*, desarrollado en el *SRI* (*Stanford Research Institute*). Su aspecto se puede observar en la Figura 1.1c. Estaba provisto de una diversidad de sensores así como una cámara de visión, pudiendo desplazarse por el suelo. El proceso se llevaba en dos computadores conectados por radio, uno a bordo encargado de controlar los motores y otro remoto para el procesamiento de imágenes. *SHAKEY* puede ser considerado como el primer robot controlado por Inteligencia Artificial.

La Inteligencia Artificial (IA) es una ciencia que intenta la creación de programas para máquinas que imiten el comportamiento y la comprensión humana. La investigación en el campo de la IA se caracteriza por la producción de máquinas para la automatización de tareas que requieran un comportamiento inteligente. Es un término inventado en 1956 por John McCarthy, Marvin Minsky y Claude Shannon en la Conferencia de Darmouth. Posteriormente, en 1959, los dos primeros establecieron el *Artificial Intelligence Laboratory* en el *MIT* (Massachussets Institute of Technology), con el objetivo de dar un empuje a esta ciencia. La IA permite al hombre emular en las máquinas el comportamiento humano, tomando como base el cerebro y su funcionamiento, de manera tal que se pueda alcanzar cierto razonamiento y grado de autonomía. Existen numerosos métodos basados en la IA, destacando las redes neuronales, el razonamiento basado en casos y la computación evolutiva, empleándose en la presente Tesis las dos últimas técnicas.

El interés y afán por imaginar cómo serían los robots con una inteligencia y autonomía similar a la humana se pone de manifiesto en dos obras maestras del cine. La primera corresponde a Stanley Kubrick, *2001: Una Odisea del Espacio*. Esta cinta está basada en la novela homónima del autor de ciencia ficción Arthur C. Clarke [Cla68]. En ella aparece un ordenador, *HAL*, que a veces nos hace pensar que ha superado incluso al hombre en cuanto a capacidad cognitiva. La segunda tiene a George Lucas como su creador. Es la reconocida saga *La Guerra de las Galaxias*, donde, además de plantear un futuro espacial muy alejado de la realidad actual, aparecen dos androides, *R2-D2* y *C-3P0*. Ambos poseen una inteligencia que soporta su capacidad para tomar decisiones en las situaciones más insospechadas y peligrosas. Incluso se diría que poseen emociones como los humanos. Uno de ellos, *C-3PO*, posee un aspecto y comportamiento físico bastante cercano al de la raza humana.

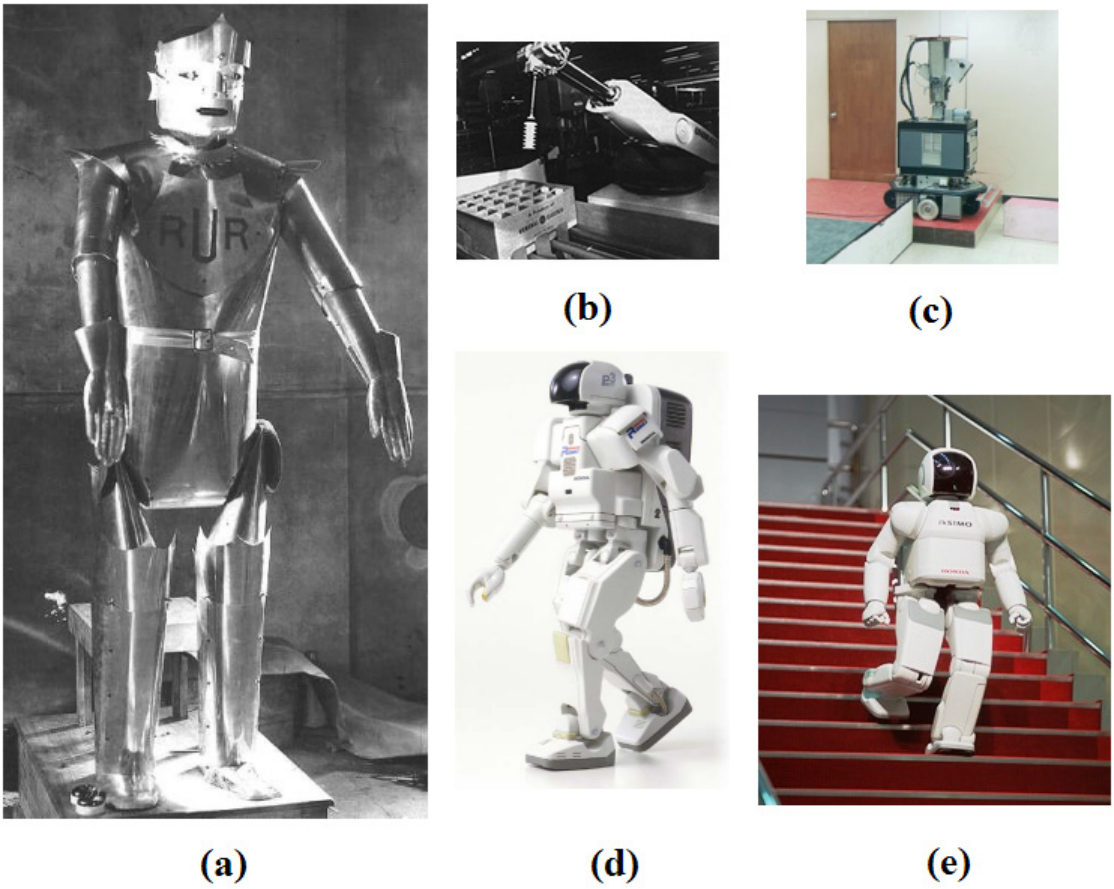

Figura 1.1: (a) Robot de *Rossum's Universal Robots* de Capek; (b) robot industrial *Unimate*; (c) robot autónomo *SHAKEY*; (d) robot humanoide P3; (e) robot humanoide ASIMO.

El anhelo irrefrenable de construir seres artificiales parecidos a los humanos en todos los sentidos, físico, cognitivo y emocional, es una constante desde hace muchos años, incluso se diría que desde hace bastantes siglos. Este deseo tuvo un impulso definitivo en los años 80, cuando Honda se embarcó en un programa de investigación cuyo punto de partida es la cooperación de los robots con los humanos en las tareas que estos no puedan resolver (1986), analizando las necesidades para poder caminar dinámicamente. Sus investigaciones pioneras tuvieron su fruto en 1996, cuando el robot humanoide *P3* fue presentado (Figura 1.1d). El descendiente aventajado de este robot es *Asimo*, que apareció en el año 2000. Este nuevo robot tiene el aspecto que se observa en la Figura 1.1e, donde se le puede ver bajando una escalera. De hecho, es capaz de marchar a dos pies, subir y bajar escaleras y de otra serie de proezas de locomoción bípeda. Comentar también que otro aspecto importante de los robots humanoides es el tema de las emociones. Destaca de entre todos ellos *Kismet*, un robot social capaz de expresar enfado, sorpresa, miedo, cansancio, alegría y otras emociones [Bre03].

No podríamos terminar este apartado sin dedicar algunos comentarios a otro ambicioso deseo del ser humano, la conquista del espacio. La necesidad de utilizar robots en el espacio está justificada por las características del entorno de operación, hostil y peligroso. Además, los

entornos de trabajo están a muchos kilómetros de distancia de la Tierra, lo que hace inviable en la actualidad el desplazamiento de seres humanos. Ello hace necesario el uso de robots exploradores de planetas [PKWN03].

Debido a su cercanía, uno de los primeros objetivos espaciales del hombre fue la Luna. El hombre aterrizó en la Luna en 1969, haciendo realidad la fantasía que un siglo antes había tenido Julio Verne en su novela *De la Tierra a la Luna* en 1865. A pesar de que el hombre ya había posado sus pies en este satélite, los vehículos soviéticos *Lunakhod 1* y *Lunakhod 2* exploraron la Luna en 1970 y 1973, respectivamente. El aspecto del primero se muestra en la Figura 1.2a, convirtiéndose en el primer vehículo controlado de forma remota en aterrizar en otro mundo.

Una vez conquistada la Luna, los esfuerzos de la carrera espacial se encaminaron hacia Marte, un planeta relativamente cercano a la Tierra. Los primeros agentes exploradores en Marte fueron los gemelos *Viking 1* y *Viking 2* (Figura 1.2b), en Julio y Septiembre de 1976, respectivamente. Ambos vehículos formaban parte del proyecto *Viking* de la *NASA*. Constaban de cámara, un brazo robótico articulado para tomar muestras (el *Stanford Arm* de Victor Schinman), y un conjunto de instrumentos meteorológicos, químicos y biológicos. Con todo ello, los vehículos fueron capaces de tomar fotografías, recopilar otros datos científicos en la superficie marciana y conducir tres experimentos biológicos diseñados para la búsqueda de posibles signos de vida.

La exploración planetaria tiene una grave inconveniente, las grandes distancias que hay que recorrer para llevar al agente autónomo hasta la superficie del planeta escogido. Este hecho conlleva un enorme tiempo, con el retraso que esto supone para los científicas por la espera infructuosa en sus investigaciones. Por este motivo, ha habido intentos de estudiar ambientes hostiles similares a los que se podrían encontrar en los planetas. Así, en 1993 la *Carnegie Mellon University* desarrolló en colaboración con la *NASA* el robot *Dante*, con 8 patas y capaz de caminar. Su objetivo era descender al Monte Erebus, un volcán activo de la Antártida, para recoger datos. Sin embargo, tras 7 metros de descenso el cable que lo sostenía se rompió, cayendo al infierno del volcán, conduciendo al fallo en la misión. Una versión más robusta fue enviada al Monte Spurr en Alaska al año siguiente. Esta segunda misión fue llevada a cabo por el robot *Dante II* (Figura 1.2c). Fue muy difícil moverse en el volcán debido a los depósitos de ceniza suave, las piedras sueltas y el hielo. La expedición realizada por *Dante II* fue considerada un éxito debido a la cantidad de datos y experiencia que se acumuló.

El siguiente gran hito en la historia de los robots exploradores planetarios lo fijó el *Sojourner* en 1997 (Figura 1.2d), al moverse por la superficie de Marte. Este hecho forma parte de la misión *Mars Pathfinder*, cuyo objetivo era demostrar que es posible enviar una carga de instrumentos científicos a otro planeta con un sistema simple y barato. El *Sojourner* estaba equipado con cámaras estereoscópicas y diversos sensores para realizar análisis de la morfología superficial, la geología, la petrología y la geoquímica de los materiales superficiales. Posteriormente, en Enero de 2004 comenzaron su funcionamiento sobre la superficie marciana los dos últimos ingenios robóticos que han explorado Marte. Se trata de los robots gemelos *Spirit* y *Opportunity*, cuyo

aspecto se muestra en la Figura 1.2e. Estos vehículos fueron enviados con el objetivo específico de localizar y examinar el suelo y las rocas de la superficie marciana para tratar de encontrar pistas de la existencia de agua en el pasado. Esta labor de investigación fue exitosa, pues se encontraron evidencias tangibles de que en la superficie de Marte hubo agua en estado liquido hace miles de años.

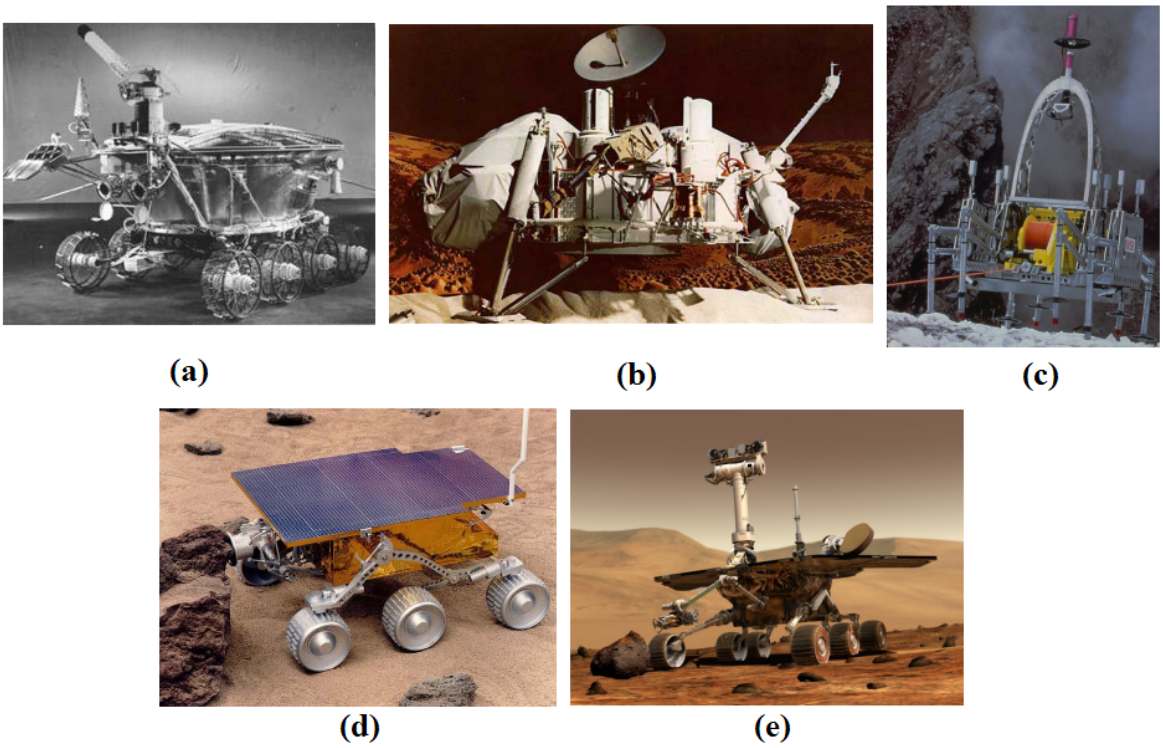

Figura 1.2: Robots exploradores: (a) *Lunakhod 1*; (b) *Viking 1* y *2*; (c) *Dante II*; (d) *Sojourner*; (e) *Spirit* y *Opportunity*.

La exploración planetaria se caracteriza porque los entornos en los que operan los agentes autónomos móviles son exteriores. No obstante, también existen otros tipos de entornos, los de interior, para los cuales están surgiendo un sinfín de aplicaciones. Entre otras destacamos la exploración, limpieza, recuperación, vigilancia, detección, guiado y asistencia a la discapacidad, encuadradas dentro de la Robótica de servicio y la Robótica de seguridad.

1.2 Situación actual y tendencias

La Robótica se está convirtiendo poco a poco en un sector estratégico, puesto que está asociada con el desarrollo tecnológico. De hecho, en los países más desarrollados las inversiones en tecnologías robóticas han crecido muy por encima que en otros sectores. La dirección hacia la que nos lleva esta situación es la de una sociedad robótica a partir del año 2010, donde se supone que la robótica de servicio va a eclosionar en muchos ámbitos, como la asistencia personal a niños y ancianos, la educación, el entretenimiento y la vigilancia.

Los sectores hacia los que está orientada la Robótica son muy amplios. Una clasificación

habitual de estos sectores es la que plantea el *EUROP* (*EUropean RObotics Platform*), que clasifica la Robótica en tres grupos de aplicaciones, las cuales se analizan a continuación:

- Robótica industrial.
- Robótica de servicio.
- Robótica de seguridad y para el espacio.

1.2.1 Robótica industrial

La Robótica industrial se define como el estudio, diseño y uso de robots para la ejecución de procesos industriales. Es el campo de aplicación que ha sido el principal receptor de los robots durante largo tiempo. Sin embargo, en los últimos años ha habido avances muy importantes en la Robótica que han evidenciado la tendencia al desarrollo de la Robótica de servicio.

A finales del año 2004 el parque de robots industriales activos en el mundo era de 847.764, según el informe *UNECE/IFR World Robotics 2005*. Aunque Japón sigue concentrando el 42% de los robots instalados (356.483), ha habido un importante crecimiento en el número de robots. Significativo es también el hecho de que, si bien Japón es el líder en el número de robots, las empresas europeas dominan el mercado mundial, siendo la primera de todas ellas la sueca *ABB Robotics*.

En el citado informa España se sitúa en un significativo 7º lugar mundial en cuanto al número de robots instalados, con 21.893. Solamente se encuentra por detrás de Japón, Alemania, Estados Unidos, Italia, Corea y Francia, tal y como se observa en la Figura 1.3. Este hecho es debido a que España es uno de los mayores fabricantes de coches del mundo, donde se encuentra el porcentaje más alto de robots de nuestro país, un 68.6% en 2004. En la industria del automóvil la mayoría de los robots se emplean para la soldadura, una actividad prioritaria en el sector. En España llega incluso a superar el 50%, siendo el país con un mayor porcentaje.

Al margen del sector del automóvil, existen muchos sectores potenciales donde la Robótica puede mejorar los procesos industriales. Así, unos campos de aplicación donde los robots están apareciendo progresivamente son la pequeña industria manufacturera y la industria de la alimentación, considerando la manipulación de piezas, aplicación de material, aplicaciones de mecanizado y las aplicaciones de control de calidad.

También existen otros sectores menos desarrollados técnicamente por distintas circunstancias, pero con un potencial elevado de robotización. Tenemos en primer lugar el sector agroalimentario. Actualmente está siendo objeto de especial atención en cuanto a la incorporación de tecnologías avanzadas, debido a una creciente carestía de mano de obra. La Robótica está participando cada vez más de forma activa en las cuatro tareas básicas del sector: i) preparación del cultivo; ii) siembra; iii) producción; y iv) recolección. En segundo lugar tenemos el sector de la construcción, donde tenemos uno de los índices de uso de robots más bajos entre los sectores productivos.

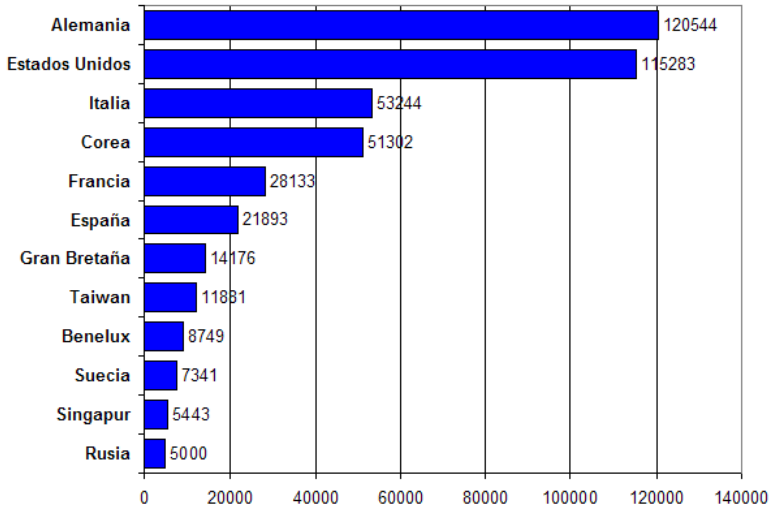

Figura 1.3: Parque mundial de robots en el año 2004, excluido Japón.

1.2.2 Robótica de servicio

Según la definición del Comité Técnico de Robots de Servicios del Instituto Americano IEEE, los robots de servicios son aquéllos que de forma semiautomática o totalmente automática realizan servicios en beneficio de los humanos o para el mantenimiento de infraestructuras y equipos, excluidas las operaciones de fabricación. Sus características más importantes deben ser la movilidad, manipulabilidad, percepción, inteligencia, interfaz amigable y tener un coste relativamente bajo.

El *Informe de la Robótica de Servicio 2005* en España pone de manifiesto que en el año 2004 había 160 empresas con actividad en este ámbito. La mayoría de ellas pertenecen a Estados Unidos, Japón, Alemania y Gran Bretaña. Es destacable que no hay conocimiento de ninguna empresa en España dedicada a esta actividad. En el mismo informe se recoge que el número de robots de servicio en el mundo dedicado a actividades profesionales era de unos 21.000 en el año 2003. Para las aplicaciones domésticas ese número rondaba los 1.3 millones de unidades. Quizás el hecho más reseñable es la previsión de enorme crecimiento de este área en los próximos años, de incluso un 1.000%.

A continuación se presenta un breve análisis sobre las áreas más destacadas de actividad de la Robótica de servicio.

1. Robots para usos domésticos. Representan el principal exponente de las posibilidades de crecimiento de la Robótica a corto plazo. Existen, por ejemplo, aplicaciones comerciales dedicadas a la limpieza doméstica (*Roomba*), utilizando tecnología de navegación inteligente. Para el futuro se prevé que los asistentes domésticos y las aplicaciones de cuidado de ancianos y discapacitados serán las que acaparen mayor atención.

2. Robots en educación y entretenimiento. Ha dado lugar a un nuevo término, *edutainment*,

que agrupa la Robótica encaminada a interaccionar con personas, sobre todo niños y jóvenes. Destacan en este área tres tipos de aplicaciones:

- Educación infantil y juvenil. En España existen grupos de investigación que han desarrollado robots para el cuidado y entretenimiento de niños. Cuentan con una movilidad aceptable en entornos escolares y domésticos, pudiendo mover la cabeza, los brazos, los labios y los párpados. La educación también puede estar encaminada a la construcción y manejo de robots. Uno de los sistemas más famosos de este tipo es el sistema *LEGO Mindstorms*, compuesto por fichas de lego, un microprocesador y algunos sensores.
- Juguetes y mascotas. El factor diferenciador de estas máquinas es su interacción con el usuario, siendo capaces de comunicarse con él, junto con la implantación de técnicas de Inteligencia Artificial, que consiguen que el juguete-robot aprenda. A pesar de que España posee un sector del juguete de destacada tradición, son los fabricantes extranjeros los que han tomado la iniciativa, aunque nuestro país está perfectamente capacitado para entrar en el mercado. Entre los juguetes-robot destacan el *Furby*, el *Aibo* y el *Robosapien*. El *Furby* es un juguete con aspecto de peluche tristón que tiene su propio lenguaje, introducido por *Tiger Electronics* en 1998 (Figura 1.4a). Aprende nuestro idioma, pudiendo decirse a ciencia cierta que a veces nos deja con la boca abierta por las reacciones que tiene y su capacidad de aprendizaje. El *Aibo* es un perro mascota introducido por *Sony* en 1999. Dispone de sensores que le evitan chocar contra objetos, pudiendo tratarse como si fuera un compañero interactivo (Figura 1.4b). El aspecto de *Robosapien* se aprecia en la Figura 1.4c. Es una mascota con un aspecto parecido a los humanos que puede andar, coger y lanzar objetos.

  Al margen de las mascota-robot comentadas, existen algunas aplicaciones curiosas de la Robótica para el entretenimiento. Como ejemplo, un robot capaz de resolver el célebre *Cubo de Rubik* [Gar00]. Este rompecabezas fue inventado por el húngaro Erno Rübik en 1974. El hombre se ha superado a sí mismo, fijando el récord en tan solo 9.86 segundos. El robot más rápido puede resolver el puzzle en 15 segundos. El problema en este caso está en los temas mecánicos, ya que en los aspectos algorítmicos la solución hace mucho que se ha alcanzado. Aun así, se puede decir que el hombre está llegando a su límite, mientras que el avance tecnológico seguro que hará que la máquina supere al humano.
- Robots en parques temáticos y exposiciones. En los parques temáticos de España abundan los autómatas con apariencia humana o animal. Aunque poseen unos movimientos y actitudes muy cercanas a la realidad, no son interactivos, para lo que sería preciso incorporar técnicas de Inteligencia Artificial. Este aspecto está siendo investigado actualmente en centros extranjeros y nacionales.

  Una actividad cercana al uso de los robots en los parques temáticos es su empleo como guías para museos y salas de exposiciones. Dos ejemplos representativos de este campo son *RHINO* [TBB+98] y *MINERVA* [TBB+99]. En España ya se han

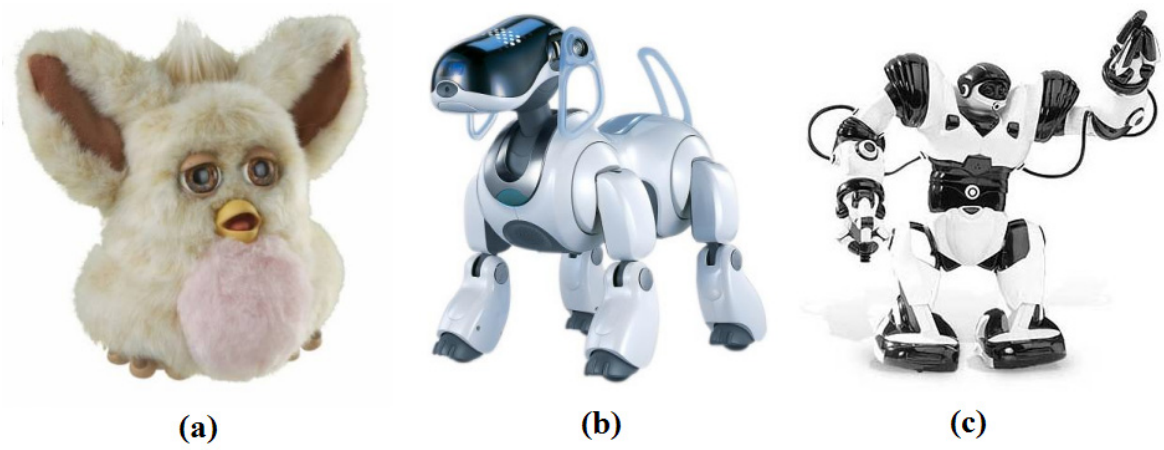

Figura 1.4: Juguetes-robot: (a) *Furby*; (b) *Aibo*; (c) *Robosapien*.

desarrollado prototipos para visitas guiadas. Sin embargo, Estados Unidos y algunos países de Europa llevan la delantera. Por lo tanto, sería interesante que en España se impulsara este área, para no quedar relegados a un segundo plano.

3. Robots en la sanidad. La medicina es uno de los campos de aplicación más tangibles de la Robótica, por los beneficios sociales obtenidos. Las aplicaciones existentes son muchas, siendo su principal limitación el elevado coste. Nos encontramos con aplicaciones en:

   - Cirugía. Incluye la cirugía robotizada guiada por imagen y el uso de telerobots en cirugía mínimamente invasiva (robot *daVinci* de *Intuitive Surgical*). Fundamentalmente son las empresas estadounidenses las que comercializan estos robots a precios muy elevados, lo que hace difícil su adquisición. En España también hay varios grupos de investigación que desarrollan actividad en este tema, aunque no llegan al nivel de las soluciones comerciales.
   - Rehabilitación. Al aumentar la esperanza de vida también lo hace la necesidad de atender dolencias o patologías asociadas a la edad. La tecnología robótica actual proporciona soluciones adecuadas para ayudar en muchas de las actividades de rehabilitación.
   - Sillas autónomas y robots asistenciales. El objetivo de las sillas autónomas es dotar de mayor nivel de independencia a las personas con movilidad limitada. La investigación en este campo es alta, participando de ella el grupo de investigación en el que se ha desarrollado la presente Tesis. La mera monitorización de las sillas de ruedas es insuficiente para responder a las necesidades del paciente. Sería interesante que la silla actuara en función del estado del paciente, estando encaminadas las investigaciones en este sentido [FEPUS07a]. Por otro lado, los robots asistenciales están destinados a auxiliar en sus actividades diarias a aquellas personas que carecen de la movilidad necesaria para su autonomía. En España existen también desarrollos de este tipo de robots, aunque los recursos empleados son mucho menores que en el caso de las sillas.

1.2.3 Robótica de seguridad y para el espacio

Este grupo de aplicaciones abarca, por un lado, los temas de seguridad individual y colectiva y, por otro, la Robótica espacial, ya introducida anteriormente.

1. Robótica de seguridad. La sociedad actual se encuentra inmersa en una gran problemática medioambiental, lo que ha motivado que la Robótica haya extendido sus brazos hacia la mejora de la calidad del entorno en el que vivimos. Principalmente se dedica a la vigilancia de superficies para la prevención de catástrofes naturales mediante robots de vigilancia aérea o robots marinos y submarinos. En cualquier caso, se trata de equipos cuya misión es la captura inteligente de información de variables ambientales para su tratamiento e interpretación de manera manual o automática, con el objetivo de actuar en consecuencia.

   Otras aplicaciones de la Robótica de seguridad son el rescate de personas en ambientes hostiles y de difícil acceso para los especialistas, así como la desactivación de minas. En este último caso, además de la autonomía, es crítico el diseño físico del robot para conseguir la protección suficiente frente a las detonaciones.

2. Robótica espacial. El estado de la Robótica espacial nos ha demostrado que es posible recabar información de ambientes lejanos y hostiles donde el hombre todavía no puede llegar. El desarrollo de estos sistemas autónomos requiere de características adicionales determinadas por el entorno de operación. Entre otras, navegación en entornos dinámicos y desconocidos, manipulación de objetos sin dañar al robot, tolerancia a condiciones ambientales extremas y autonomía, incluyendo subsistemas de supervisión y autodiagnóstico.

2 Motivación y Justificación

La Robótica se usa hoy en día de forma extendida en multitud de aplicaciones, muchas de ellas comerciales. Muchas de estas aplicaciones están basadas en plataformas que disponen de ruedas para desplazarse por el entorno. Su operación se desarrolla en entornos dinámicos y, potencialmente, no estructurados. Por lo tanto, es necesario llevar a cabo su exploración para adquirir conocimiento del mismo. Con la presente Tesis se pretende contribuir a este tipo de aplicaciones.

Por otro lado, se puede afirmar, sin temor a equivocarnos, que España tiene un potencial investigador significativo en el campo de la Robótica. Es más, algunos de los grupos de investigación son pioneros y líderes a nivel europeo y mundial en algunas líneas de investigación. Teniendo en cuenta las líneas de investigación existentes en nuestro país, el número de grupos, su fortaleza internacional y el interés futuro, el Libro Blanco de la Robótica [GTR07] establece cuáles son las fortalezas de la I+D+i de la Robótica española, estudio avalado por la Red Nacional de Robótica y subvencionado por el Ministerio de Educación y Ciencia. Del análisis se desprenden siete líneas fuertes, ordenadas según su importancia. La primera de ellas es la *Robótica Autónoma*, temática a la que se contribuye con la presente Tesis. Es el área de la

Robótica que desarrolla robots capaces de desplazarse, actuar y tomar decisiones independientes sin intervención humana.

Además de las siete líneas fuertes comentadas, existen otras en las cuáles también se debería centrar el esfuerzo en I+D+i en base a cuatro criterios [GTR07]: i) las fortalezas y debilidades del tejido robótico; ii) las prioridades de la Robótica internacional; iii) la demanda futura empresarial; y iv) las necesidades de nuestro entorno social. Todo ello lleva a reorganizar las prioridades de la Robótica española en diez líneas, donde la *Robótica Autónoma* sigue siendo la más importante. Dentro de estas prioridades nos encontramos también con otra línea relacionada con la presente Tesis, el comportamiento cognitivo y aprendizaje.

Dentro de esas dos líneas estratégicas existen contenidos que justifican por sí mismos la presente Tesis. Así, en el campo de la *Robótica Autónoma*, el análisis presentado en el Libro Blanco de la Robótica recomienda enfocar los esfuerzos, entre otros, hacia el desarrollo de nuevos algoritmos de navegación robusta en entornos reales de interior. En cuanto a los sistemas cognitivos se indica que sería conveniente desarrollar sistemas capaces de efectuar un aprendizaje evolutivo y responder a la falta de conocimiento de forma razonable. En la presente Tesis se realizan aportaciones a ambos contenidos concretos, justificando su desarrollo.

La navegación es una de las características principales que un sistema autónomo móvil debe poseer. El origen del término hay que buscarlo en la naútica, donde se aplicaba al proceso de dirigir un barco hacia su destino. Este mismo sentido es el que adquirió cuando se comenzó a emplear en la Robótica, donde se define de muchas maneras. Una de ellas es la que considera la navegación como un problema para alcanzar un objetivo prefijado de una manera segura. Tradicionalmente también se ha definido como el proceso de determinar y mantener una trayectoria hasta un objetivo [Gal90]. Quizás una de las definiciones más conocidas sea la de Levitt y Lawton [LL90]. Definen la navegación como el proceso de respuesta de tres preguntas: i) ¿dónde está el robot?; ii) ¿dónde están el resto de localizaciones con respecto al robot?; y iii) ¿cómo puede el robot ir a un punto desde su posición?.

La definición de Levitt y Lawton implica que el robot, para poder navegar, debe mantener una representación del entorno basada en su percepción sensorial a partir de la cual se decidan los movimientos a realizar. Muchos sistemas autónomos se basan en esta aproximación, aunque ninguno de ellos ha alcanzado la flexibilidad y los comportamientos de navegación de los animales. Sin embargo, muchos animales, incluidos los humanos, llevan a cabo tareas de navegación sin necesidad de responder a todas las preguntas de Levitt y Lawton e, incluso, sin responder a ninguna de ellas. Por lo tanto, para poder llegar a imitar a los organismos vivos se debe ampliar el concepto de navegación, ya que la mayoría de los mecanismos biológicos de navegación están excluidos de la idea de Levitt y Lawton.

Los comportamientos de navegación inspirados en los seres vivos han sido profundamente estudiados porque la naturaleza es el mejor modelo para estos sistemas y porque se pueden examinar, probar y refinar las teorías en las que se basa un diseño [FND03]. El esfuerzo ha estado encaminado a la consecución de sistemas autónomos que simulan o imitan, de uno u

otro modo, la inteligencia que poseen las criaturas vivas. Los comportamientos de navegación que implementan estos sistemas se pueden clasificar de acuerdo a la Jerarquía de Navegación de Franz y Mallot [FM00], uno de los esquemas más populares. La Jerarquía de Navegación es una extensión de los trabajos de Trullier [TWBM97]. Los comportamientos de navegación se clasifican de acuerdo a la complejidad de la tarea que llevan a cabo. La Jerarquía de Navegación admite dos grupos de comportamientos, Locales y Globales o de Búsqueda de Caminos:

1. *Comportamientos de Navegación Local.* Son reactivos, decidiendo el agente sus acciones en función de la información sensorial captada en cada momento, sin necesidad de poseer una representación del entorno. Se dividen en cuatro niveles: *Búsqueda*, *Seguimiento*, *Apuntado* y *Guiado*.

2. *Comportamientos de Búsqueda de Caminos.* Requieren del reconocimiento de lugares en el entorno, así como la representación de las relaciones entre estos lugares, basándose en un modelo del entorno. Es importante resaltar que se basan en los Comportamientos de Navegación Local o reactiva para desplazarse de un lugar a otro. Esto va a permitir al agente encontrar lugares que no se podrían hallar empleando únicamente la navegación local. Hay tres clases de técnicas globales o de Búsqueda de Caminos:

   - *Respuesta Disparada por Reconocimiento.* Mediante estas técnicas se conectan dos localizaciones usando un método de navegación local [GZ95]. Para ello se emplea un par *(localización inicial, método de navegación local)* para representarlos. Una vez reconocida la localización inicial, se dispara el método de navegación local.
   - *Navegación Topológica.* Este tipo de navegación se basa en un mapa topológico del entorno. En este caso sí se puede realizar una navegación global, planificando rutas gracias a este mapa. Sin embargo, las zonas no exploradas no se incluyen en la representación. Por este motivo, un mapa topológico no se puede emplear para planificar rutas a través de zonas desconocidas.
   - *Navegación por Inspección.* Requiere de la inserción de todos los lugares y de sus relaciones espaciales en un marco común de referencia. La representación es accesible como un todo. De ahí que se pueda deducir a partir de la misma la relación entre cualquier pareja de lugares representados. Cuando un robot soporta la Navegación por Inspección puede, por tanto, planificar rutas a través de zonas desconocidas.

Un agente que posea las capacidades de un determinado nivel también posee todas las capacidades de los niveles inferiores de su grupo. Sin embargo, un agente que posea comportamientos de Búsqueda de Caminos no tiene necesariamente que ser capaz de desarrollar todos los comportamientos de Navegación Local. En el reino animal nos encontramos con muchos comportamientos de Navegación Local, Respuesta Disparada por Reconocimiento y Navegación Topológica, mientras que la Navegación por Inspección está limitada a los vertebrados, lo que nos indica la dirección seguida durante la evolución. Quizás por esta razón existen imitaciones

de los organismos biológicos mediante sistemas reales de navegación que implementan todos los comportamientos de la Jerarquía de Navegación con excepción del nivel de Navegación por Inspección. Ésta es la motivación que nos ha llevado al desarrollo de la presente Tesis, el alcanzar un sistema autónomo de navegación que implemente el nivel más alto de la Jerarquía de Navegación de Franz y Mallot, la Navegación por Inspección. Este nivel requiere de un modelo del entorno que represente sus zonas no exploradas. Sin embargo, este no es un problema obvio, al cual se contribuye con el trabajo desarrollado en la presente Tesis. Otro de los problemas a los que se realizan aportaciones en esta Tesis es el del comportamiento reactivo de Navegación Local a escoger en cada momento, que implica el reconocimiento rápido de lugares a bajo nivel.

3 Objetivos

El objetivo de esta Tesis es el desarrollo de un sistema eficiente de navegación basado en comportamientos para la exploración completa de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil. Las características de este sistema lo deben dotar de la funcionalidad más elevada dentro de la Jerarquía de Navegación, la Navegación por Inspección. Esto implica que nuestro sistema deberá estar dotado de la capacidad de planificar rutas a través de zonas desconocidas. Para ello se necesita representar las áreas desconocidas del entorno. En esta Tesis se propone una estructura que integra los paradigmas métrico y topológico de modelado de entornos, para así aprovechar las ventajas de ambos. Por otro lado, el sistema debe adaptarse a los cambios que supone la progresiva exploración que el agente lleva a cabo.

Tal y como se ha comentado en la Sección 2, la Navegación por Inspección es el nivel superior del grupo de comportamientos de Búsqueda de Caminos. Todos los comportamientos que pertenecen a este grupo comparten el hecho de que se basan en los Comportamientos de Navegación Local o reactiva para moverse por el entorno. Por lo tanto, se necesita algún comportamiento de navegación reactiva para conseguir alcanzar nuestro objetivo. En nuestro caso, el sistema está destinado a entornos de interior. Se va a basar en tres comportamientos, habitualmente utilizados cuando el robot está explorando un área desconocida [MMBB06]: Seguir Pared, Seguir Pasillo y Cruzar Puerta, buscando su cooperación en todo momento. Esto exige el reconocimiento rápido de los lugares por los que navega el robot. Además, para asegurar la integración en el entorno, el robot deberá disponer de capacidad de razonamiento y aprendizaje a partir de la información sensorial que reciba. Con todas estas premisas, a partir del objetivo planteado y de cara a la implementación del sistema completo establecemos unos objetivos específicos más concretos:

- Diseñar e implementar los tres Comportamientos de Navegación Local, incorporando mecanismos de razonamiento y de aprendizaje que permitan su adaptación a diferentes plataformas robóticas con pequeños cambios.
- Diseñar e implementar un algoritmo para la planificación de rutas de exploración completa en entornos total o parcialmente desconocidos. Este algoritmo deberá estar basado en un

modelo del entorno donde las zonas no exploradas estén explícitamente representadas.

- Desarrollar un mecanismo de cooperación de los comportamientos de navegación reactiva que se adapte a las circunstancias y particularidades del entorno en cada momento.
- Integrar las distintas partes que conforman el sistema y validar su funcionalidad sobre una plataforma robótica real.

Para alcanzar los objetivos propuestos habrá que tener en cuenta las siguientes consideraciones:

- El agente móvil funcionará de manera autónoma para realizar las tareas de navegación.
- Los entornos hacia los que va destinado el sistema son interiores y dinámicos.
- El funcionamiento del sistema deberá ser eficiente en el uso de los recursos existentes en cuanto al tiempo y la memoria, permitiendo la adaptación a los cambios que sufre el entorno.
- La plataforma robótica empleada constará únicamente de sensores sonar para adquirir información del entorno y un sistema de posicionamiento basado en odometría. El sistema diseñado pretende ser independiente de la plataforma, de ahí que se realicen los experimentos en dos distintas. En esta Tesis se realizan pruebas simuladas y reales. Para las pruebas simuladas se emplea el simulador de la plataforma robótica *Nomad 200* de *Nomadics*, donde el robot consta de un anillo de 16 sensores sonar equiespaciados. Para las pruebas reales se utiliza la plataforma robótica real *Pioneer P2AT*, equipada con 8 sensores sonar *Polaroid* frontales.

4 Organización del texto

Además de esta introducción, la presente Tesis está organizada en los siguientes capítulos:

- *Capítulo 2. Arquitectura del Sistema.* En este capítulo se presenta la arquitectura de nuestro sistema, consistente en la integración de un conjunto de módulos y capas que definen la estructura de la misma. De todas las capas del sistema, en este capítulo se presentan tres, las que soportan el modelado del entorno y las interfaces entre el robot y el sistema, así como entre el sistema y el usuario.
- *Capítulo 3. Comportamientos para Navegación Reactiva.* Este capítulo está dedicado a la implementación realizada de los Comportamientos de Navegación Local de nuestro sistema: Seguir Pared, Seguir Pasillo y Cruzar Puerta. En primer lugar se presenta la implementación según un modelo analítico. Sin embargo, se pretende evitar el tener que efectuar análisis cinemáticos y dinámicos cuando se cambia de plataforma y/o sensores.

Por lo tanto, nuestro sistema debe incorporar mecanismos de razonamiento y aprendizaje, por lo que también se ha optado por desarrollar estos comportamientos según el razonamiento basado en casos (*CBR*). Los resultados que se presentan demuestran el correcto funcionamiento de ambas aproximaciones, superando el *CBR* al modelo analítico en algunos aspectos.

- *Capítulo 4. Fusión de Comportamientos.* Este capítulo presenta una aportación novedosa de la presente Tesis, una técnica para la cooperación o fusión de los tres comportamientos reactivos introducidos en el Capítulo 3. El método tiene en cuenta la situación particular del entorno en cada momento y proporciona una vía eficiente para ponderar en todo instante la contribución de cada uno de los tres comportamientos a la navegación reactiva y así seguir de la manera más adecuada la ruta de exploración completa planificada mediante el algoritmo que se analiza en el Capítulo 5.

- *Capítulo 5. Exploración Completa de Entornos.* En este capítulo se propone un método para el cálculo eficiente de rutas de exploración completa para entornos total o parcialmente desconocidos. En primer lugar se presenta la estructura jerárquica que soporta el método, donde están explícitamente representadas las zonas no exploradas. En segundo lugar se introduce el algoritmo desarrollado para la planificación eficiente de rutas que exploren todas las zonas no exploradas del entorno, todo ello gracias a la estructura jerárquica. Por último, se presentan los resultados que validan nuestra propuesta.

- *Capítulo 6. Resultados.* En este capítulo se muestran las pruebas globales mediante las cuales se verifica el correcto funcionamiento del sistema propuesto para la exploración completa de entornos total o parcialmente desconocidos.

- *Capítulo 7. Conclusiones y Líneas Futuras.* Por último, este capítulo presenta las conclusiones extraídas después de la realización de esta Tesis, haciendo hincapié en las aportaciones efectuadas. Así mismo, también se abren futuras líneas de investigación sobre el tema tratado.

Finalmente, se ha considerado conveniente incluir dos apéndices que contemplan aspectos adicionales del desarrollo de la presente Tesis:

- *Apéndice A. Descripción del Software del Sistema.* Este apéndice incluye una descripción y explicación del *software* desarrollado para dar sustento al sistema que se presenta en la presente Tesis.

- *Apéndice B. Cuestionarios de Usuario del Control Asistido.* En este último apéndice se presenta el modelo de cuestionario y los cuestionarios reales rellenados por los distintos voluntarios participantes en los experimentos del control asistido.

Capítulo 2

Arquitectura del Sistema

En este capítulo se describe la arquitectura del sistema desarrollado para la exploración completa de forma eficiente de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil.

Antes de analizar la estructura empleada se presenta en la Sección 1 una visión panorámica de las arquitecturas de control, comenzado por los inicios de las arquitecturas deliberadas, pasando por las reactivas y llegando a las usadas actualmente, las arquitecturas híbridas. Seguidamente se presentan en la Sección 2 los mecanismos de interacción empleados para la correcta integración de los distintos módulos del sistema, es decir, el estilo usado. Llegados a este punto se introduce en la Sección 3 la estructura del sistema desarrollado para cumplir nuestros objetivos. En este sistema existen varios módulos que, por su importancia, son analizados por separado en los Capítulos 3, 4 y 5. El resto de los módulos son necesarios para soportar la funcionalidad del sistema, siendo analizados en este capítulo. Finalmente, la Sección 4 presenta las conclusiones que se derivan de este capítulo.

1 Introducción

La complejidad es una característica común a todos los sistemas autónomos móviles. Puesto que los objetivos son o pueden ser más o menos difíciles, se necesita interactuar con entornos dinámicos y complejos, y todo ello manteniendo un nivel de seguridad y reacción ante situaciones inesperadas. Para facilitar el diseño de estos sistemas, la Robótica necesita de mecanismos que simplifiquen su desarrollo. Esta ayuda proviene de las arquitecturas de control. Permiten definir las unidades funcionales del sistema, conocidas como módulos, así como sus interconexiones. También suministran los mecanismos de interacción entre módulos para alcanzar la correcta integración de todos ellos.

A continuación se realiza un recorrido por los principales tipos de arquitecturas de control que nos podemos encontrar: las arquitecturas deliberadas, las arquitecturas reactivas y las arquitecturas híbridas. Este estado del arte se presenta según la natural evolución cronológica, desde las arquitecturas deliberadas, pasando por las reactivas y llegando a las empleadas en la

actualidad, las híbridas.

1.1 Arquitecturas deliberadas

Hasta los años 80 la tendencia generalizada para controlar un sistema autónomo móvil era aquélla que lo descomponía en tres elementos funcionales [Nil82]:

- Sistema sensorial. Se encarga de obtener una representación del entorno a partir de la entrada sensorial, normalmente sonar o visión.
- Sistema de planificación. Su misión es generar un plan para alcanzar un objetivo a partir del modelo de entorno obtenido por el sistema sensorial.
- Sistema de ejecución. Su labor es generar las acciones que el agente debe ejecutar tomando el plan proporcionado por el sistema de planificación.

Este esquema se conoce como aproximación *SPA* (*Sense-Plan-Act*), el empleado en las denominadas arquitecturas deliberadas. La planificación deliberada se basa en una metodología *top-down*, conocida como descomposición horizontal [Alb91, HB96]. Estos esquemas se dividen en varios niveles sucesivos de un modo jerárquico, los cuales intercambian información mediante flujos predefinidos. Mientras que los niveles más altos de la jerarquía soportan la planificación y el razonamiento, los niveles inferiores controlan las acciones a ejecutar sobre el *hardware*. Sin embargo, los flujos son unidireccionales y lineales, haciendo que la comunicación entre niveles ocurra de una manera predecible y determinista.

A pesar de que los sistemas deliberados eran los usados en las primeras etapas del desarrollo de arquitecturas para agentes autónomos móviles, poseen muchas limitaciones que hicieron que fueran cayendo en desuso:

- Todos los módulos del sistema deben estar funcionales para poder llevar a cabo una prueba sobre él. Si algún módulo fallara, todo el sistema se vendría abajo, por la linealidad de la aproximación.
- Del mismo modo, ese flujo lineal provoca que el tiempo de respuesta del sistema sea elevado, ya que un módulo debe esperar a que el inmediatamente superior finalice su ejecución.
- Obviamente, el elevado tiempo necesario para que el sistema actúe a partir de la información de los sensores hace que no pueda reaccionar rápidamente ante cambios o situaciones inesperadas del entorno de trabajo.

1.2 Arquitecturas reactivas

En los años 80 se demostró claramente que el ciclo *SPA* no era una buena política a la hora de gestionar la complejidad de las tareas de los agentes autónomos móviles. Varios investigadores

sugirieron por la época que se necesitaban mecanismos de ejecución diferentes a los de las arquitecturas deliberadas [Pay86, Fir87, AC87].

Frente a los sistemas deliberados aparecieron los sistemas reactivos, basados en una descomposición *bottom-up*. Los esquemas reactivos se basan en la asociación directa entre los sensores y las acciones que se llevan a cabo, sin usar un modelo de entorno. En las arquitecturas reactivas existe un conjunto de módulos denominados comportamientos. Los comportamientos se ejecutan de forma concurrente, interactuando cada uno de ellos con el entorno. El resultado de la ejecución simultánea de todos los comportamientos es una acción global que combina uno o más comportamientos reactivos, denominada comportamiento emergente.

Al contrario que los sistemas deliberados, los sistemas reactivos son rápidos y pueden tratar con varios sensores a la vez. Además, son bastante robustos frente a los errores de los sonar y al ruido, pudiendo ser modificados de manera sencilla para tratar con diferentes plataformas robóticas e incluso distintas tareas. Así mismo, el sistema puede reaccionar de forma rápida ante cambios en el entorno o situaciones imprevistas, no siendo necesario para ello el tener una representación del entorno.

El máximo exponente de los sistemas reactivos es la arquitectura de subsunción de Rodney Brooks [Bro86], quien fue el primero en presentar un sistema basado únicamente en comportamientos reactivos muy sencillos que se combinan entre sí. La arquitectura de subsunción posee una estructura en capas, constituidas por módulos que cooperan entre sí para alcanzar un determinado comportamiento. Las capas están distribuidas de forma jerárquica, permitiendo la adición de nuevas funcionalidades a los comportamientos básicos. Para interactuar, los comportamientos emplean los mecanismos de supresión e inhibición temporal. Al ser los flujos de información asíncronos, se asegura que todos los módulos operan con los datos más recientes. Esta arquitectura alcanzó su tope con el robot *Herbert* [Con89], cuya tarea era encontrar y recoger latas en un entorno de oficinas.

A pesar de las ventajas que las arquitecturas reactivas tienen sobre las deliberadas, también adolecen de algunos inconvenientes que se han hecho más patentes a lo largo de los años:

- Los comportamientos emergentes pueden ser impredecibles.
- Por la limitación anterior, el esquema de operación del robot puede no ser eficiente, pudiendo quedar atrapado en trampas de mínimos locales.
- El desarrollo de tareas complejas no puede realizarse con un funcionamiento adecuado.

1.3 Arquitecturas híbridas

Los sistemas híbridos surgen ante la necesidad de solventar los problemas que por separado presentan tanto los sistemas deliberados como los reactivos. Combinando ambas aproximaciones se consiguen mejores prestaciones, ya que se aprovechan los beneficios de la reactividad y la deliberación. Así, surgieron arquitecturas de control que intentaban superar las limitaciones de

la arquitectura de subsunción [RP89], las denominadas arquitecturas híbridas. Normalmente, el control de bajo nivel se implementa de un modo reactivo mientras que el procesamiento y razonamiento de alto nivel sigue un esquema deliberado. Así, estos sistemas son capaces de alcanzar una navegación eficiente en entornos dinámicos total o parcialmente desconocidos. El principal obstáculo a superar a la hora de diseñar sistemas híbridos será, por tanto, el encontrar la correcta integración de la respuesta reactiva con el razonamiento para alcanzar la funcionalidad deseada.

Uno de los primeros robots implementados con la nueva filosofía fue *Tooth* [Ang89], equipado con sensores muy simples. Estaba programado para buscar objetos pequeños, cogerlos y llevarlos hasta un punto de destino. Si bien *Tooth* era un robot para entornos interiores, también nos podemos encontrar por la misma época agentes autónomos móviles para exteriores. Así, aparece *Rocky III* [MDG+92], con la misma funcionalidad que *Tooth*. Ambos sistemas eran muy fiables, superando a *Herbert* en su operación, ya que podían trabajar durante mucho tiempo sin fallo.

Tanto *Tooth* como *Rocky III* compartían un severo inconveniente. En ambos casos no era posible cambiar la tarea para la que habían sido diseñados sin reescribir su programa de control. La solución a este problema llegó de la mano de tres grupos de investigadores coetáneos, que resolvieron el enigma de forma independiente [Con92, Bon91, Gat91]. La arquitectura *SSS* de Connell [Con92] estaba basada en la arquitectura de subsunción de Brooks. El sistema de Bonasso se conoce como arquitectura *3T* [Bon91], siendo su base los trabajos de Kaelbling [Kae88]. Por último, la arquitectura *ATLANTIS* de Gat [Gat91] usa las investigaciones de Firby [Fir89] en su desarrollo. Estas tres arquitecturas eras similares en su concepción, habiendo pasado a la historia de la Robótica como el punto de partida de las arquitecturas de tres capas (*Three-Layer Architectures*), estándar de referencia de los sistemas híbridos [Gat98].

Las arquitecturas de tres capas son sistemas híbridos que se modelan con tres capas independientes [OC03], como la arquitectura que se propone en la presente Tesis:

- Capa reactiva. Se encarga de realizar el control de bajo nivel.
- Capa deliberada. Incluye el razonamiento, el procesamiento y el control de alto nivel.
- Capa intermedia. Su misión es actuar de interfaz entre la capa reactiva y la capa deliberada.

Normalmente, la capa reactiva de un sistema híbrido se basa en comportamientos. El sistema consta de varios comportamientos simples que producen respuestas independientes ante la entrada sensorial. Todos ellos operan de forma concurrente, siendo sus respuestas fusionadas en un único comando de movimiento. Además, la lectura dada por los sensores se suele integrar para extraer una representación del entorno. Ambos aspectos, fusión de comportamientos e integración de sensores, son empleados en el sistema que se presenta en esta Tesis.

Las arquitecturas híbridas son las empleadas en la actualidad para diseñar agentes autónomos móviles, toda vez que los resultados han demostrado que es la mejor opción. Sin embargo, las soluciones que han surgido son particulares y específicamente diseñadas para una

aplicación concreta. Así, los principales sistemas desarrollados son *AuRA* [Ark90], *CIRCA* [MDS93], *TCA* [Sim94], *Saphira* [KM98], *OrCCAD* [BCME+98], *LAAS* [ACF+98] y *BERRA* [LOC00]. Un análisis de las características de estas arquitecturas híbridas se puede encontrar en [Gat98, CMS00, OC03].

2 La arquitectura *DLA*

Cualquier arquitectura de control se puede dividir en dos niveles de descripción estrechamente relacionados [CMS00]:

- Estructura. Hace referencia a la descomposición del sistema en módulos funcionales, definiendo las interacciones entre ellos.
- Estilo. Tiene en cuenta el problema de la integración de los módulos del sistema, definiendo los mecanismos computacionales empleados para el adecuado intercambio de información.

Es esta Tesis se emplea el estilo de la arquitectura *DLA* (*Distributed and Layered Architecture*), presentado en [Pér06]. Esta arquitectura permite implementar sistemas cooperativos mediante la interacción de procesos libremente distribuidos. Su diseño simplifica el desarrollo de agentes autónomos móviles, siendo su objetivo prioritario la transparencia en su uso, concretada en dos aspectos. El primero, la sencillez, es decir, que la carga de trabajo necesaria para su aprendizaje sea mínima. El segundo, la portabilidad, esto es, que permita el uso del mayor número de plataformas de desarrollo y lenguajes de programación.

El desarrollo de la arquitectura *DLA* viene motivado por el hecho de que ninguna de las arquitecturas de control híbridas existentes satisface las necesidades de sencillez y portabilidad necesarias para permitir una filosofía de trabajo basada en la libre elección por parte del diseñador. Las principales características de la arquitectura *DLA* son: sencillez, escalabilidad, extensibilidad a nivel *hardware* y *software*, portabilidad a nivel *hardware* y *software*, flexibilidad, depuración y eficiencia. Gracias a estas características es posible implementar distintos tipos de sistemas cooperativos para agentes autónomos móviles. Si bien en [Pér06] la estructura de la arquitectura fue la destinada a la implementación de una infraestructura básica de navegación, en esta Tesis la estructura tiene un objetivo bien distinto. Usando el estilo de la arquitectura *DLA* se ha implementado una estructura eficiente de navegación destinada a la exploración completa de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil.

Las posibilidades que el estilo de la arquitectura *DLA* proporciona son muy amplias. Por las características del sistema diseñado, el entorno de trabajo y la plataforma robótica empleada, únicamente se han utilizado aquéllas que permiten un desarrollo eficiente. A continuación se realiza una revisión del estilo de la arquitectura *DLA*, centrando el análisis en las facilidades que se han considerado.

2.1 El estilo de la arquitectura *DLA*

Los mecanismos de interacción entre módulos que definen el estilo de la arquitectura *DLA* están diseñados en base a cuatro especificaciones de partida:

- No bloqueantes. El sistema estará descompuesto en una serie de módulos que operan de forma asíncrona, operando a la velocidad que necesiten e intercambiando datos con el resto cuando lo requieran. Por lo tanto, deben ser no bloqueantes para garantizar que ningún módulo se queda a la espera de recibir la respuesta solicitada por el resto de módulos.
- Tipos de datos. Se debe permitir el intercambio de toda clase de información sin imponer restricciones en cuanto a algún tipo predefinido de datos.
- Portabilidad. Para garantizar la posibilidad de que los mecanismos de interacción trabajen con módulos heterogéneos desarrollados con diferentes lenguajes y sobre diversas plataformas.
- Distribuidos. Posibilitando la distribución de los módulos entre varias plataformas.

Teniendo en mente estas premisas, la alternativa más eficiente la constituye la interacción entre procesos [SC96]. Consiste en implementar los distintos módulos del sistema de forma que operen independiente y concurrentemente, intercambiando entre sí los datos necesarios para la correcta operación de cada uno de ellos. Aunque es más complejo de implementar, el esquema apropiado es el que emplea una interconexión indirecta entre los módulos mediante un agente central, ya que así la modificación de cualquiera de ellos no implica la modificación del resto. En cuanto a los mecanismos de intercambio de datos, dependiendo del esquema de funcionamiento, se emplean *sockets* o memoria compartida.

El estilo de la arquitectura *DLA* presenta tres esquemas diferentes, resultado de la evolución realizada para mejorar sus prestaciones:

- Esquema distribuido básico. Usado cuando los módulos del sistema se distribuyen entre varias máquinas.
- Esquema distribuido síncrono. Usado cuando los módulos del sistema se distribuyen entre varias máquinas.
- Esquema local síncrono. Usado cuando todos los módulos del sistema residen en la misma máquina.

El esquema distribuido básico es el menos eficiente de los tres. Está basado en la arquitectura cliente-servidor propuesta por Dulimarta [DJ96]. En este esquema se incorporan varios servidores específicos para la gestión del intercambio de información entre módulos en lugar de emplear únicamente uno. Existe un servidor central *DLAServer* que se encarga de gestionar

de manera transparente la creación y destrucción de estos servidores específicos. Esta aproximación emplea un sistema de memoria compartida distribuida basada en los *sockets*. Las zonas de memoria se denominan conexiones, gracias a las cuales los módulos comparten los datos entre sí. Cada servidor está destinado a atender a una de las conexiones del sistema, gestionando cualquier posible acceso de los distintos módulos. La arquitectura se ha diseñado para optimizar de modo automático el uso de los recursos disponibles, minimizando la memoria empleada y la carga del procesador.

El esquema distribuido síncrono surge ante la necesidad de añadir la capacidad de sincronización al esquema distribuido básico. A pesar de que el estilo de la arquitectura es asíncrono, se incluyen mecanismos de sincronización entre módulos para optimizar el funcionamiento del sistema bajo determinadas circunstancias. El mecanismo de sincronización es abstracto. Está basado en los buzones (*mailboxes*), cuya implementación interna usa semáforos. Un buzón es un contenedor de datos que puede almacenar cualquier valor simple de tipo entero. Cualquier módulo puede bloquearse en un buzón hasta que lleguen datos nuevos al mismo, no consumiendo tiempo de ejecución ni generando tráfico mientras está bloqueado. Cuando cualquier módulo escribe datos en un buzón, todos los módulos que estaban bloqueados sobre dicho buzón se activan automáticamente, continuando con el procesamiento de datos para el que fueron diseñados. Al igual que el esquema distribuido básico, el esquema distribuido síncrono usa los *sockets* como mecanismo de intercambio de datos. Sin embargo, es mucho más eficiente, aun en el caso de que no sea necesaria la sincronización de los módulos entre sí. Como el tamaño de los datos a intercambiar mediante el uso de los buzones es muy pequeño (un entero), únicamente se emplea un servidor para gestionarlos todos, al contrario de lo que ocurre con las conexiones.

Por último, el esquema local síncrono es el apropiado cuando los módulos que intercambian datos entre sí residen en la misma máquina. La alternativa más eficiente la constituye el uso de memoria compartida local. En este caso, no es necesaria la utilización del servidor central *DLAServer*, sino que todos los módulos se conectan directamente a través de la memoria compartida. Este mecanismo incluye tanto a las conexiones como a los buzones, por lo que los mecanismos de sincronización introducidos por el esquema distribuido síncrono siguen estando vigentes. El desarrollo de este esquema facilita su uso por parte de un usuario de forma transparente, gestionando el sistema la complejidad inherente de la memoria compartida.

En la presente Tesis todas las pruebas se han llevado a cabo empleando el esquema local síncrono, haciendo uso de los buzones siempre que ha sido posible. Como todos los módulos del sistema son ejecutados en la misma máquina, esta es la opción más eficiente. La máquina no se veía sobrecargada en exceso siguiendo este esquema, quizás, por el uso de los buzones, lo cual reduce significativamente el tiempo en el que los módulos están activos y, con ello, la carga del procesador. Por este motivo, no ha sido necesario el uso del esquema distribuido síncrono, donde los módulos residirían en varias máquinas.

Al usar el esquema local síncrono, el servidor central *DLAServer* no es necesario. Sin embargo, sí se ha hecho uso del administrador remoto *DLAAdmin* para monitorizar y analizar

en tiempo real el flujo de intercambio de datos existente en el sistema y así facilitar su depuración. En cuanto al lenguaje de programación, se ha empleado *C* para *Linux*, aunque la potencia y capacidad de la arquitectura *DLA* es muy grande, permitiendo el desarrollo de aplicaciones en *JAVA* y *Matlab*.

3 Estructura de la arquitectura DLA para el sistema propuesto

Usando el estilo de la arquitectura *DLA* se ha diseñado e implementado un sistema eficiente de navegación destinado a la exploración completa de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil. El sistema desarrollado es híbrido, puesto que la combinación de comportamientos reactivos y deliberados es actualmente la mejor alternativa al trabajar con agentes autónomos móviles. Consta de varios módulos agrupados en capas, donde se distingue la capa reactiva, la capa deliberada y la capa intermedia [OC03] . Como ya sabemos, un módulo es una unidad funcional de menor complejidad que el sistema completo, que tiene asignada una determinada tarea. Así mismo, una capa es una entidad abstracta que agrupa a todos los módulos que contribuyen a una misma tarea u objetivo dentro del sistema.

A continuación se describe la descomposición del sistema en módulos y capas. Algunos de los módulos, por su importancia, son analizados por separado en capítulos sucesivos. El resto de los módulos son usados para soportar la funcionalidad deseada del sistema, siendo presentados después del esquema general.

3.1 Esquema general

El sistema consta de distintas capas y módulos, que se muestran en el esquema general de la Figura 2.1. Aunque en la Figura 2.1 se muestre la interacción directa entre los módulos, debemos recordar que dicha comunicación se realiza mediante el esquema local síncrono del estilo de la arquitectura *DLA*. Las capas y módulos del sistema de exploración completa son los siguientes:

- *Capa InterfazRobot.* La misión de esta capa es la de servir de interfaz entre el robot y el resto del sistema. Consta de un único módulo, *ControlRobot.* Se encarga de dos tareas. Una primera es recibir la información dada por los sensores sonar y la posición y orientación que la odometría del agente proporciona. La segunda consiste en mandar los comandos de movimiento que se deben aplicar a los motores del robot para su movimiento por el entorno. Cada comando de movimiento es doble. Incluye la velocidad de rotación y la velocidad de traslación que el robot debe tomar en cada instante de tiempo. Este módulo permite también la combinación de los comandos de movimiento que el robot intenta aplicar y las órdenes que un humano envía a través de un joystick, para así alcanzar un control asistido.

  Los comandos de movimiento que son aplicados al agente incluyen de forma cooperativa aquéllos que cada uno de los tres módulos de la capa *Comportamientos* intenta desarrollar de forma aislada, dirigiendo el robot hacia el objetivo marcado por la capa *Planificación.*

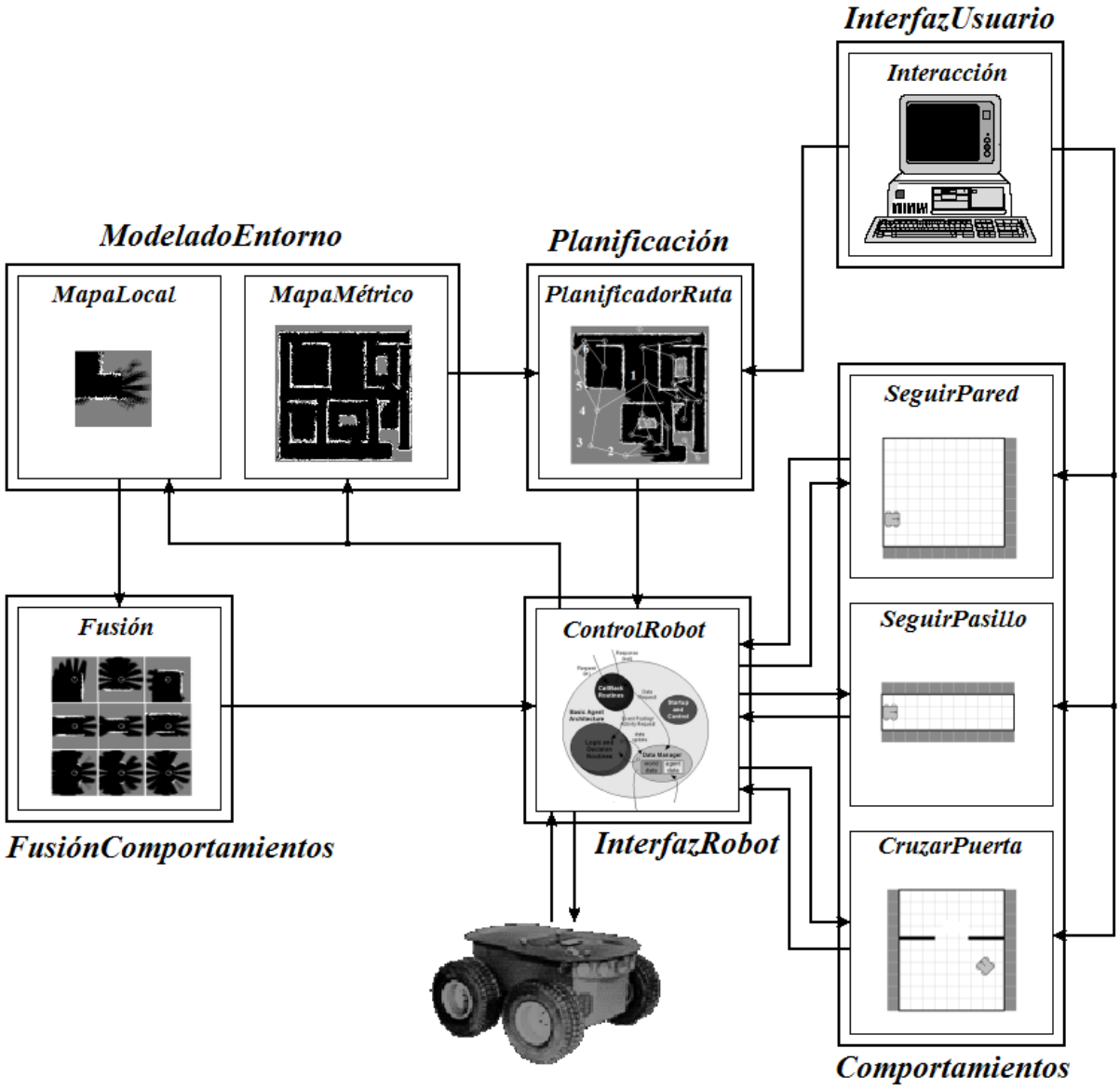

Figura 2.1: Esquema general del sistema de exploración.

La cooperación entre los comportamientos es la señalada en cada momento por la capa *FusiónComportamientos*.

- *Capa Comportamientos.* El objetivo de esta capa es el desarrollo de los comportamientos de navegación reactiva que implementan la Navegación Local de la Jerarquía de Navegación [FM00]. Los tres comportamientos implementados son puramente reactivos. En respuesta a una determinada lectura de los sensores sonar, en cada instante, su salida es la velocidad de rotación y la velocidad de traslación que el robot debería aplicar para implementar la funcionalidad deseada. En esta Tesis se han seguido dos esquemas diferentes para cada uno de los tres comportamientos, uno analítico y otro que tiene su origen en el razonamiento basado en casos. Esta capa consta de tres módulos, uno por comportamiento diseñado:

  - *SeguirPared.* Este módulo implementa un comportamiento reactivo que trata de desplazar el robot de manera paralela a la pared que se encuentra a su derecha, siguiendo su contorno a una distancia prefijada. En el diseño se incluyen mecanismos de evitación de obstáculos.
  - *SeguirPasillo.* Este segundo módulo desarrolla otro comportamiento reactivo, mediante el cual el agente navega a lo largo de un pasillo por su centro. También se incluyen mecanismos de evitación de obstáculos.
  - *CruzarPuerta.* Por último, el tercer módulo de la capa *Comportamientos* implementa de manera reactiva el cruce de una puerta por parte del agente, para que pueda entrar o salir de una habitación. Al igual que para las dos módulos anteriores, la evitación de obstáculos ha sido considerada.

- *Capa ModeladoEntorno.* Al estar basado nuestro sistema en una filosofía híbrida, existe una capa deliberada encargada de la planificación, la capa *Planificación.* Para poder planificar es necesaria una representación del entorno. Esta es la misión de la capa *ModeladoEntorno*, proporcionar una representación lo más fiable posible del entorno por el que se desplaza el agente, y así poder desarrollar comportamientos deliberados. También es responsable de generar y mantener un modelo de entorno adecuado para servir como punto de partida al sistema de ayuda a la cooperación de comportamientos. Consta de dos módulos:

  - *MapaMétrico.* A partir de la lectura de los sensores sonar, implementa el algoritmo de construcción de un mapa métrico probabilístico [Elf87] del entorno en el que se desenvuelve el robot. Este modelo del entorno es el que será usado por la capa *Planificación* para desarrollar la deliberación del sistema. El mapa métrico tiene que ser continuamente actualizado para que la planificación de la capa *Planificación* sea adecuada.
  - *MapaLocal.* Al igual que el módulo *MapaMétrico*, implementa el algoritmo de construcción de un mapa métrico probabilístico. La diferencia con aquél es que en este caso la representación es local, incluyendo únicamente el entorno más cercano del robot. Este modelo es usado por la capa *FusiónComportamientos.* Aunque ésta podría entenderse como una capa que implementa comportamientos deliberados, quizás es más adecuada la interpretación que la considera como una entidad que ayuda a la fusión de los comandos de los comportamientos reactivos [OC03]. Al igual que en el caso del mapa métrico, el mapa local también tiene que ser continuamente actualizado para mantener de un modo fiel la representación del entorno cercano al robot.

- *Capa FusiónComportamientos.* El objetivo de esta capa es proporcionar los factores de ponderación que la capa *InterfazRobot* tiene que aplicar a los comandos de movimiento de cada uno de los tres comportamientos reactivos implementados, de manera que la navegación del robot sea eficiente y se adapte en todo momento a la configuración del entorno. Se implementa a través del módulo *Fusión*, cuya entrada es el mapa local construido por

el módulo *MapaLocal*. A partir de esta entrada produce una salida que consiste en los pesos a aplicar a los tres comportamientos. Aceptamos la interpretación de esta capa como módulo de fusión de comandos. En el diseño se ha tenido en cuenta que no es conveniente integrar en el mismo módulo el algoritmo de fusión con la fusión efectiva de los comandos de movimiento [OC03]. Por este motivo, la fusión de los comandos se realiza en el módulo *ControlRobot* y la obtención de los factores de ponderación se realiza por separado en el módulo *Fusión*.

- *Capa Planificación*. Es la encargada del desarrollo de los comportamientos deliberados del sistema que soportan el nivel de Navegación por Inspección de la Jerarquía de Navegación. Consta de un único módulo, *PlanificadorRuta*, que toma como entrada el mapa métrico construido por el módulo *MapaMétrico* de la capa *ModeladoEntorno*. Esta representación es usada para construir una estructura jerárquica de la que se extrae un mapa topológico con las regiones del entorno y las conexiones entre ellas. A partir del mapa topológico se planifica, de forma eficiente, la ruta de exploración completa mediante un algoritmo específicamente diseñado a tal efecto.

- *Capa InterfazUsuario*. Esta capa sirve de interfaz entre el usuario de la aplicación y el robot. Se implementa por medio del módulo *Interacción*. Tiene dos objetivos principales. En primer lugar, poder iniciar y parar el proceso de exploración completa en cualquier momento. En segundo lugar, servir de interfaz visual para el usuario y así poder observar en todo instante el estado del robot y de la exploración.

El esquema general de la Figura 2.1 se corresponde con el de un sistema híbrido, donde aparecen las tres capas típicas de estos sistemas [OC03]. Así, la capa *Planificación* se corresponde con el nivel deliberado. La capa *Comportamientos* con el nivel reactivo, consistente en varios comportamientos separados. Y la capa *FusiónComportamientos* con el nivel intermedio que actúa de interfaz entre el nivel reactivo y el nivel deliberado. En el diseño del sistema se ha tenido en cuenta el mantener la independencia tanto del nivel deliberado como del reactivo. Así, si se desea cambiar alguno de ellos para que el sistema exhiba una funcionalidad distinta, se puede hacer sin tener que cambiar el otro.

Existen tres capas fundamentales dentro del sistema, la que implementa los comportamientos reactivos, la que implementa la deliberación y la que actúa como elemento de ayuda a la cooperación o fusión de los comportamientos. Cada una de ellas incluye aportaciones fundamentales de la presente Tesis. Por este motivo, se analizan por separado en sucesivos capítulos. Así, el Capítulo 3 está dedicado a la capa *Comportamientos* o nivel reactivo, el Capítulo 4 a la capa *FusiónComportamientos* o nivel intermedio, y el Capítulo 5 a la capa *Planificación* o nivel deliberado,. Si bien el resto de las capas quizás no tenga esta importancia, también son necesarias para la correcta operación del sistema de exploración. Son analizadas en este capítulo. Concretamente, la capa *InterfazRobot* se introduce en la Sección 3.2, la capa *ModeladoEntorno* se presenta en la Sección 3.3 y la capa *InterfazUsuario* se estudia en la Sección 3.4.

3.2 La capa *InterfazRobot*

Esta capa consta de un único módulo, *ControlRobot*, que implementa la interfaz entre el robot y el sistema. El objetivo de este módulo es doble:

- Capturar del robot la lectura de los sensores sonar y la información odométrica relativa a su posición y orientación, poniendo estos datos a disposición de todos los módulos.
- Enviar los comandos de movimiento al robot. Consisten en la velocidad de rotación y la velocidad de traslación a las que el robot debe desplazarse en cada instante.

Los comandos de movimiento incluyen una combinación lineal de los comandos de movimiento que cada uno de los comportamientos de navegación reactiva de la capa *Comportamientos* proporciona como entrada del módulo *ControlRobot*. La combinación se hace en función de los pesos calculados en el módulo *Fusión* de la capa *FusiónComportamientos*. El agente se dirige al siguiente punto de destino de la ruta de exploración completa proporcionado por el módulo *PlanificadorRuta* de la capa *Planificación*.

Un aspecto importante que debemos comentar es el relativo al sistema de odometría del robot. La odometría se basa en ecuaciones simples que se pueden implementar fácilmente y que utilizan datos de *encoders* situados en las ruedas del robot, contando el número de vueltas que dan. Así, se puede conocer la posición del robot en el entorno con respecto a su punto de partida. A pesar de que es una solución muy barata, tiene el problema de la pérdida de exactitud con la distancia recorrida, principalmente por el deslizamiento (*slippage*) y la acumulación de los errores. Por lo tanto, se necesita algún método de corrección para aumentar la exactitud en la estimación de la posición del robot. El método usado es un filtro de Kalman [Kal60]. Esta técnica se emplea para resolver los problemas de la odometría de un robot, tema de investigación muy extenso y activo. En esta Tesis nos basamos en el trabajo implementado en [Pér06], que utiliza un procedimiento sencillo y claramente mejorable. El sistema está preparado para incorporar métodos más complejos y precisos, mediante los algoritmos de localización que se desee.

3.3 La capa *ModeladoEntorno*

Tal y como se ha comentado al describir el esquema general del sistema, la capa *ModeladoEntorno* es la encargada de generar y mantener la representación del entorno necesaria para soportar los comportamientos deliberados que se implementan en la capa *Planificación*. También es responsable de generar y mantener un modelo de entorno adecuado para servir como punto de partida al sistema de ayuda a la fusión o cooperación de comportamientos (capa *FusiónComportamientos*).

Existen diversas alternativas a la hora de abordar la representación de un entorno. La aproximación más empleada es la de los mapas métricos probabilísticos [Elf87, TBB+98], que establecen una correspondencia geométrica absoluta entre la representación y el entorno. Estos mapas son los empleados en la presente Tesis. Se basan en dividir el entorno en una rejilla o *grid*

de celdas de un tamaño suficientemente pequeño para obtener una representación del entorno con la resolución deseada, asignándole a cada celda una probabilidad de ocupación. Cada celda tiene una correspondencia directa con una determinada posición del espacio.

Para la obtención del mapa métrico probabilístico habrá que integrar en el tiempo y en el espacio las medidas obtenidas de uno o varios sensores, convirtiéndolas en probabilidades de ocupación. Los sensores más habituales a la hora de construir un mapa probabilístico son los sonar [LC94], aunque es posible emplear otro tipo de sensores como el láser [Jon93], los sensores de infrarrojos [GT94] o sensores táctiles [BEF96].

A pesar de que los mapas métricos probabilísticos son los más comunes, también existen otros métodos para la representación geométrica de un entorno. Entre estos nos encontramos con métodos basados en el espacio de configuración [LP81], en los diagramas de Voronoi [Mil85], modelos basados en segmentos [Cro85] o vértices [Koc85], e incluso modelos basados en aproximaciones poligonales [CL85]. Todos estos métodos comparten con los mapas probabilísticos la ventaja de la correspondencia entre el modelo del entorno y la geometría.

A la hora de diseñar el algoritmo de construcción de un mapa métrico probabilístico existe un compromiso que hay que considerar. Si se emplea un mapa de muchas celdas se podrá abarcar una mayor extensión de terreno. Sin embargo, a medida que aumenta el tamaño de dicho mapa también lo hará el tiempo de proceso requerido para su construcción. Además, hay que tener en cuenta el tamaño de celda empleado. Un tamaño de celda pequeño dará lugar a una representación del entorno con mayor resolución, pero al mismo tiempo el área del entorno abarcada será menor. Por lo tanto, antes de construir el mapa probabilístico hay que decidir el tamaño de celda y el tamaño del mapa.

El proceso de construcción del mapa métrico probabilístico debe tener en cuenta dos aspectos, la probabilidad de que una celda esté ocupada en función de la lectura de los sensores sonar y la integración temporal de las sucesivas lecturas. Existen diferentes aproximaciones para modelar los sensores sonar, todas ellas basadas en una u otra función de probabilidad. Así, nos encontramos con el modelo de Oriolo [OVU95], el modelo de Pagac [PNDW98] o el modelo de Matía [MJ98]. En cuanto al tema de la integración sensorial, la regla de Bayes es la más utilizada [TBB$^+$98]. Otra de las posibilidades de integración se presenta en [BK91b], donde el autor emplea el método de las rejillas de histograma (*Histogramic In-Motion Mapping (HIMM).*

En [Urd99] se efectúa un análisis de los distintos modelos de construcción de mapas métricos probabilísticos, comparando sus características. Así mismo, se propone un nuevo modelo más sencillo y que implica un cálculo computacional menor, basado en realizar medidas frecuentes en vez de en complejos modelos de actualización. Debido a las buenas prestaciones de este modelo, será el empleado en esta Tesis para construir los dos mapas probabilísticos implementados en los módulos *MapaMétrico* y *MapaLocal*. La probabilidad de ocupación en el instante t, $P^t_{ocup}(\theta, \rho)$, de una celda dentro del campo de visión de los sensores sonar se modela mediante una distribución de probabilidad muy simple que tiene en cuenta la probabilidad de ocupación en el instante previo, $P^{t-1}_{ocup}(\theta, \rho)$. Viene dada por la expresión:

$$P_{ocup}{}^{t}(\theta,\rho) = P_{ocup}^{t-1}(\theta,\rho) + \begin{cases} -P_{espacio} & \text{si } 0 \leq \rho < d, 0 \leq |\theta| \leq \beta/2 \\ P_{ocupado} & \text{si } \rho = d, 0 \leq |\theta| \leq \beta/2 \\ 0 & \text{en otro caso} \end{cases} \quad (2.1)$$

donde ρ es la distancia desde una determinada posición hasta el robot, θ es el ángulo entre la misma posición y el eje del lóbulo principal del sonar, d la distancia al obstáculo detectado y β la anchura del lóbulo principal. $P_{espacio}$ y $P_{ocupado}$ son las constantes correspondientes a una celda libre y una ocupada, respectivamente. El modelo parte de la suposición de que todas las celdas hasta la distancia d están libres y que las lecturas sonar se integran a lo largo del tiempo.

El algoritmo de construcción del mapa probabilístico parte de un mapa inicial completamente inexplorado donde la probabilidad de ocupación de todas sus celdas es del 50%. Los valores de $P_{espacio}$ y $P_{ocupado}$ se han fijado experimentalmente al 10% y 20%, respectivamente. Igualmente, se considera una anchura del lóbulo principal del sensor sonar, β, de 10º. A medida que el robot navega se actualiza la representación probabilística, para así mantener en todo momento un modelo actualizado del entorno. En función de la lectura dada por los sensores sonar, las celdas del mapa probabilístico dentro del campo de visión de estos sensores ven actualizada su probabilidad de ocupación. La actualización consiste en restar una probabilidad $P_{espacio}$ o sumar una probabilidad $P_{ocupado}$. En el primer caso se realiza para las celdas ocupadas por el robot y para aquéllas que se encuentran entre el sensor sonar y el obstáculo detectado. En el segundo, se modifican las celdas que se encuentran a la distancia del obstáculo detectado. Obviamente, la probabilidad de ocupación no podrá superar en ningún caso el 100% ni podrá ser inferior a 0. Para representar un mapa probabilístico se le asocia al nivel de probabilidad de ocupación de cada celda un nivel de la escala de grises de 0 a 255. Así, las zonas libres se muestran en negro, los obstáculos en blanco y las áreas no exploradas en gris.

La capa *ModeladoEntorno* consta de dos módulos, *MapaMétrico* y *MapaLocal*. Ambos módulos implementan el algoritmo de construcción de mapas probabilísticos presentado en esta sección. La principal diferencia entre ellos radica en el tamaño de la representación. A continuación se presentan las peculiaridades de ambos módulos dentro del esquema general de nuestro sistema.

3.3.1 El módulo *MapaMétrico*

Este módulo implementa el algoritmo de construcción del mapa probabilístico presentado en la Sección 3.3. Gracias al modelo de entorno que construye, el módulo *PlanificadorRuta* de la capa *Planificación* puede planificar rutas eficientes para la exploración completa de un entorno total o parcialmente desconocido. Normalmente, a lo largo de esta Tesis se emplean tamaños de celda de $Tam\_Celda = 40mm$, que proporcionan una resolución bastante buena para conseguir nuestros objetivos. Del mismo modo, los mapas empleados suelen tener un tamaño $Tam\_MapaMetrico = 256x256celdas$. Un mapa de 256x256 celdas de $40mm$ abarca una extensión de $10x10m^2$, un área bastante grande del entorno que se encuentra representada en el modelo. Si el tamaño del

entorno fuera mayor que el área abarcada por el mapa, este módulo mantiene en todo momento la representación de la zona en la que se encuentre el agente para que éste siempre esté dentro del mapa. Durante el desarrollo del trabajo también se emplean en algunos casos mapas de 512x512 celdas e incluso de 1024x1024 celdas, como en el Capítulo 5.

Este módulo también es el encargado de activar el proceso de planificación de rutas del módulo *PlanificadorRuta* de la capa *Planificación*, mediante el uso de los buzones del estilo de la arquitectura *DLA*. Para ello, se tiene en cuenta el número de celdas del mapa métrico que varía. Cuando este valor supera un umbral $U_{metrico}$, se considera que el entorno ha cambiado sustancialmente y que hay que planificar una nueva ruta. En este caso, el módulo *PlanificadorRuta* calcularía una nueva ruta de exploración completa para así adaptarse a los últimos cambios que la representación del entorno ha sufrido. Así, se reduce la carga del procesador, puesto que la planificación solamente se lleva a cabo cuando el modelo de entorno ha variado. En esta Tesis, normalmente se emplea un umbral $U_{metrico} = 200$.

En la implementación de este módulo se ha seguido un esquema de almacenamiento circular. Cuando la memoria asignada para la generación y actualización del mapa métrico se llene al máximo, el mapa continuará su proceso de actualización, descartando la región más antigua almacenada. De este modo, se asegura que la zona del entorno representada en el modelo es en todo momento la más recientemente explorada, es decir, aquélla en la que se encuentra el robot.

En la Figura 2.2 se presentan ejemplos de mapas probabilísticos construidos con el modelo de la Sección 3.3. El tamaño de cela empleado es $Tam\_Celda = 40mm$, constando los mapas de 256x256 celdas. La Figura 2.2a muestra la planta de un entorno simulado de $10x10m^2$. El simulador empleado es el correspondiente a la plataforma robótica *Nomad 200* de *Nomadics*. Durante la exploración del entorno se ha guiado al robot manualmente por medio de un teclado. La Figura 2.2b presenta la representación probabilística durante un estado de esta exploración. La representación incluye la mayoría del entorno, excepto la habitación noreste, una pequeña zona en el suroeste, la habitación central al sur y el pasillo este. Podemos observar en el Figura 2.2b que las zonas del entorno representadas en el mapa métrico se corresponden de manera fidedigna con el entorno de la Figura 2.2a.

La Figura 2.2 también muestra un entorno real (Figura 2.2c), consistente en una habitación y un pasillo. En dicho entorno, se han generado y actualizado distintos mapas probabilísticos durante la exploración manual empleando la plataforma robótica real *Pioneer P2AT*. En este caso se aprecian dos mapas probabilísticos en dos instantes diferentes de la exploración del escenario de la Figura 2.2c. El primero (Figura 2.2d) presenta la exploración casi íntegra de la habitación, la zona central del pasillo y la zona norte sin llegar a incluir en el modelo la pared. En un instante posterior la representación de la Figura 2.2d se ve actualizada, incrementándose el conocimiento que el sistema tiene del entorno bajo estudio. De este modo, surge el modelo de la Figura 2.2e, donde ya sí se ha detectado la pared norte del pasillo. Por lo tanto, la representación en este último caso es más completa.

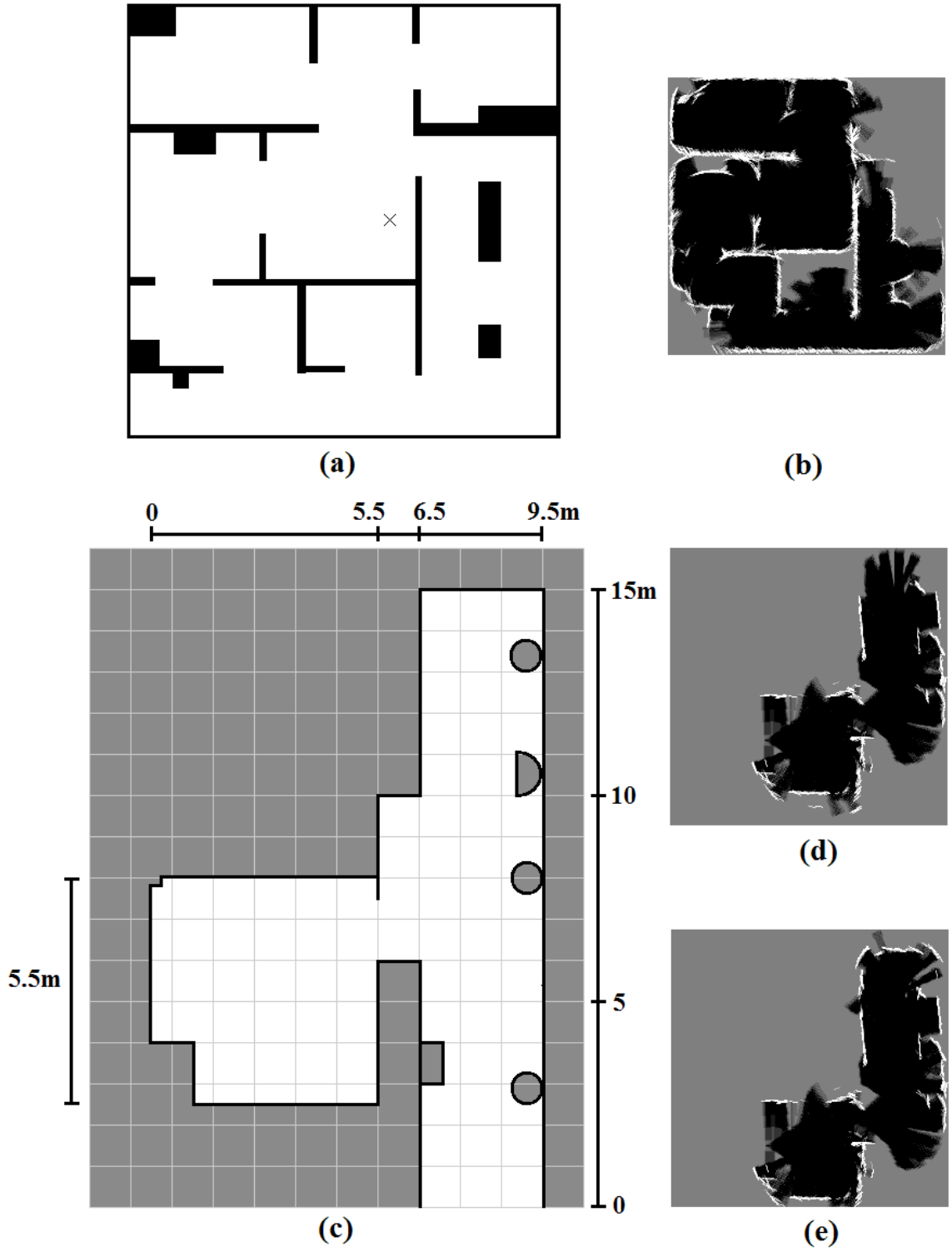

Figura 2.2: Mapas métricos: (a) entorno simulado; (b) mapa construido durante la exploración del entorno en (a); (c) entorno real; (d) mapa construido en un instante de la exploración del entorno en (c); (e) mapa construido en otro instante de la exploración del entorno en (c).

3.3.2 El módulo *MapaLocal*

Al igual que el módulo *MapaMétrico*, este módulo también implementa el algoritmo de construcción del mapa probabilístico de la Sección 3.3. Sin embargo, en este caso el modelo no

se utiliza como punto de partida para la planificación de rutas de exploración completa. La representación que construye el módulo *MapaLocal* es la entrada del módulo *Fusión*, donde se obtienen los factores de ponderación que la capa *InterfazRobot* (a través del módulo *ControlRobot*), tiene que aplicar a los comandos de movimiento de los tres comportamientos reactivos de la capa *Comportamientos*. Como en el caso de la representación del módulo *MapaMétrico*, el tamaño de celda habitualmente utilizado es $Tam\_Celda = 40mm$. En cuanto al tamaño del mapa, hay una diferencia sustancial. Como la cooperación entre los comportamientos se debe hacer para adaptarse al entorno más próximo al robot, con mantener una representación local es más que suficiente. De ahí que el mapa local tenga unas dimensiones bastante menores a las de su homólogo el mapa métrico. A lo largo de la presente Tesis se trabaja con mapas de tamaño $Tam\_MapaLocal = 113x113celdas$. Considerando que tanto el robot simulado *Nomad 200* como la plataforma robótica real *Pioneer P2AT* tienen unas dimensiones de unos $50x50cm^2$, 113 celdas se corresponden con un rango de $2m$ en torno al robot. Además, el robot siempre estará centrado en la representación. Por lo tanto, el mapa local es un modelo del entorno más próximo al robot a una distancia máxima de $2m$.

Al contrario que en el caso del módulo *MapaMétrico*, este módulo no activa ningún otro. Se podría haber incluido un umbral para considerar que el mapa local ha cambiado significativamente. Sin embargo, los comportamientos reactivos con los que vamos a trabajar requieren que nos adaptemos a la configuración del entorno en cualquier momento. Por ello, se ha preferido variar la adaptación a dicho entorno en todo momento en función de los factores de ponderación que el módulo *Fusión* extrae del mapa local, sin esperar a que dicho mapa varíe por encima de un umbral.

La Figura 2.3 presenta mapas locales construidos con el modelo de la Sección 3.3. La Figura 2.3b muestra el mapa local adquirido cuando el robot llega a la posición marcada con una cruz en el entorno simulado de la Figura 2.2a, apuntando el robot hacia el norte. Igualmente, las Figuras 2.3c y d constituyen el mapa local adquirido por la plataforma real *Pioneer P2AT* en las posiciones A y B, respectivamente, del entorno real de la Figura 2.3a. Observamos claramente que los mapas locales únicamente modelan de una forma fidedigna una pequeña zona del entorno completo, aquélla más cercana al robot.

3.4 La capa *InterfazUsuario*

Esta capa consta de un único módulo, *Interacción*, que implementa la interfaz entre el usuario del sistema y la plataforma robótica, tanto simulada como real. Tal y como se ha comentado antes, dos son los objetivos fundamentales que este módulo persigue:

- Dotar al usuario de un mecanismo para iniciar y parar el proceso de exploración completa en cualquier momento.
- Servir de interfaz visual para mostrar en todo instante el estado del robot y de la exploración del entorno de trabajo.

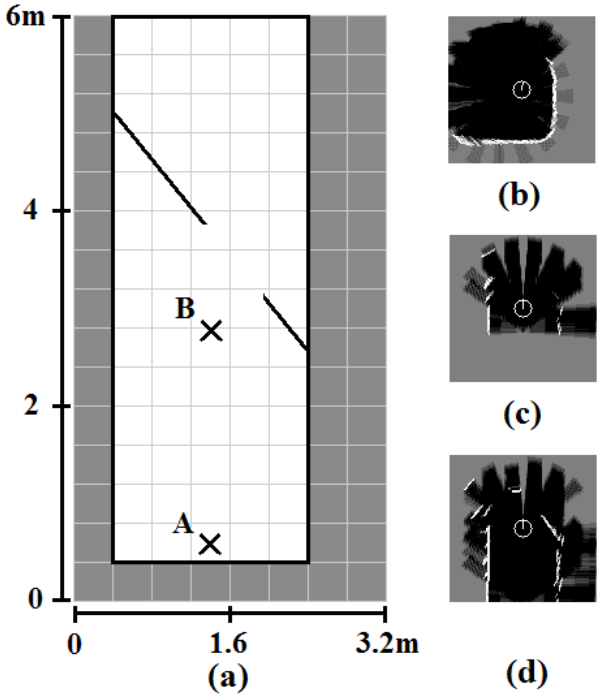

Figura 2.3: Mapas locales: (a) entorno real; (b) capturado en el entorno de la Figura 2.2a; (b) capturado en la posición A del entorno en (a); (c) capturado en la posición B del entorno en (a).

El módulo *Interacción* se ha desarrollado utilizando la biblioteca *GTK* para *Linux*, ampliamente utilizada hoy día para desarrollar aplicaciones *GUI* (*Graphical User Interface*) para *Linux*. Al tratarse de un módulo visual, la carga computacional que añade al sistema es considerable. Por este motivo, se ha tratado de incluir la máxima cantidad de información y de prestaciones en el módulo, pero sin sobrecargarlo en exceso para evitar la ralentización en la ejecución del sistema completo. No olvidemos que en la presente Tesis se emplea el esquema local síncrono del estilo de la arquitectura *DLA* con los 9 módulos del esquema general de la Figura 2.1 ejecutándose en la misma máquina. Este módulo toma como entradas el mapa construido por el módulo *MapaMétrico*, el mapa local del módulo *MapaLocal*, la ruta planificada por el módulo *PlanificadorRuta* y la posición y orientación del robot proporcionada por el módulo *ControlRobot*. La salida es un comando que se aplica al resto de los módulos para indicar si se debe iniciar/continuar la exploración completa o no.

El aspecto de la interfaz de usuario del sistema se muestra en la Figura 2.4. Los distintos controles de la interfaz y su funcionalidad son los siguientes:

- Mapa métrico. Esta zona de la interfaz sirve para mostrar en todo momento, por medio de una imagen, el mapa métrico actualizado que el módulo *MapaMétrico* construye incrementalmente. Así mismo, también puede mostrar información relevante sobre el sistema:
  - Robot. Mediante un círculo blanco en su posición actual y una línea blanca para indicar su orientación en el entorno de trabajo.

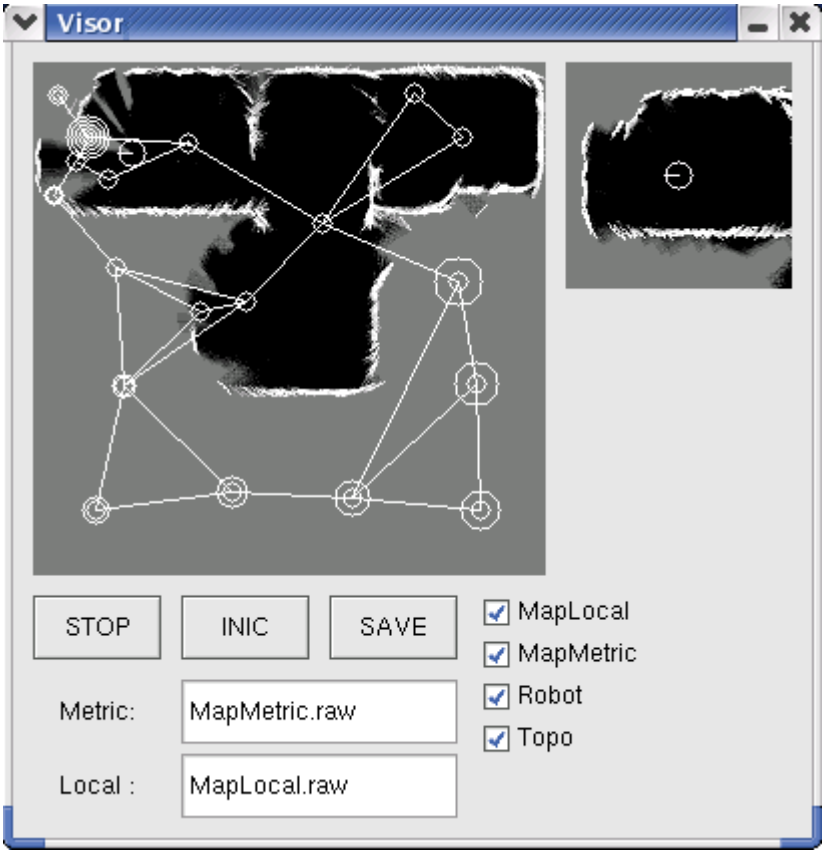

Figura 2.4: Interfaz de usuario del módulo *Interacción*.

- Mapa topológico, superpuesto al mapa métrico para facilitar su lectura. Es el modelo de entorno construido en el módulo *PlanificadorRuta* y que dicho módulo emplea para obtener la ruta de exploración completa. Se representa mediante un grafo. Los nodos de este grafo vienen identificados con un círculo blanco y las conexiones entre ellos con líneas también de color blanco.
- Ruta planificada. El módulo *PlanificadorRuta* obtiene de forma eficiente la ruta de exploración completa que el robot debe seguir. Esta ruta comprende las regiones inexploradas del mapa topológico, representadas por el nodo correspondiente. Así pues, la ruta es un conjunto ordenado de nodos del mapa topológico. La información que se muestra es esta ruta. El nodo inicial de la ruta al que habría que dirigir el robot se representa con varios círculos concéntricos al nodo del mapa topológico. El resto de nodos de la ruta se identifican con un círculo concéntrico al nodo del mapa topológico, de un tamaño proporcional al orden en que ese nodo debe ser visitado por el robot.

- Mapa local. En esta zona se muestra en todo momento, también mediante una imagen, el mapa local actualizado que el módulo *MapaLocal* genera. La representación puede tener superpuesta el robot, dibujado con un círculo blanco en su posición actual y una línea blanca para indicar su orientación en el entorno.

- Botones. Utilizados para posibilitar al usuario la interacción directa con la interfaz y ordenarle así la ejecución de una serie de tareas. Los botones existentes y las acciones ejecutadas cuando se pulsan son:
  - *STOP*. Sirve para detener el robot, anulando la exploración completa.
  - *INIC*. Este botón tiene una doble utilidad. Al arrancar el sistema el botón todavía no se ha pulsado. Con la primera pulsación se inicia la exploración completa, mandando esta indicación al resto de módulos. Si se vuelve a pulsar, se detiene temporalmente al agente, sin anular el comando de exploración completa. Para reanudar la exploración no habrá más que pulsar una tercera vez. La utilidad de esta doble funcionalidad del botón es la depuración del sistema.
  - *SAVE*. Al emplear este botón se salvan en un fichero en formato *RAW* las imágenes correspondientes al mapa métrico y al mapa local. El nombre de los ficheros salvados es el que se introduce en las cajas de texto que están en la zona inmediatamente inferior a los botones. Las imágenes guardadas se corresponden fielmente con la representación que se muestra en la interfaz en el instante de la pulsación.
- Casillas de verificación. Sirven para seleccionar la información que se desea mostrar en las imágenes del mapa métrico y del mapa local. Esta información es diversa:
  - *MapLocal*. Mapa local del entorno más cercano al robot.
  - *MapMetric*. Mapa métrico del entorno.
  - *Robot*. Posición actual y orientación del agente en el entorno de trabajo.
  - *Topo*. Mapa topológico del entorno y la ruta planificada por el módulo *Planificador-Ruta*.
- Cajas de texto. Permiten introducir el nombre con el que las imágenes del mapa métrico y del mapa local se guardarán en un fichero con formato *RAW*. Las imágenes almacenadas serán aquéllas que la interfaz visual muestre en el instante en el que se pulse el botón *SAVE*. Existen dos cajas de texto:
  - *Metric*. Para indicar el nombre con el que guardar la imagen del mapa métrico.
  - *Local*. Para indicar el nombre con el que guardar la imagen del mapa local.

4 Conclusiones

En este capítulo se ha presentado la arquitectura diseñada para el sistema eficiente de exploración completa de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil.

El sistema se descompone en capas y módulos. Los mecanismos de interacción que emplean los módulos para comunicarse son los correspondientes al esquema local síncrono del estilo de

la arquitectura *DLA*, empleando buzones para sincronizar el funcionamiento de todo el sistema. Estos mecanismos son fáciles de usar y transparentes para el usuario.

El esquema general del sistema se corresponde con el de una arquitectura híbrida, soportando el nivel más alto de la Jerarquía de Navegación, la Navegación por Inspección. Consta de seis capas y nueve módulos, dotándolo de una gran modularidad y flexibilidad. De las seis capas, tres son la base de nuestro sistema, siendo analizadas en capítulos sucesivos. De estas tres capas (cinco módulos en total), la primera implementa los comportamientos de navegación reactiva que soportan la Navegación Local de la Jerarquía de Navegación. La segunda implementa los procesos de planificación deliberada que sustentan la Navegación por Inspección. Por último, tenemos el sistema de ayuda a la fusión de los comportamientos reactivos, el nivel intermedio que actúa de interfaz entre el nivel reactivo y el deliberado. El resto de las capas son necesarias para la correcta operación del sistema. Se trata de las capas *InterfazRobot*, *ModeladoEntorno* e *InterfazUsuario*, habiendo sido presentadas en este capítulo.

La capa *InterfazRobot* consta de un único módulo, *ControlRobot*, que implementa la interfaz entre la plataforma robótica y el sistema. Su objetivo es doble. Por un lado, adquirir del robot la lectura de los sensores sonar y la información sobre su posición y orientación. Por otro, enviar al robot la velocidad de rotación y de traslación que debe aplicar a sus motores. En esta capa también se realiza la fusión de los comandos de movimiento de los distintos comportamientos reactivos y se efectúa una corrección sobre los datos odométricos.

La capa *ModeladoEntorno* es la responsable de construir una representación válida para planificar de forma deliberada rutas de exploración completa y de proporcionar un punto de partida al sistema de ayuda a la fusión de comportamientos (módulo *Fusión*). Consta de dos módulos, *MapaMétrico* y *MapaLocal*. Ambos construyen un mapa probabilístico, siendo el segundo una representación local del entorno más cercano al robot. Tanto para un caso como para el otro se han presentado representaciones construidas en entornos reales y simulados que demuestran la validez del método empleado.

Por último, la capa *InterfazUsuario* consiste en un módulo, *Interacción*, que sirve de interfaz entre el usuario y la aplicación. Tiene dos objetivos. En primer lugar, dar al usuario un mecanismo para iniciar, parar o finalizar el proceso de exploración. Y, en segundo lugar, mostrar en todo momento la posición y orientación del robot en el entorno, el mapa métrico, el mapa local, el mapa topológico y la ruta de exploración completa planificada.

Capítulo 3

Comportamientos para Navegación Reactiva

En este capítulo se presentan los tres comportamientos reactivos implementados para navegar por el entorno, habitualmente usados cuando un robot está explorando un área desconocida. Durante el desarrollo se han seguido dos estrategias, una analítica y otra que tiene su origen en el razonamiento basado en casos, esta última para que el sistema se adecúe a las particularidades del entorno mediante el aprendizaje. Ambas se corresponden con el grupo de Navegación Local de la Jerarquía de Navegación. Los tres comportamientos se implementan en la capa *Comportamientos*. Al ser reactivos, los únicos datos de entrada que reciben para calcular la acción que el robot debe ejecutar provienen de la información sensorial proporcionada por el módulo *ControlRobot*.

En primer lugar se efectúa una introducción al tema tratado en la Sección 1, realizando una revisión de los métodos más habituales de navegación reactiva. En la Sección 2 se describen los tres comportamientos implementados en el sistema siguiendo un esquema analítico, Seguir Pared, Seguir Pasillo y Cruzar Puerta. A continuación se pasa al desarrollo de los tres mismos comportamientos, pero en su implementación según una técnica de razonamiento basado en casos (Sección 3). En cuarto lugar se presenta en la Sección 4 un método de aprendizaje asistido, donde el humano y el robot colaboran de forma activa en la navegación. El sistema desarrollado permite esta posibilidad porque en el módulo *ControlRobot* se pueden combinar las órdenes del robot con las órdenes de un humano. Los resultados obtenidos con la plataforma robótica real *Pioneer P2AT*, tanto con los comportamientos como con el método de aprendizaje asistido se muestran en la Sección 5. Finalmente, la Sección 6 presenta las conclusiones extraídas.

1 Introducción

En este capítulo se presenta la manera en la que se han diseñado e implementado los tres comportamientos de navegación reactiva de nuestro sistema, Seguir Pared, Seguir Pasillo y Cruzar Puerta, que suelen ser utilizados cuando el robot está explorando un área desconocida

[MMBB06, RJ05, ZLY06]. Se corresponden con el grupo de comportamientos de Navegación Local de la Jerarquía de Navegación [FM00], puesto que la acción ejecutada por el robot depende únicamente de la información sensorial captada en cada momento. Mediante estos comportamientos reactivos se consigue navegar de forma autónoma entre las zonas no exploradas de la ruta de exploración obtenida mediante la técnica que se presenta en el Capítulo 5. Por lo tanto, y tal y como hemos comentado anteriormente, nuestro sistema incluye tanto la navegación reactiva como la planificación de alto nivel para aprovechar las ventajas de ambos esquemas.

Antes de analizar las dos estrategias seguidas para implementar los comportamientos, se realiza a continuación una revisión de los métodos más comunes de navegación reactiva.

1.1 Navegación reactiva

Los sistemas reactivos son relativamente recientes [Bro86, Kae86, Pay86, Ark89]. Por definición un sistema reactivo asocia la percepción con la acción sin el uso de ninguna representación abstracta ni de la historia temporal [Ark98]. La respuesta que proporciona el sistema es, por tanto, muy rápida, debido a que no lleva a cabo ningún tipo de planificación. Sin embargo, la navegación exclusivamente reactiva suele ser ineficiente y propensa a caer en trampas locales.

Existen dos formas habituales de diseñar la navegación reactiva de un agente autónomo, mediante un modelo analítico o empleando técnicas de Inteligencia Artificial [WL04]. Las primeras técnicas modelan la respuesta reactiva del robot mediante una expresión analítica que suele depender de varios parámetros que caracterizan el entorno, haciéndose difícil el modelado de cualquier posible situación. Entre todas ellos destacamos los campos de potencial [Kha86], el histograma del vector de campo [BK91a], las bandas elásticas [QK93], la ventana dinámica [FBT97] y el diagrama de proximidad [MM00] (Figura 3.1).

Los campos de potencial (*Potential Fields Method*, *PFM*) son una de las aproximaciones más conocidas. Fueron introducidos por Khatib en 1986 [Kha86]. Se basan en la creación de los campos de atracción hacia el objetivo y de repulsión de obstáculos en el espacio de trabajo. Gracias al campo generado se puede llegar desde el lugar en el que se encuentra el robot hacia el destino siguiendo el gradiente descendiente, lo cual evita los obstáculos. A pesar de su sencillez y eficiencia, tiene algunos inconvenientes significativos: i) tiende a oscilar cuando el robot está cerca de los obstáculos, por ejemplo, en corredores estrechos o cerca de una pared; ii) se queda atrapado en mínimos locales; y iii) tiene dificultades para moverse entre obstáculos cercanos, como por ejemplo una puerta.

El histograma del vector de campo, (*Vector Field Histograma*, *VFH*) usa un histograma que depende de la orientación del robot para representar la densidad de los obstáculos [BK91a]. Este histograma es empleado para mover que el robot se mueva en la dirección donde haya menos obstáculos. El método construye dos histogramas, el cartesiano y el polar. El primero es en dos dimensiones, siendo actualizado continuamente. El segundo es unidimensional, proporcionando los denominados valles candidato. El valle más cercano a la dirección del objetivo es seleccionado, girando el robot hasta alinearse con el centro de dicho valle. Este método es un poco más

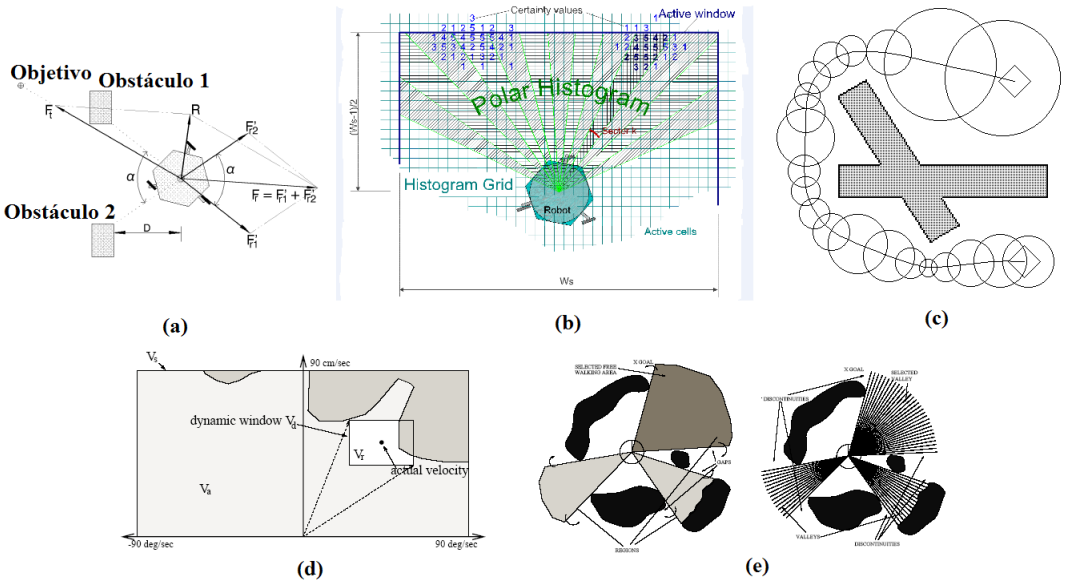

Figura 3.1: Navegación reactiva, modelos analíticos: (a) campos de potencial; (b) histograma del vector de campo; (c) bandas elásticas; (d) ventana dinámica; (e) diagrama de proximidad.

complejo y costoso que el *PFM*, llegando incluso a ser considerado como un método que no es puramente reactivo por algunos autores. Además, posee algunos inconvenientes: i) no tiene en cuenta el tamaño del robot; y ii) no es adecuado para entornos con gran densidad de obstáculos.

Las bandas elásticas (*Elastic Bands*, *EB*) consisten en un planificador de movimientos que obtiene un camino libre de obstáculos desde la posición del robot hasta el punto de destino [QK93]. La idea es implementar movimientos basados en la información sensorial mediante la deformación en tiempo real del camino calculado por el planificador, lo que se denomina una banda elástica. Las burbujas son el concepto clave de la técnica, las cuales definen el camino libre de obstáculos en el espacio de trabajo. El camino generado es suave, manteniéndose alejado de los obstáculos. Aunque el método funciona bien en entornos conocidos, no es adecuado para entornos desconocidos o dinámicos.

El cuarto de los métodos comentados es el de la ventana dinámica (*Dynamic Window Approach*, *DWA*). Es un método que evita los choques, donde la búsqueda de los comandos de movimiento se lleva a cabo directamente en el espacio de las velocidades [FBT97]. Los comandos de movimiento se seleccionan a partir de la maximización de una función objetivo. Esta función considera la dirección del objetivo, la velocidad de avance del robot y la distancia a los obstáculos. El problema radica en la selección de los parámetros adecuados para que el método sea válido bajo cualquier circunstancia. De hecho, el ajuste de los parámetros es un problema común a todos los esquemas presentados hasta el momento (*PFM*, *VFH*, *EB* y *DWA*), puesto que es complicado encontrar valores que permitan una navegación reactiva acorde a todas las diferentes situaciones que pueden aparecer.

El diagrama de proximidad (*Nearness Diagram*, *ND*) es una aproximación reactiva a la

navegación de un agente autónomo [MM00]. A partir de la información disponible se extrae una descripción de las regiones libres de obstáculos. Se escoge una de estas regiones, seleccionando una de las cinco posibles situaciones que aparecen. Con estos datos se generan los comandos de movimiento. El método es apropiado para navegar en entornos desconocidos y dinámicos. Quizás su gran ventaja es la posibilidad de trabajar en entornos densos y complejos, algo que no se podía hacer con el método *VFH*. Como inconveniente comparte la parametrización de la navegación, aunque en este caso únicamente hay que ajustar un parámetro. También debemos comentar que, al igual que el método *VFH*, el *ND* no es una técnica puramente reactiva, puesto que se necesita un mapa del entorno para trabajar, por lo que existe un cierto grado de planificación.

Al contrario de lo que ocurre en los métodos analíticos, las técnicas de Inteligencia Artificial no efectúan un modelado analítico del comportamiento reactivo, sino que llevan a cabo un aprendizaje paulatino de la navegación. Este aprendizaje suele requerir de una etapa de entrenamiento. La principal característica de estos métodos es la capacidad de adaptación a circunstancias cambiantes sin supervisión humana. Por otro lado, el entrenamiento de la técnica que se propone en esta Tesis evita la necesidad de tener en cuenta restricciones cinemáticas o la dinámica del robot. Esto hace que el método sea válido para diferentes plataformas después de pequeños cambios y que un mismo esquema pueda ser utilizado para conseguir distintos comportamientos del agente autónomo.

El abanico de técnicas no analíticas empleadas para implementar comportamientos reactivos es muy amplio. Tenemos métodos basados en redes neuronales [RPW$^+$03] y lógica borrosa [SDC05]. También existen aproximaciones soportadas por el razonamiento basado en casos [RAMC97, LKKA05], de especial interés para nosotros, puesto que una de las dos estrategias seguidas para implementar nuestros tres comportamientos se basa en dicho esquema. El razonamiento basado en casos (*Case Based Reasoning*, *CBR*) es una técnica de aprendizaje y adaptación para resolver problemas actuales mediante la recuperación y adaptación de experiencias pasadas. Debido a que el *CBR* será tratado con profundidad en la Sección 3, no comentamos ningún aspecto más por el momento.

Además de la aproximación novedosa basada en el *CBR*, también se han implementado los tres comportamientos mediante una técnica analítica, con objeto de comparar los resultados obtenidos con ambos procedimientos. Nuestra propuesta puede decirse que se basa en los campos de potencial, habiendo sido escogidos estos por su sencillez, aunque su funcionamiento se verá afectado por las oscilaciones y la dificultad para moverse entre obstáculos cercanos. Posteriormente se analizará que ambos métodos funcionan correctamente a la hora de implementar nuestros tres comportamientos, superando el *CBR* al modelo analítico en algunos aspectos.

2 Comportamientos analíticos

Nuestro sistema de exploración está basado en tres comportamientos, presentados en [PUS07]:

- Seguir Pared (SP). El robot se desplaza de forma paralela a la pared que se encuentra a

su derecha, siguiendo su contorno a una distancia determinada.

- Seguir Pasillo (SC). En este caso el agente navega a largo de un pasillo por su centro.
- Cruzar Puerta (CP). El tercer comportamiento permite que el robot cruce una puerta por su centro, con el objetivo de entrar o salir de una habitación.

Se han escogido estos tres comportamientos porque: i) son ampliamente conocidos; ii) han sido extensamente tratados a lo largo de los años; y iii) suelen ser utilizados cuando el robot está explorando un área desconocida [MMBB06, RJ05, ZLY06], como es el caso de la presente Tesis. Así, para el comportamiento Seguir Pared nos encontramos con métodos analíticos [BMT00, AY95, YKY98, CO03], o técnicas que emplean la Inteligencia Artificial, como por ejemplo la lógica borrosa [MMBB06, BSE95]. También para el comportamiento Seguir Pasillo se han empleado modelos analíticos [CO03] o métodos de Inteligencia Artificial como las redes neuronales [Cha05]. Por último, para el comportamiento Cruzar Puerta los autores utilizan, bien un modelo analítico [AGMR03], bien métodos como la lógica borrosa [MYM05].

Nuestra nueva aproximación analítica está basada en los campos de potencial [Kha86], en el sentido de que se emplea una suma ponderada de distintas fuerzas. Debido al diseño realizado que incluye una fuerza de evitación de obstáculos, se incorpora automáticamente para los tres comportamientos el repulsor típico de los campos de potencial. En los tres casos el comportamiento obtiene, a partir de la lectura de sensores sonar (proporcionada por el módulo *ControlRobot*), las velocidades de rotación, v_r, y de traslación, v_t, que el robot debe aplicar a sus motores. Ente enfoque en el que se obtienen las velocidades a partir de sensores sonar no es nuevo [ZGPP02, SY04], al margen de la forma en la que se establece la correspondencia.

A continuación se presentan los detalles de implementación de nuestros tres comportamientos. Este método será además empleado para adaptar los comportamientos según la segunda estrategia que se introduce en esta Tesis, el razonamiento basado en casos (ver Sección 3). Por esta razón se analiza en primer lugar la solución analítica.

2.1 Seguir Pared

Es deseable que cualquier comportamiento de seguimiento de paredes presente tres características [MMBB06]:

1. Mantenimiento adecuado de la pared a una distancia fija.
2. Alta velocidad de traslación, siempre que sea posible.
3. Cambios de dirección suaves y paulatinos.

Estas tres premisas se han tenido en cuenta a la hora de diseñar nuestro comportamiento analítico Seguir Pared. En primer lugar se calcula la velocidad de rotación a partir de la suma de tres fuerzas (Figura 3.2):

- $f_{SPparal}$ mantiene el robot paralelo a la pared.
- f_{SPdist} intenta mantener la distancia prefijada de la pared que se encuentra a su derecha.
- $f_{SPavoid}$ evita los obstáculos que aparecen mientras el robot navega.

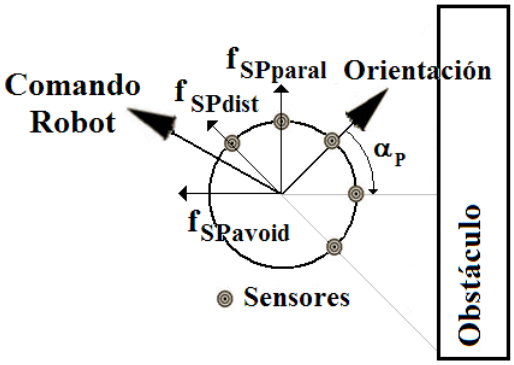

Figura 3.2: Fuerzas del comportamiento Seguir Pared.

Sean $k_{SPparal}$, k_{SPdist} y $k_{SPavoid}$ tres constantes que ponderan $f_{SPparal}$, f_{SPdist} y $f_{SPavoid}$, respectivamente. Considerando que α_P es el ángulo a la pared más cercana a la derecha del robot respecto de su orientación, que d_P es la distancia mínima actual a la pared derecha y que d_F es la distancia a los obstáculos frontales del robot, la velocidad de rotación, v_{SPr} se calcula como:

$$\begin{aligned} v_{SPr} &= f_{SPparal} + f_{SPdist} + f_{SPavoid} = \\ &= k_{SPparal} \cdot (\alpha_P + 90^o) + k_{SPdist} \cdot (d_D - d_P) + k_{SCavoid} \cdot (d_{SPSEC} - d_F) \end{aligned} \tag{3.1}$$

donde d_D es la distancia que el robot debe mantener con la pared derecha y d_{SPSEC} es una distancia umbral para considerar que un obstáculo afecta a la navegación del robot. v_{SPr} puede tomar valores positivos o negativos dependiendo de la situación particular del robot en cada momento Por otro lado, v_{SPr} nunca podrá superar en módulo la máxima velocidad de rotación que se haya programado para la navegación del agente autónomo, v_{rMAX}. Del mismo modo, la fuerza que evita los obstáculos, $f_{SPavoid}$, solamente se considerará en el caso de que $d_F < d_{SPSEC}$. Las constantes de ponderación de las tres fuerzas se escogen con criterio para que la contribución de cada una de esas fuerzas sea máxima bajo determinados supuestos. Así, $k_{SPparal}$ se escoge para que $f_{SPparal}$ sea máxima cuando la orientación del robot es paralela al vector perpendicular a la pared:

$$k_{SPparal} = \frac{v_{rMAX}}{90^o} \tag{3.2}$$

k_{SPdist} se elige para que f_{SPdist} sea máxima cuando la distancia a la pared es nula:

$$k_{SPdist} = \frac{v_{rMAX}}{d_D} \tag{3.3}$$

y $k_{SPavoid}$ se selecciona para que $f_{SPavoid}$ sea máxima cuando la distancia a los obstáculos es mínima, es decir, es la distancia a la que se quiere seguir la pared. Su signo dependerá de la dirección de giro que se deba aplicar, siendo ésta la que conduzca al robot hacia la zona izquierda o derecha en función de la distancia a los obstáculos a ambos lados del robot. Su valor será:

$$k_{SPavoid} = \pm\frac{v_{rMAX}}{d_{SPSEC} - d_D} \tag{3.4}$$

El método se basa, además de en estas tres constantes escogidas con criterio, en dos valores heurísticos, d_D y d_{SPSEC}. Su valor variará dependiendo de la distancia a la que se quiera seguir la pared y de la aplicación bajo estudio, aunque se ha verificado empíricamente que unos valores de d_D entre $400mm$ y $700mm$ y de d_{SPSEC} en torno a los $1000mm$ son adecuados para trabajar en los entornos hacia los que va destinado el trabajo de la presente Tesis.

La velocidad de traslación, v_{SPt}, se obtiene a partir de v_{SPr}, según la expresión:

$$v_{SPt} = v_{tMAX} \cdot \left[1 - \frac{v_{SPr}}{v_{rMAX}}\right] \tag{3.5}$$

donde v_{tMAX} es la máxima velocidad de traslación programada en el robot. Por lo tanto, cuanto mayor sea la velocidad de rotación menor será la de traslación. De este modo, si $v_{SPr} = v_{rMAX}$, la velocidad de traslación será nula. Cuando el robot está girando a una velocidad elevada tiende a trasladarse a baja velocidad.

En las Figuras 3.3 y 3.4 se muestran varios entornos simulados donde el agente debe seguir la pared que se encuentra a su derecha. La máxima velocidad de rotación y de traslación es, respectivamente, $v_{rMAX} = 15^o/s$ y $v_{tMAX} = 100mm/s$. En todas estas pruebas se ha fijado la distancia de seguridad d_{SPSEC} a $1000mm$. En cuanto a la distancia d_D a la que se sigue la pared, ésta será de $500mm$ para el entorno de la Figura 3.3a y de $700mm$ para el resto. Para cada uno de los entornos se puede apreciar la configuración del mismo, la trayectoria seguida por el robot y el mapa métrico del entorno que se construye durante la operación. Al ser nuestros comportamientos puramente reactivos, no existe planificación durante estas pruebas. Por lo tanto, el mapa métrico únicamente se emplea para representar gráficamente el modelo de entorno que el agente adquiere mientras navega.

En el entorno de la Figura 3.3a el robot consigue explorar de forma casi total el escenario, según se aprecia en el mapa métrico de la Figura 3.3d, pero hay una pequeña parte del mismo que no se puede alcanzar únicamente con nuestro comportamiento Seguir Pared. Además, podemos apreciar perfectamente las oscilaciones típicas de los modelos analíticos basados en los campos de potencial, como es nuestro caso. Aunque estas oscilaciones no presentan una gran amplitud, sí que se aprecian y no son deseables. Por otro lado, se puede presentar algún problema si se desea seguir un pasillo como si fuera una pared. Así, por ejemplo, en el pasillo norte del

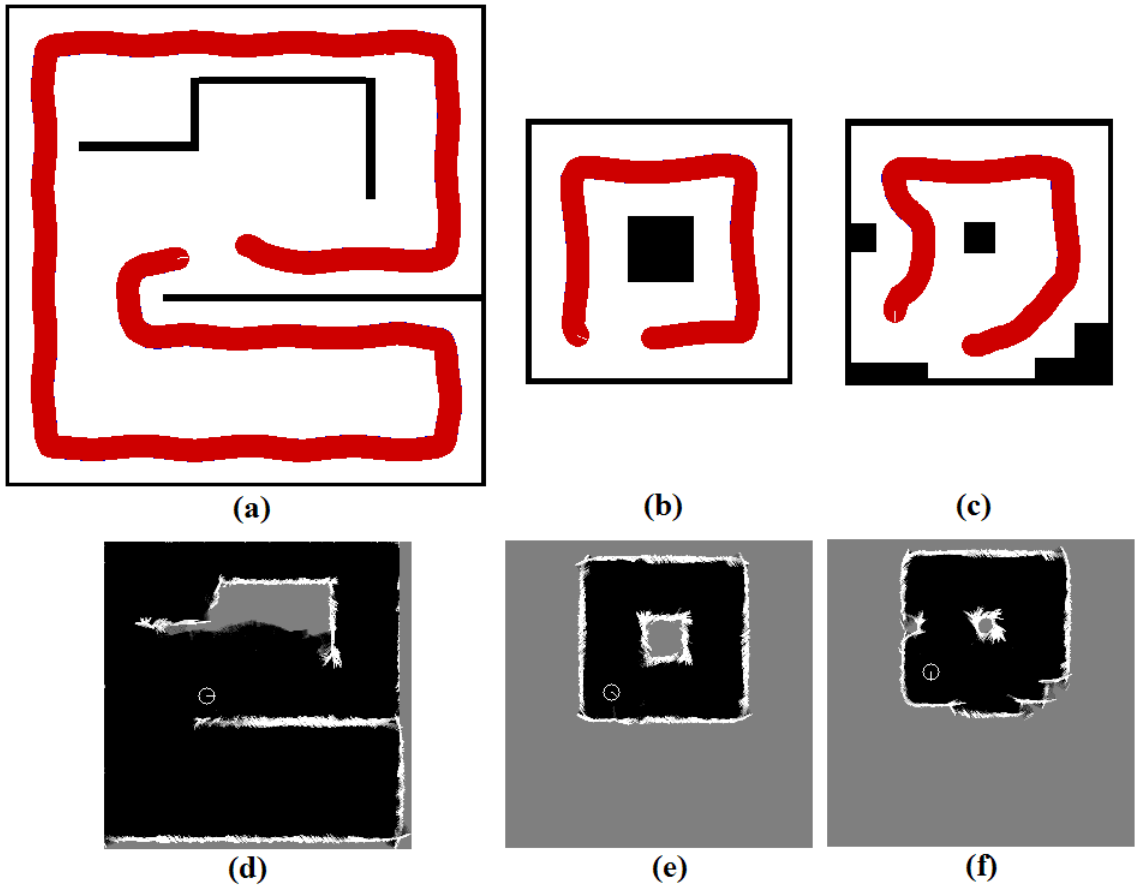

Figura 3.3: Seguir Pared, modelo analítico: (a) entorno simulado 1; (b) entorno simulado 2; (c) entorno simulado 3; (d) mapa métrico obtenido en (a); (e) mapa métrico obtenido en (b); (f) mapa métrico obtenido en (c).

entorno el robot sigue la pared, al margen de la presencia de oscilaciones. Sin embargo, si el pasillo fuera más estrecho o la distancia a la que se quiere seguir la pared fuera más grande, el robot no podría atravesar el pasillo. Una de las fuerzas intentaría alejarlo de la pared derecha para seguirla a la distancia programada. Otra, trataría de evitar la pared izquierda, que sería vista como un obstáculo. El robot comenzaría a oscilar en la abertura del pasillo, que puede ser considerada como una puerta, sin poder avanzar, y evitando así que se siguiera explorando el entorno. Este hecho ha sido probado, manifestándose este efecto cuando $d_D = 700mm$, siendo la anchura del pasillo de tan solo $1.5m$. Obviamente, lo lógico es que para recorrer el pasillo se empleara el comportamiento Seguir Pasillo que, como veremos en la siguiente sección, adapta el movimiento del robot a la anchura del pasillo sin pretender engancharse a una pared a una distancia prefijada. Todo esto nos hace pensar que un único comportamiento no es suficiente para explorar de forma total un entorno desconocido, ya que no es posible adaptarse a una configuración arbitraria.

Los entornos de las Figuras 3.3b y c son muy parecidos. Ambos constan de un obstáculo en el medio de una habitación. Quizás el de la Figura 3.3c es algo más complicado porque el robot se encuentra también con un obstáculo al seguir la pared oeste del mismo. En ambos

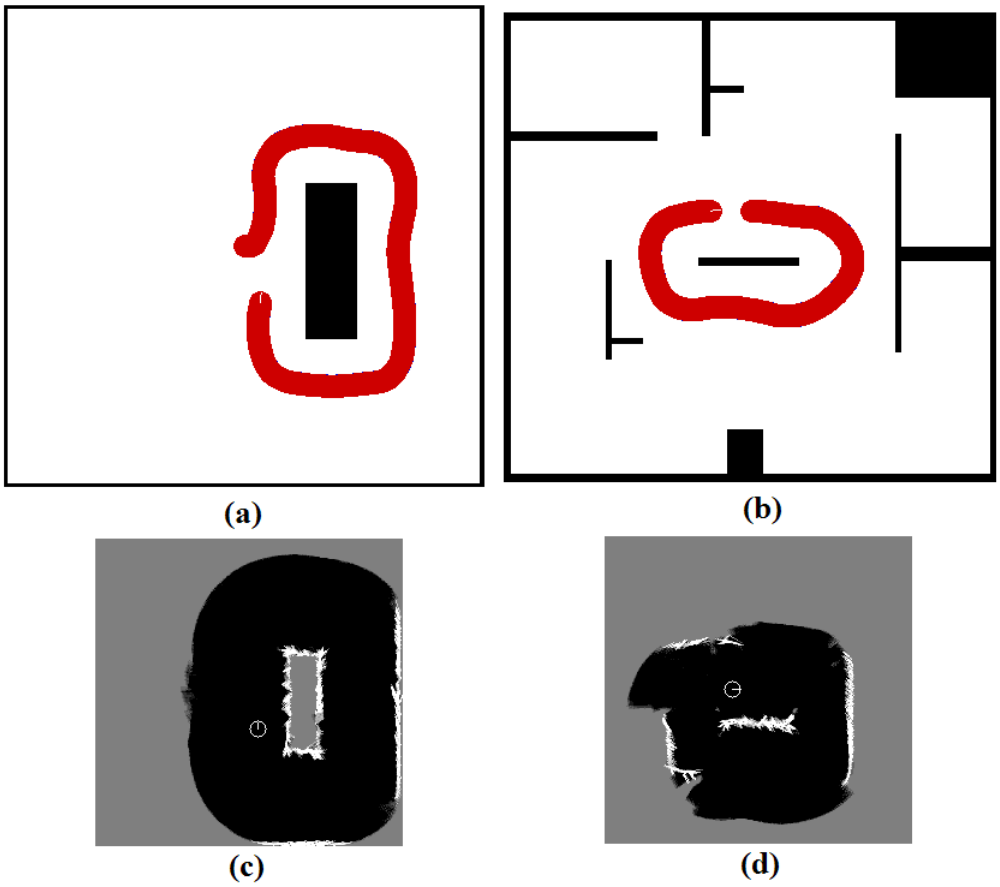

Figura 3.4: Seguir Pared, seguimiento de obstáculo con modelo analítico: (a) entorno simulado 4; (b) entorno simulado 5; (c) mapa métrico obtenido en (a); (d) mapa métrico obtenido en (b).

casos el entorno se explora de forma íntegra, tal y como se observa en las Figuras 3.3e y f. Sin embargo, las oscilaciones nuevamente están presentes durante el movimiento del agente. Así mismo, al analizar la configuración del entorno que el robot se encuentra en su camino, puede llegar a pensarse que, incluso llevando a cabo un movimiento correcto, quizás el comportamiento Seguir Pared no es el más adecuado en determinadas situaciones. De hecho, en esta Tesis nos planteamos como uno de los objetivos el adaptarnos a las circunstancias y particularidades del entorno en cada momento mediante el método de aprendizaje que se propone.

Si bien en los entornos de la Figura 3.3 el entorno se explora total o casi totalmente, existen ocasiones en las que el comportamiento Seguir Pared es incapaz de explorar un entorno en su totalidad (Figuras 3.4c y d). Esta situación se muestra en los dos entornos de la Figura 3.4. En ambos el robot se queda enganchado a un obstáculo, bordeándolo de forma infinita sin llegar a adquirir ningún conocimiento del entorno al margen del que consigue al dar vueltas alrededor del obstáculo. El entorno de la Figura 3.4a es muy simple, pues consta solamente de un obstáculo dentro de una habitación vacía. El robot es incapaz de desengancharse del obstáculo, debido al comportamiento Seguir Pared para el que ha sido programado. Aunque el entorno de la Figura 3.4b es algo más complicado, la conclusión a la que se llega es la misma. El robot sigue la pared

del centro del escenario hacia la derecha, después hacia la izquierda y a continuación repetiría el proceso. Obviamente, las oscilaciones siguen estando presentes para ambos entornos, pues son inherentes al método implementado. Del mismo modo, es evidente la conclusión de que con este comportamiento no es suficiente para alcanzar el objetivo de exploración completa de un entorno total o parcialmente desconocido. Por este motivo, en esta Tesis se presenta una técnica que busca la cooperación o fusión de tres comportamientos para conseguir una adaptación adecuada a la configuración del entorno.

2.2 Seguir Pasillo

La filosofía adoptada para implementar este comportamiento es similar a la de nuestro Seguir Pared. Se obtiene en primer lugar la velocidad de rotación que hay que aplicar al robot a partir de la suma de tres fuerzas, como se puede apreciar en la Figura 3.5:

- $f_{SCparal}$ intenta mantener el robot paralelo a las paredes.
- f_{SCdist}, al contrario que en el comportamiento Seguir Pared, no intenta mantener el robot a una distancia prefijada de la pared derecha, debido a que el robot debe navegar por el centro del pasillo. En este caso se tiene en cuenta dinámicamente el ancho del pasillo para colocar al robot en su centro en todo momento.
- $f_{SCavoid}$ evita los obstáculos que aparecen mientras el robot navega.

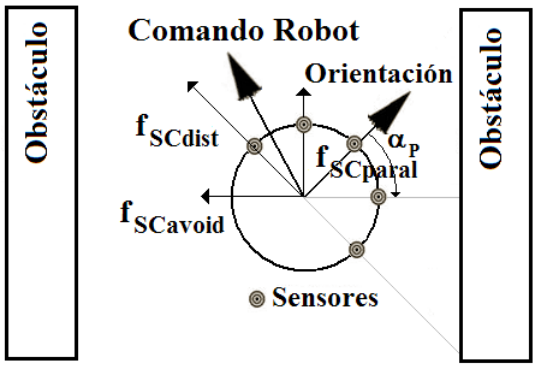

Figura 3.5: Fuerzas del comportamiento Seguir Pasillo.

Sean $k_{SCparal}$, k_{SCdist} y $k_{SCavoid}$ las tres constantes empleadas en este caso para ponderar las fuerzas $f_{SCparal}$, f_{SCdist} y $f_{SCavoid}$, respectivamente. Si bien para el comportamiento Seguir Pared las constantes no variaban, para el Seguir Pasillo la situación es diferente. Como hay que tener en cuenta la anchura del pasillo, su valor es adaptado a la configuración del entorno que el robot detecte en cada momento. Tal y como se comentó en el comportamiento SP, α_P es el ángulo respecto de la pared derecha más próxima, d_P es la distancia mínima actual a la pared derecha y d_F es la distancia a los obstáculos frontales. Entonces, si d_C es la mitad de la anchura del pasillo, la velocidad de rotación v_{SCr} viene definida por:

$$\begin{aligned} v_{SCr} &= f_{SCparal} + f_{SCdist} + f_{SCavoid} = \\ &= k_{SCparal} \cdot (\alpha_P + 90^{\text{o}}) + k_{SCdist} \cdot (d_C - d_P) + k_{SCavoid} \cdot (d_{SCSEC} - d_F) \end{aligned} \tag{3.6}$$

donde d_{SCSEC} es un umbral de distancia empleado para tener en cuenta los obstáculos cuando están muy cercanos al robot. Nuevamente la velocidad de rotación v_{SCr} puede tomar valores positivos o negativos, no pudiendo su módulo superar en ningún caso la máxima velocidad de rotación establecida, v_{rMAX}. Así mismo, la fuerza que evita los obstáculos, $f_{SCavoid}$, solamente se considerará en el caso de que $d_F < d_{SCSEC}$. Otra vez se eligen las constantes de ponderación siguiendo el criterio de su maximización bajo determinadas circunstancias. Sin embargo, en este caso dos de ellas tienen un valor variable que depende de la anchura del pasillo. $k_{SCparal}$ es un valor invariante, que hace máxima la fuerza $f_{SCparal}$ cuando el robot está perpendicular a la pared derecha del pasillo:

$$k_{SCparal} = \frac{v_{rMAX}}{90^{\text{o}}} \tag{3.7}$$

k_{SCdist} se elige para que f_{SCdist} sea máxima cuando la distancia a la pared tiende a cero, para así situar el robot en el centro del pasillo:

$$k_{SCdist} = \frac{v_{rMAX}}{d_C} \tag{3.8}$$

y $k_{SCavoid}$ se selecciona para que $f_{SCavoid}$ sea máxima cuando la distancia a los obstáculos es mínima, considerando la anchura del pasillo. Nuevamente, su signo dependerá de la dirección de giro que se deba aplicar, siendo ésta la que conduzca al robot hacia el centro del pasillo. Su valor viene dado por:

$$k_{SCavoid} = \pm \frac{v_{rMAX}}{d_{SCSEC} - d_C} \tag{3.9}$$

Si bien el valor de estas constantes son escogidas para que se su contribución a v_{SCr} sea la comentada, el método se basa en una constante heurística, d_{SCSEC}. Este valor debe ser escogido dependiendo de la repulsión que se pretenda tener respecto de los obstáculos. Teniendo en cuenta los pasillos potenciales por los que el robot va a navegar, se ha comprobado experimentalmente que si d_{SCSEC} es igual a unos $400mm$ ó $500mm$ el método funciona apropiadamente para la presente Tesis, por supuesto, considerando en cualquier caso los problemas inherentes a las aproximaciones analíticas basadas en los campos de potencial. Se puede observar que este parámetro es menor que en el caso del comportamiento Seguir Pared para poder atravesar pasillos estrechos.

La velocidad de traslación, v_{SCt}, se calcula a partir de v_{SCr} con una expresión similar a la del comportamiento SP:

$$v_{SCt} = v_{tMAX} \cdot \left[1 - \frac{v_{SCr}}{v_{rMAX}}\right] \tag{3.10}$$

recordando que v_{tMAX} es la máxima velocidad a la que se puede trasladar el robot. La Ecuación (3.10) implica que a mayor velocidad de rotación, menor velocidad de traslación. Si $v_{SCr} = v_{rMAX}$, ocurrirá que $v_{SCt} = 0$.

En las Figuras 3.6, 3.7 y 3.8 se muestran varios entornos donde el robot recorre pasillos con diferentes características. Al igual que para el caso del comportamiento Seguir Pared, la máxima velocidad de rotación y de traslación es, respectivamente, $v_{rMAX} = 15^o/s$ y $v_{tMAX} = 100mm/s$. En todas estas pruebas se ha fijado la distancia de seguridad d_{SCSEC} a $400mm$. Para cada uno de los entornos se puede apreciar la configuración del mismo, la trayectoria seguida por el robot y el mapa métrico del entorno que se construye durante la operación. Del mismo modo, y tal como se comentó para el comportamiento Seguir Pared, el mapa métrico únicamente se emplea para representar gráficamente el modelo de entorno que el agente adquiere.

En la Figura 3.6 se presenta el seguimiento de tres pasillos de diferente anchura. En todos los casos se observan claramente las oscilaciones propias de los métodos de navegación reactiva basados en los campos de potencial. El efecto es similar a si el robot fuera rebotando de una pared a otra al mismo tiempo que recorre el pasillo. En los entornos de las Figuras 3.6a y 3.6c el robot parte centrado respecto de los pasillos. Sin embargo, para el entorno simulado 2 (Figura 3.6b), el agente se encuentra más cerca de la pared inferior que de la superior. Gracias a la implementación analítica realizada, al comienzo el robot tiende a centrarse en el pasillo, para pasar posteriormente a recorrerlo. Los tres mapas métricos muestran el modelo de entorno que se adquiere durante la operación (Figuras 3.6d, e y f).

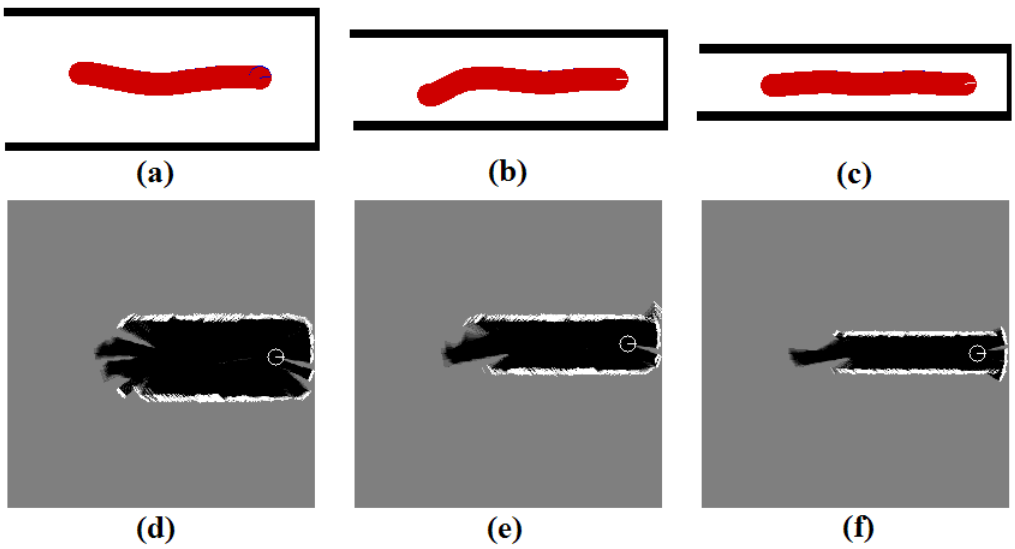

Figura 3.6: Seguir Pasillo, diferentes anchuras con modelo analítico: (a) entorno simulado 1; (b) entorno simulado 2; (c) entorno simulado 3; (d) mapa métrico obtenido en (a); (e) mapa métrico obtenido en (b); (f) mapa métrico obtenido en (c).

La Figura 3.7 presenta tres escenarios algo más complejos. En este caso, el pasillo consta de uno o dos obstáculos. Estas situaciones pueden ser entendidas como la concatenación de pasillos de diferentes anchuras, ya que al aparecer un obstáculo el pasillo se ve estrechado. El primero de los entornos de la Figura 3.7 consta de un obstáculo en la parte superior del pasillo (entorno simulado 4 de la Figura 3.7a). El robot comienza su movimiento, tratando de centrarse en el pasillo. Cuando el obstáculo es detectado, se debe desplazar hacia la pared sur, para así adecuarse a la anchura de la configuración que en ese momento se encuentra. Al dejar el obstáculo atrás, el robot nuevamente tratará de centrarse, tendiendo su movimiento a ir hacia el norte. En la Figura 3.7b se observa una situación análoga, con un obstáculo en la parte inferior del pasillo. Como al comienzo el robot está ligeramente más cerca de la pared superior que de la inferior, tratará de centrarse en el pasillo. Sin embargo, al hacerlo e ir avanzando, se detecta el obstáculo inferior. Por lo tanto, el agente se dirige hacia la parte norte para abordar el estrechamiento del pasillo. Una vez atravesado el obstáculo, nuevamente se centra en el pasillo, considerando la anchura del mismo. En el tercero de los entornos (Figura 3.7c), el robot se encuentra con dos obstáculos en su camino, uno primero en la parte superior y, posteriormente, otro en la inferior. El entorno es una especie de combinación de los dos anteriores. El robot, por tanto, recorre el pasillo por su centro, adaptando su movimiento a la anchura que en cada momento se encuentra. En principio tiene que ir hacia el sur para pasar el primer obstáculo. Después hacia el norte, para dejar atrás el segundo. Y, por último, tiende a centrarse al final del pasillo.

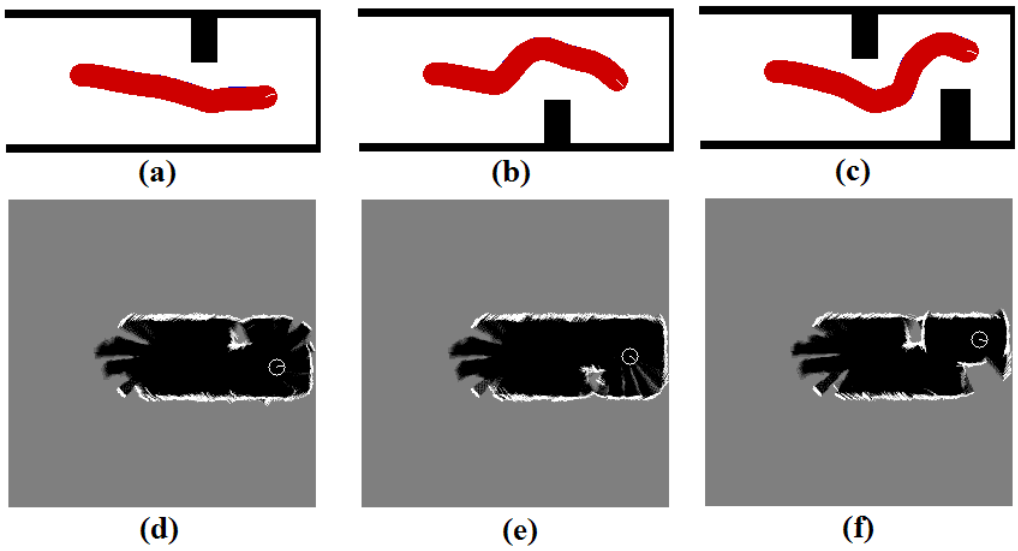

Figura 3.7: Seguir Pasillo, pasillo con obstáculos, modelo analítico: (a) entorno simulado 4; (b) entorno simulado 5; (c) entorno simulado 6; (d) mapa métrico obtenido en (a); (e) mapa métrico obtenido en (b); (f) mapa métrico obtenido en (c).

La principal característica de la configuración de los entornos que se presentan en la Figura 3.8 es su forma en L. En ambos casos, el robot se encuentra en su camino dos pasillos perpendiculares. Al margen de las oscilaciones inherentes al método que se observan, el robot es capaz de seguir ambos pasillos, girando los 90 grados necesarios para orientarse según el eje del segundo

pasillo. Por lo tanto, debido al diseño realizado para este comportamiento, en estas situaciones el robot perfectamente podría seguir ambos pasillos con únicamente este comportamiento. Sin embargo, parece lógico que no es lo más adecuado. Cuando el robot finaliza el seguimiento del primer pasillo desaparece la pared a su izquierda, encontrándose con una situación para la que es más adecuado el comportamiento Seguir Pared. Más aún, al tratar de enfilarse con el segundo pasillo, la situación que el robot ve es parecida a una puerta. Por lo tanto, sería conveniente el emplear más de un comportamiento para explorar de forma total un entorno en previsión de que puedan aparecer configuraciones como las de la Figura 3.8. Por último, comentar que, a pesar de que el pasillo inicial del entorno de la Figura 3.8b tiene una anchura de $1.2m$, el robot lo recorre adecuadamente, al margen de las oscilaciones.

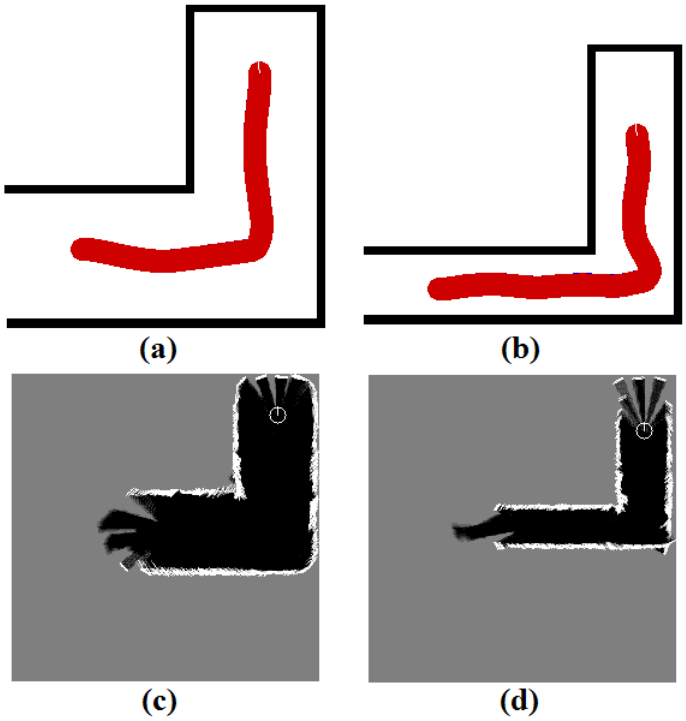

Figura 3.8: Seguir Pasillo, pasillo en L con modelo analítico: (a) entorno simulado 7; (b) entorno simulado 8; (c) mapa métrico obtenido en (a); (d) mapa métrico obtenido en (b).

2.3 Cruzar Puerta

Nuestro tercer comportamiento también se ha implementado a partir de una combinación lineal de fuerzas, igual que los dos anteriores, SP y SC. En el diseño se han tenido en cuenta tres consideraciones [MYM05]:

1. La distancia frontal a la puerta.
2. La distancia a la parte izquierda y derecha del marco de la puerta.
3. Atravesar la puerta por el centro.

La velocidad de rotación es calculada en primer lugar, para pasar acto seguido a la obtención de la de traslación. La principal diferencia con respecto a los dos comportamientos anteriores estriba en el hecho de que ahora únicamente se suman dos fuerzas (Figura 3.9):

- f_{CPort} permite que el robot permanezca de forma perpendicular al marco de la puerta.
- $f_{CPavoid}$ evita el marco de la puerta, previniendo posibles colisiones con él.

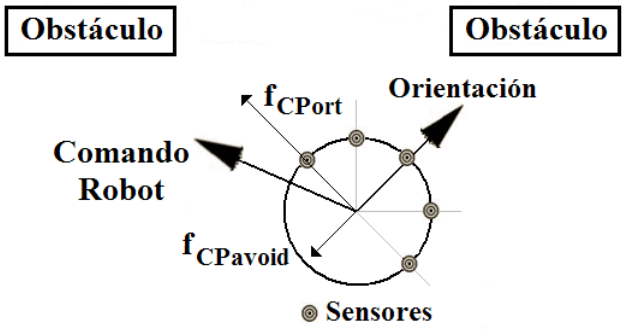

Figura 3.9: Fuerzas del comportamiento Cruzar Puerta.

Sean k_{CPort} y $k_{CPavoid}$ dos constantes que ponderan las fuerzas f_{CPort} y $f_{CPavoid}$, respectivamente. Al contrario que para el comportamiento Seguir Pared, estas dos constantes no varían nunca. Del mismo modo, sean d_L y d_R las distancias a la parte izquierda y derecha del marco de la puerta, respectivamente. Si, como se ha comentado anteriormente, d_F es la distancia a los obstáculos frontales del robot, la velocidad de rotación v_{CPr} será:

$$v_{CPr} = f_{CPort} + f_{CPavoid} = k_{CPort} \cdot (d_L - d_R) + k_{CPavoid} \cdot (d_{CPSEC} - d_F) \tag{3.11}$$

donde d_{CPSEC} es el umbral de distancia necesario para considerar que los obstáculos están cercanos al robot. v_{CPr} puede ser positiva o negativa, al igual que en los dos comportamientos anteriores, no pudiendo superar en ningún caso su módulo la máxima velocidad de rotación, v_{rMAX}. Del mismo modo, $f_{CPavoid}$ se tendrá en cuenta cuando se verifique que $d_F < d_{CPSEC}$. k_{CPort} se ha fijado para tener en cuenta la distancia de seguridad d_{CPSEC} y el máximo valor que retornan los sensores sonar, dado por el fabricante, d_{MAX}. Así, su valor será:

$$k_{CPort} = \frac{v_{rMAX}}{d_{MAX} - d_{CPSEC}} \tag{3.12}$$

y $k_{SCavoid}$ se selecciona para que $f_{SPavoid}$ sea máxima cuando la distancia a los obstáculos sea mínima. Nuevamente, su signo dependerá de la dirección de giro que se deba aplicar. Su valor viene dado por:

$$k_{CPavoid} = \pm \frac{v_{rMAX}}{d_{CPSEC}} \tag{3.13}$$

Además de en las dos constantes cuyo criterio de elección ha sido comentado, el método también se basa en un valor heurístico, d_{CPSEC}. Se ha comprobado experimentalmente que estableciendo este valor a unos $300mm$ ó $400mm$ se consigue que el modelo funcione adecuadamente en los entornos de trabajo de esta Tesis. Este valor debe ser pequeño para poder atravesar las puertas, las cuales pueden ser potencialmente estrechas.

Una vez estamos en disposición de la velocidad de rotación se calcula la de traslación, v_{CPt}. Su expresión es análoga a la de los dos comportamientos anteriores:

$$v_{CPt} = v_{tMAX} \cdot \left[1 - \frac{v_{CPr}}{v_{rMAX}}\right] \tag{3.14}$$

siendo v_{tMAX} la máxima velocidad de traslación fijada para el agente autónomo. Ya que la Ecuación (3.14) es similar a las Ecuaciones (3.5) y (3.10), comparte sus características. Así, a mayor velocidad de rotación, menor velocidad de traslación.

La Figura 3.10 presenta 15 entornos simulados distintos donde el robot debe atravesar, en cada uno de ellos, una puerta. Al igual que para los comportamientos Seguir Pared y Seguir Pasillo, la máxima velocidad de rotación y de traslación es, respectivamente, $v_{rMAX} = 15^o/s$ y $v_{tMAX} = 100mm/s$. La distancia d_{CPSEC} se ha fijado en todos los casos a $300mm$. Para cada uno de los entornos se presenta la trayectoria seguida por el robot durante su operación.

En todos los casos que se muestran en la Figura 3.10 el robot atraviesa correctamente la puerta que se encuentra en su camino. Sin embargo, hay diferencias cualitativas en el movimiento llevado a cabo en cada entorno que deben ser comentadas. En primer lugar, si el robot está centrado respecto del centro de la puerta (misma distancia a la parte izquierda del marco que a la derecha), se esperaría que la atravesara sin oscilar. Sin embargo, esto no es así, según se observa en las Figuras 3.10a, g y l, pues aparecen unas pequeñas oscilaciones. Se debe a la propia limitación que el método implementado tiene por basarse en los campos de potencial. En segundo lugar, si el robot no se encuentra centrado, es decir, si está más cerca de la parte superior o inferior del marco, no consigue centrarse del todo para atravesarla. Así, en las Figuras 3.10c y f el robot parte desde la parte superior, atravesando la puerta más cerca de esta parte del marco. Igualmente, en las Figuras 3.10b, e, i y k, el agente comienza el movimiento desde la parte inferior, por lo que la puerta se atraviesa más cerca de esta parte del marco.

Por último, a veces se realiza un movimiento que puede considerarse peligroso, a pesar de atravesar la puerta. Si bien en todos los entornos de la Figura 3.10 el robot atraviesa la puerta, una de estas situaciones se presenta en la Figura 3.10n. La puerta solamente tiene $90cm$ de ancho. Teniendo en mente que el robot tiene unas dimensiones de $50x50cm^2$, se aprecia la dificultad del movimiento. Y todavía más si no está centrado, como ocurre en este entorno. Se observa que el agente no choca, pero casi lo hace. Todo esto nos sugiere la dificultad para cruzar puertas estrechas, y no solamente en cuanto a la seguridad del agente para evitar su choque. Debido a la implementación realizada que se basa en los campos de potencial, se pueden dar circunstancias que hagan que el robot comience a oscilar delante de la puerta sin poder atravesarla.

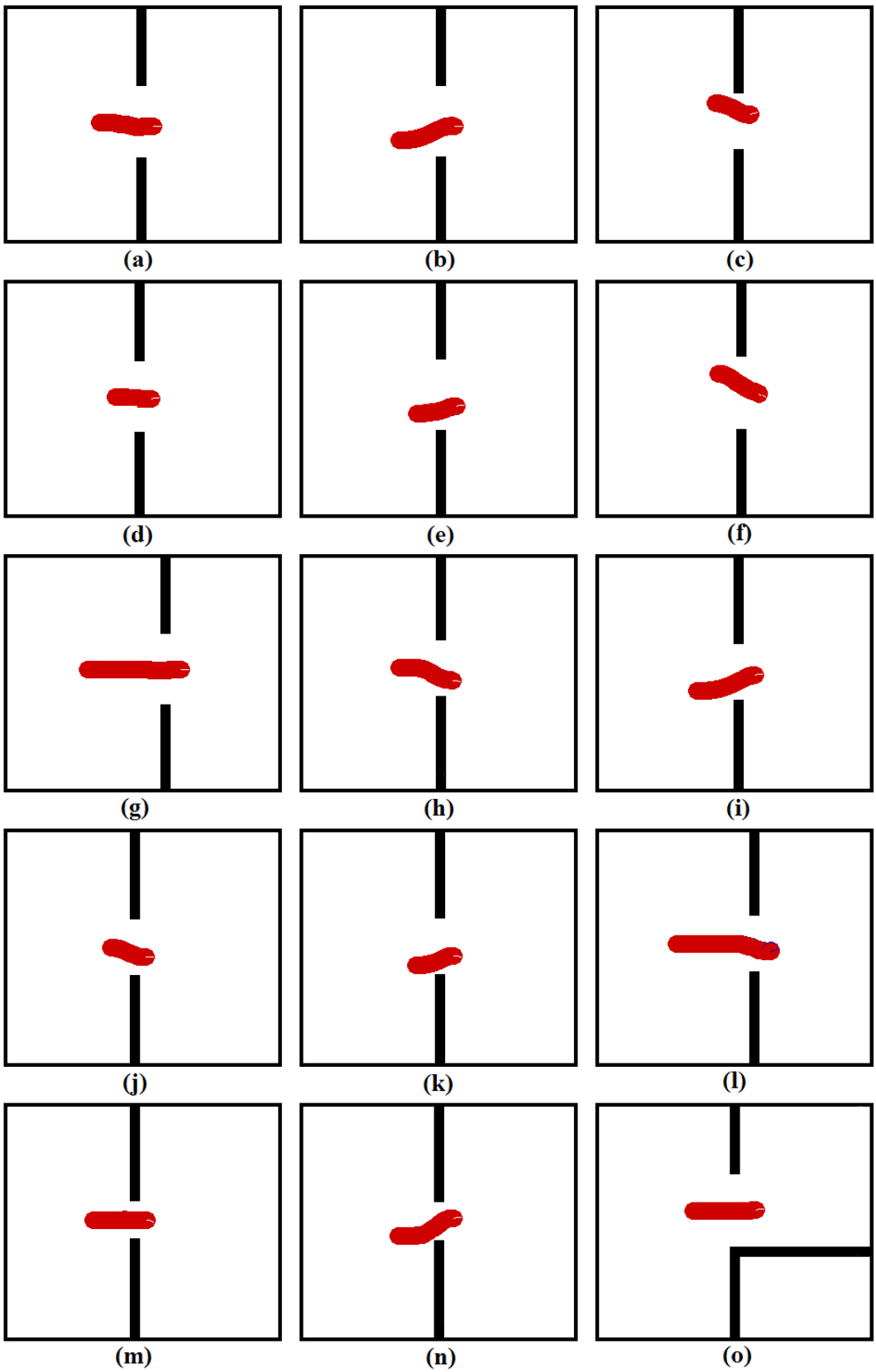

Figura 3.10: Cruzar puerta, entornos simulados con modelo analítico: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7; (h) 8; (i) 9; (j) 10; (k) 11; (l) 12; (m) 13; (n) 14; (o) 15.

3 Comportamientos mediante CBR

Nuestros tres comportamientos también han sido desarrollados siguiendo una técnica de Inteligencia Artificial, el razonamiento basado en casos (*Case Base reasoning*, *CBR*). Como los comportamientos analíticos de la Sección 2 se emplean además para adaptar los implementados mediante el *CBR*, aquéllos han sido presentados en primer lugar. Así pues, una vez introducidos los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta según el método analítico, ya sí podemos analizar los basados en el *CBR*, los cuales se presentan en [PUS07].

3.1 El *CBR*

El razonamiento basado en casos (*CBR*), es una técnica de aprendizaje y adaptación para revolver problemas actuales mediante la recuperación y adaptación de problemas pasados [AP94]. Su origen hay que buscarlo en el trabajo de Schank y Abelson de 1977 [SA77] sobre la memoria dinámica y el papel central del recuerdo de situaciones y patrones pasados en la resolución y aprendizaje de problemas [Sch82].

El primer sistema basado en el *CBR* fue *CYRUS* [Kol83], desarrollado en la Universidad de Yale. *CYRUS* estaba basado en el modelo de memoria dinámica de Schank, siendo su esquema la base de otros sistemas basados en el *CBR*, como *MEDIATOR* [Sim85], *PERSUADER* [Syc88], *CHEF* [Ham89], *JULIA* [Hin92] y *CASEY* [Kot89], todos ellos desarrollados en Estados Unidos. Otros ejemplos de sistemas desarrollas según la filosofía del *CBR*, también en Estados Unidos, son *PROTOS* [Bar89], *GREBE* [Bra91], *HYPO* [Ash91] y *CABARET* [SR92].

El comienzo del uso del paradigma *CBR* en Europa data de unos años más tarde del inicio del mismo en Estados Unidos. Uno de los primeros sistemas fue *MOTLKE* [Alt89], desarrollado por Michael Richter y Klaus Dieter Althoff en la Universidad de Kaiserslautern. Este sistema dio lugar, posteriormente, al sistema *PATDEX* [RW91]. También nos encontramos con los sistemas *REFINER* [SS88], *IULIAN* [Oeh92] y *CREEK* [Aam91]. Incluso en España se desarrollaron sistemas siguiendo el razonamiento basado en casos, como el de Enric Plaza y Ramón López [PL90]. Actualmente, el *CBR* es una técnica muy conocida que se emplea en multitud de campos de investigación como son, entre otros, el diagnóstico, la clasificación, la interpretación, la planificación, la minería de datos y las aplicaciones comerciales. Tan importante ha llegado a ser este paradigma que incluso hay conferencias exclusivamente dedicadas a él, como es la *European Conference on Case-Based Reasoning*.

En la Robótica también se aplica el paradigma *CBR* para la resolución de problemas. Así, nos encontramos con su uso para realizar tareas de planificación [PH92], navegación guiada por visión [WC95] y localización [MPSU01]. En la presente Tesis se contribuye a la navegación autónoma mediante el empleo del *CBR*, aprendiendo comportamientos reactivos específicos. Este paradigma ha sido previamente usado para resolver problemas de esta índole. Existen aproximaciones válidas para entornos estáticos [FL95]. También existen sistemas que tratan con entornos dinámicos, usando un mapa topológico de un entorno conocido *a priori* [HV95],

aunque el mapa topológico tiene que volver a ser construido cuando se modifica el entorno. Para solventar este inconveniente, Kruusmaa propone el uso de mapas probabilísticos [Kru03]. Sin embargo, en su trabajo se afirma que el sistema de navegación global solamente es válido cuando los obstáculos son grandes y numerosos. Así mismo, existen trabajos relacionados con la navegación basada en comportamientos que emplean el *CBR* [RAMC97, LA01]. Si bien el primero únicamente presenta resultados simulados [RAMC97], el segundo es una de las pocas aproximaciones basadas en el *CBR* probada en robots reales [LA01]. Este último esquema se basa en la acumulación de experiencia sobre una ventana temporal mientras se navega, para así conseguir que los comportamientos cooperen. Sin embargo, no es un esquema adecuado para entornos dinámicos donde la configuración cambia frecuentemente.

Aunque todos los métodos anteriores hacen uso del *CBR*, ninguno de ellos se puede considerar como puramente reactivo. Además, excepto la técnica de integración temporal de Likhachev [LA01], ninguno de los métodos ha sido probado en robots reales, sino tan solo por medio de simulaciones. En esta Tesis se propone un nuevo esquema para el diseño e implementación de comportamientos para navegación reactiva que emplea la filosofía del *CBR*. Si se mantiene la aproximación analítica puramente reactiva no se pueden eliminar problemas como las oscilaciones, por lo que se aprovechan las ventajas que el *CBR* proporciona mediante el aprendizaje. Además de por ser una aproximación novedosa, el nuevo desarrollo también destaca por haber dado resultados satisfactorios tanto en pruebas simuladas como en pruebas reales [PUS07]. El método es válido para cualquier entorno arbitrario, ya que al ser puramente reactivo no se necesita información global sobre el mismo. Por lo tanto, el sistema no almacena información de una configuración determinada. Además, el sistema:

- Se adapta de manera dinámica a los cambios o situaciones inesperadas.
- Es capaz de aprender las peculiaridades del entorno de trabajo.
- Elimina la necesidad de considerar la cinemática del robot, pues ésta está inherentemente tenida en cuenta. Por lo tanto, se puede adoptar la misma filosofía para otra plataforma robótica con pequeños cambios.

A continuación se analiza la manera en la que se han diseñado e implementado los tres comportamientos, Seguir Pared, Seguir Pasillo y Cruzar Puerta, según el razonamiento basado en casos. Como la filosofía de diseño de estos tres comportamientos es análoga, se presenta en primer lugar la manera en la que cualquier comportamiento genérico sería diseñado en la Sección 3.2. Posteriormente se detallan las particularidades de cada uno en la Sección 3.3.

3.2 Diseño e implementación

En el Departamento de Tecnología Electrónica [1] de la Universidad de Málaga se usa el trabajo realizado por el grupo *KEMLg (Knowledge Engineering and Machine Learning Group)* de la

[1] Departamento en el que se realiza el trabajo de la presente Tesis

Universidad Politécnica de Cataluña para implementar el paradigma *CBR* [SMCR+97]. Este trabajo desarrolla un servidor *CBR* cuya objetivo es devolver la situación más parecida para una determinada situación de entrada, a través de una conexión remota. El servidor dota de una gran potencialidad a los diseñadores, ya que se puede configurar y adaptar al problema particular a tratar, la implementación de los comportamientos reactivos en nuestro caso. Sin embargo, la implementación del *CBR* que se utiliza en esta Tesis es propia, sin hacer uso de un servidor *CBR* en otra máquina. Se ha desarrollado toda la infraestructura local necesaria para el correcto funcionamiento de los comportamientos reactivos que emplean el *CBR* en el lenguaje de programación *C*. Se ha hecho así para que todo el sistema se ejecute en una misma máquina, al margen del simulador o la plataforma robótica real, recordando además que se utiliza el esquema local síncrono del estilo de la arquitectura *DLA*. Aun así, se ha seguido el formato del servidor *CBR* del grupo *KEMLg* por si en un futuro se quisiera hacer uso del mismo. De hecho, al usar el servidor *CBR* el sistema era más lento, puesto que ante una situación de entrada había que conectarse de manera remota al servidor para recuperar la solución en vez de obtenerla de forma local en la misma máquina.

A la hora de diseñar nuestros comportamientos con la filosofía del *CBR* hay que tener en cuenta la forma en la que se representa y almacena el conocimiento, el modo de operación del sistema o ciclo *CBR*, así como la posibilidad de clasificar la base de datos con el conocimiento. Todos estos aspectos son tratados con detalle a continuación.

3.2.1 Representación y almacenamiento de los casos

El *CBR* se basa en la recuperación y adaptación de problemas pasados para resolver situaciones actuales. Cualquier situación que aporte experiencia al sistema se almacena, constituyendo lo que se denomina caso. Un caso es una porción de conocimiento que representa una experiencia [Wat95]. Un caso debe contener, al menos, la descripción del problema y la solución a este problema. La solución de los casos servirá como información a usar para resolver situaciones similares a las almacenadas.

A la hora de diseñar un sistema basado en el *CBR* hay que seleccionar adecuadamente la información que va a estar contenida en cada uno de los casos. Esta decisión de diseño es de vital importancia, pues condiciona el funcionamiento de todo el sistema. La información no debe ser excesiva, para impedir que el coste computacional de recuperación de la experiencia sea muy alto, y tampoco puede ser escasa, puesto que entonces muchas situaciones distintas podrían estar contenidas en un mismo caso. Por lo tanto, la definición del caso es crucial.

Los comportamientos de navegación implementados en nuestro sistema son puramente reactivos. Un comportamiento reactivo es aquél que asocia una determinada acción en respuesta a la información de entrada. La única información de entrada que poseemos proviene de la lectura de los sensores sonar, proporcionada por el módulo *ControlRobot*. Por lo tanto, la descripción de la entrada del caso estará constituida por las lecturas sonar, según se muestra en la definición del caso empleada en la presente Tesis (Figura 3.11). En cuanto a su salida, estará formada

por la velocidad de traslación, v_t, y la velocidad de rotación, v_r, que el robot debe aplicar a sus motores en respuesta a la lectura de los sensores sonar. Con esta salida se consigue implementar, siguiendo otra filosofía completamente distinta, un comportamiento similar al desarrollado mediante el modelo analítico. Es decir, se proporcionan las velocidades de traslación y rotación a aplicar ante una determinada configuración del entorno detectada mediante los sensores. La orientación del robot no se ha incluido en la descripción de la entrada del caso porque no afecta al problema, ya que el robot debe reaccionar a una determinada configuración del entorno, independientemente de su orientación.

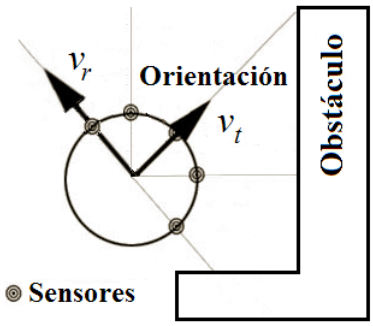

Figura 3.11: Definición del caso para el *CBR*.

Formalmente, un caso $\overrightarrow{CA}$ será un vector constituido por dos subvectores, uno con la entrada del caso, $\overrightarrow{CA}^{IN}$, y el otro con su salida, $\overrightarrow{CA}^{OUT}$:

$$\overrightarrow{CA} = [\overrightarrow{CA}^{IN}, \overrightarrow{CA}^{OUT}] \tag{3.15}$$

donde la entrada del caso incluye la lectura de los n sensores sonar, s_i:

$$\overrightarrow{CA}^{IN} = [s_1, s_2, \ldots, s_n] \tag{3.16}$$

y la salida del caso viene determinada por la velocidad de traslación, v_t, y de rotación, v_r, que el agente debería aplicar:

$$\overrightarrow{CA}^{OUT} = [v_t, v_r] \tag{3.17}$$

Los casos se almacenan en la denominada base de casos (*Case Base*), o biblioteca de casos (*Case Library*). Las dos principales aproximaciones en cuanto a la organización de la base de casos son la memoria plana y la memoria jerárquica [SMCR+97].

Las memorias planas siempre devuelven el caso más parecido de la base de casos ante una situación determinada. Esto constituye una gran ventaja, puesto que nunca se deja de explorar ningún caso durante la búsqueda de la solución del problema actual, ya que se compara nuestra situación actual con todos los casos de la base de casos. Más aún, son estructuras fáciles de implementar, donde la inclusión de un nuevo caso en la base de casos es simple. Sin embargo, su

gran inconveniente estriba en el hecho de que el tiempo necesario para obtener el mejor caso es potencialmente elevado, siendo éste dependiente del tamaño de la base de casos. Por lo tanto, para un uso eficiente de este esquema hay que asegurar que el tamaño de la base de casos no es demasiado grande.

En las memorias jerárquicas el proceso de comparación y el tiempo de recuperación es bastante más eficiente que en el caso de las memorias planas, siendo una aproximación adecuada para sistemas con muchos casos. Es debido a que únicamente se consideran unos cuantos casos durante este proceso, después de una selección de la búsqueda dentro de la estructura jerárquica. Al margen de esta ventaja, también existen varios inconvenientes. Así, la organización de la estructura es mucho más complicada que en las memorias dinámicas, puesto que se necesita agrupar los casos en función de su semejanza. Por lo tanto, la inclusión de un nuevo caso en la base de casos en más costosa computacionalmente. Quizás más grave aún es el hecho de que, por la propia configuración del esquema, se dejan de explorar casos durante el proceso de búsqueda, pudiendo devolver el sistema como mejor caso uno que no lo es en realidad.

Considerando las ventajas y los inconvenientes tanto de las memorias planas como de las memorias jerárquicas, en esta Tesis se ha optado por la implementación de la estructura plana para la base de casos. Tres son los motivos que han llevado a tomar esta decisión. El primero, la sencillez de la implementación. El segundo, asegurar que el caso devuelto para una determinada situación es siempre el mejor. Y, por último, porque nuestra base de casos será clasificada, entre otras razones, para reducir el tiempo invertido en la recuperación del mejor caso.

3.2.2 El ciclo *CBR* aplicado al diseño de comportamientos reactivos

De forma genérica, cualquier sistema *CBR* se rige por del denominado ciclo *CBR* [AP94], que descompone los procesos involucrados en cuatro fases (Figura 3.12):

1. Recuperar. Su objetivo es obtener el caso más parecido (mejor caso) de la base de casos para el problema actual.
2. Reusar. Consiste en emplear la información del caso recuperado para solucionar el problema actual.
3. Revisar. Su misión es analizar la validez de la solución propuesta en la fase anterior como solución para la situación actual.
4. Retener. Por último, si la solución al nuevo problema es válida, se almacena en la base de casos, pasando a formar parte de la experiencia para situaciones futuras.

El ciclo *CBR* de la Figura 3.12 se conoce como el esquema de las cuatro *REs*. La descripción de una situación nueva define un nuevo caso. Este caso se usa para recuperar el mejor caso de la base de casos con la experiencia pasada. En la fase Reusar, el caso recuperado se combina con el nuevo caso, proporcionando la solución propuesta para la nueva situación. Mediante la

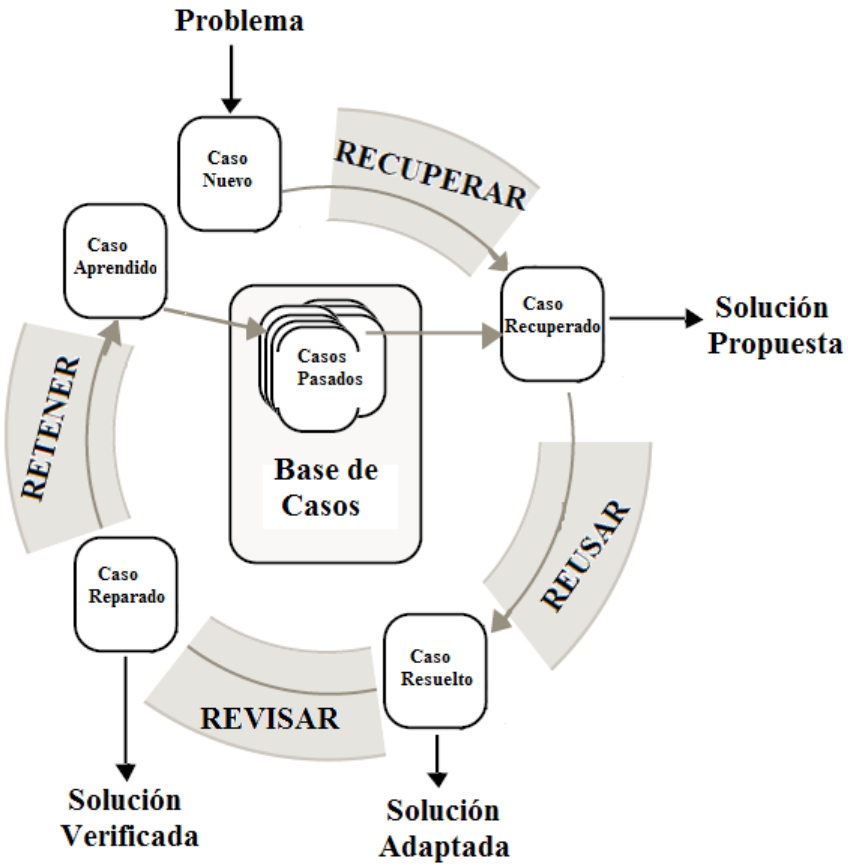

Figura 3.12: El ciclo *CBR*.

fase Revisar se evalúa la calidad de la solución propuesta, modificándola en el caso de que sea necesario. Finalmente, durante la etapa Retener la nueva experiencia es almacenada en la base de casos para poder usarla en un futuro para resolver otros problemas.

A continuación se analizan las cuatro fases del ciclo *CBR* en su aplicación al diseño de comportamientos reactivos. Tay y como se ha comentado anteriormente, la filosofía de diseño de nuestros tres comportamientos es análoga. Si se deseara incluir nuevos comportamientos habría que seguir el mismo procedimiento, añadiendo un módulo nuevo a la capa *Comportamientos* por cada comportamiento añadido. Gracias a la modularidad del sistema, la extensibilidad en el futuro es sencilla y está garantizada.

3.2.2.1 Recuperación del caso: Recuperar El objetivo de esta primera etapa es, para una nueva situación de entrada o caso nuevo, recuperar de la base de datos el mejor caso posible para ese nuevo problema, es decir, el caso más parecido. Para poder evaluar el mejor caso es necesaria una medida de la similitud entre casos. El mejor caso sería el que presentara la máxima similitud respecto del nuevo caso. Al ser la estructura de la base de casos plana se requiere la comparación de todos los casos de la base de casos con el nuevo caso. A partir de todas estas comparaciones se deduce cuál es el mejor caso, siendo éste el recuperado por el sistema.

Si recordamos, cualquier caso de la base de casos se describe con una entrada y la salida a ejecutar para esa entrada. La entrada está constituida por las lecturas dadas por los sensores sonar. La salida consiste en las velocidades de traslación y de rotación a aplicar para implementar el comportamiento deseado. Pequeñas diferencias entre las lecturas de los sensores pueden derivar en casos diferentes. Sin embargo, estas pequeñas diferencias suelen corresponder a situaciones similares. Como además se ha demostrado que la discretización de los casos es una decisión recomendada en los sistemas *CBR* [SMCB$^+$99], la descripción de un caso se puede mejorar si se discretizan las lecturas sonar. Por lo tanto, nuestra base de casos será discretizada, manteniendo el sistema tanto la base de casos sin discretizar como la que lo está.

El primer paso del proceso de discretización de la base de casos consiste en la definición del número de intervalos o rangos de discretización y el establecimiento de dichos rangos. Como este paso se efectúa de forma independiente para cada una de las lecturas que constituyen la entrada de un caso, se podría emplear distinto número y margen de rangos para cada sensor. Sin embargo, en nuestro sistema no se desea favorecer ninguna orientación respecto a cualquier otra. Por lo tanto, todas las entradas (sensores sonar), se van a discretizar siguiendo el mismo esquema, es decir, con el mismo número de rangos y con una idéntica definición de estos márgenes. En nuestro sistema se van a emplear 5 rangos para cada entrada. Si la entrada de un caso se describe con la lectura s_i dada por n sensores sonar, el vector $\overrightarrow{NR}$ con el número de rangos para cada entrada será un vector de n componentes de valor 5:

$$\overrightarrow{NR} = [nr_1, nr_2, \ldots, nr_n] = [5, 5, \ldots, 5] \tag{3.18}$$

Los 5 intervalos o márgenes que se han definido para cada una de las entradas del caso son:

- Rango 0. Crítico, de 0 a 200mm.
- Rango 1. Cerca, de 200 a 500mm.
- Rango 2. Medio, de 500 a 1000mm.
- Rango 3. Lejos, de 1000 a 1500mm.
- Rango 4. Sin influencia, de 1500mm hasta el valor máximo devuelto por el sensor sonar.

En el diseño de los rangos se ha tenido en cuenta que, como los comportamientos son reactivos, los obstáculos más alejados no influyen, mientras que cuanto más cercano sea un obstáculo, más hay que tenerlo en cuenta.

El segundo paso consiste en la ejecución del proceso de discretización para cada uno de los casos de la base de casos. Obviamente, en la discretización de un caso solamente están involucradas las correspondientes entradas, puesto que la salida del caso discretizado sigue siendo la misma que la del caso sin discretizar. Cuando se finaliza la discretización de un caso $\overrightarrow{CA}$ se obtiene un vector $\overrightarrow{CA}_D$, con dos partes. Por un lado, la entrada del caso $\overrightarrow{CA}^{IN}$ discretizada,

dando lugar al subvector $\overrightarrow{CA}_D^{IN}$. Por otro, la salida del caso, la cual no varía, $\overrightarrow{CA}^{OUT}$. Si la entrada del caso incluye la lectura de los n sensores sonar s_i, el caso discretizado $\overrightarrow{CA}_D^{IN}$ consistirá en el número de rango r_i al que pertenece cada s_i junto con la misma salida que el caso sin discretizar:

$$\overrightarrow{CA}_D = [\overrightarrow{CA}_D^{IN}, \overrightarrow{CA}^{OUT}] = [r_1, r_2, \ldots, r_n, v_t, v_r] \tag{3.19}$$

Por ejemplo, si el número de sensores del sistema es $n = 4$ y un caso almacenado en la base de casos es $\overrightarrow{CA} = [150, 300, 400, 1100, 85.4, 102.3]$, el caso discretizado correspondiente que se almacenaría en la base sería $\overrightarrow{CA}_D = [0, 1, 1, 3, 85.4, 102.3]$.

Una vez discretizada la base de casos, se compara el caso actual con la base de casos para recuperar el mejor de todos los almacenados. Antes de realizar esta comparación se discretiza el caso nuevo según el proceso comentado. De este modo, ya se puede comparar el caso nuevo discretizado con todos los almacenados en la base de casos discretizada. La comparación hay que efectuarla de acuerdo a alguna medida de similitud que nos proporcione un criterio cuantitativo para seleccionar el mejor caso. Nuestra técnica se encuadra dentro de los métodos de vecindad [Wat95], habitualmente empleados. Estos métodos establecen la similitud entre los casos almacenados y el nuevo caso en base a una determinada función distancia que tiene en cuenta la ponderación de cada una de las entradas. En nuestro sistema nos basamos en la distancia de Manhattan discretizada para evaluar la similitud entre casos. Sea $\overrightarrow{CA}^{new} = [s_1^{new}, s_2^{new}, \ldots, s_n^{new}, ??, ??]$ el caso que representa una situación nueva en el sistema, siendo $\overrightarrow{CA}_D^{new} = [r_1^{new}, r_2^{new}, \ldots, r_n^{new}, ??, ??]$ el mismo caso discretizado. En ambos, la salida es indeterminada, pues dependerá del mejor caso de la base de casos, seleccionado durante esta fase de recuperación. Sea también $\overrightarrow{CA}_D^i = [r_1^i, r_2^i, \ldots, r_n^i, v_t^i, v_r^i]$ un caso almacenado en la base de casos discretizada y $\overrightarrow{W} = [w_1, w_2, \ldots, w_n]$ el vector de pesos que pondera cada una de las entradas de los casos. Por último, recordemos que $\overrightarrow{NR} = [nr_1, nr_2, \ldots, nr_n] = [5, 5, \ldots, 5]$ es el vector con el número de rangos de cada sensor de entrada. La distancia Manhattan discretizada entre $\overrightarrow{CA}_D^{new}$ y $\overrightarrow{CA}_D^i$ se define como:

$$D(\overrightarrow{CA}_D^{new}, \overrightarrow{CA}_D^i) = \frac{\sum_{k=1}^{n} w_k \times \left[\frac{|r_k^{new} - r_k^i|}{nr_k - 1}\right]}{\sum_{k=1}^{n} w_k} \tag{3.20}$$

Cuanto mayor sea el peso w_i de un determinado sensor de entrada i, mayor será su influencia en el proceso de recuperación. En esta Tesis, como no se pretende favorecer ninguna orientación respecto de las demás, todos los pesos son unitarios. Por lo tanto, el vector de pesos será $\overrightarrow{W} = [w_1, w_2, \ldots, w_n] = [1.0, 1.0, \ldots, 1.0]$. Aun así, se ha considerado la posibilidad de ponderar las distintas entradas por si en un futuro se desea modificar el sistema o hacer uso de esta facilidad. Debido a que ya está implementado el mecanismo, el conseguir darle más importancia a alguna entrada requeriría de un esfuerzo mínimo por parte del diseñador.

La distancia, definida como en la Ecuación (3.20), es 0 cuando existe máxima similitud entre $\overrightarrow{CA}_D^{new}$ y $\overrightarrow{CA}_D^{i}$. Y será 1.0 cuando la similitud sea mínima. Por lo tanto, el mejor caso será aquél cuya distancia al caso nuevo sea mínima. Este caso será el recuperado por el sistema para intentar solucionar la situación planteada por el nuevo caso. Si la base de casos consta de M casos, el caso recuperado $\overrightarrow{CA}^{recup}$ será aquél que cumpla que:

$$D(\overrightarrow{CA}_D^{new}, \overrightarrow{CA}_D^{recup}) = \min_{1 \leq k \leq M} D(\overrightarrow{CA}_D^{new}, \overrightarrow{CA}_D^{k}) \tag{3.21}$$

3.2.2.2 Adaptación del caso: Reusar Esta fase se basa en emplear la información del caso recuperado para solucionar el problema actual. Cuando el caso recuperado de la base de datos no cuadra con el caso nuevo, la solución propuesta por el caso recuperado debe ser adaptada a la nueva situación. Existen tres aproximaciones principales al problema de la adaptación de la solución dada por un caso de la base de casos [Kol93]:

- Adaptación nula. La solución devuelta por el caso recuperado se toma como solución del nuevo caso. Es una estrategia válida en sistemas que requieren de acciones simples.
- Adaptación estructural. Se aplica el proceso de adaptación a la solución devuelta por el caso recuperado. La solución del caso almacenado en la base de casos no es directamente la solución adoptada por el nuevo caso, sino que esa solución es modificada de algún modo para obtener la que se aplica al caso nuevo.
- Adaptación derivada. En vez de usar la solución devuelta por el caso recuperado, se utiliza un modelo analítico para obtener la solución del nuevo caso.

El proceso de adaptación usado en esta Tesis para los comportamientos es el siguiente. Si la distancia de Manhattan discretizada entre el caso nuevo, $\overrightarrow{CA}^{new}$, y el caso recuperado de la base de casos, $\overrightarrow{CA}^{recup}$, es menor o igual que un umbral U_{reusar}, se adopta como solución del nuevo caso la solución propuesta por el caso recuperado:

$$\text{Si } D(\overrightarrow{CA}_D^{new}, \overrightarrow{CA}_D^{recup}) \leq U_{reusar},\ \overrightarrow{CA}^{newOUT} = \overrightarrow{CA}^{recupOUT} = [v_t^{recup}, v_r^{recup}] \tag{3.22}$$

En el caso de que sea mayor que el umbral U_{reusar}, nos basamos en una adaptación derivada. Se adopta como solución del caso la obtenida, para la entrada del caso nuevo, por el modelo analítico del comportamiento correspondiente analizado en la Sección 2. Nos basamos en este tipo de adaptación porque se ha demostrado que el modelo analítico implementado funciona a la hora de que el robot realice una determinada tarea, con las limitaciones inherentes al propio método. Además, es una solución puramente reactiva que proporciona la acción a ejecutar ante una determinada entrada sensorial. Es importante hacer notar que el hecho de emplear el modelo analítico para solucionar nuevas situaciones encontradas por el agente no implica que

al usar la filosofía del *CBR* se consigue un movimiento del robot similar al descrito para los comportamientos siguiendo dicho modelo. Esto es así porque el agente únicamente empleará la adaptación en situaciones nuevas, lo que no ocurrirá, ni mucho menos, a lo largo de toda la navegación, sino tan solo en algunos momentos. Se ha probado experimentalmente que un umbral U_{reusar} en torno a 0.1 es un valor razonable para considerar que el robot ha encontrado una nueva situación.

3.2.2.3 Evaluación del caso: Revisar La misión de esta tercera fase del ciclo *CBR* es evaluar la validez de la solución adaptada que el sistema obtiene como solución para el caso nuevo tras aplicar la fase anterior. Es posible que el proceso de evaluación haga que se requiera una adaptación adicional de la solución adaptada, denominada reparación. La fase de evaluación puede realizarse mediante diferentes estrategias. Entre otras, las más interesantes son la supervisión de un humano experto, la simulación automática con algún algoritmo que evalúe la bondad de la solución adaptada y la observación de los resultados obtenidos tras aplicar la solución adaptada.

En nuestro sistema, en el caso de que la solución adaptada se corresponda con la solución del caso recuperado, no tiene sentido evaluar la calidad de la solución. Y esto es así porque se supone que si el caso estaba almacenado en la base de casos ha sido previamente considerado positivo en la implementación del comportamiento tratado. Del mismo modo, si la solución recuperada de la base de casos es adaptada según el modelo analítico, se supone que esa nueva solución adaptada es adecuada para resolver la nueva situación detectada por el robot. Esta decisión de diseño se toma porque el modelo analítico ha demostrado su validez en la consecución del comportamiento deseado (ver Sección 2). Por lo tanto, las soluciones adaptadas con el modelo analítico son evaluadas positivamente, almacenándose esta nueva experiencia en la base de casos, tal y como se analiza posteriormente. Se podría haber considerado un sistema automático de evaluación de la eficiencia de un caso en cuanto a la corrección de su solución para implementar un determinado comportamiento. Sin embargo, no se ha hecho porque, bien mediante el entrenamiento inicial, bien mediante el modelo analítico usado para adaptar los casos cuando se detecta una nueva situación, la solución almacenada en la base de casos implementa el comportamiento deseado.

3.2.2.4 Aprendizaje del caso: Retener La última fase del ciclo *CBR* es la de retener. Durante esta fase se incorporan a la base de casos los casos adaptados que han sido evaluados positivamente, para así ir aumentando el conocimiento y la experiencia del sistema para resolver problemas futuros. En nuestro caso, se considera que los casos adaptados están evaluados positivamente, por lo que pasan a formar parte de la base de casos. Esta fase de aprendizaje es una de las más importantes dentro de un sistema *CBR*. Existen dos tipos de aprendizaje diferenciados en todo sistema *CBR*:

1. Aprendizaje por observación. Se produce cuando la base de casos del sistema es iniciada con un conjunto de casos obtenidos de situaciones reales por un experto o por observación

directa. El aprendizaje efectuado en un sistema *CBR* es preferible al llevado a cabo por una red neuronal porque, gracias a este tipo de aprendizaje, se puede iniciar la base de casos con un conocimiento adquirido previamente [Kru03]. Así, se puede adaptar el comportamiento a la situación específica que se desee.

2. Aprendizaje por experiencia. Se produce cuando la base de casos del sistema es incrementada al incorporar un caso adaptado que da respuesta a una nueva situación detectada, realizándose tras cada ciclo *CBR*.

Aunque se pueden usar de forma independiente, es una buena política el emplearlos de forma complementaria en el diseño de un sistema *CBR*. Así, ambas aproximaciones son combinadas para implementar cada uno de los tres comportamientos (Seguir Pared, Seguir Pasillo y Cruzar Puerta), siguiendo la filosofía del *CBR*.

En primer lugar se lleva a cabo un aprendizaje por observación. Durante esta etapa *off line*, un humano guía el robot por medio de un teclado para alcanzar el comportamiento deseado. Por ejemplo, si se deseara aprender el comportamiento Seguir Pared, el humano guiaría al robot de forma paralela a la pared a la distancia prefijada a la que se deseara seguir. A medida que el robot navega, se almacena en la base de casos cada uno de los casos que se van capturando. Si recordamos, un caso consta de la entrada sensorial que los sonar producen junto con las velocidades de traslación y rotación que el robot aplica a sus motores. Bajo la suposición de que el humano está desarrollando correctamente el comportamiento especificado, todos los casos se consideran positivos en cuanto a la ampliación del conocimiento que se posee. Por este motivo, es recomendable que el humano que guía el agente lo haga concienzudamente y con cuidado para no almacenar en el sistema información que no es válida. Aun así, este hecho no es tan crítico como puede parecer, puesto que si se posee suficiente experiencia en el trabajo con agentes autónomos móviles, desarrollar tareas simples como Seguir Pared, Seguir Pasillo y Cruzar Puerta, no es difícil.

En vez de usar un teclado para guiar al robot manualmente, también podría haberse usado un joystick. Este pequeño cambio no introduciría ninguna modificación en el aprendizaje y, por ende, en el funcionamiento del sistema, puesto que es el humano el que en última instancia determina el conocimiento almacenado en la base de casos tras el aprendizaje por observación. Sí supondría una variación algo mayor el efectuar un aprendizaje automático por medio de algún modelo analítico. Como en esta Tesis también se han implementado los tres comportamientos según un modelo analítico, fue una opción considerada. Sin embargo, esta elección presenta algunos inconvenientes: i) no se incluye el punto de vista subjetivo e intuitivo del humano; ii) no existe margen de maniobra ante situaciones como mínimos locales, oscilaciones, malfuncionamientos, etc, puesto que el aprendizaje es automático; y iii) no se manejan restricciones cinemáticas y dinámicas al mover el agente, algo que sí se incorpora al aprender del movimiento de un humano. Estas razones han hecho que en el sistema se haya decidido realizar un aprendizaje por observación basado en un operador humano.

Otra posibilidad de aprendizaje por observación es aquélla que combina el guiado por parte de un operador humano con la operación autónoma del robot mediante un modelo analítico. Así, se pueden evitar simultáneamente los inconvenientes que cada uno de los dos aprendizajes tiene por separado. Combinando ambos esquemas se consiguen eliminar los casos debidos a un guiado incorrecto por parte de un humano, así como los problemas derivados de los modelos analíticos, como por ejemplo las oscilaciones, ya que el humano participa del aprendizaje. En esta Tesis también se propone un método de aprendizaje asistido que combina linealmente los comandos de movimiento que un humano introduce mientras guía el robot mediante un joystick con los comandos de movimiento que el agente intenta aplicar autónomamente. Esta técnica se presenta en la Sección 4, y ha sido de utilidad para que usuarios sin conocimiento previo acerca del sistema puedan guiar el robot, aumentando la diversidad del aprendizaje. El objetivo del método es poder aprender de ambos, del humano y del movimiento autónomo, para mejorar el aprendizaje. La combinación se realiza en función de una medida de eficiencia definida.

El aprendizaje por experiencia es usado en cualquier sistema *CBR* para incrementar el conocimiento que posee la base de casos, mediante la incorporación de los casos adaptados en respuesta a una nueva situación detectada. Este aprendizaje puede ser positivo o negativo. En el primer caso, la solución adaptada es evaluada positivamente durante la fase Revisar, por lo que se añade la nueva experiencia a la base de casos. En el segundo, la evaluación durante la fase Revisar es negativa. Por lo tanto, hay que incluir en el sistema mecanismos para no considerar los casos fallidos en el futuro.

En esta Tesis el aprendizaje por experiencia siempre es positivo. A medida que el agente navega va aprendiendo los comportamientos gracias a su propia experiencia. Cada vez que un caso es adaptado, la nueva situación es almacenada en la base de casos para aumentar el conocimiento del sistema. Según se comentó al analizar la fase Revisar, las soluciones adaptadas se consideran adecuadas para resolver la nueva situación porque están basadas en un modelo analítico del cual se ha demostrado su viabilidad, de ahí que el aprendizaje sea siempre positivo. Por lo tanto, todos los casos adaptados son incorporados automáticamente a la base de casos.

Al igual que para los comportamientos implementados según el modelo analítico, los desarrollados según la filosofía del *CBR* también incluyen mecanismos de evitación de obstáculos. En aquel caso la evitación de obstáculos se consigue mediante la incorporación de una fuerza incluida para tal efecto en los tres comportamientos. Cuando se emplea el *CBR*, la evitación de obstáculos se alcanza tanto con el aprendizaje por observación como con el aprendizaje por experiencia. Durante la fase de aprendizaje por observación el humano entrena al agente para implementar un determinado comportamiento. Durante este entrenamiento se está teniendo en cuenta de forma implícita la evitación de los obstáculos para que el agente no colisione. Por otro lado, cuando se detecta una nueva situación se adapta el caso de entrada mediante el modelo analítico, aprendiendo así el robot de su propia experiencia. Por lo tanto, como se hace uso del modelo analítico se está teniendo en cuenta de manera explícita, por medio de la fuerza correspondiente, la evitación de los obstáculos. La posibilidad de que el robot se acerque peligrosamente a los obstáculos está considerada en el sistema, puesto que se trata de un escenario

para el cual no ha sido entrenado. Por lo tanto, este conocimiento no está almacenado en la base de casos. Ante dicha situación el sistema adaptaría el caso con el modelo analítico, forzando al robot a evitar los obstáculos.

3.2.3 Clasificación de la base de casos

En esta Tesis se ha optado por implementar una estructura plana de la base de casos. Se ha hecho de este modo por la sencillez de su implementación y para asegurar que se explora todo el espacio de búsqueda durante la fase Recuperar y así obtener el mejor caso para la situación actual. Sin embargo, el tiempo de recuperación del mejor caso se incrementa con el tamaño de la base de casos. Si progresivamente se van añadiendo casos a la base de casos durante la fase Retener gracias al aprendizaje por experiencia, también se va incrementando el tiempo necesario para obtener el mejor caso. Por lo tanto, si se desea que la reactividad del sistema no se vea mermada, no conviene que el tamaño de la base de casos crezca indiscriminadamente.

La reducción del tiempo invertido en la fase Recuperar es uno de los motivos principales por el que se ha implementado una clasificación de la base de casos. Este proceso se puede realizar *off line* en cualquier momento y a cualquier base de casos almacenada. La base de casos se clasifica con un algoritmo *MaxMin* [Mar93]. Se ha escogido este algoritmo heurístico por las siguiente razones:

- Es sencillo de implementar.
- Da la posibilidad de escoger entre tomar al azar el elemento inicial del algoritmo o escogerlo por cualquier otro criterio, como puede ser la distancia al resto de elementos.
- El número final de clases no se define *a priori*, sino que depende de los elementos involucrados en la clasificación.
- Únicamente depende de un parámetro, la distancia $Dist\_MaxMin$. Es la distancia mínima que debe existir entre las clases.

La métrica empleada para evaluar la similitud entre los casos de la base de casos es la distancia Manhattan, pero sin discretizar. Durante la clasificación se consideran las entradas (lectura de los sensores sonar), para evaluar la distancia entre casos. Las salidas no se tienen en cuenta porque los casos se deben agrupar en función de la descripción del problema que representan, no en función de la solución que aportan. Al finalizar la clasificación, los M casos de la base de casos se clasifican en S clases. Cada clase S_i tiene un prototipo $\overrightarrow{PT}_i$, con unas entradas y unas salidas. Las entradas del prototipo de una clase consisten en el promedio de las entradas de los casos que pertenecen a esa clase. En cuanto a las salidas, serán también el promedio de las salidas de los casos pertenecientes a la clase, aunque no hayan participado en el proceso de clasificación. Formalmente:

$$\overrightarrow{PT}_i = [\overrightarrow{PT}_i^{IN}, \overrightarrow{PT}_i^{OUT}] = [\sum_{\substack{k=1 \\ \forall \overrightarrow{CA}_k \in S_i}}^{M} \frac{\overrightarrow{CA}_k^{IN}}{ns_i}, \sum_{\substack{k=1 \\ \forall \overrightarrow{CA}_k \in S_i}}^{M} \frac{\overrightarrow{CA}_k^{OUT}}{ns_i}] \tag{3.23}$$

donde ns_i es el número de casos pertenecientes a la clase S_i. Al finalizar la clasificación se obtiene una base de casos de menor tamaño. Los casos almacenados serán los prototipos de las distintas clases. Estos prototipos serán los casos con los que se comparan los casos nuevos durante la navegación. Es necesario hacer notar que los prototipos no se corresponden con ningún caso capturado durante el aprendizaje, sino que consisten en el promediado de los casos que pertenecen a la clase que representa.

Además de reducir el tiempo empleado para recuperar el mejor caso, al clasificar una base de casos también se consiguen otras ventajas. Como durante el proceso de aprendizaje no se ha impuesto ninguna restricción en cuanto a la cercanía de los casos capturados, puede ocurrir que en la base de casos estén almacenados varios casos muy cercanos en el espacio. Estos casos tan cercanos en el entorno tienen, con toda probabilidad, una distancia muy cercana entre ellos. Por lo tanto, sería redundante su presencia simultánea en la base de casos. Mediante la clasificación se consiguen agrupar en la misma clase, promediando sus soluciones. Así mismo, dos casos capturados en lugares alejados del entorno también pueden representar situaciones similares. Mediante la clasificación también serían agrupados en la misma clase, aun estando alejados espacialmente. De este modo, y gracias al proceso de clasificación, casos que describen situaciones muy parecidas se agrupan en una misma clase. Este hecho también tiene la ventaja de evitar oscilaciones entre casos que se reflejan en el comportamiento emergente. La similitud entre las situaciones puede provenir, bien por una cercanía en el espacio entre dos casos, bien porque dos casos representan situaciones similares aun estando alejados.

El único parámetro que influye en el proceso de clasificación es la distancia mínima entre clases, $Dist\_MaxMin$. Obviamente, cuanto mayor sea este parámetro, menor será el número de clases obtenidas. Igualmente, cuanto menor sea, mayor será el número de clases. A la hora de elegir este parámetro existe un compromiso entre el número de clases generadas y la calidad de la agrupación que se consigue. Si el número de clases en que se clasifica la base de casos es muy pequeño, se corre el riesgo de que casos distintos y, por tanto, que describen situaciones diferentes, se clasifiquen como pertenecientes a la misma clase. Por el contrario, si la reducción de la base de casos inicial no es lo suficientemente intensa, se pueden obtener muchas clases donde casos similares pertenezcan a clases distintas. Para nuestro sistema se ha comprobado experimentalmente que un valor del parámetro $Dist\_MaxMin$ entre 500 y 1000 conduce a una base de casos clasificada que funciona adecuadamente para nuestros objetivos.

La clasificación de una base de casos no es un proceso estático. Como durante la navegación del robot se van adquiriendo nuevos casos al aprender el agente de su propia experiencia, el conocimiento que posee el sistema se va ampliando poco a poco. Sin embargo, durante una prueba no se puede emplear la experiencia adquirida a lo largo de la misma. Para hacer uso de esa nueva experiencia habría que clasificar de nuevo *off line* la base de casos ampliada. En

una nueva prueba ya sí que se incorpora el conocimiento adquirido durante la prueba anterior y además ese conocimiento estaría disponible, pues se habría empleado en una nueva clasificación de la base de casos.

Comentar en último lugar que, al margen del algoritmo *MaxMin*, se han considerado otras opciones para clasificar la base de casos. Así, se pensó en un algoritmo que clasificara la base de casos en un número determinado de clases. Algoritmos de este tipo son, entre otros, el *k medias* [Mar93] y la red de Kohonen [Koh82]. De estos dos algoritmos se ha implementado una red de Kohonen. Los resultados obtenidos no han sido satisfactorios porque, al ser el número de clases conocido *a priori*, la clasificación no es adecuada. Dependiendo del número de casos de la base de casos el número de clases prefijado está dimensionado por exceso o por defecto, dando lugar a una agrupación de casos que no es correcta. Conseguir casar el número de clases con el número de casos de la base de casos es difícil. Además, dota de una rigidez innecesaria al sistema, puesto que cualquier incorporación a la base de casos ya haría que el ajuste realizado no fuera adecuado. Por otro lado, el algoritmo de la red de Kohonen es mucho más parametrizado que el algoritmo *MaxMin*, lo que exigiría un estudio más profundo en relación al valor de los parámetros a usar en el sistema.

3.3 Los comportamientos

Una vez analizado el proceso de diseño de un comportamiento genérico, pasamos a describir cada uno de los tres comportamientos de nuestro sistema, Seguir Pared, Seguir Pasillo y Cruzar Puerta. Para los tres, la representación y almacenamiento de los casos es similar. Se emplea una estructura plana de la base de casos. Cada caso consta de las entradas sensoriales y las salidas en forma de velocidad de traslación y rotación en respuesta a esas lecturas. En el diseño se aplica el ciclo *CBR* y el algoritmo de clasificación de la base de casos. Una de las ventajas de emplear esta filosofía en el diseño es que la metodología es similar para cualquier comportamiento. Si en vez los tres comportamientos empleados en nuestro sistema se utilizaran otros, mediante el aprendizaje se podría conseguir que el robot ejecutara otra tarea. Además, se sabe en todo momento qué ha aprendido el robot gracias al análisis de los casos almacenados en la base de casos.

A continuación se presentan aspectos particulares para cada uno de los tres comportamientos. Como el proceso de diseño es similar para los tres y ha sido analizado en la Sección 3.2, no se va a comentar de nuevo todo el proceso ni se van a analizar las decisiones tomadas. Sin embargo, sí se presentan distintas pruebas que demuestran la validez del método desarrollado y su viabilidad para ejecutar la tarea para la que el robot es entrenado. Las máximas velocidades de traslación y de rotación fijadas para el robot son, respectivamente, $v_{tMAX} = 100mm/s$ y $v_{rMAX} = 15^o/s$. En todos los tests se emplean bases de datos clasificadas con una distancia $Dist\_MaxMin$ entre 500 y 1000. Así mismo, el umbral usado para adaptar los casos es $U_{reusar} = 0.1$. Los mapas métricos que se muestran únicamente se emplean para presentar gráficamente los resultados, puesto que no existe planificación al ser los comportamientos reac-

tivos. Además de poder ver el modelo de entorno generado, se presenta en gris la trayectoria recorrida por el agente. Aquellos puntos de dicha trayectoria donde los casos tuvieron que ser adaptados se representan con círculos blancos.

3.3.1 Seguir Pared

En este primer comportamiento el robot se debe desplazar de forma paralela a la pared que se encuentra a su derecha, siguiendo su contorno a una distancia determinada.

En una primera prueba se muestra el proceso de aprendizaje y su efecto a la hora de navegar realizando la tarea de seguir una pared. En la Figura 3.13 se presenta el entorno de trabajo donde se realiza un entrenamiento muy simple, lo que dota al agente de un cierto aprendizaje por la observación directa del humano que guía el robot. La trayectoria seguida se encuentra superpuesta en gris al modelo de entorno generado durante dicho aprendizaje. Podemos observar que al robot aprende a seguir de forma paralela una pared, así como a girar hacia la izquierda para seguirla cuando tiene este cambio de dirección. El número de casos almacenados en la base de datos durante este aprendizaje es de 393, clasificándose en 37 clases ($Dist\_MaxMin = 1000$). La base de datos clasificada es la empleada durante la operación autónoma del robot.

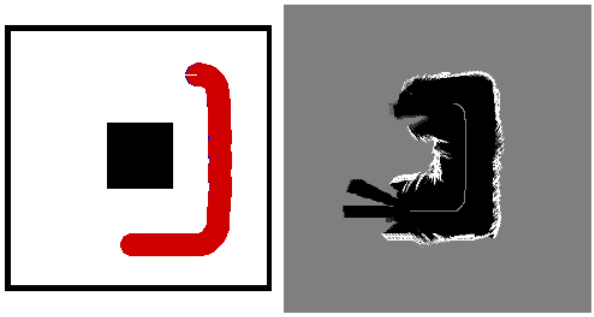

Figura 3.13: Seguir Pared con *CBR*, aprendizaje por observación.

El objetivo del agente en la Figura 3.14 es seguir la pared que se encuentra al iniciar su recorrido. El robot está situado en la parte norte de la pared. Además de dicha pared, existe otra pared dispuesta de forma perpendicular a su derecha. Partiendo del aprendizaje obtenido en el entorno de la Figura 3.13, comienza la navegación. En una primera situación se ha considerado que solamente se puede emplear el conocimiento almacenado en la base de casos, eliminando la posibilidad de usar el aprendizaje por experiencia mediante la adaptación. Es como si en en el sistema se usara una adaptación nula, aunque sabemos que en nuestro sistema se ha implementado un mecanismo de adaptación derivada basado en el modelo analítico. El hecho de probar en primer lugar sin adaptación es para mostrar los problemas que pueden surgir. Se puede observar en la Figura 3.14a que el robot empieza a seguir la pared de una forma casi paralela. Llega un punto en el que no sabe cuál es la situación que se encuentra, puesto que el entorno le exigiría girar hacia la derecha (sentido de las agujas del reloj). Sin embargo, el entrenamiento únicamente ha dotado al agente de conocimiento en relación al seguimiento paralelo de la pared

y cuando hay que ejecutar un giro hacia la izquierda. Por lo tanto, el robot no es capaz de resolver esta situación, según se aprecia perfectamente en su trayectoria recorrida.

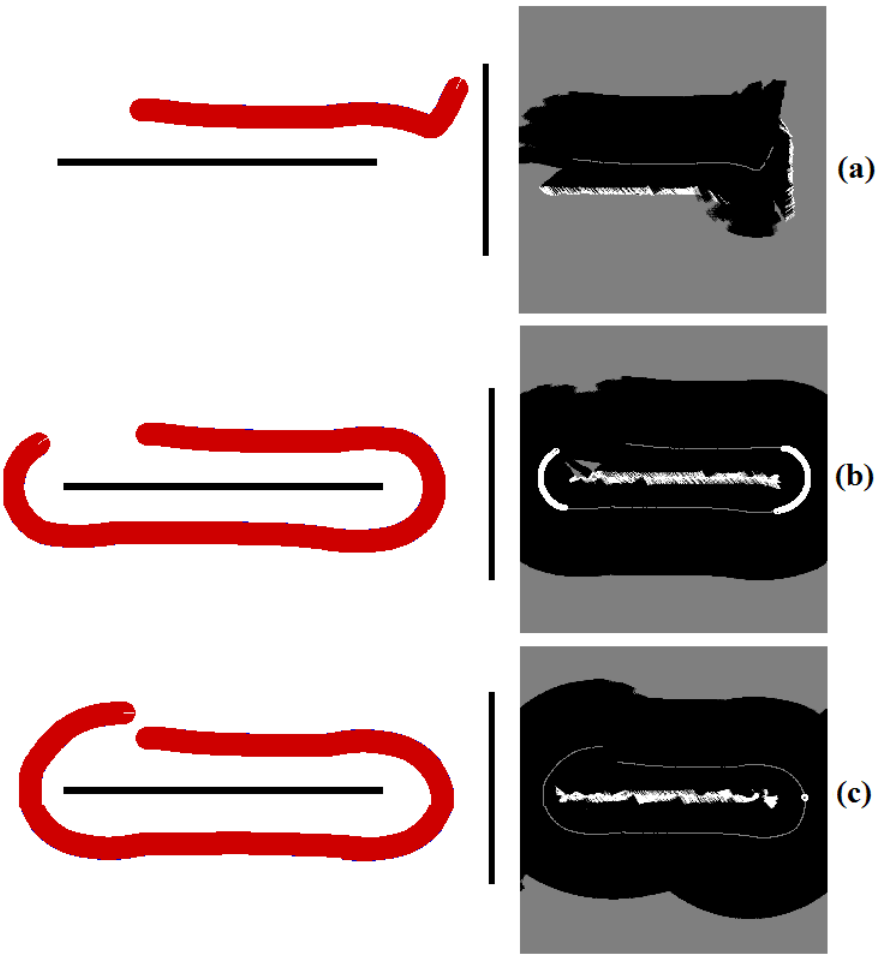

Figura 3.14: Seguir Pared con *CBR*, adaptación: (a) sin adaptación; (b) con adaptación; (c) con adaptación teniendo en cuenta el aprendizaje por experiencia en (b).

Para evitar el problema que surge si la adaptación es nula, se habilita esta posibilidad. Observamos en la Figura 3.14b que ahora sí se sigue la pared completamente. Al inicio de la navegación el agente ejecuta un movimiento casi paralelo a la pared. Cuando llega al punto en el que debe girar a la derecha, no tiene suficiente conocimiento a partir de la base de casos para decidir su acción. Sin embargo, mediante la adaptación, se produce el aprendizaje de nuevos casos porque la distancia del mejor caso almacenado en la base de casos al caso actual es mayor que el umbral U_{reusar}. En la Figura 3.14b se aprecia que la trayectoria recorrida por el agente tiene círculos blancos durante todo el giro hacia la derecha. Una vez paralelo a la pared, nuevamente se puede ver que ya no hay que adaptar los casos, porque ha sido entrenado en el entorno de la Figura 3.13. Cuando se alcanza el final oeste de la pared, se reproduce de nuevo la necesidad de girar hacia la derecha. Por este motivo, se vuelven a adaptar los casos, ampliando el conocimiento que se posee. Durante esta prueba se pone de manifiesto que el sistema ha sido entrenado para una determinada situación, por lo que no puede resolver

situaciones novedosas. Sin embargo, como el *CBR* aprende progresivamente, sí que se amplía el conocimiento almacenado. De hecho, en la Figura 3.14b se adaptan 284 casos nuevos. Si se suman estos casos a los 397 adquiridos en el entorno de la Figura 3.13, hacen un total de 677 casos. Al clasificar esta nueva base de casos se obtienen 56 clases.

Para evaluar el efecto que tiene en el sistema el aprendizaje por experiencia realizado en la Figura 3.14, se repite de nuevo la prueba. Se emplea ahora la nueva base de casos clasificada con 56 clases, que contiene el conocimiento para poder seguir una pared girando hacia la izquierda y hacia la derecha. El resultado se muestra en la Figura 3.14c. Se observa que el agente sigue la pared sin prácticamente necesitar adaptar nuevos casos, tomando la solución almacenada en la base de casos para girar hacia la derecha en vez de adaptar esta situación con el modelo analítico. De hecho, únicamente se adquieren 4 nuevos casos. Es debido a que el aprendizaje es un proceso dinámico, por lo que siempre existe la posibilidad de incrementar la base de casos con nuevas situaciones y así poder mejorar el funcionamiento del sistema en el futuro.

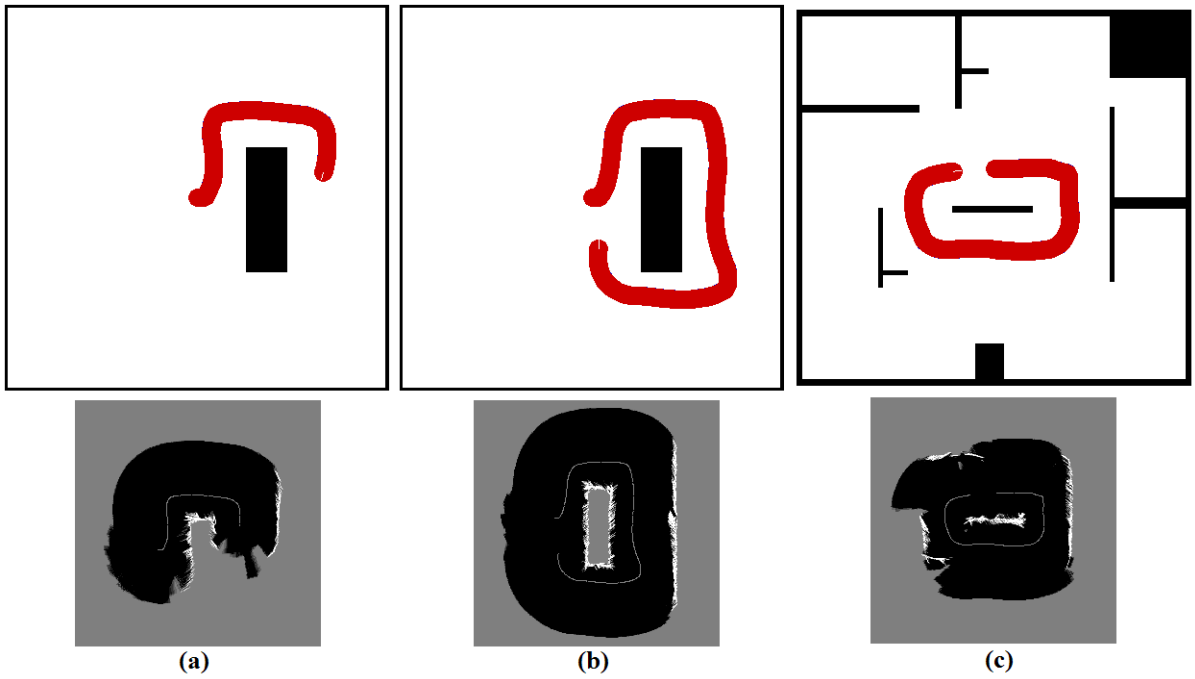

Figura 3.15: Seguir Pared con *CBR*, seguimiento de obstáculo: (a) aprendizaje por observación; (b) operación en el mismo entorno que (a); (c) operación en un entorno distinto de (a).

Uno de los inconvenientes del comportamiento Seguir Pared es su imposibilidad de explorar de forma total un entorno cuando se queda enganchado a un obstáculo dando vueltas a su alrededor, tal y como se analizó al presentar esta situación en la implementación según el modelo analítico. No por el hecho de usar el *CBR* el robot va a dejar de exhibir este comportamiento, como se muestra en la Figura 3.15. En este caso se ha entrenado al robot para seguir la pared y girar hacia la derecha, es decir, en el sentido de las agujas del reloj (Figura 3.15a). La base de casos resultante de la fase de aprendizaje por observación directa de un humano consta de 350 casos. Al clasificarla se obtienen 49 clases, las cuales conformarán la base de casos durante

la operación del agente. En las Figuras 3.15b y c se observa que el obstáculo que el robot se encuentra es recorrido indefinidamente, imposibilitando al agente para explorar todo el entorno. También debemos comentar que, a pesar de que el entrenamiento ha sido breve, el agente es capaz de ejecutar la tarea de forma adecuada, siguiendo el contorno del obstáculo.

Si bien el entorno de la Figura 3.15b es el mismo usado durante la fase de entrenamiento, el de la Figura 3.15c es totalmente distinto. Por lo tanto, se cumple una de las premisas fundamentales del sistema *CBR*, la posibilidad de trabajar con el conocimiento almacenado en entornos distintos del empleado para adquirir ese conocimiento. Es evidente, al igual que se comentó al tratar de la implementación según el modelo analítico, que quizás el comportamiento Seguir Pared no es el más adecuado en determinadas situaciones, como las de la Figura 3.15, ya que no se alcanza el objetivo de la exploración completa. Por lo tanto, con un único comportamiento no es suficiente, y se necesita del concurso de algún otro para conseguir la adaptación a la configuración del entorno.

Se presenta en último lugar una comparativa del modelo analítico y del basado en el *CBR* cuando el robot opera siguiendo una pared (Figura 3.16). La Figura 3.16a muestra la operación del agente mediante el método analítico, mientras que la Figura 3.16b presenta el correspondiente al *CBR*. El entrenamiento seguido en este último caso es el de la Figura 3.13. El robot parte en ambos casos desde la parte superior de la pared que está en mitad del entorno. El comportamiento Seguir Pared hace que el agente explore la parte superior de esta pared inicial, para después continuar por la pared este, la pared norte, la pared oeste, la pared sur, nuevamente la pared este y la parte inferior de la pared central. Una vez recorrido el entorno de esta manera, se continuaría la operación del mismo modo. Podemos observar que el modelo analítico provoca una operación correcta del robot, pero con las oscilaciones típicas de la aproximaciones basadas en los campos de potencial (Figura 3.16a). Sin embargo, al operar con el *CBR*, se reducen las oscilaciones, consiguiendo un movimiento más suave. Se aprecia la forma tan paralela en la que el agente sigue en este caso la pared (Figura 3.16b). Gracias al aprendizaje por observación el sistema tiene en consideración los aspectos dinámicos y cinemáticos, alcanzando un mejor funcionamiento global.

En la Figura 3.16c se presenta el mapa métrico construido durante la operación del robot con el *CBR*. La trayectoria seguida se encuentra superpuesta en gris. Se observan los círculos blancos que determinan los puntos en los que se han adaptado las situaciones no almacenadas en la base de casos. Como el entrenamiento solamente considera los giros hacia la izquierda (Figura 3.13), los giros hacia la derecha no están incluidos, de ahí la adaptación. Este aprendizaje por experiencia podría ser usado en el futuro, incrementando el conocimiento del sistema, como se ha puesto de manifiesto en el entorno de la Figura 3.14.

3.3.2 Seguir Pasillo

Al contrario que en el comportamiento Seguir Pared, cuando el robot sigue un pasillo no debe seguir de forma paralela una pared, sino recorrerlo por su centro.

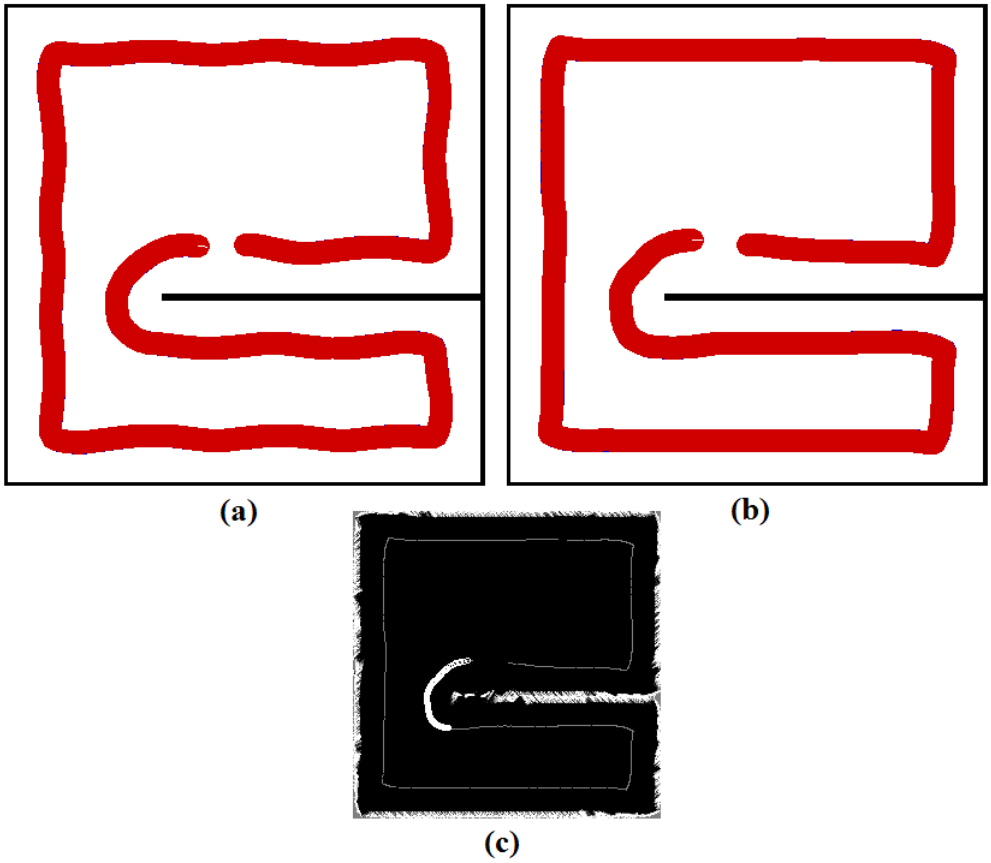

Figura 3.16: Seguir Pared, comparación entre el modelo analítico y el basado en el *CBR*: (a) funcionamiento con el modelo analítico; (b) funcionamiento con el *CBR*; (c) trayectoria recorrida por el robot en (b).

En primer lugar se presenta en la Figura 3.17 el entrenamiento realizado en un pasillo recto. Durante esta fase de aprendizaje por observación un humano ha guiado el robot, tratando de centrarlo en el pasillo. El agente parte centrado en el pasillo, por lo que si se guía el robot de forma recta será suficiente para adquirir los casos necesarios que implementen el comportamiento. Sin embargo, se aprecia en la Figura 3.17 que el humano no ha sido capaz de dirigir el agente de forma recta, aunque esto no supone ningún problema a la hora de operar con este entrenamiento. El número de casos capturados es de 145, lo cual destaca por su brevedad. Al clasificar la base de casos con $Dist\_MaxMin = 1000$ se obtienen tan solo 6 clases.

La Figura 3.18 muestra la operación del robot en dos pasillos de anchura diferente, considerando el entrenamiento efectuado en la Figura 3.17. El pasillo de la Figura 3.18a es el mismo usado para el entrenamiento. Sin embargo, se aprecia que, partiendo desde su centro, se sigue de una forma rectilínea. Gracias al aprendizaje por observación, el proceso de clasificación y los mecanismos involucrados en todo sistema *CBR*, se consigue una operación adecuada del robot. A pesar de que el humano no guió de forma totalmente recta al agente, éste sí es capaz de ejecutar el movimiento de forma apropiada gracias a la adaptación, rescatando de la base de casos el conocimiento que más le conviene en cada situación. Al final de la trayectoria aparecen casos

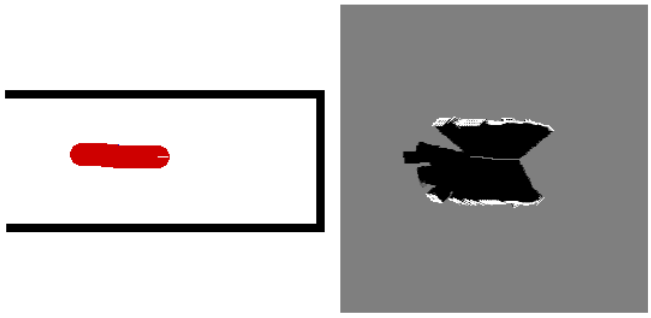

Figura 3.17: Seguir Pasillo con *CBR*, aprendizaje por observación.

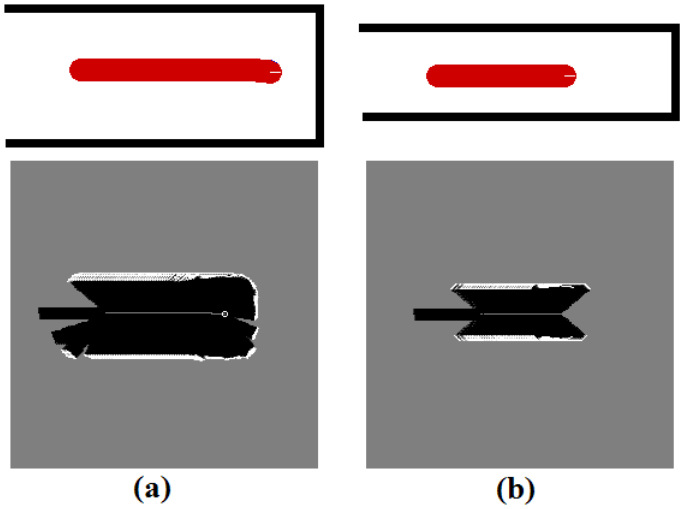

Figura 3.18: Seguir Pasillo con *CBR*, diferentes anchuras: (a) entorno simulado 1; (b) entorno simulado 2.

adaptados porque el robot detecta la pared frontal, situación para la que no ha sido entrenado. El pasillo de la Figura 3.18b ya no es el mismo que el empleado durante el entrenamiento. La principal diferencia es su anchura, pues es inferior. Aun así, el robot sigue rectilíneamente el pasillo por su centro. Para ello se apoya en el conocimiento almacenado en la base de casos clasificada, consiguiendo una ejecución adecuada del comportamiento.

En una segunda prueba se pone de manifiesto la adaptación y el aprendizaje por experiencia que el agente adquiere mientras navega (Figura 3.19). Se parte del conocimiento almacenado en la base de casos tras el entrenamiento de la Figura 3.17 que, si recordamos, se clasifica en tan solo 6 clases. Para observar la diferente operación del robot ante distintos supuestos, se considera en primer lugar únicamente la base de casos, no permitiendo la adaptación. La trayectoria recorrida por el robot es rectilínea, como se observa en la Figura 3.19a. Sin embargo, cuando la pared frontal al robot es detectada, el movimiento no es el adecuado. Esto es así porque la base de casos solamente almacena situaciones en las que el robot debe ir aproximadamente en línea recta. Se le permite entonces al agente adaptar los casos cuando se considere pertinente. Así, se repite la prueba, siguiendo el robot la trayectoria de la Figura 3.19b. Los círculos blancos sobre el mapa métrico ilustran los puntos donde el robot adapta casos, situación que se produce

al detectar la pared para intentar centrarse en el segundo pasillo. Los casos adaptados son 69. Estos 69 casos se añaden a los 145 adquiridos en el entrenamiento inicial, obteniéndose un total de 214. Al ser clasificados la base de casos útil contiene 19 clases.

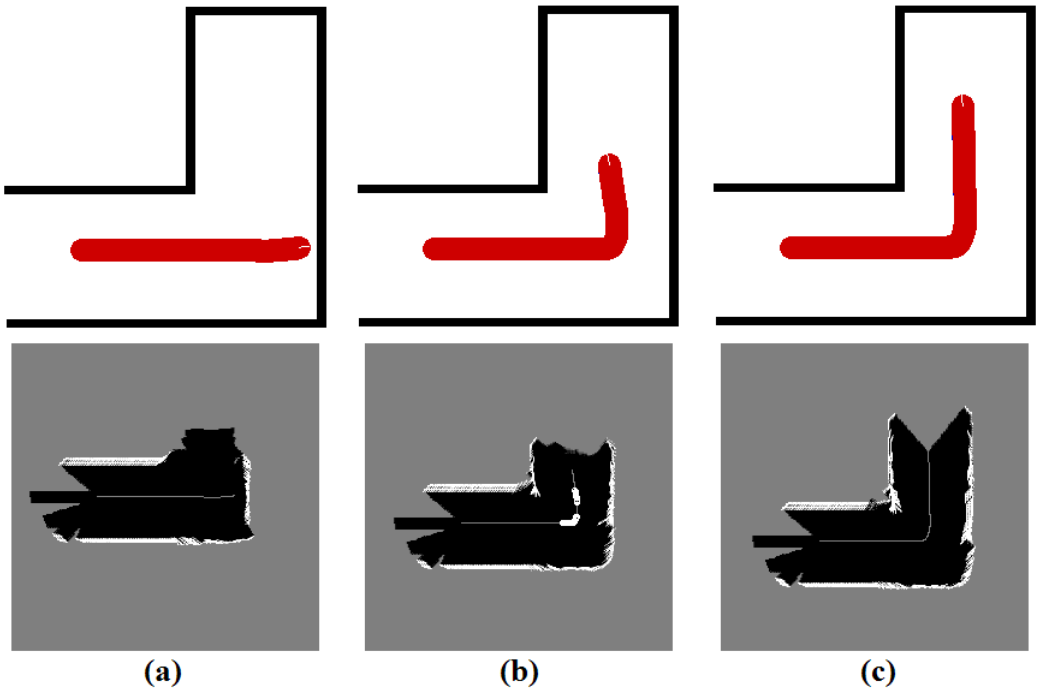

Figura 3.19: Seguir Pasillo con *CBR*, adaptación: (a) sin adaptación; (b) con adaptación; (c) con adaptación teniendo en cuenta el aprendizaje por experiencia en (b).

Se repite la prueba con esta nueva base de casos, dando lugar al movimiento presentado en la Figura 3.19c. Es destacable que el robot sigue ambos pasillos de una forma esencialmente rectilínea. Además, es capaz de girar 90º al detectar la pared, para después centrarse en el segundo pasillo. Este último hecho se consigue gracias al aprendizaje por experiencia almacenado en la prueba anterior, y que se emplea de forma efectiva en ésta. También debemos hacer notar que ya no se adaptan casos en esta prueba, puesto que la nueva base de casos ya considera las situaciones que el agente se encuentra mientras navega. Al igual que se hizo al comentar el modelo analítico del comportamiento Seguir Pasillo, el entorno de la Figura 3.19 nos permite hacer una reflexión. Quizás, al detectar el robot una pared frontal y girar para centrarse en el segundo pasillo, el comportamiento más adecuado no sea el Seguir Pasillo. Al finalizar el seguimiento del primer pasillo desaparece la pared izquierda del pasillo, siendo en esta situación el comportamiento Seguir Pared el más apropiado. E incluso al girar los 90º y disponerse para navegar por el centro del segundo pasillo, el robot ve una puerta. Por todo ello, parece claro que solamente con el comportamiento Seguir Pasillo no se alcanza una navegación eficiente para explorar de forma total un entorno. Aun así, gracias a la filosofía del *CBR* se puede conseguir que el robot aprenda estas situaciones, alcanzando un funcionamiento apropiado del robot durante su operación. Por otro lado, si comparamos la trayectoria del agente en la Figura 3.19c con la que sigue cuando emplea el modelo analítico (Figura 3.8c), se puede observar la mejora cualitativa

obtenida al emplear el método basado en el *CBR*.

En otra prueba se pretende poner de manifiesto la operación el agente en pasillos donde existen obstáculos que el robot debe evitar. Se han realizado dos entrenamientos, durante los cuales un humano guía al robot. El primero se efectúa en el entorno de la Figura 3.21a, donde el robot recorre el pasillo esquivando el obstáculo en su parte superior (Figura 3.20a). El número de casos almacenados en la base de casos durante este aprendizaje asciende a 328, clasificándose en 20 clases. El segundo se lleva a cabo en el entorno de la Figura 3.21c. En este caso el agente recorre el pasillo sin chocar con el obstáculo en su parte inferior (Figura 3.20b). En este segundo entrenamiento se capturan 523 casos que dan lugar a 22 clases.

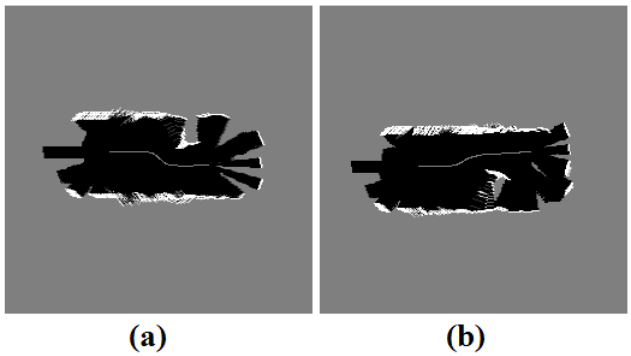

Figura 3.20: Seguir Pasillo con *CBR*, aprendizaje por observación: (a) obstáculo en la parte superior; (b) obstáculo en la parte inferior.

Con los dos aprendizajes por observación realizados se efectúan distintas pruebas (Figura 3.21). En primer lugar, se considera el entorno que tiene un obstáculo en la parte superior del pasillo. La Figura 3.21a muestra la operación del agente con el entrenamiento realizado en este mismo entorno. Se observa que es capaz de seguir el pasillo esquivando el obstáculo perfectamente. No se adaptan los casos durante la evitación del obstáculo, puesto que ha sido entrenado para ello. Al final del pasillo, cuando se detecta la pared frontal, sí que se adaptan algunos casos al no haber sido entrenado para considerar esta situación. Sin embargo, en la Figura 3.21b sí se observa la adaptación de casos durante la evitación del obstáculo, puesto que ahora se usa el entrenamiento realizado en el entorno de la Figura 3.21c. Esto implica que la base de casos no almacena situaciones similares a las que se encuentra, salvo aquéllas en las que se debe seguir de forma recta el pasillo. Algo parecido ocurre en la Figura 3.21c. El entrenamiento empleado en esta prueba es el del entorno de la Figura 3.21a. Como el robot ha aprendido a evitar el obstáculo en la parte superior, cuando se encuentra con ese mismo obstáculo en la parte inferior no sabe cómo esquivarlo a partir de la base de casos. Por lo tanto, tiene que adaptar esos casos para poder solucionar la situación que se le presenta.

Por último, se consideran dos entornos más complejos (Figura 3.22). Constan de dos obstáculos, uno en la parte superior y otro en la inferior. Para resolver la navegación del agente en estos dos entornos hacemos uso de los dos entrenamientos efectuados con anterioridad (Figura 3.20). Se mezclan las dos bases de casos antes de clasificar, haciendo un total de 851 casos. Al

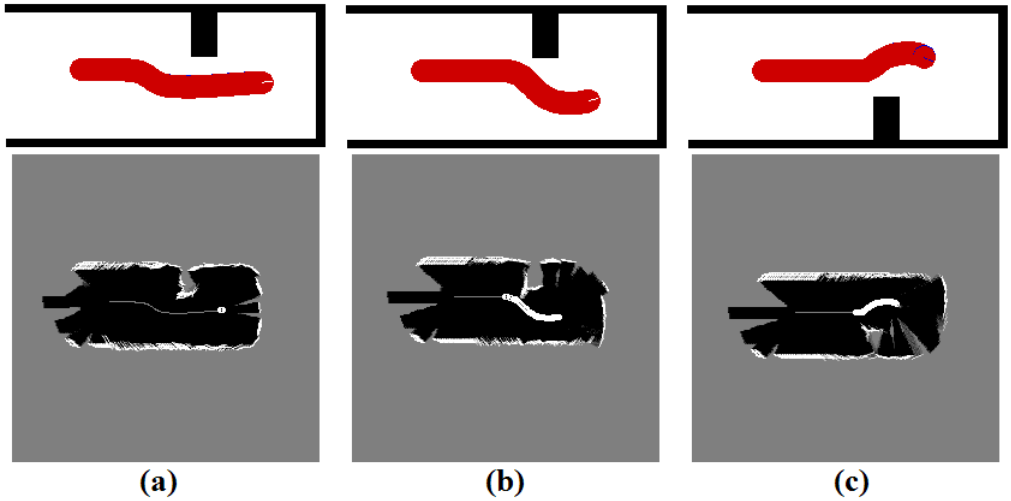

Figura 3.21: Seguir Pasillo con *CBR*, funcionamiento con obstáculos: (a) aprendizaje por observación con obstáculo en la parte superior; (b) aprendizaje por observación con obstáculo en la parte inferior; (c) aprendizaje por observación con obstáculo en la parte superior.

clasificar esta nueva base de casos se obtienen 41 clases. No se han unido las dos bases de casos clasificadas porque potencialmente existen redundancias entre ambas bases de casos, por lo que es mucho más apropiada la clasificación con todos los casos de partida de ambos entrenamientos, para evitar que dos casos similares pertenezcan a dos clases diferentes.

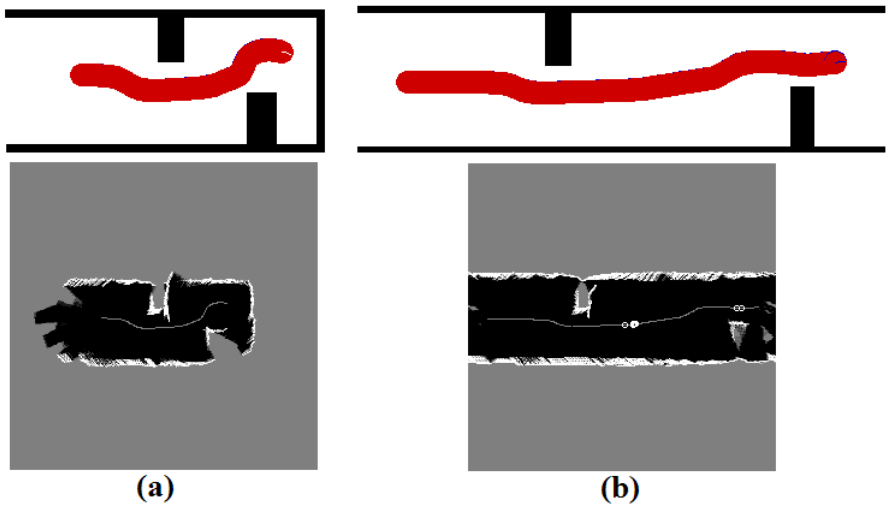

Figura 3.22: Seguir Pasillo con *CBR*, entornos con un obstáculo en la parte superior y otro en la parte inferior: (a) entorno simulado 3; (b) entorno simulado 4.

En el entorno de la Figura 3.22a el agente se centra en los pasillos, evitando los dos obstáculos que aparecen, uno en la parte superior y otro en la inferior. Este comportamiento es posible gracias a que la base de casos que se usa almacena las situaciones necesarias para que la evolución del agente durante la navegación sea correcta. El entorno de la Figura 3.22b también consta de dos obstáculos. Así mismo, uno está situado en la parte superior del pasillo y el otro

en la parte inferior. Sin embargo, en este caso los obstáculos están bastante más alejados entre sí que en el entorno de la Figura 3.22a. Se aprecia a partir de la trayectoria descrita por el agente que el movimiento se ejecuta por el centro del pasillo, esquivando los obstáculos. Aunque se podría pensar que no se deben adaptar casos en esta situación porque la base de casos ya tiene todo el conocimiento necesario, esto no es así. Al pasar por el primer obstáculo se adquieren unos pocos casos, 11 en concreto. Este hecho pone de manifiesto que los sistemas *CBR* son dinámicos en su concepción, puesto que nunca se puede asegurar que el aprendizaje ha finalizado, ya que pueden aparecer situaciones nuevas que en el pasado no han sido consideradas.

3.3.3 Cruzar Puerta

El último de los comportamientos debe permitir al robot cruzar una puerta por su centro para entrar o salir de una habitación.

En este tercer comportamiento partimos del entrenamiento realizado en el entorno de la Figura 3.23. Se trata de un aprendizaje breve, donde se adquieren 103 casos. Esta base de casos se clasifica en 13 clases, siendo $Dist\_MaxMin = 500$. En este caso se emplea una distancia inferior a la usada para los comportamientos Seguir Pared y Seguir Pasillo. Es así porque el entrenamiento es tan breve que el hacer uso de una distancia algo superior, por ejemplo $Dist\_MaxMin = 1000$, provocaría tan pocas clases que no serían útiles. Por lo tanto, se ha optado por disminuir la distancia del algoritmo para poder alcanzar una clasificación adecuada de la información. En el entorno de la Figura 3.23 el robot es guiado a través de la puerta desde una posición no centrada respecto de la misma. De hecho, en la configuración de la Figura 3.23 el robot parte desde la parte inferior de la puerta, aunque al final la atraviesa por su centro.

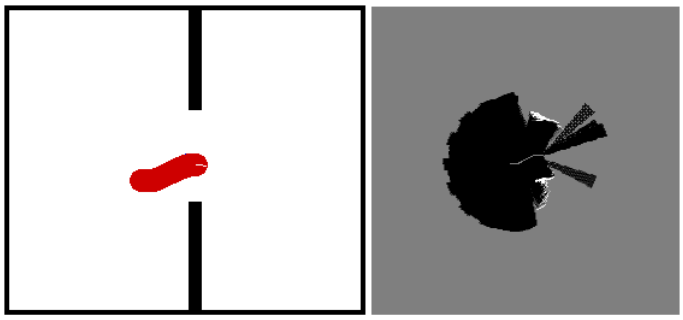

Figura 3.23: Cruzar Puerta con *CBR*, aprendizaje por observación.

Con la base de casos generada gracias al aprendizaje por observación realizado en el entorno de la Figura 3.23, se han efectuado una serie de pruebas que se presentan en las Figuras 3.24 y 3.25. En los entornos de estas figuras existen diferentes situaciones que hay que comentar. El robot puede estar centrado respecto del marco de la puerta, como ocurre en las Figuras 3.24a, d y g, y en las Figuras 3.25c, f y g. En todos los casos la puerta es atravesada con la suficiente seguridad para el robot, independientemente de su anchura. Dependiendo de la prueba el robot intenta centrarse inicialmente o no con la puerta. Esto sugiere que, quizás, en algunos casos no

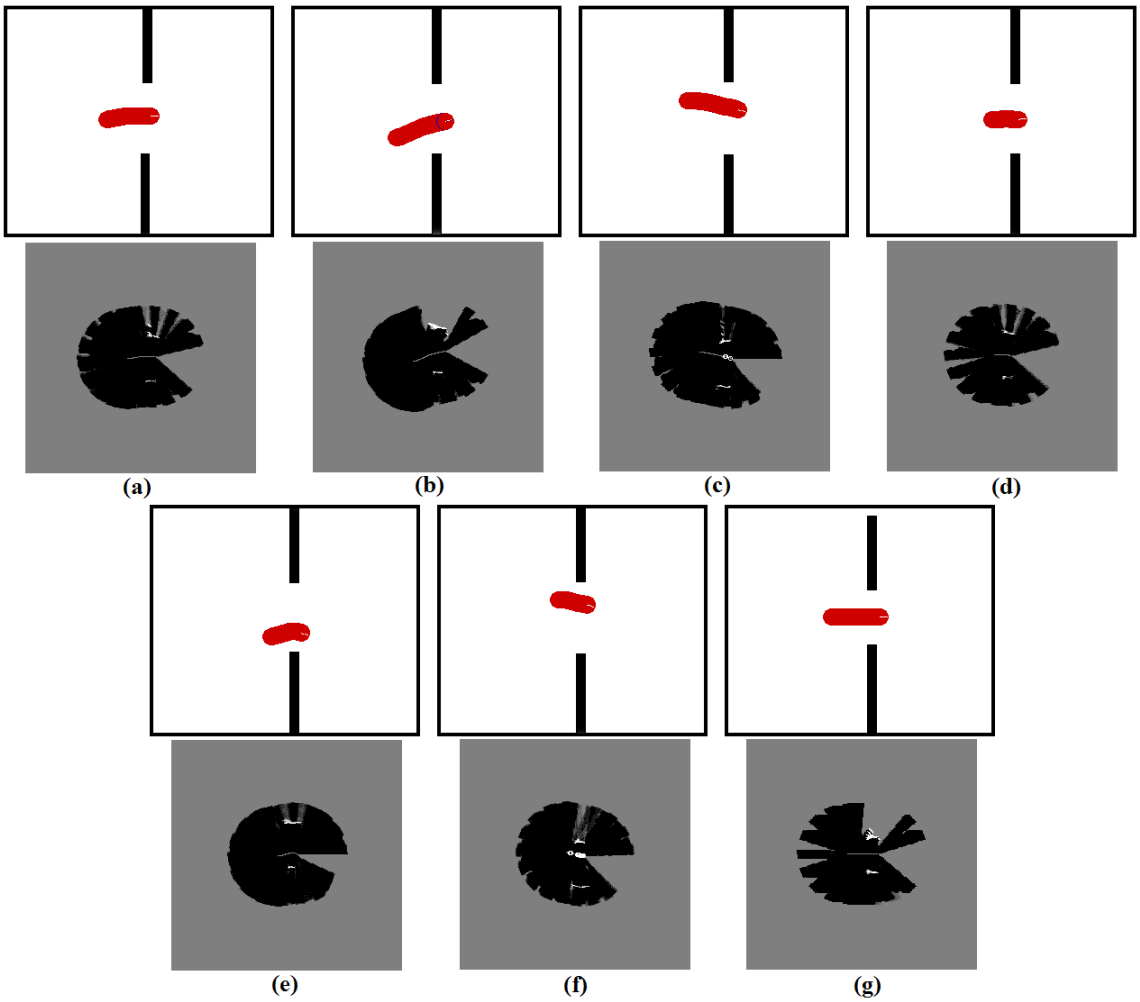

Figura 3.24: Cruzar Puerta con *CBR*, operación en entornos simulados: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7.

estaba tan centrado como se creía. Sin embargo, no supone un problema porque el sistema *CBR* considera estas situaciones (Figuras 3.24a y 3.25f y g).

Una segunda situación se produce cuando el robot no está centrado en la puerta, lo que sucede en el resto de los entornos de las Figuras 3.24 y 3.25. En este caso también hay dos supuestos. El primero, que el robot esté más cerca de la parte superior del marco de la puerta que de la parte inferior (Figuras 3.24c y f, y Figuras 3.25b y d). En este caso sucede que el robot no ha sido entrenado para las situaciones que aparecen, puesto que el aprendizaje por observación realizado parte desde la parte inferior de la puerta, tal y como se muestra en la Figura 3.23. Por lo tanto, el agente adapta casos a medida que navega cuando se da cuenta que está demasiado cerca de la puerta. Esto provoca que le cueste centrarse en la misma, aunque se podría solucionar realizando un entrenamiento que incluyera esta situación.

El segundo supuesto es justo el contrario, que el agente esté más cerca de la parte inferior que de la superior (Figuras 3.24b y e, y Figuras 3.25a y e). La base de casos tiene almacenada información en relación a estas situaciones. Por lo tanto, el agente hace uso de ese conocimiento

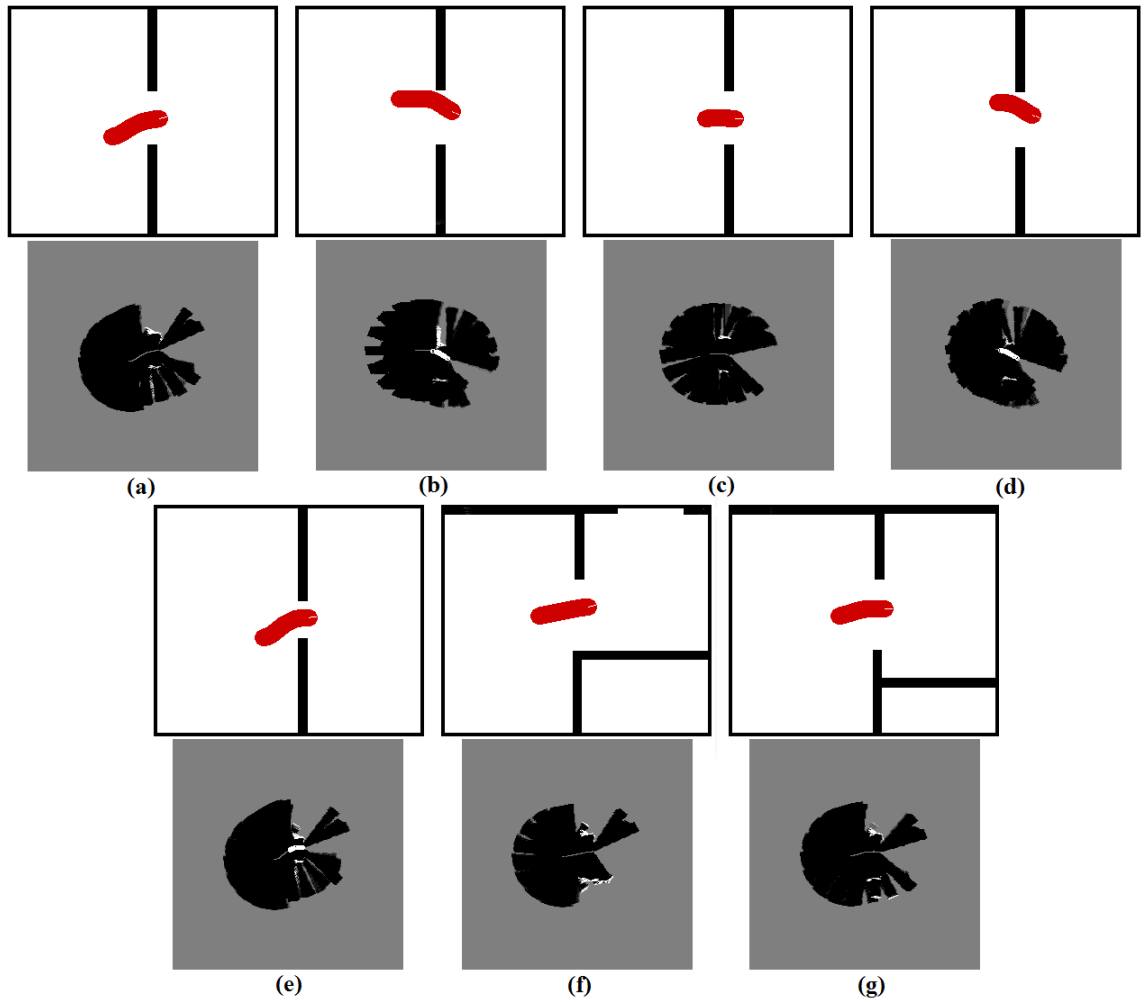

Figura 3.25: Cruzar Puerta con *CBR*, operación en entornos simulados: (a) 8; (b) 9; (c) 10; (d) 11; (e) 12; (f) 13; (g) 14.

para atravesar las puertas que encuentra lo más centrado posible en ellas. Sin embargo, aunque la situación de la Figura 3.25e es similar a la del entrenamiento de la Figura 3.23, el agente adapta casos para poder cruzar la puerta por el centro. Esta adaptación ocurre porque el tamaño de la puerta es mucho menor, tan solo de $90cm$, al de la atravesada durante la fase de aprendizaje por observación (Figura 3.23). Aun así, se aprecia perfectamente que la puerta se cruza por el centro, gracias a la ayuda combinada de aprendizaje por observación y aprendizaje por experiencia. Los casos adaptados recogen las nuevas situaciones detectadas, pudiendo ser usados en el futuro como conocimiento a aplicar para el cruce de otras puertas.

Si comparamos los resultados obtenidos mediante la técnica basada en el *CBR* (Figuras 3.24 y 3.25) con los obtenidos al emplear el modelo analítico (Figura 3.10), se aprecia la mayor suavidad y seguridad conseguida con la filosofía del *CBR*. Cuando se emplea el modelo analítico, el agente atraviesa la puerta más cerca de su marco que en el caso del *CBR*. Así, para la mismas pruebas, se observa una mayor seguridad con el método basado en el *CBR*, como se puede comparar en la Figura 3.10 respecto de las Figuras 3.24 y 3.25. Así, la operación del robot es

más adecuada en las pruebas de las Figuras 3.24c y g, y las Figuras 3.25a y c, en comparación con las mismas pruebas de la Figura 3.10c, h, i y j, respectivamente. Mención especial merece la prueba de la Figura 3.25e. Al analizar el resultado obtenido con el modelo analítico (Figura 3.10n), se observa la dificultad del movimiento, puesto que el robot casi colisiona con el marco de la puerta, no atravesándola por su centro. Bien es cierto que dicha puerta es muy estrecha en relación al tamaño del robot ($50x50cm^2$), pero el mismo comportamiento implementado con el *CBR* sí que consigue centrar el robot en la puerta y cruzarla de un modo bastante más seguro.

Finalmente, se analiza en una última prueba la influencia del aprendizaje por observación en la adaptación de los casos. El entrenamiento realizado en el entorno de la Figura 3.23 provoca que cuando el robot parte desde una posición superior al eje perpendicular al marco de la puerta, se adapten casos. Esta situación se ha analizado, mostrándose su efecto en las Figuras 3.24c y f, así como en las Figuras 3.25b y d. Se espera, por tanto, que cuando se entrene al robot partiendo desde una posición superior al eje perpendicular al marco de la puerta no se adapten casos. Este hecho se presenta en la Figura 3.26. La Figura 3.26a muestra el entrenamiento efectuado, que ha derivado en la captura de 361 casos o 35 clases tras la clasificación. Este aprendizaje por observación se emplea en el mismo entorno, dando lugar a la trayectoria de la Figura 3.26b. El movimiento del agente es más suave, como era de esperar, en la operación que en el entrenamiento, puesto que se adapta mejor a la configuración del entorno que el guiado llevado a cabo por el operador humano. Como se esperaba, no aparecen nuevas situaciones, es decir, no se adaptan casos, al contrario de lo que ocurría con esta misma prueba en la Figura 3.24c.

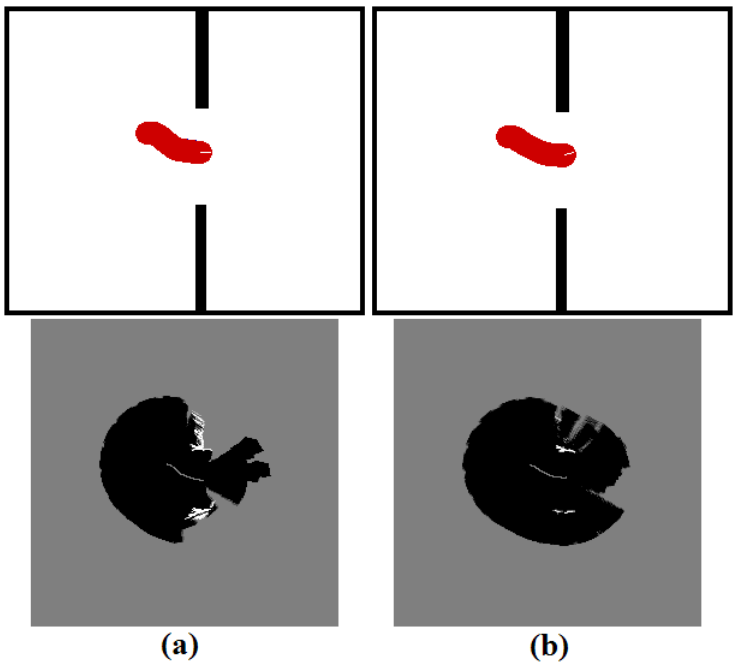

Figura 3.26: Cruzar Puerta con *CBR*: (a) aprendizaje por observación; (b) operación en entorno simulado.

4 Método de aprendizaje asistido

Un móvil se considera autónomo cuando puede llevar a cabo una determinada tarea en un entorno desconocido sin una supervisión continua por parte de un humano. En este capítulo se han presentado previamente tres comportamientos autónomos implementados de dos formas distintas, siguiendo un modelo analítico y mediante una técnica basada en el *CBR*. Cada uno de los comportamientos lleva al robot a ejecutar una tarea diferente, seguir una pared de forma paralela, recorrer un pasillo por su centro y cruzar una puerta.

Una de las claves de la filosofía del *CBR* aplicada al diseño de comportamientos reactivos es el aprendizaje que se puede realizar. En la Sección 3 se propone, para el desarrollo de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta, llevar a cabo un aprendizaje en dos fases, por observación y por experiencia. En esta Tesis, el aprendizaje por observación se alcanza mediante el guiado del robot por un humano por medio de un teclado. Sin embargo, este guiado puede introducir información errónea debido al propio humano involucrado en el proceso. Por este motivo, se propone en esta Tesis un nuevo mecanismo de aprendizaje asistido que combina las órdenes que un humano aplica mediante un joystick con el movimiento autónomo del robot. El sistema desarrollado en la presente Tesis da la posibilidad de alcanzar, gracias al módulo *ControlRobot*, esta combinación de comandos para ayudar al humano a conducir correctamente y, por tanto, evitar en la medida de lo posible el aprendizaje de casos erróneos.

Antes de pasar a describir el método propuesto se presenta una revisión de los estudios que nos podemos encontrar sobre la cooperación entre un humano y un robot a la hora de navegar por un entorno.

4.1 Cooperación humano/robot

Las situaciones en las que las máquinas y las personas cooperan para alcanzar un mismo fin se encuadran dentro del campo del control colaborativo. En estos casos existe la necesidad de decidir cuánto control debería tener el robot a la hora de interactuar con el humano y viceversa [BFB+05, KWLV05, HS05, AM00]. Existen muchas aproximaciones al control colaborativo. Normalmente se basan en darle el control, bien al usuario humano, bien a la máquina, de acuerdo a algún algoritmo más o menos complejo.

En un primer método el móvil es controlado completamente por el humano. Sin embargo, bajo determinadas circunstancias el robot puede tomar el control para evitar un peligro inminente. Esta técnica suele emplearse en medios en los que la comunicación con el agente no es buena en términos de retardos e interrupciones [MDK+03, KBRS97] o cuando el control que efectúa el humano no es el adecuado [PGKO05, PGKO04, MAL+05]. En otras ocasiones, el humano puede decidir por sí mismo cuándo ceder el control al robot para efectuar tareas complejas como atravesar puertas o girar 180^o [BFB+05, HS05]. En una situación extrema, si el humano se ve incapacitado para el control, éste puede quedar en su totalidad en manos del robot, el cual se vería obligado a calcular las trayectorias y seguirlas [SL02].

Un segundo grupo de aproximaciones [CV90, SL98, Mil98, RCJ+02] están basadas en un conjunto de primitivas como EvitarObstáculos, SeguirPared y CruzarPuerta, que pueden ser usadas para asistir a una persona en maniobras difíciles, mediante una selección manual o algún mecanismo automático de disparo. Por lo tanto, el operador podría guiar directamente el robot o conmutar entre varios comportamientos autónomos con el objetivo de superar situaciones complicadas. La principal diferencia entre todas las técnicas estriba en la implementación de los comportamientos.

Otros sistemas funcionan como un robot autónomo convencional. El usuario proporciona un destino y el robot se encarga de ejecutar esta tarea de forma autónoma [FLD05, SL97, CC98, NCOA95]. Así, podemos encontrar sistemas como *SENARIO* [KST+97], un robot que emplea una arquitectura híbrida que combina un planificador de alto nivel basado en un mapa topológico con la navegación local. El sistema tiene en cuenta los riesgos e incluso la condición del usuario.

El principal inconveniente de los sistemas descritos estriba en el hecho de que el control no es en realidad compartido, puesto que en cada momento lo ostenta la máquina o el humano. Esto puede provocar cambios bruscos al pasar el control de uno a otro, los cuales no son convenientes para personas con dificultades. Sería mucho mejor distribuir el control entre el robot y el usuario en todo momento, aumentando o disminuyendo el peso de cada uno de ellos en función de la eficiencia de la conducción por parte del usuario. Sin embargo, es necesario decidir cómo combinar de manera eficiente los comandos propuestos por el robot con los comandos introducidos por el usuario en cada punto de la navegación.

En la presente Tesis se propone una técnica de aprendizaje asistido para fusionar las órdenes de un humano, proporcionadas por medio de un joystick, con los comandos de movimiento que el robot está intentando aplicar (Figura 3.27). Ambos comandos son combinados mediante una suma ponderada. El peso de cada uno de ellos viene determinado por la eficiencia del comando que tratan de imponer en cada momento de la trayectoria.

Figura 3.27: Combinación de los comandos del robot y el humano.

4.2 Aproximación reactiva para el aprendizaje asistido

La técnica propuesta fusiona en una única orden los comandos del humano con los que el agente autónomo intenta aplicar. Dependiendo del experimento, los comandos de movimiento del robot son proporcionados por uno de los tres comportamientos implementados, Seguir Pared, Seguir Pasillo o Cruzar Puerta. El aprendizaje asistido se alcanza mediante una combinación lineal ponderada de los comandos del robot y los comandos introducidos por el humano mediante un joystick.

El algoritmo es análogo para los tres comportamientos. Tal y como hemos visto en la Sección 2, cada comportamiento proporciona las velocidades de rotación y traslación que el robot debe aplicar como sus comandos de movimiento. Al margen del comportamiento que el robot esté ejecutando, supongamos que v_{rR} y v_{tR} son, respectivamente, la velocidad de rotación y de traslación que el robot proporciona como su propio comando de movimiento. El humano, al mismo tiempo, trata de aplicar gracias al joystick la velocidad de rotación v_{rH} y la velocidad de traslación v_{tH}. La combinación de la orden del usuario con la del agente autónomo proporciona el comando de movimiento que en realidad se aplica al robot. Consiste en la velocidad de rotación, v_{rC}, y de traslación, v_{tC}, resultado de la colaboración del humano y el robot, es decir, del control asistido. Estas velocidades se definen como:

$$v_{rC} = 0.5 \cdot \eta_R \cdot v_{rR} + 0.5 \cdot \eta_H \cdot v_{rH} \tag{3.24}$$

$$v_{tC} = 0.5 \cdot \eta_R \cdot v_{tR} + 0.5 \cdot \eta_H \cdot v_{tH} \tag{3.25}$$

donde η_R es la eficiencia del comando de movimiento del robot y η_H es la eficiencia del comando de movimiento introducido por el usuario mediante el joystick. La eficiencia del comando que combina el humano y el robot se define como η_C. Las eficiencias están en el rango de 0 a 1, siendo 1 la máxima eficiencia. Es evidente que η_C no es igual ni a η_R ni a η_H. Como v_{rC} y v_{tC} combinan linealmente las velocidades del robot y las del humano, η_C tiene a ser la media de η_R y η_H. Podemos observar también en las Ecuaciones (3.24) y (3.25) que, al margen de las eficiencias, cada uno de los dos agentes implicados contribuye con un 50% al comando de movimiento final, por medio de los factores 0.5. Estos factores pueden ser usados para reflejar el estado del usuario, consiguiendo así que el humano aumente o reduzca su contribución al control del sistema, según se presenta en [FEPUS07b].

Las eficiencias, genéricamente η, aúnan la contribución de tres factores locales que tienen influencia en las mismas. Estos tres factores son la suavidad (η_{su}), la longitud de la trayectoria (η_{lt}) y la seguridad (η_{se}), cuyo valor varía entre 0 y 1.

La suavidad se evalúa en función del ángulo diferencia entre la orientación actual del robot y el vector del comando de movimiento. Está pensada para tener en cuenta que muchos robots son no holonómicos y que es mejor mantener una trayectoria tan suave como sea posible con el

objetivo de evitar el deslizamiento (*slippage*) y las oscilaciones. Supongamos que para un comportamiento cualquiera las velocidades de rotación y de traslación son v_r y v_t, respectivamente. En estas condiciones el ángulo entre la orientación actual del robot y el vector del comando, α_{dif}, se obtiene como:

$$\alpha_{dif} = \arctan \frac{v_r}{v_t} \tag{3.26}$$

Si C_{su} es una constante, el factor de suavidad η_{su} de la eficiencia η será:

$$\eta_{su} = e^{-C_{su} \cdot |\alpha_{dif}|} \tag{3.27}$$

La longitud de la trayectoria no se puede evaluar de manera global en un instante de tiempo puntual, puesto que la navegación es reactiva al no existir ningún tipo de planificación. Por ello, la medimos localmente en términos del ángulo conformado por la orientación del robot y la dirección que el robot debería tener para implementar el comportamiento seleccionado, α_{comp}. Si C_{lt} es una constante, el factor de longitud de la trayectoria, η_{lt}, se calcula como:

$$\eta_{lt} = e^{-C_{lt} \cdot |\alpha_{comp} - \alpha_{dif}|} \tag{3.28}$$

El factor de seguridad, η_{se},se evalúa en términos de la distancia al obstáculo más cercano en cada momento. Cuanto más se acerque el robot a los obstáculos, más peligrosa será la trayectoria. Considerando que C_{se} es una constante y que α_{min} es el ángulo entre la orientación actual del robot y el obstáculo más cercano, η_{se} será:

$$\eta_{se} = 1 - e^{-C_{se} \cdot d_{min} \cdot |\alpha_{min} - \alpha_{dif}|} \tag{3.29}$$

donde d_{min} es la distancia al obstáculo más próximo.

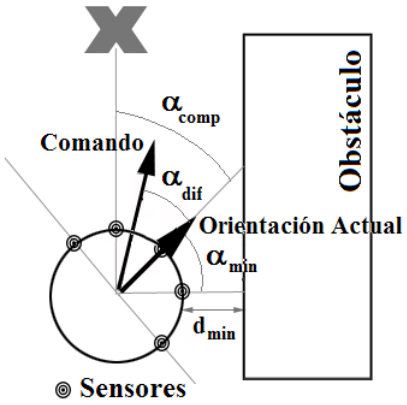

Figura 3.28: Definición de los tres factores involucrados en la eficiencia.

Los parámetros que definen cada uno de los tres factores se pueden observar en la Figura

3.28, donde se ha considerado que el robot está ejecutando el comportamiento Seguir Pared. Estos tres factores son combinados para obtener, finalmente, la eficiencia η:

$$\eta = \frac{\eta_{su} + \eta_{lt} + \eta_{se}}{3} \tag{3.30}$$

η, definida como en la Ecuación (3.30), nos proporciona un mecanismo para evaluar las diferentes opciones (robot, humano, control asistido), con el propósito de compararlas. Más aún, los comandos de movimiento pueden ser evaluados en términos del efecto inmediato de cada uno de los factores en la eficiencia. Las constantes, C_{su}, C_{lt} y C_{se}, se han fijado empíricamente a los siguientes valores (para más detalles, ver [Pér06]):

$$\begin{aligned} C_{su} &= 2.3 \cdot 10^{-2} \\ C_{lt} &= 2.3 \cdot 10^{-2} \\ C_{se} &= 5.8 \cdot 10^{-5} \end{aligned} \tag{3.31}$$

5 Experimentos y resultados

En este apartado se presentan los resultados obtenidos con la plataforma real *Pioneer P2AT*. Por un lado se muestran los experimentos realizados con los comportamientos implementados tanto con el modelo analítico como con la filosofía del *CBR*. Por otro, se analiza cualitativa y cuantitativamente la operación del robot mediante el método de aprendizaje asistido propuesto.

5.1 Pruebas de los comportamientos

Al analizar la implementación realizada de los comportamientos, bien por el método analítico en la Sección 2, bien con un método basado en el *CBR* en la Sección 3, se presentaron pruebas en entornos simulados llevadas a cabo con el simulador de la plataforma robótica *Nomad 200* de *Nomadics*. Con estas pruebas se ha estudiado el funcionamiento de cada comportamiento, así como los problemas que pueden surgir, comparando ambas estrategias desde el punto de vista del funcionamiento del robot a la hora de ejecutar una determinada tarea. En el caso de la implementación realizada con el *CBR* también se analiza la influencia de ambos tipos de aprendizaje, por observación y por experiencia, en la operación del agente. Del mismo modo, se ha recalcado en todos los tests la necesidad de emplear más de un comportamiento para alcanzar una exploración completa eficiente de un entorno, puesto que con uno solo de ellos es insuficiente (Figura 3.4a o Figura 3.15b), o incluso ineficiente (Figura 3.8a o Figura 3.19c).

Como se considera que los comportamientos han sido convenientemente estudiados en entornos simulados, se presentan en este apartado únicamente experimentos llevados a cabo en entornos reales con la plataforma robótica real *Pioneer P2AT*, equipada con 8 sensores sonar *Polaroid* frontales. La máxima velocidad de traslación y de rotación empleada en estos experimentos es $v_{tMAX} = 100mm/s$ y $v_{rMAX} = 15^o/s$, respectivamente. Como los tres comportamientos

son reactivos no existe planificación en estas pruebas. Por lo tanto, los mapas métricos que se presentan se usan solamente para mostrar gráficamente la trayectoria seguida por el robot sobre el modelo de entorno generado. Esta trayectoria se presenta en gris. Cuando se emplea la técnica basada en el *CBR* y se adaptan casos, éstos se dibujan con un círculo blanco.

Aunque el simulador de la plataforma robótica *Nomad 200* de *Nomadics* modela los posibles errores que los sensores sonar tienen, no alcanza el nivel de realidad que estos sensores exhiben en la práctica. Por lo tanto, al realizar pruebas en entornos reales es de esperar que los resultados no sean tan buenos como en escenarios simulados. Y todo ello debido a los errores presentes en las lecturas de los sonar. Estos errores incluyen la inexactitud de la lectura debido al ángulo de incidencia del haz del sonar que puede producir lecturas erróneas detectando obstáculos donde no los hay, e incluso que no se detecten obstáculos que sí aparecen. En los entornos reales de prueba usados en esta Tesis aparecen diferentes materiales como son paredes, metal, madera y cartón. Este hecho también influye en las lecturas, pues éstas dependen del material sobre el que incide el haz del sonar.

La influencia de los errores es mayor en los comportamientos implementados con el modelo analítico que en aquéllos que se basan en el *CBR* porque la velocidad de traslación y de rotación instantánea depende únicamente de la lectura proporcionada por los sensores sonar en ese instante. Sin embargo, si se usa la filosofía del *CBR* los errores son tratados por el sistema, al estar presentes en la fase de aprendizaje por observación. Aunque el error aparezca, el humano que guía el robot durante el entrenamiento abstrae esta información y aplica las velocidades adecuadas al robot para la tarea y configuración del entorno que se encuentra, independientemente de que la lectura haya sido errónea o no. Es decir, la base de casos que se emplea durante la navegación con un determinado comportamiento contempla los errores de los sonar, teniéndolos en cuenta el sistema de forma automática y siendo tratados por el mismo.

5.1.1 Seguir Pared

Como ya sabemos, en este primer comportamiento el robot debe seguir la pared que se encuentra a su derecha a una distancia determinada, partiendo desde su posición inicial. Si el movimiento se hace según el modelo analítico de la Sección 2, se han fijado para todas las pruebas los parámetros del modelo a los valores $d_D = 500mm$ y $d_{SPSEC} = 1000mm$. En el caso de que se emplee la filosofía del *CBR*, el umbral de adaptación de los casos y la distancia mínima entre clases se han establecido a los valores $U_{reusar} = 0.1$ y $Dist\_MaxMin = 1000$, respectivamente. Las bases de casos usadas siempre están clasificadas.

En la Figura 3.29 se presentan dos entornos reales de prueba para el comportamiento Seguir Pared. En el entorno de la Figura 3.29a se realiza un entrenamiento muy simple, lo que le permite al sistema adquirir aprendizaje por observación útil durante la navegación con el modelo basado en el *CBR*. La trayectoria seguida por el agente durante el guiado de un operador humano se encuentra superpuesta en gris al modelo de entorno generado durante el entrenamiento (Figura 3.30a). Este entrenamiento dota al agente de conocimiento para seguir de forma paralela una

pared y para girar hacia la izquierda cuando hay un cambio de dirección. El número de casos almacenados en la base de casos durante el entrenamiento es de 543, clasificándose en 62 clases.

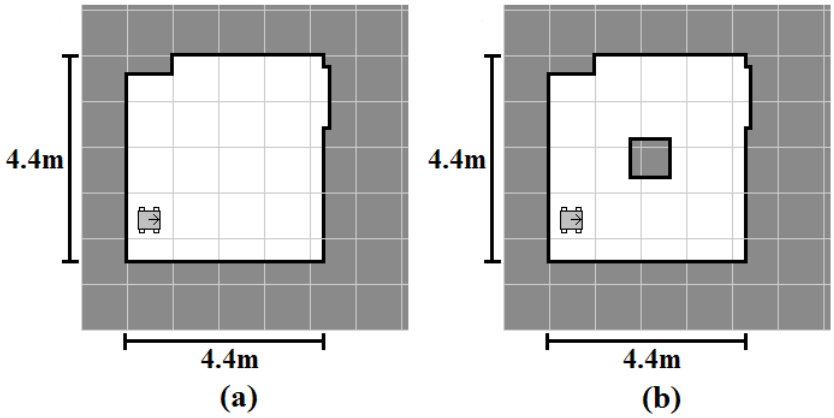

Figura 3.29: Seguir Pared, entornos reales de prueba: (a) 1; (b) 2.

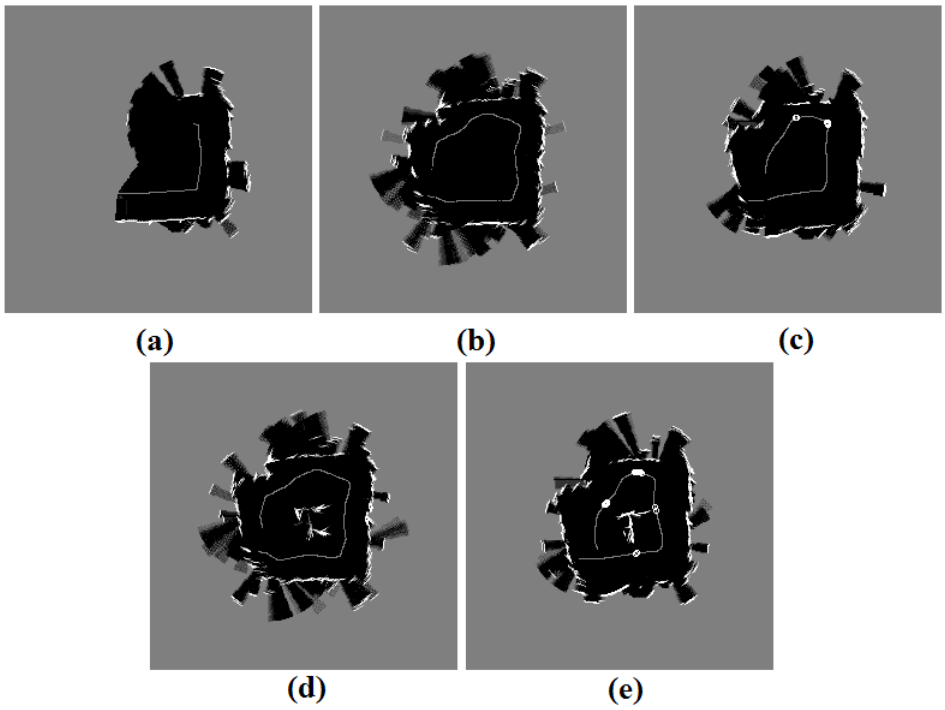

Figura 3.30: Seguir Pared, entorno real de la Figura 3.29a: (a) aprendizaje por observación; (b) operación con el modelo analítico; (c) operación con el *CBR*. Entorno real de la Figura 3.29b: (d) operación con el modelo analítico; (e) operación con el *CBR*.

La operación del robot en el entorno de la Figura 3.29a con el modelo analítico se muestra en la Figura 3.30b. Se aprecian perfectamente las oscilaciones típicas de los modelos analíticos basados en los campos de potencial, como es nuestro caso. Aunque estas oscilaciones no presentan una gran amplitud, tampoco son deseables. Aun así, se puede decir que el movimiento se ejecuta correctamente. En la Figura 3.30c se muestra la operación en el mismo escenario, pero con la aproximación basada en el *CBR*. La base de casos usada en esta prueba es la obtenida tras

clasificar el aprendizaje por observación adquirido durante el entrenamiento que se presenta en la Figura 3.30a. Se observa que la trayectoria seguida presenta menos giros y menos oscilaciones que en el caso del modelo analítico (Figura 3.30b). Sin embargo, también se aprecia que se adaptan algunos casos. Aunque pueda parecer una desventaja, no tiene por qué ser así. Si se adaptan nuevos casos se aumenta la experiencia del sistema, aumentando el conocimiento para sucesivas pruebas. En la Figura 3.30c se adaptan casos al detectar la esquinas. Esto no nos debe extrañar, puesto que los sensores sonar pueden provocar errores en su lectura al encontrarse con esquinas. Por lo tanto, lo que está sucediendo es que la base de casos no tiene almacenado ese conocimiento y es necesario recurrir al modelo analítico (adaptación empleada en esta Tesis), para solucionar la nueva situación. Sin embargo, desde un punto de vista subjetivo, durante las pruebas en el entorno de la Figura 3.29a se observó que el movimiento con el modelo basado en el *CBR* era bastante más suave, realizando el robot menos cambios de dirección que cuando se usa el modelo analítico.

El segundo entorno de prueba (Figura 3.29b) es similar al primero (Figura 3.29a). La única diferencia es que tiene un obstáculo en el medio de la habitación de $70x70cm^2$. Cuando se emplea el modelo analítico el robot da lugar a la trayectoria representada sobre el mapa métrico de la Figura 3.30d. La pared que se encuentra a la derecha del robot se sigue perfectamente, aunque el agente oscile debido al modelo empleado que se basa en los campos de potencial. Estas oscilaciones se eliminan si se usa la implementación con el *CBR* (Figura 3.30e), donde la base de casos usada es la que se obtiene tras clasificar el aprendizaje por observación adquirido durante el entrenamiento representado en la Figura 3.30a. Gracias a esta técnica la trayectoria presenta menos oscilaciones, lo cual supone una ventaja del método basado en el *CBR* respecto del modelo analítico. Sin embargo, también hay que decir que se adaptan nuevos casos, en concreto 39. Se aprecia en la Figura 3.30e que cuando esto sucede el robot, además de tener la pared que siempre se encuentra a su derecha, también detecta el obstáculo a su izquierda. El sistema no tiene contemplada esta posibilidad, pues el robot solamente ha sido entrenado para una situación en la cual aparece una pared a su derecha. No obstante, esto no supone un problema, puesto que al adaptar nuevos casos se aumenta la experiencia del sistema para el futuro.

Un tercer entorno real de prueba se presenta en la Figura 3.31a. Es la misma habitación de los entornos de la Figura 3.29, pero con un obstáculo en el medio del mismo. Además, el robot no parte desde la posición inicial de aquellos entornos. En este caso, el agente está inicialmente cerca del obstáculo. Por lo tanto, es esperable que se siga el contorno de este obstáculo. De hecho, esto es lo que ocurre. La operación con el modelo analítico se muestra en la Figura 3.31b. El robot es incapaz de explorar otra zona al margen del alcance que tiene mientras bordea el obstáculo. Si se dejara al robot navegando indefinidamente, siempre se repetiría la misma secuencia de movimientos, con el robot siguiendo el contorno del obstáculo. Esta prueba se ha repetido con el modelo basado en el *CBR*. Partiendo del aprendizaje realizado en el entorno de la Figura 3.29a, comienza la navegación. En una primera situación se ha considerado que solamente se puede considerar el conocimiento almacenado en la base de casos, eliminando la

posibilidad de adaptar casos. Es decir, es como si el sistema tuviera adaptación nula. Bajo estas condiciones, el robot no es capaz de ejecutar el comportamiento adecuadamente. Cuando tiene que girar hacia la derecha no puede hacerlo, puesto que en la base de casos solamente hay experiencia sobre el seguimiento paralelo de una pared y los giros a la izquierda. Por lo tanto, al ser el giro hacia la derecha una situación novedosa, el agente no puede resolverla, dando lugar a la trayectoria de la Figura 3.31c.

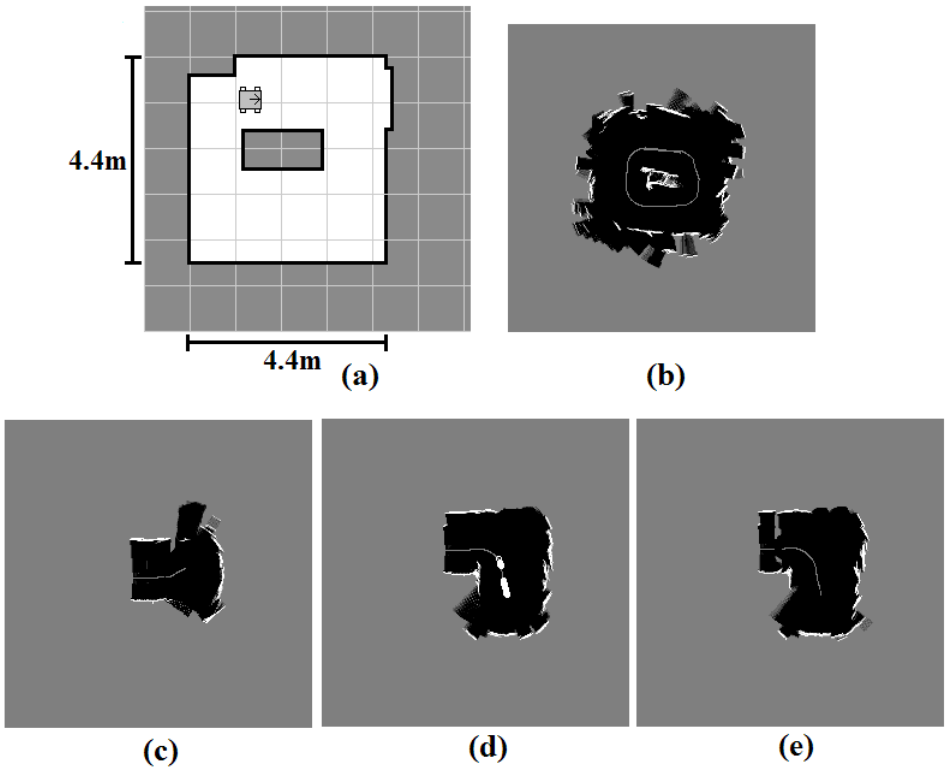

Figura 3.31: Seguir Pared, entorno real de prueba: (a) planta; (b) operación con el modelo analítico. Operación con el *CBR*: (c) sin adaptación; (d) con adaptación; (e) con adaptación teniendo en cuenta el aprendizaje por experiencia en (d).

Al habilitar la posibilidad de aprender de la propia experiencia, la situación es radicalmente distinta. La base de casos no tiene experiencia sobre los giros hacia la derecha. Por lo tanto, cuando hay que ejecutar este giro, los casos se adaptan porque la menor distancia del caso actual a la base de casos es superior al umbral U_{reusar}. En la Figura 3.31d se presenta la trayectoria recorrida y los casos adaptados. El número de casos adaptados es de 36, aumentando la experiencia que se tiene con las nuevas situaciones. Se repite de nuevo la prueba para evaluar el efecto que tiene en el sistema el aprendizaje por experiencia adquirido. Si los 36 casos adaptados se suman a los 543 que ya había en la base de casos, se obtienen 579. Al clasificar esta base de casos se obtienen 72 clases. En esta nueva base de casos ya hay experiencia acumulada para poder girar a la izquierda, a la derecha y para seguir de forma paralela la pared. El resultado de la navegación con la nueva base de casos clasificada se muestra en la Figura 3.31e. Se aprecia que la pared se sigue perfectamente, sin necesidad de adaptar nuevos casos, puesto que las situaciones que se le presentan al robot ya estaban almacenadas en la base de casos.

Para evaluar el efecto que en el sistema tiene el aprendizaje por observación se repite la prueba en el entorno de la Figura 3.31a, pero partiendo de un entrenamiento diferente. El mapa métrico de la Figura 3.32a tiene la trayectoria seguida por el robot durante el entrenamiento manual que un operador humano lleva a cabo. La base de casos que se adquiere contiene 428 casos, que se clasifican en 78 clases. Al emplear esta base de casos durante la navegación se obtiene la trayectoria de la Figura 3.32b. Se aprecia que el obstáculo se sigue perfectamente. Por otro lado, en esta trayectoria se adaptan tan solo 7 casos. Si observamos, esta adaptación se produce cuando se detectan las esquinas, situaciones en las que las lecturas de los sensores sonar pueden ser erróneas. Sin embargo, se puede decir que el número de casos adaptados es mínimo. Este resultado era esperable porque la base de casos empleada contiene las situaciones que el robot se encuentra durante la navegación en ese entorno.

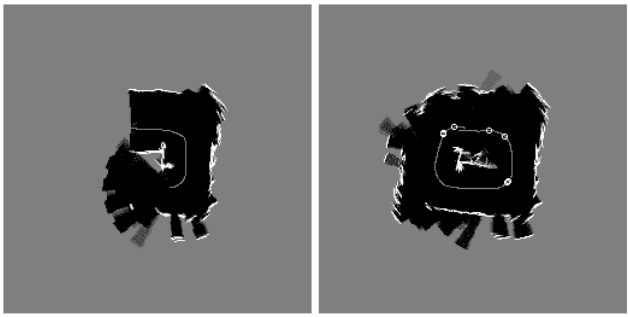

Figura 3.32: Seguir Pared, entorno real de la Figura 3.31a: (a) aprendizaje por observación; (b) operación con el *CBR*.

5.1.2 Seguir Pasillo

Para este segundo comportamiento, la tarea que debe ejecutar el robot es recorrer un pasillo de anchura arbitraria por su centro. Si la navegación se realiza con el modelo analítico de la Sección 2, todas las pruebas se llevan a cabo con el valor $d_{SCSEC} = 400mm$. En el caso de que se emplee la filosofía del *CBR*, el umbral de adaptación de los casos y la distancia mínima entre clases se han establecido a los valores $U_{reusar} = 0.1$ y $Dist\_MaxMin = 1000$, respectivamente. Las bases de casos usadas siempre están clasificadas.

En una primera prueba el robot opera en un pasillo de $2.5m$ de ancho (Figura 3.33a). La operación con el modelo analítico se muestra en la Figura 3.33c. En esta trayectoria se observan las oscilaciones típicas de las aproximaciones puramente reactivas basadas en los campos de potencial. La misma prueba se repite con la filosofía del *CBR*. Antes de realizar dicha prueba se efectúa un aprendizaje por observación en el mismo entorno. Un humano guía el robot con un teclado a lo largo de la trayectoria de la Figura 3.33b. Durante este entrenamiento se adquieren 152 casos. Estos casos son clasificados, dando lugar a la base de casos que se emplea en la prueba, con 18 clases. La operación del agente con el modelo basado en el *CBR* se presenta en la Figura 3.33d. Se aprecia que la trayectoria es mucho más suave, con menos oscilaciones

y menos giros que cuando se hace uso del modelo analítico, realizando el robot menos cambios de dirección. No se adaptan casos, salvo cuando se detecta la pared frontal, puesto que esta situación no estaba recogida en la base de casos.

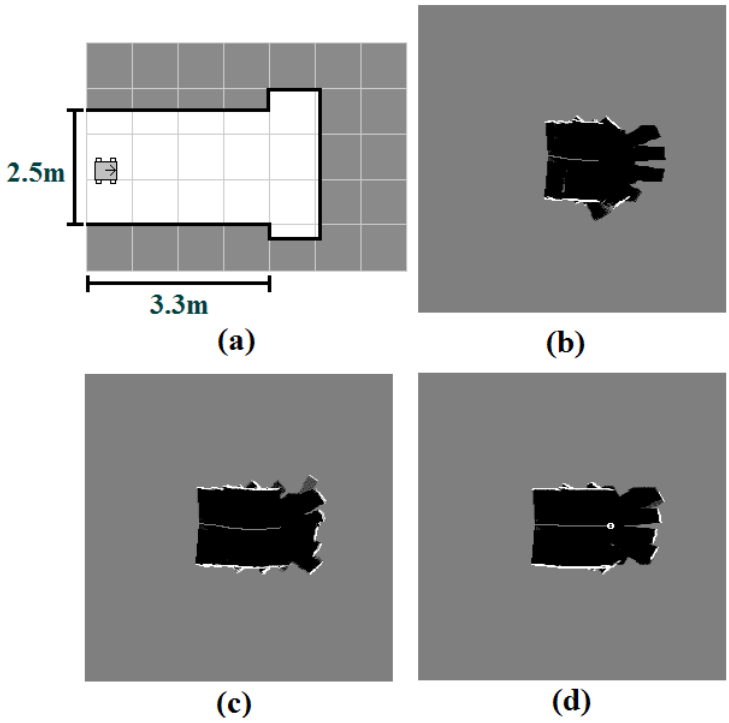

Figura 3.33: Seguir Pasillo, entorno real de prueba: (a) planta; (b) aprendizaje por observación; (c) operación con el modelo analítico; (d) operación con el *CBR*.

Un pasillo de menor anchura, $1.8m$, se muestra en la Figura 3.34a. Nuevamente, la operación con el modelo analítico presenta oscilaciones, según se aprecia en la Figura 3.34b. Si bien este hecho no es deseable, es un resultado de la aproximación que se está empleando que toma como base los campos de potencial. Sin embargo, cuando se hace uso del *CBR* la trayectoria recorrida es bastante más rectilínea, suave y con menos cambios de dirección (Figura 3.34c). La base de casos usada para este experimento es la clasificada obtenida tras el entrenamiento en el entorno de la Figura 3.33a, con 18 clases. Al margen de los casos que se adaptan al final del pasillo porque la detección de la pared frontal no está contenida en la base de casos, solamente se adaptan 20 casos nuevos. Aunque el seguimiento del pasillo sí forma parte del conocimiento de la base de casos, el entrenamiento fue realizado en un pasillo de mayor anchura. Por este motivo se adaptan esos nuevos casos. Aun así, en la mayor parte de la trayectoria con el conocimiento almacenado en la base de casos es más que suficiente, y todo ello dando lugar a una trayectoria bastante recta.

En otra prueba se pretende poner de manifiesto la adaptación y el aprendizaje por experiencia que el agente adquiere mientras navega (Figura 3.35). La principal característica del entorno de la Figura 3.35a es su forma en L. La trayectoria seguida por el robot cuando se emplea el modelo analítico se muestra en la Figura 3.35b. La operación del agente es correcta,

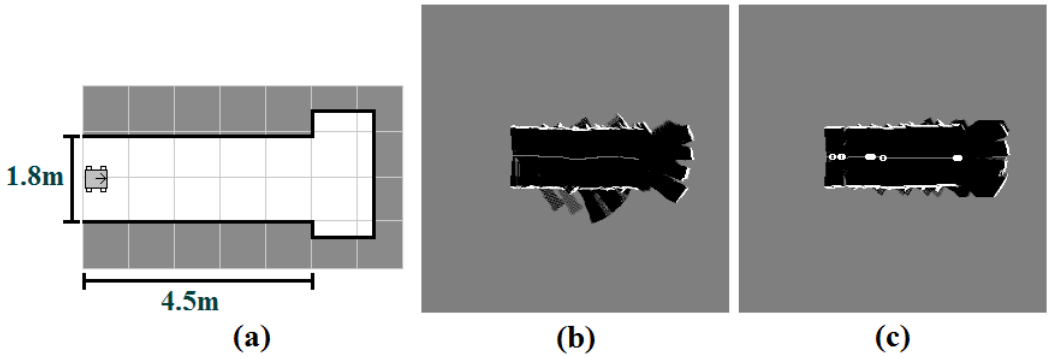

Figura 3.34: Seguir Pasillo, entorno real de prueba: (a) planta; (b) operación con el modelo analítico; (c) operación con el *CBR*.

al margen de las oscilaciones que aparecen. La misma prueba se lleva a cabo considerando el modelo basado en el *CBR*. El conocimiento del que se parte es el almacenado en las 18 clases de la base de casos clasificada adquirida en el entorno de la Figura 3.33a. Para estudiar la diferente operación del robot ante distintos supuestos, se considera en primer lugar la base de casos únicamente, sin permitir la adaptación de nuevos casos. La trayectoria recorrida por el agente autónomo móvil se muestra en la Figura 3.35c. Como la base de casos solamente tiene en cuenta el movimiento rectilíneo del robot, no es capaz de girar cuando debe. Si la prueba no se hubiera detenido, el robot habría chocado con la pared este del entorno. Para evitar este problema se le permite al agente adaptar casos. En este nuevo supuesto la trayectoria recorrida por el robot es la de la Figura 3.35d. La adaptación de casos se produce durante el giro, cuando el agente intenta centrarse en el segundo pasillo. El número de casos adaptados es de 91 que, sumados a los 152 del entrenamiento anterior, hacen un total de 243 casos. Al clasificar la nueva base de casos se obtienen 36 clases, que serán las usadas para repetir el experimento.

Al lanzar la prueba con la nueva base de casos se obtiene la trayectoria representada en la Figura 3.35e. Se observa que para esta situación no se adaptan nuevos casos, al contrario que en la prueba de la Figura 3.35d. Esto ocurre porque la nueva base de casos ya contiene información sobre el giro de 90^o que el robot debe realizar para orientarse con el segundo pasillo. Un resultado importante desde el punto de vista cualitativo es que las pruebas llevadas a cabo con el *CBR* dan lugar a un movimiento más suave del robot, con menos giros y cambios de dirección que la realizada con el modelo analítico, lo cual es una ventaja. También debemos hacer notar que, al margen del método empleado, quizás el comportamiento Seguir Pasillo no sea el más adecuado en determinadas circunstancias en las pruebas en el entorno de la Figura 3.35a. Aunque el movimiento del robot es correcto, cuando se finaliza el seguimiento del primer pasillo quizás sería más adecuada la utilización del comportamiento Seguir Pared hasta centrarse en el segundo pasillo. Y, más aún, cuando el robot se está intentando centrar en este segundo pasillo, la situación que se presenta es parecida a la de una puerta. Por lo tanto, es razonable el pensar que para explorar de forma completa un entorno total o parcialmente desconocido se necesite más de un comportamiento. De este modo el robot se puede adaptar mejor a la configuración

del entorno en todo momento.

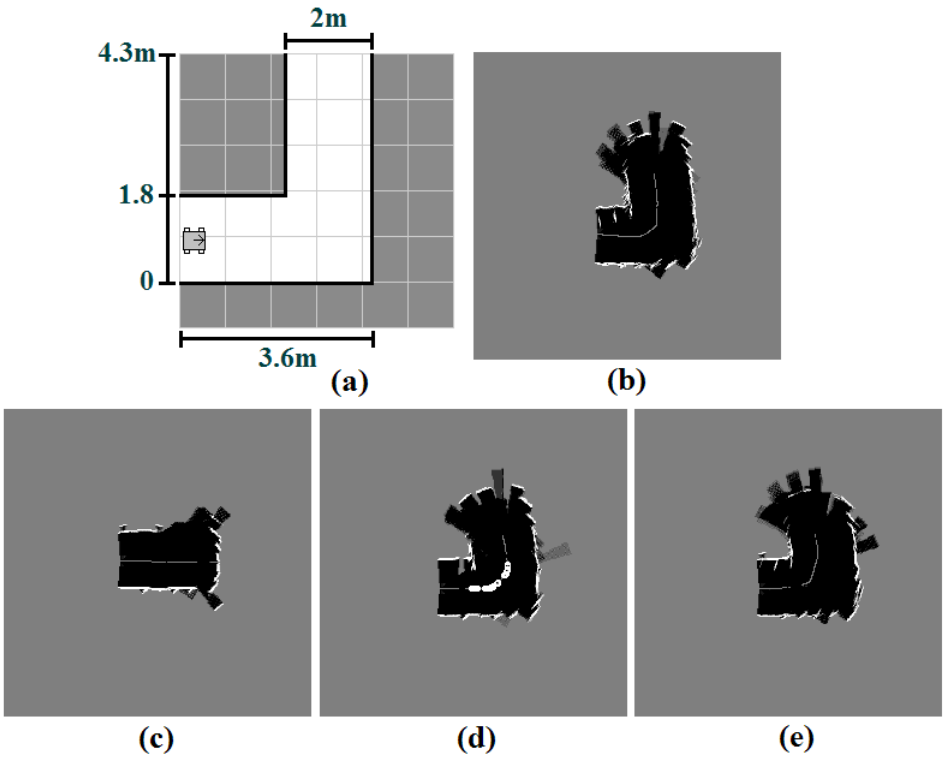

Figura 3.35: Seguir Pasillo, pasillo en L: (a) planta; (b) operación con el modelo analítico; (c) operación con el *CBR*, sin adaptación; (d) operación con el *CBR*, con adaptación; (e) operación con el *CBR*, teniendo en cuenta el aprendizaje por experiencia en (d).

El objetivo del siguiente experimento es poner de manifiesto la operación del agente en pasillos donde existen obstáculos que el robot debe evitar. Esta situación es similar a la concatenación de pasillos de diferentes anchuras. En la Figura 3.36 se muestran dos entornos de prueba. Ambos consisten en un pasillo de $2m$ de ancho con un obstáculo. En el primer entorno (Figura 3.36a), el obstáculo se encuentra en la parte superior, mientras que en el segundo está situado en la parte inferior del pasillo (Figura 3.36b). En estos dos entornos se han realizado dos entrenamientos. La Figura 3.36c muestra la trayectoria recorrida durante el aprendizaje por observación en el entorno de la Figura 3.36a, capturándose 226 casos. La clasificación de esta base de casos da lugar a 36 clases. El segundo entrenamiento se lleva a cabo en el entorno de la Figura 3.36b, cuya trayectoria se muestra en la Figura 3.36d. Este aprendizaje da lugar a 219 casos, clasificándose en 37 clases.

La operación con el modelo analítico en los entornos de las Figuras 3.36a y b se muestra en las Figuras 3.37a y c, respectivamente. Aunque el robot sigue el pasillo de forma adecuada, la trayectoria presenta las oscilaciones habituales de las aproximaciones puramente reactivas basadas en los campos de potencial. Más aún, cualitativamente se observa durante la realización de las pruebas una cierta falta de suavidad y que los cambios de dirección son más abundantes que en las mismas pruebas realizadas con el *CBR*. Cuando se emplea esta segunda estrategia, los resultados que se obtienen permiten decir que la trayectoria es más suave.

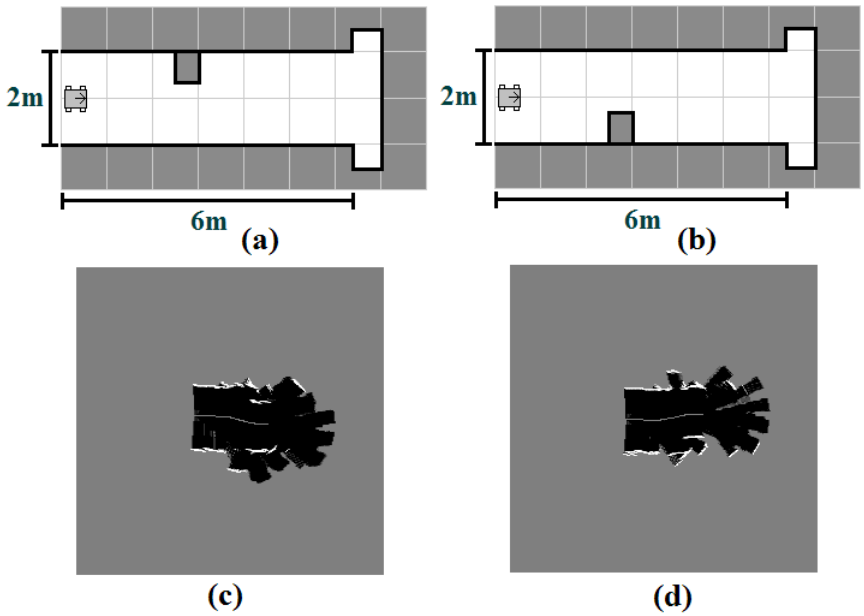

Figura 3.36: Seguir Pasillo, funcionamiento con obstáculos: (a) planta 1; (b) aprendizaje por observación en (a); (c) planta 2; (d) aprendizaje por observación en (b).

En la Figura 3.37b se observa el movimiento del robot en el entorno de la Figura 3.36a cuando se utiliza la base de casos clasificada adquirida en ese mismo entorno. Se aprecia que el robot se centra en el pasillo al reducirse su anchura por la presencia del obstáculo. Además, es importante hacer notar que no se adaptan casos salvo al final de la trayectoria, debido a la detección de la pared frontal. Si este entrenamiento se usa en el entorno de la Figura 3.36b, los resultados difieren en gran medida. Como la base de casos tiene conocimiento sobre la aparición de un obstáculo en la parte superior, cuando éste aparece en la parte inferior del pasillo se adaptan casos. Esta situación se aprecia en la Figura 3.37d. Cuando el obstáculo es detectado, se empiezan a adaptar casos. Cuando el obstáculo es superado, se dejan de adaptar casos. Si se repite la misma prueba pero con el entrenamiento realizado en ese mismo entorno (Figura 3.36d), la trayectoria del robot ya no presenta casos adaptados al detectar el obstáculo, puesto que esta información ya está contenida en la base de casos gracias al aprendizaje por observación realizado. Sí sucede que se adaptan algunos casos, en concreto 6, debido a nuevas situaciones detectadas, fundamentalmente por errores en las lecturas de los sonar.

5.1.3 Cruzar Puerta

Por último, en el tercer comportamiento el robot debe cruzar una puerta por su centro para entrar o salir de una habitación. Si se emplea el modelo analítico de la Sección 2, todas las pruebas se llevan a cabo con el valor $d_{CPSEC} = 300mm$. En el caso de que se emplee la filosofía del *CBR*, el umbral de adaptación de los casos y la distancia mínima entre clases se han establecido a los valores $U_{reusar} = 0.1$ y $Dist\_MaxMin = 1000$, respectivamente. Las bases de casos usadas siempre están clasificadas.

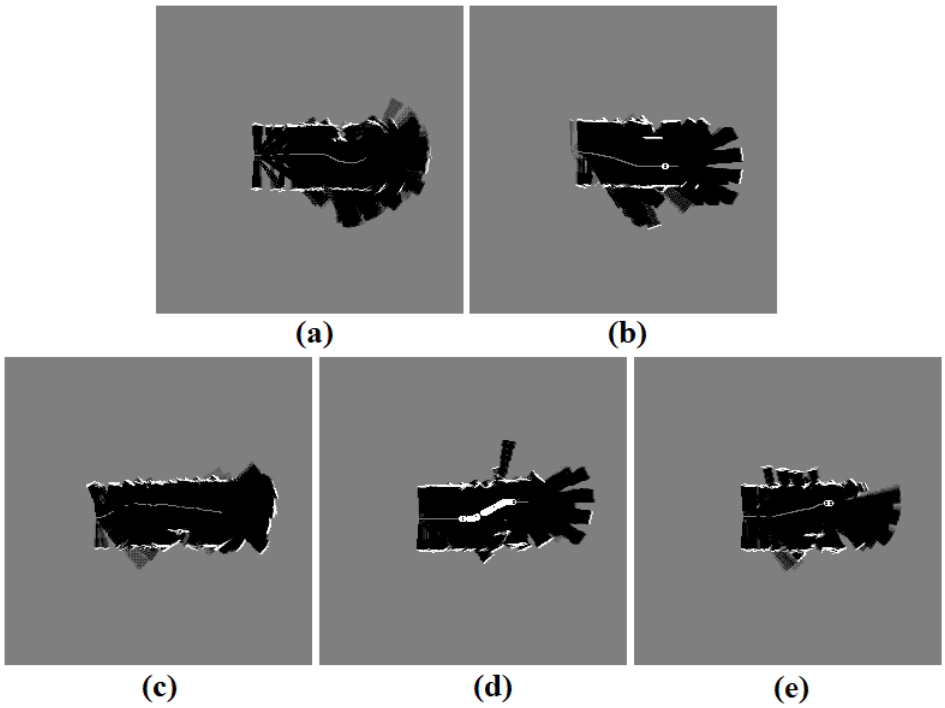

Figura 3.37: Seguir Pasillo, funcionamiento con obstáculos. Entorno de la Figura 3.36a: (a) operación con el modelo analítico; (b) operación con el *CBR*, entrenamiento con obstáculo en la parte superior. Entorno de la Figura 3.36b: (c) operación con el modelo analítico; (d) operación con el *CBR*, entrenamiento con obstáculo en la parte superior; (e) operación con el *CBR*, entrenamiento con obstáculo en la parte inferior.

Las pruebas con el modelo analítico parten del entrenamiento realizado en el entorno de la Figura 3.38a. El aprendizaje es bastante breve, como se observa en la trayectoria recorrida por el agente en dicho entorno (Figura 3.38b). Se adquieren tan solo 174 casos. Tras la clasificación, la base de casos se segmenta en 23 clases. Durante este aprendizaje por observación el operador humano guía al robot a través de una puerta de $1.3m$. El agente parte desde una posición lateral respecto de la puerta, girado unos 30^o respecto del eje perpendicular a su marco. El operador humano ejecuta un movimiento recto y posteriormente un giro hacia la derecha, para centrar el robot en la puerta y así poder atravesarla.

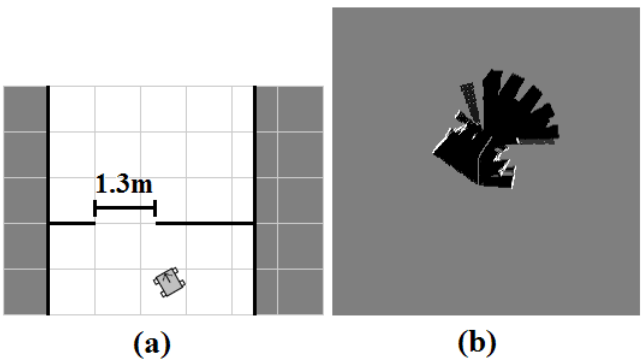

Figura 3.38: Cruzar Puerta, entorno real: (a) planta; (b) aprendizaje por observación.

En la Figura 3.39 se muestran diferentes escenarios donde el robot debe atravesar una puerta empleando el modelo analítico. Existen puertas de diferente anchura, partiendo el robot desde distintas posiciones. Así, tenemos puertas de $1.5m$ (Figuras 3.39a, b y c), de $1.2m$ (Figuras 3.39d, e y f) y de $1m$ (Figuras 3.39g y h). En todos los casos el robot atraviesa la puerta, unas veces más centrado, como en el entorno de la Figura 3.39b, y otras menos, como en el entorno de la Figura 3.39e. Aunque quizás no se aprecia del todo con claridad por la corta longitud de la trayectoria, ésta se obtiene tras una serie de oscilaciones que provocan que el robot deba cambiar varias veces de dirección. Este hecho es debido a la propia implementación del método analítico, basado en los campos de potencial. Aquéllas situaciones en las que el agente se acerca demasiado a uno de los lados de la puerta, por ejemplo la puerta de $1m$ de la Figura 3.39h, nos sugieren que el método puede tener dificultades para atravesar puertas estrechas. Posteriormente se muestra un ejemplo más clarificador a este respecto.

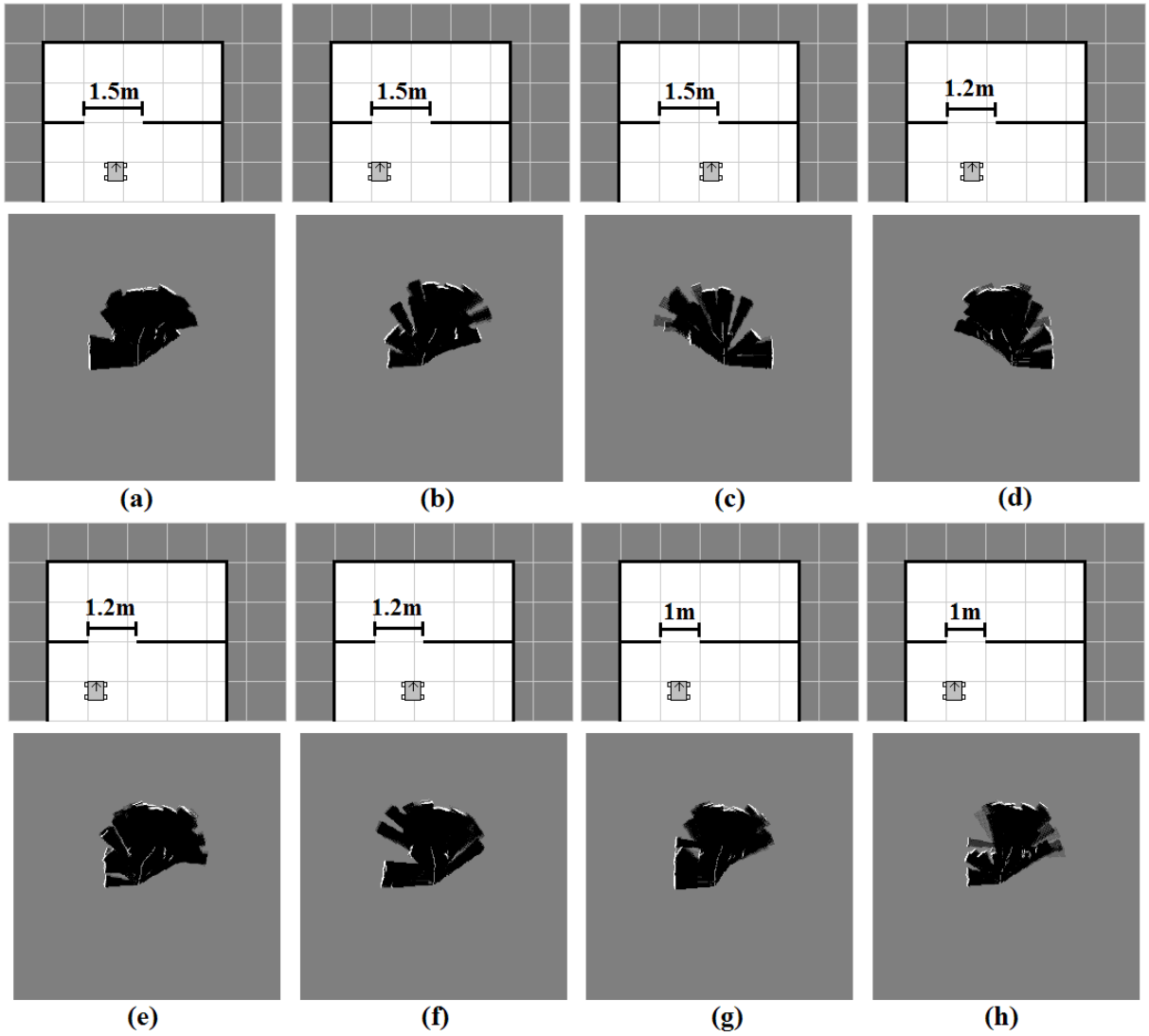

Figura 3.39: Cruzar Puerta, entornos reales con el modelo analítico: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7; (h) 8.

Las pruebas en los entornos de la Figura 3.39 se han repetido con la técnica basada en el *CBR*. La base de casos utilizada para todos los experimentos es la clasificada obtenida tras el aprendizaje por observación realizado en el entorno de la Figura 3.38a. Los resultados de estos tests se presentan en la Figura 3.40.

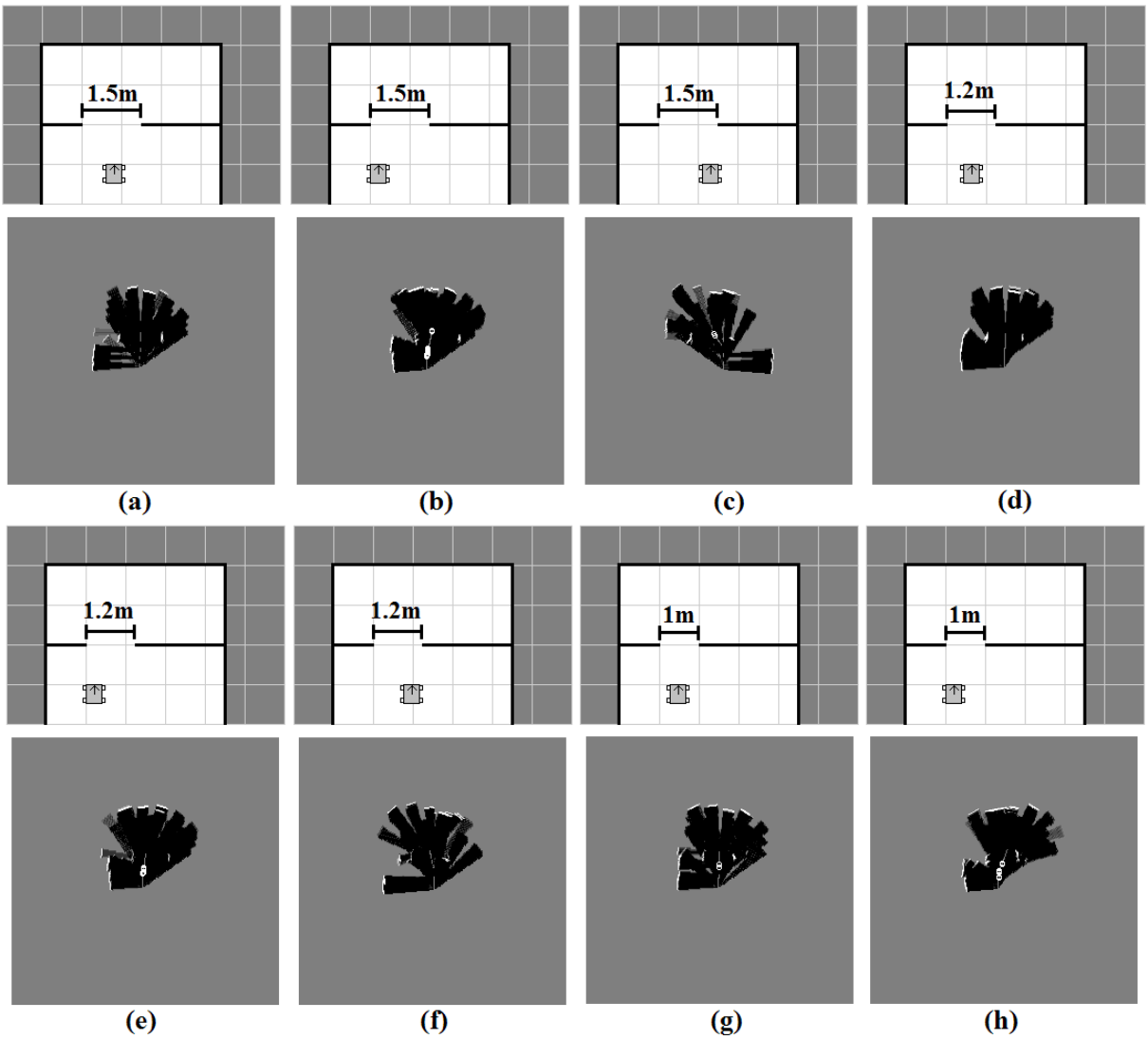

Figura 3.40: Cruzar Puerta, entornos reales con el *CBR*: (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6; (g) 7; (h) 8.

En el aprendizaje por observación que se utiliza en estos tests el robot parte desde una posición no centrada respecto de la puerta, ya que inicia el movimiento más cerca del marco derecho de la puerta que del izquierdo. Este hecho es importante, tal y como se aprecia en los entornos de la Figura 3.40. Se pueden distinguir tres situaciones distintas. La primera es aquélla en la que el robot está centrado respecto de la puerta (Figuras 3.40a, d y g). El robot atraviesa la puerta por su centro, sin oscilar y adaptando casos únicamente cuando la puerta tiene $1m$ de anchura (5 casos en la Figura 3.40g). La base de casos tiene información sobre esta primera situación. Como el entrenamiento realizado es para una puerta de $1.3m$, cuando

la anchura difiera bastante, como en la puerta de $1m$, se adaptan casos. Aun así, el número de casos adaptados es mínimo.

La segunda situación se da cuando el robot está más cerca de la parte derecha del marco de la puerta que de la izquierda (Figuras 3.40c y f). En este caso solamente se adaptan 4 casos en el entorno de la Figura 3.40c. Esta mínima adaptación es debida a que el entrenamiento realizado tiene en consideración la configuración del entorno que el agente se encuentra, partiendo desde la parte derecha de la puerta. Sin embargo, cuando el robot parte desde la parte izquierda de la puerta se adaptan bastantes casos, tal y como se observa en las Figuras 3.40b, e y h. Esto sucede porque, al partir el agente desde la parte izquierda, la situación que el robot encuentra no ha sido aprendida previamente por observación. No obstante, esto no es un problema para el sistema, puesto que este aprendizaje por experiencia se puede usar en el futuro para considerar las situaciones que representa. Por último, comentar que en las pruebas realizadas con el modelo basado en el *CBR* es una constante la suavidad en el movimiento respecto al modelo analítico, así como el menor número de cambios de dirección.

En otra prueba se pretende analizar la influencia del aprendizaje por observación en la adaptación de los casos. Para ello, repetimos la prueba en el entorno de la Figura 3.40b, partiendo de una base de casos distinta. En el entorno de la Figura 3.41a se lleva a cabo un aprendizaje por observación de un operador humano. La trayectoria recorrida por el agente (Figura 3.41b), da lugar a una base de casos con 145 casos que se clasifican en 30 clases. Esta base de casos clasificada es la usada en esta nueva prueba. La trayectoria recorrida por el agente en el entorno de la Figura 3.40b se muestra en la Figura 3.41c. El número de casos adaptados es despreciable, pues es de 3. Como el robot parte desde la parte izquierda de la puerta y la base de casos usada se adquirió con el agente iniciando el movimiento también desde esa parte izquierda, las situaciones que éste encuentra durante la navegación están representadas en la base de casos, al contrario de lo que ocurre en la prueba de la Figura 3.40b.

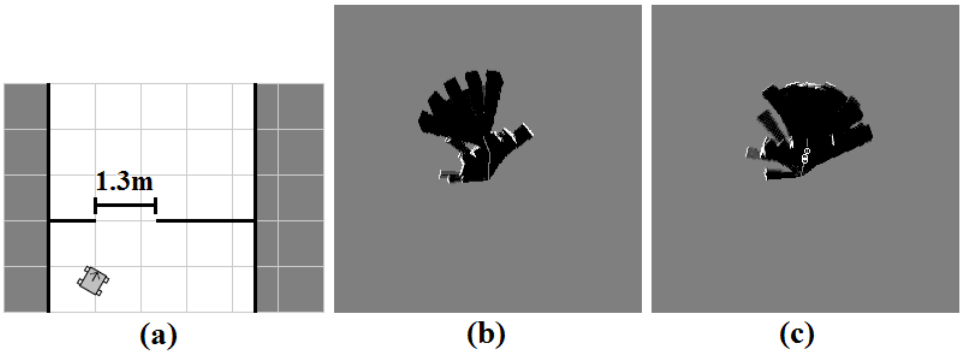

Figura 3.41: Cruzar Puerta, entorno real de prueba; (a) planta; (b) aprendizaje por observación; (c) operación en el entorno de la Figura 3.40b.

En un último experimento se pone de manifiesto una de las ventajas del modelo basado en el *CBR* respecto del modelo analítico. Anteriormente se ha sugerido que el modelo analítico puede tener dificultades a la hora de atravesar una puerta estrecha. Esta situación se muestra en

la Figura 3.42b. Este modelo no permite atravesar la puerta estrecha de 75cm del entorno de la Figura 3.42a. La trayectoria recorrida por el agente es muy corta porque, cuando se encuentra cerca de la puerta, comienza a oscilar indefinidamente a uno y otro lado, impidiendo su paso. Sin embargo, cuando se usa la filosofía del *CBR* sí es posible atravesar la puerta. La base de casos usada para este experimento es la clasificada obtenida tras el entrenamiento en el entorno de la Figura 3.38a. La puerta es atravesada por su centro. Los casos adaptados surgen porque la puerta es bastante más estrecha que la usada para el entrenamiento.

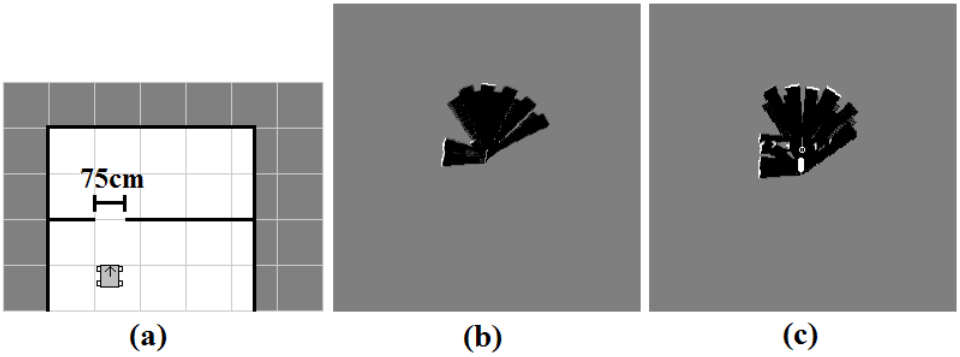

Figura 3.42: Cruzar Puerta, entorno real con puerta estrecha: (a) operación con el modelo analítico; (b) operación con el *CBR*.

5.2 Pruebas del método de aprendizaje asistido

Para comprobar el funcionamiento de la propuesta de control asistido se han realizado varios experimentos donde usuarios voluntarios llevan a cabo, durante una sesión de test, cuatro tareas que se corresponden con los tres comportamientos implementados, usando la plataforma robótica real *Pioneer P2AT* equipada con 8 sensores sonar frontales. A propósito se han incluido personas que nunca han conducido un robot e incluso, en algunos casos, tampoco han conducido un vehículo. De los experimentos se derivan unos resultados desde un punto de vista cuantitativo y cualitativo. A la vez que navega, el robot construye un mapa métrico [Mor88] para mostrar dónde se encuentran los obstáculos y representar gráficamente el entorno. La trayectoria seguida por el robot se muestra superpuesta en color blanco al mapa métrico. Así mismo, comentar que los experimentos se realizaron con $v_{rMAX} = 15^o/s$ y $v_{tMAX} = 200mm/s$. El joystick usado es un *WingMan Attack 2* de *Logitech* para personas diestras. Este último dato no es baladí, puesto que en los experimentos participaron 2 usuarios zurdos, así como gente sin experiencia con videojuegos, pues la elección del humano influye en los resultados.

El objetivo de las pruebas es comprobar:

- Si los resultados usando el control asistido son mejores que los obtenidos únicamente con los comandos del robot.
- Si los usuarios logran aprender a guiar el sistema de forma segura.

- Si los usuarios se sienten cómodos con el control asistido.

Para alcanzar los objetivos planteados se emplea la métrica definida en la Sección 4.2, consistente en la medida de la eficiencia de los comandos de movimiento, η, junto con los tres factores que la conforman, η_{su}, η_{lt} y η_{se}. También se hace uso de un cuestionario que los usuarios rellenan al finalizar su sesión de test, para conocer su opinión sobre el funcionamiento del sistema.

5.2.1 Estudio con usuarios

Los tests se limitan a 13 usuarios voluntarios, ya que investigaciones anteriores sobre el control colaborativo indican que un número de usuarios reducido es suficiente [FTB03]. La edad de los usuarios varía de 23 a 35 años. Se combinan personas relacionadas con la robótica junto con personas que no tienen nada que ver con la disciplina. Se tuvo en cuenta también el incluir un grupo significativo de usuarios sin permiso para conducir coches. Igualmente, se incluyen dos personas zurdas en el experimento a propósito. Finalmente, ya que el robot se conduce con un joystick, se incluyen personas habituadas a los videojuegos y personas que no lo están.

A todos los usuarios se les pidió, durante su sesión de test, que condujeran el robot en cuatro entornos diferentes, de acuerdo a unas guías cualitativas. La Figura 3.43 muestra los cuatro entornos reales de test usados en nuestros experimentos. Desde este momento nos referiremos a ellos como FW, NC, DNC y PD. En el entorno FW los usuarios tenían que seguir el contorno de la pared derecha (Figura 3.43a). En los entornos NC y DNC se les pidió que recorrieran los pasillos por el centro (Figuras 3.43b y c). Finalmente, en el entorno de test PD (Figura 3.43d) debían cruzar la puerta sin chocar. Los comandos del robot en los tests provienen de alguno de los tres comportamientos que se han implementado en la presente Tesis, Seguir Pared, Seguir Pasillo y Cruzar Puerta, para así aprender de forma asistida los mismos.

Todos los voluntarios llevaron a cabo cinco veces cada test, dando lugar a 20 trayectorias. Los usuarios realizaron los tests en distinto orden, porque se quería probar si los conductores que empezaban con las pruebas más complicadas lo hacían mejor que aquéllos que comenzaban por las más sencillas o, por el contrario, lo hacían peor. No se informó a los usuarios sobre el control asistido, sino que únicamente se les dio unas guías que debían seguir para conducir el robot en cada entorno, como si todo el control fuera de ellos sin la ayuda del agente. Sin embargo, en algún punto de los tests la mayoría de ellos se dio cuenta de que estaban recibiendo alguna ayuda en la conducción, sin saber su procedencia.

Al finalizar la sesión de test todos los usuarios rellenaron un cuestionario para conocer su opinión al respecto del funcionamiento del sistema y obtener así una realimentación sobre las pruebas. En el Apéndice B se puede ver el aspecto del cuestionario (Tabla B.1), así como los cuestionarios exactamente reproducidos que cada uno de los usuarios completó (Tablas B.2-B.14). A partir de los datos de los cuestionarios se obtienen las características de los usuarios participantes en las pruebas. Se presentan en la Tabla 3.1. Se debe señalar que casi existe

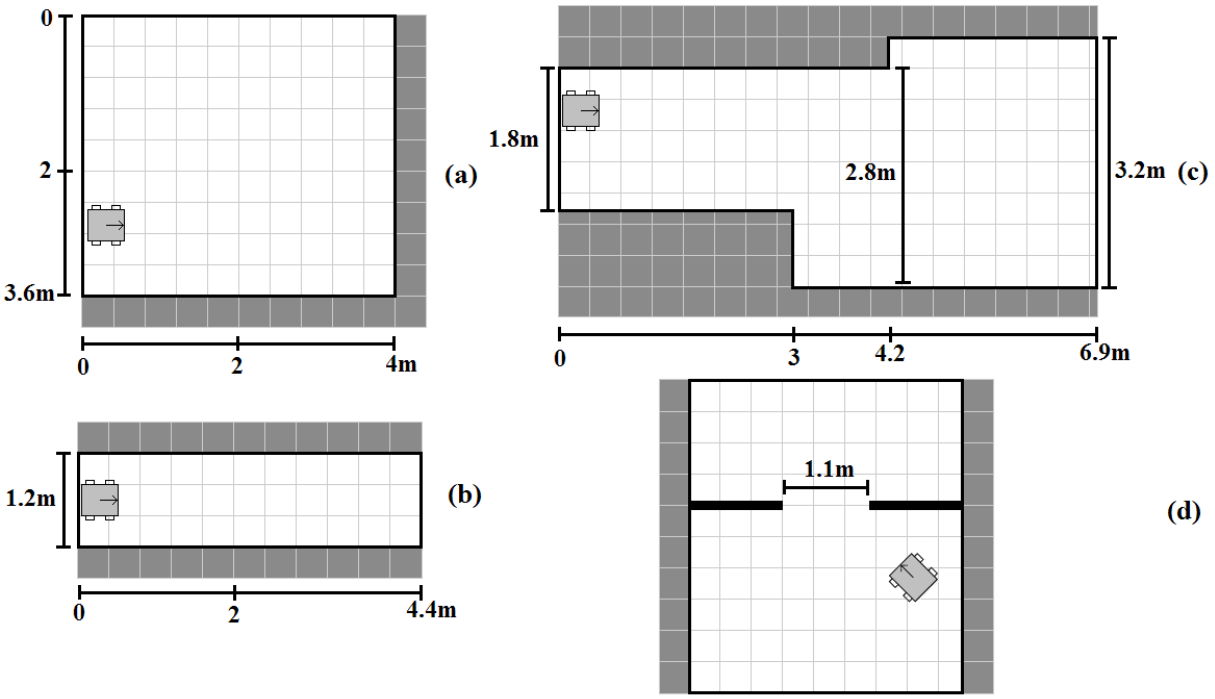

Figura 3.43: Entornos reales: (a) Seguir Pared (FW); (b) Pasillo (NC); (c) Doble Pasillo (DNC); (d) Cruzar Puerta (PD).

paridad, al realizar las pruebas 5 hombres y 8 mujeres, siendo uno de los hombres y una de las mujeres usuarios zurdos. Según se ha comentado anteriormente, se han introducido a propósito personas que no disponen de permiso para conducir coches, en concreto 5. También es significativo el hecho de que 9 personas nunca u ocasionalmente usan videojuegos. Finalmente, hay 8 usuarios que no tienen o tienen muy poca relación con la robótica, tal y como se buscaba.

Sexo	Hombre	5	38.5%
	Mujer	8	61.5%
Permiso para conducir coches	Sí	8	61.5%
	No	5	38.5%
Pulso	Regular	5	38.5%
	Bueno	8	61.5%
Uso gafas habitualmente	Sí	7	53.8%
	No	6	46.1%
Uso videojuegos	Nunca	3	23.1%
	Ocasionalmente	6	46.2%
	Semanalmente	3	23.1%
	Casi diariamente	1	7.7%
Relación con la robótica	Ninguno	7	53.8%
	Poco	1	7.7%
	Bastante	1	7.7%
	Mucho	4	30.8%

Tabla 3.1: Características de los usuarios.

5.2.2 Resultados del funcionamiento del robot

En la Figura 3.44 se muestran las trayectorias seguidas por el robot en los entornos de test FW, NC, DNC y PD cuando se consideran únicamente sus comandos. Es decir, en este caso no se tienen en cuenta los comandos introducidos por el usuario mediante el joystick. La trayectoria seguida por el agente se encuentra superpuesta en color blanco al mapa métrico, según se comentó anteriormente. En la Figura 3.44a el agente debe seguir, en todo momento, la pared que se encuentra a su derecha a una distancia fija. En la Figura 3.44b se supone que el sistema autónomo debía recorrer el pasillo navegando por el centro del mismo. Esta misma tarea de navegación tenía que llevarse a cabo en el entorno de la Figura 3.44c, donde el pasillo presenta varias anchuras distintas. Finalmente, el propósito del robot en el entorno de la Figura 3.44d fue cruzar la puerta sin colisionar. Como durante estas pruebas el robot navega según el modelo analítico propuesto en esta Tesis, todas las trayectorias se ven afectadas por los problemas de los métodos basados en los campos de potencial, esto es, las oscilaciones y la dificultad para moverse entre obstáculos cercanos.

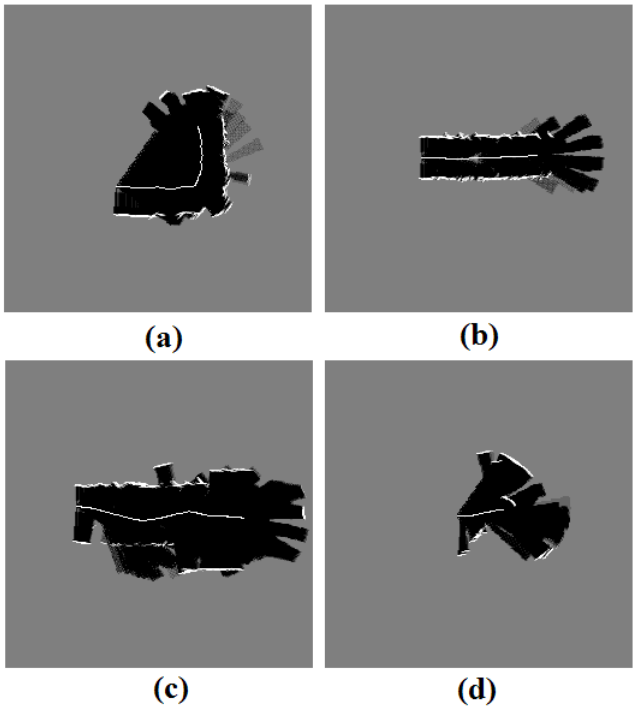

Figura 3.44: Trayectorias llevadas a cabo por el robot: (a) entorno FW; (b) entorno NC; (c) entorno DNC; (d) entorno PD.

Para evaluar la eficiencia, η, a lo largo de toda la trayectoria se usa nueva representación gráfica y estadística básica. Gráficamente, se le asigna a cada uno de los factores involucrados en la eficiencia un canal de color *RGB* (*Red Green Blue*). El rojo se emplea para el factor de suavidad, η_{su}, el verde para el factor de longitud de la trayectoria, η_{lt}, y el azul para el factor de seguridad, η_{se}. A tal efecto, se emplea una representación 3D donde el plano XY representa el espacio geométrico del entorno de trabajo, mientras que al eje Z se le asigna la eficiencia

η en cada punto de la trayectoria. Esta eficiencia vendrá representa con el color determinado por cada uno de los tres factores que la conforman. Cuando la representación se proyecta en el plano XY es posible conocer la manera en la que cada uno de los factores influye en la eficiencia total en cada punto de la trayectoria, gracias al color. Si se proyecta en el plano XZ, la altura de los puntos sirve además para conocer la relación de eficiencias entre zonas distintas de la trayectoria. Por lo tanto, esta representación 3D es muy útil. Mediante un simple vistazo a la misma se puede tener una noción bastante exacta del comportamiento del robot a lo largo de la trayectoria.

Las Figuras 3.45 y 3.46 muestran el comportamiento del robot en los cuatro entornos de la Figura 3.43 cuando el movimiento es completamente autónomo. Para cada entorno se muestran 3 gráficas: i) la representación 3D de la eficiencia en cada punto de la trayectoria; ii) la proyección sobre el plano XZ de la representación 3D; y iii) la proyección sobre el plano XY de la representación 3D. Se puede apreciar que todas las trayectorias presentan oscilaciones, típicas en los modelos analíticos basados en los campos de potencial. En las Figuras 3.45c, 3.45d, 3.46c y 3.46d se observa que, en general, la eficiencia global del control totalmente autónomo, η_R, es bastante alta. En concreto, la eficiencia media es de 74.13%, 91.90%, 80.84% y 58.84% en los tests FW, NC, DNC y PD, respectivamente (Tabla 3.2). Es esperable que los factores de la eficiencia que dependen de las oscilaciones, η_{su} y η_{lt}, decrezcan cuando el agente se mueve cerca de los obstáculos (Figuras 3.45c y 3.46d). La seguridad, en cambio, debería ser la adecuada dependiendo de los umbrales usados en los comportamientos. Su valor, η_{se}, está por encima del 75% en todos los tests, ya que el robot intenta evitar los obstáculos durante la navegación.

Test	FW	NC	DNC	PD
Eficiencia media (%)	74,13	91,90	80,84	58,84
Varianza de la Eficiencia (%)	21,89	2,79	11,81	9,01
Suma de la Función de Curvatura (^{o})	101,15	1,68	-1.22	2,64
Varianza de la Función de Curvatura (^{o})	5,89	0,56	8,98	7,02

Tabla 3.2: Resultados estadísticos del funcionamiento del robot.

Las eficiencias medias presentadas en la Tabla 3.2 cuantifican el funcionamiento del robot a lo largo de toda su trayectoria. Sin embargo, existen situaciones donde la eficiencia es menor que la media. Una de estas situaciones podemos encontrarla en el test FW. La eficiencia media decae por debajo del 50% mientras el robot está girando en la esquina. En las Figuras 3.45a, 3.45c y 3.45e, el giro da lugar a un color azul de la eficiencia, lo que se corresponde con una trayectoria segura donde los factores η_{su} y η_{lt} tienen una baja contribución a η. De hecho, la media de los factores de suavidad y de longitud de la trayectoria está por debajo del 25%, mientras que la media del factor de seguridad supera el 95%. La Tabla 3.2 presenta también la suma y la varianza de la función de curvatura obtenida para la trayectoria del robot. La función de curvatura se calcula con el método presentado en [PUdTS07]. Debido al giro en la esquina (Figura 3.45e), la suma de la función de curvatura debería ser de 90^o en el test FW. En realidad, este valor esta próximo a los 100^o, ya que el móvil oscila en el experimento debido al modelo analítico empleado para el comportamiento. Más aún, las oscilaciones también se pueden

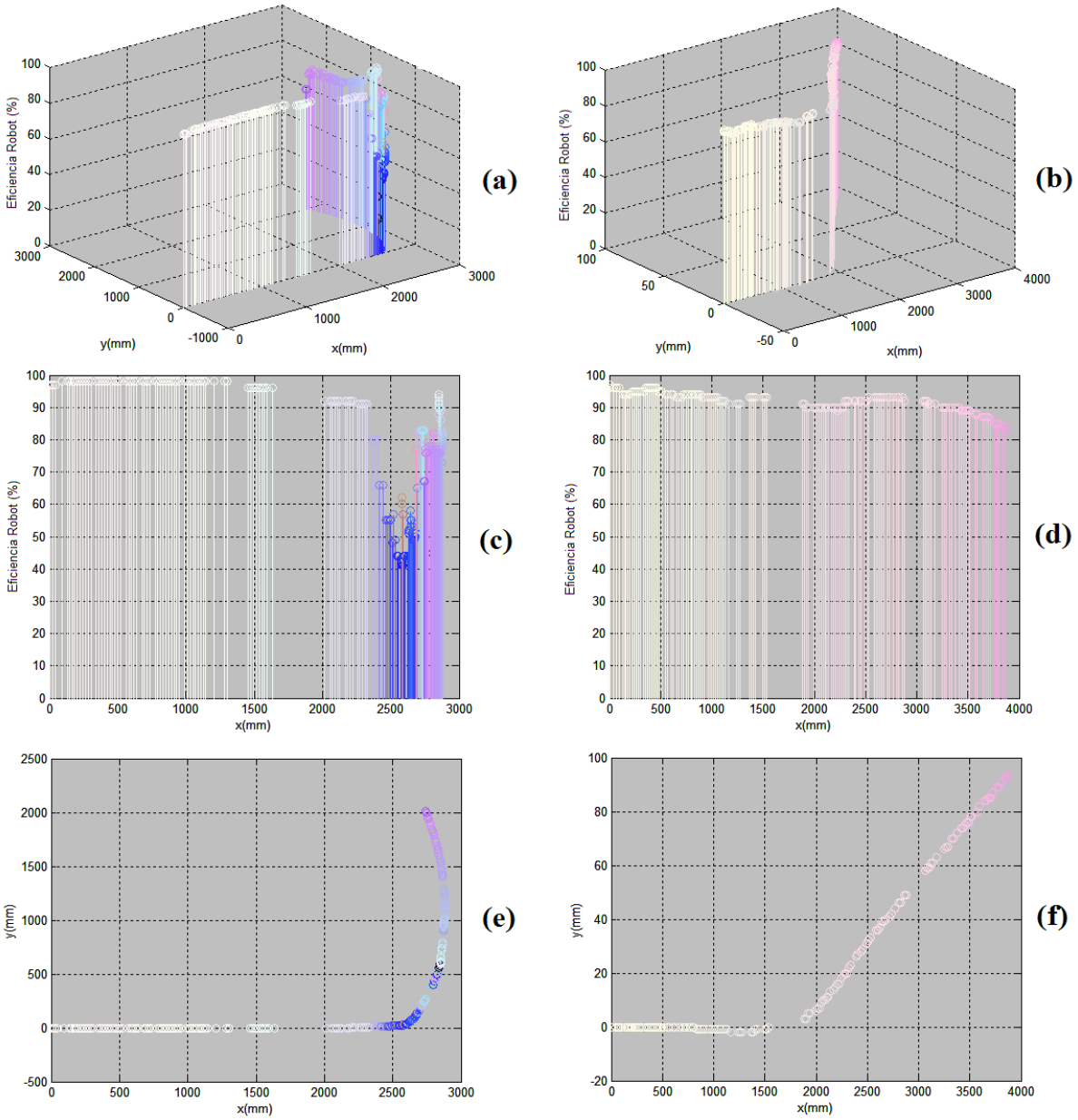

Figura 3.45: Funcionamiento del robot en los entornos de test FW y NC: (a) representación 3D de la eficiencia del robot en el test FW; (b) representación 3D de la eficiencia del robot en el test NC; (c) proyección XZ de la eficiencia del robot en (a); (d) proyección XZ de la eficiencia del robot en (b); (e) proyección XY de la eficiencia del robot en (a); (f) proyección XY de la eficiencia del robot en (b).

apreciar en el test DNC (Figura 3.44c) mientras el robot intenta centrarse en el pasillo. Estas oscilaciones tienen que ver con las dos zonas de la trayectoria con un color azul de la eficiencia en la Figura 3.46e. En ambos casos la eficiencia total es menor del 68%, es decir, significativamente inferior a la media (80.84%). Si en ambos casos se analizan los tres factores de la eficiencia η, se observa que: i) la trayectoria es segura, ya que la media del factor de seguridad es superior al 91%; y ii) la media de los factores de suavidad y de longitud de la trayectoria es inferior al 60%.

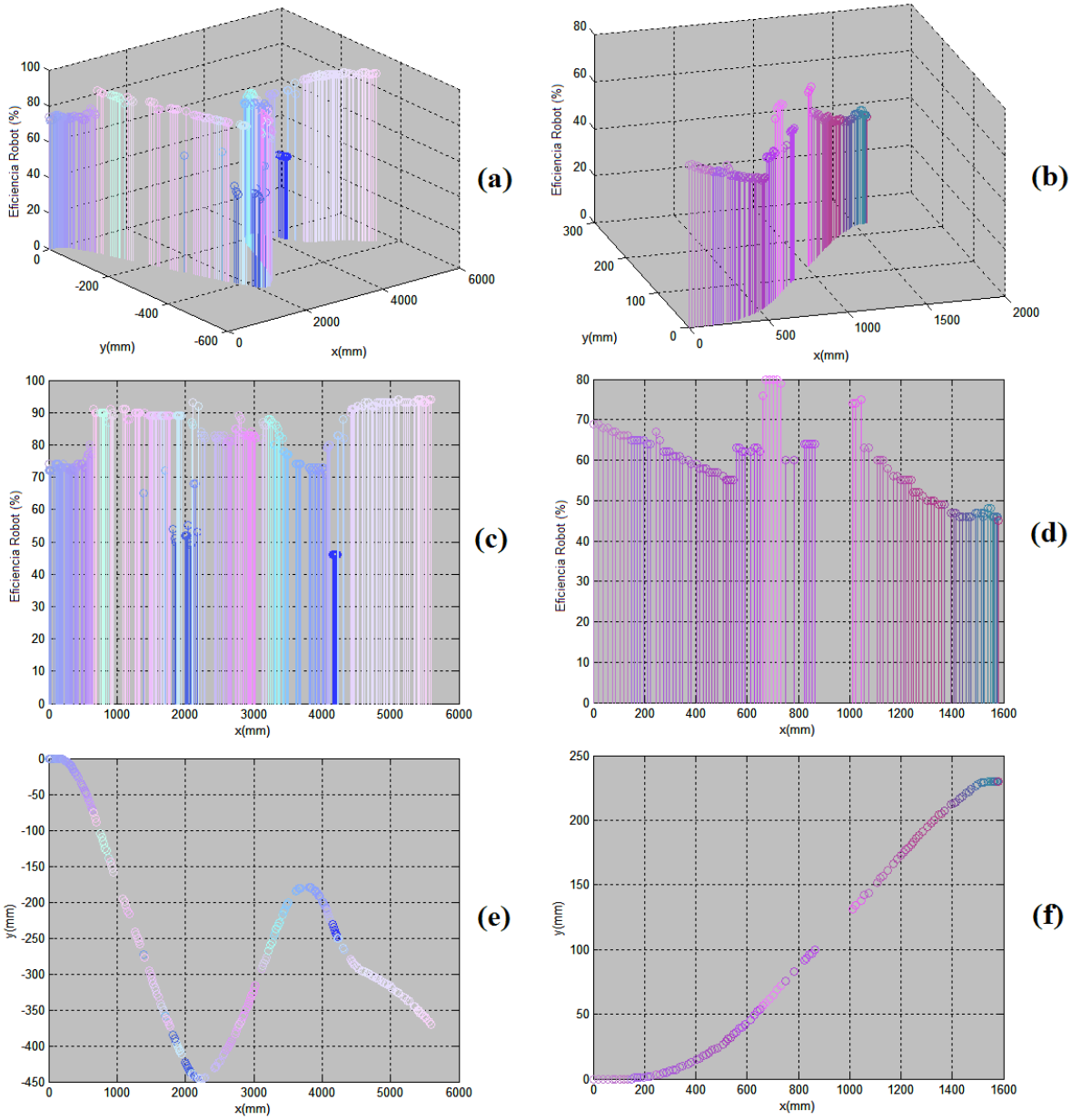

Figura 3.46: Funcionamiento del robot en los entornos de test DNC y PD: (a) representación 3D de la eficiencia del robot en el test DNC; (b) representación 3D de la eficiencia del robot en el test PD; (c) proyección XZ de la eficiencia del robot en (a); (d) proyección XZ de la eficiencia del robot en (b); (e) proyección XY de la eficiencia del robot en (a); (f) proyección XY de la eficiencia del robot en (b).

5.2.3 Resultados del funcionamiento del control asistido

Desde este punto y en lo sucesivo, los comandos de movimiento aplicados a los motores del robot serán una combinación lineal de la entrada del usuario (joystick) y de los comportamientos autónomos, lo que hemos denominado control asistido. Debemos hacer notar que estos comandos de movimiento no producirán necesariamente las trayectorias de la Figura 3.44. De hecho,

genéricamente las trayectorias serán distintas, puesto que no solo dependen en cada punto de los comandos autónomos, sino también del usuario que conduce el robot. Esto nos va permitir comparar en los experimentos el control asistido con el robot en cada punto de la trayectoria.

Un primer hecho interesante que se extrae de las pruebas es que las personas que guardan relación con la robótica realizan las cuatro pruebas de una manera similar. En media, alcanzan unas eficiencias del control asistido del 79%, 91%, 79% y 66% en los tests FW, NC, DNC y PD, respectivamente. Sin embargo, algunos individuos registraron diferencias en la suavidad del control. Para comparar la conducción de los usuarios se evaluó la variación de la curvatura en cada punto, calculada de manera simple como la diferencia entre la orientación actual del robot y la que trata de imponer el humano. A este respecto, todos los usuarios cumplieron la tarea de una forma similar en el entorno NC, donde la trayectoria se supone que tiene que ser recta. Así mismo, tampoco hubo mucha variación durante el cruce de la puerta (entorno PD). Sin embargo, se registraron diferencias significativas en los entornos FW y DNC, debidas principalmente a que existía un mayor grado de libertad a la hora de girar en el test FW y de corregir la trayectoria en el test DNC.

La curvatura considerada en el párrafo anterior es local. Para evaluar las diferencias reales en las decisiones globales también se midió la variación de la curvatura a lo largo de toda la trayectoria, calculada con la técnica presentada en [PUdTS07]. Este dato depende del guiado a medio plazo, entendiendo éste como que el humano decide empezar a girar antes o después en función de su percepción global del entorno para evitar movimientos bruscos. Estas decisiones están influenciadas principalmente por la manera de conducir. En este caso, todos los usuarios tenían permiso para conducir coches y sabían como conducir el robot, al estar relacionados con la robótica. Los cambios en la curvatura global fueron comparados con las trayectorias reactivas, encontrándose que, en todos los casos, la suavidad es mejor en los humanos que en el control autónomo para los tests FW y PD. Este dato era esperable, pues en ambos casos el modelo analítico de comportamiento es proclive a oscilar. En los pasillos (entornos NC y DNC), sin embargo, las variaciones de la curvatura del control asistido eran similares a las del robot funcionando solo. Incluso, en algunos casos, el control asistido era ligeramente peor. Esto es debido a que los humanos tratan de corregir su curvatura en todo momento para mantenerse en el centro del pasillo, mientras que el sistema autónomo consigue alcanzar un mejor equilibrio. En este sentido, es interesante hacer notar que los conductores que realizan la prueba peor que el sistema autónomo tienen trayectorias menos nerviosas, cuyo significado es que mantienen la orientación todo lo posible, pero tienen que corregirla ligeramente en todo momento. Los usuarios que superan a la máquina en estos casos tendían a giran de una manera más brusca cuando era necesario, pero no corregían su trayectoria el resto del tiempo. No obstante, en este grupo de conductores la eficiencia media del control asistido era mejor o similar a la del robot en todos los casos. La única explicación que encontramos a esta forma de conducir es que la experiencia sobre el funcionamiento del robot hace que los conductores sospechen lo que el robot va a hacer y, a propósito o no, intentan compensar las debilidades del robot cuando es necesario.

En la Figura 3.47 se comparan las trayectorias de un usuario relacionado con la robótica y

de un usuario que no lo está en el test DNC. Las Figuras 3.47a y 3.47b muestran, para el usuario relacionado con la robótica, la representación 3D de la eficiencia del control asistido a lo largo de la trayectoria y dicha trayectoria, respectivamente. Las Figuras 3.47c y 3.47d presentan los mismos gráficos para el usuario no involucrado en la robótica. La persona que sabe de robótica gira bruscamente al comienzo de la prueba para centrar al robot en el pasillo. Una vez que el robot está centrado, su trayectoria no se ve corregida, ya que gracias a la experiencia en la disciplina compensa dicha trayectoria únicamente cuando se hace necesario. Por el contrario, el otro usuario intenta centrarse en el pasillo de una manera más suave. Sin embargo, el usuario debe corregir la trayectoria porque no ha tenido en cuenta al robot. La eficiencia media del control asistido para el primer usuario es de 80.4%, similar a la del robot en su funcionamiento autónomo aislado (80.84%), mientras que esa misma eficiencia media es ligeramente inferior para el usuario que no sabe de robótica (77.5%). La pequeña diferencia se debe a que el robot ayuda a los usuarios en todo momento. Además, la eficiencia al principio de la trayectoria es mejor en la Figura 3.47a que en la Figura 3.47c, es decir, es mejor para el usuario relacionado con la robótica que para el que no lo está. Ambas zonas iniciales de la trayectoria dan lugar a una eficiencia representada en color azul. Más aún, la eficiencia media en esta zona es del 75.1% para la persona con relación con la robótica, mientras que para la que no tiene relación es de tan solo el 66.7%. El factor de suavidad, η_{su}, es del 67.9% y del 49.5%, respectivamente. Por lo tanto, la persona que conoce el funcionamiento del robot consigue que la trayectoria sea más suave, incluso cuando el giro al comienzo ha sido más brusco. Por otro lado, el usuario que no tiene relación con la robótica intenta guiar al agente de forma más suave. Sin embargo, el control que la máquina también ejerce hace que disminuya la eficiencia al principio del movimiento, con un factor de suavidad bastante más bajo.

Para evaluar el proceso de aprendizaje durante un test, definimos la tasa de aprendizaje TA. Se calcula como la relación entre la eficiencia media de los comandos de movimiento del humano y la eficiencia media de los comandos de movimiento del robot a lo largo de toda su trayectoria, para estimar el porcentaje de control que ejerce el humano y el robot:

$$TA = \frac{Media(\eta_H)}{Media(\eta_R)} \tag{3.32}$$

La Figura 3.48a muestra la evolución de la tasa de aprendizaje para un usuario relacionado con la robótica a lo largo de las cinco iteraciones del test FW. TA varía entre 0.89 y 0.97. El mayor valor se da en la primera iteración. Además, no existen diferencias significativas entre las cinco iteraciones, puesto que TA presenta un valor cercano a una constante. Esta situación nos indica que el usuario conoce cómo opera el robot desde la primera hasta la última iteración del test. Por lo tanto, y tal como se esperaba, el proceso de aprendizaje no es tal, ya que el usuario tiene en cuenta el control asistido desde el principio.

Un segundo hecho interesante que se ha observado es que las personas que no tienen permiso para conducir coches y que no son aficionadas a los videojuegos, no estando tampoco relacionadas con la robótica, tienden a realizar los experimentos mejor que el resto. Esto es así especialmente

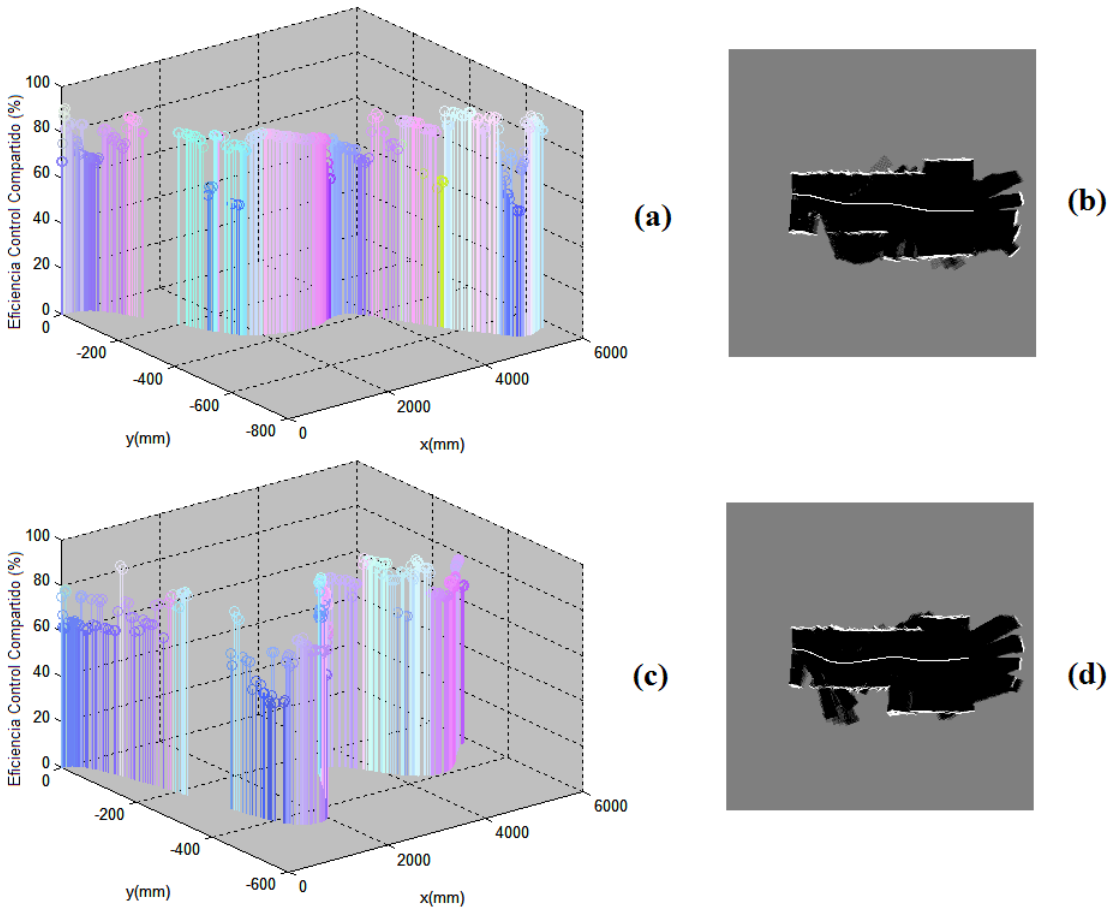

Figura 3.47: Usuario relacionado con la robótica: (a) representación 3D de la eficiencia del control asistido en el test DNC; (b) trayectoria. Usuario no relacionado con la robótica: (c) representación 3D de la eficiencia del control asistido en el test DNC; (d) trayectoria.

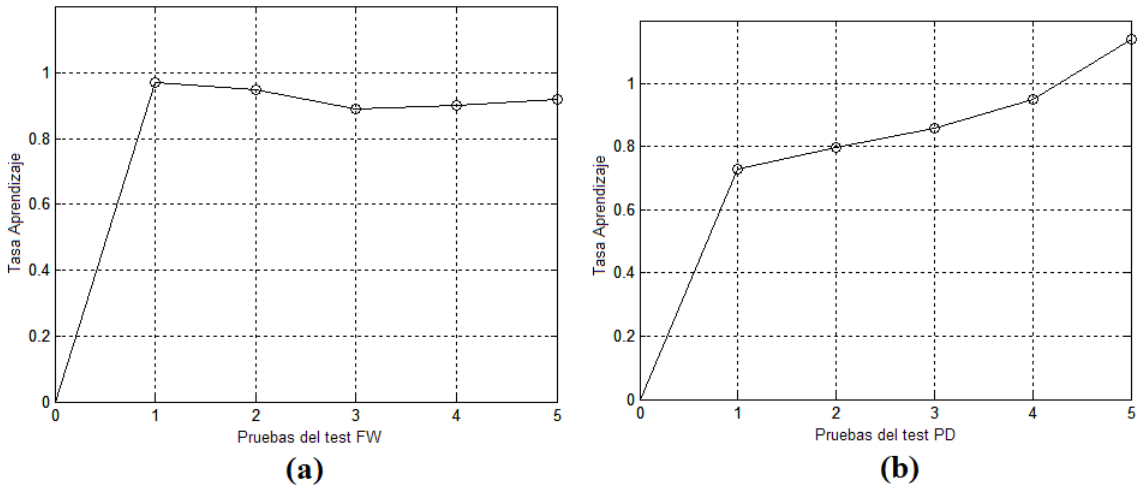

Figura 3.48: (a) Tasa de aprendizaje en el test FW; (b) tasa de aprendizaje en el test PD.

en cuanto a la adaptación, presentado una curva de aprendizaje pura. Esto nos lleva a pensar que

fueron más cuidadosos en la conducción y que, aunque típicamente condujeron de una manera más lenta, se aseguraron de mantener las premisas de los experimentos estrictamente. En media, este grupo de usuarios alcanzó una eficiencia del control asistido del 80%, 90%, 79% y 65% para los entornos de test FW, NC, DNC y PD, respectivamente. A simple vista estos resultados parece que son similares a los de los del grupo de usuarios con experiencia. Sin embargo, hay una diferencia significativa en sus curvas de aprendizaje. Además, para uno de los individuos la eficiencia media del control asistido superó al robot en todos los casos, con un 80.8%, 93.8%, 88.4% y 65.4% en los test FW, NC, DNC y PD, respectivamente (ver Tabla 3.2). En ningún otro caso se detectó una situación similar. Los demás individuos realizaron las pruebas de una forma muy parecida a la del resto, obteniendo medias superiores al robot en los experimentos FW y PD, siendo esta media ligeramente inferior en los pasillos. Aunque todos los usuarios se sintieron cómodos con el sistema, algunos de ellos se dieron cuenta de que la máquina estaba ejerciendo cierto control sobre el funcionamiento global en los test FW y PD, pero no notaron este control en las pruebas NC y DNC.

La Figura 3.49 muestra una de las cinco iteraciones en el test FW por parte de un usuario que siente que la máquina está controlando parcialmente el sistema. Cuando el usuario estaba llevando a cabo la tarea de seguimiento de la pared notó que el movimiento estaba siendo controlado, además de por su joystick, por la propia máquina. Esta situación se dio mientras estaba girando en la esquina, lo que se corresponde con la zona verde y azul de la Figura 3.49a. En esta prueba el usuario no tenía experiencia con el sistema. Esto se confirmó al comprobar que se acercaba mucho a la pared antes de girar, con una media de la eficiencia η_H en torno al 38%. En este contexto el robot intenta seguir la pared a la distancia prefijada, con una media de la eficiencia η_R del 50%. Por lo tanto, la media de la eficiencia del control asistido es de tan solo el 44%, donde el comportamiento autónomo tiene un mayor peso. Aunque el robot se mueve a una distancia suficiente de los obstáculos para no poner en peligro la seguridad, no realiza la tarea correctamente, porque no se debería haber acercado tanto a la pared a la hora de girar.

En tercer lugar, se comparó la realización de las pruebas de las personas habituadas a los videojuegos sin permiso para conducir coches y viceversa. La conclusión es que los jugadores llevaron a cabo las pruebas, en media, mejor desde un punto de vista cuantitativo. En concreto, porque la curvatura global fue mejor conservada. Creemos que este dato está influenciado principalmente por el interfaz escogido, el joystick, y porque las personas no iban montadas en el robot, sino que lo tuvieron que guiar a distancia.

En cuanto al orden de los experimentos, la observación más evidente es que los usuarios que empezaron por los más simples, FW y NC, realizaron mejor la prueba más complicada, PD, desde la primera iteración, desde el punto de vista de la curvatura y la eficiencia, que aquéllos que empezaron por los tests DNC o PD. En este sentido, la Figura 3.48b muestra el proceso de aprendizaje de un usuario en el test PD. Este usuario comenzó con los experimentos más fáciles, primero el entorno FW y después el NC. A continuación siguió por el entorno DNC, finalizando con la prueba más complicada, el test PD. La Figura 3.48b presenta una representación gráfica de la tasa de aprendizaje de este usuario a lo largo de las cinco iteraciones de su último test.

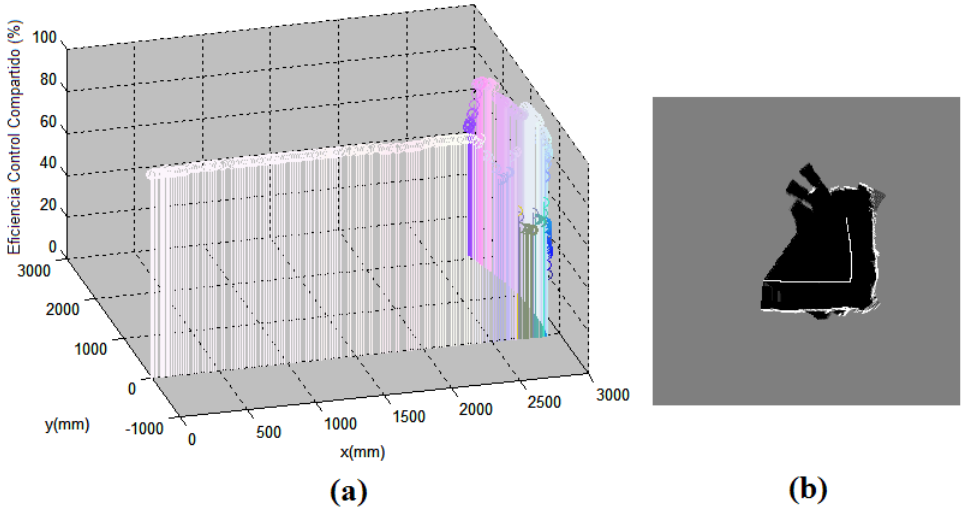

Figura 3.49: El usuario percibe el control de la máquina: (a) representación 3D de la eficiencia del control asistido en el test FW; (b) trayectoria.

Se puede apreciar que TA tiene un valor de 0.73 en la primera iteración, incrementándose hasta 1.14 en el último intento a medida que el número de iteraciones crece. Se debe hacer notar que TA crece desde la primera hasta la quinta iteración. Este hecho nos revela un proceso de aprendizaje nítido. Además, es destacable que el humano es incluso más eficiente que el robot en la última iteración ($TA = 1.14$). Por lo tanto, el humano ejerce mayor control del móvil que el comportamiento autónomo, gracias al proceso de aprendizaje que ha experimentado desde el principio de la sesión en el entorno FW hasta la finalización de la misma con el experimento PD.

Cuando el usuario comienza con uno de los experimentos más complicados (DNC o PD), la situación es diferente. La Figura 3.50 muestra el proceso de aprendizaje de un usuario en el test NC, cuya primera prueba es en el entorno PD, seguida de los entornos FW y DNC. TA varía entre 1.02 y 1.10. El mayor valor de TA se da en el primer intento (1.10), para continuar con una tasa constante a lo largo de las otras cuatro iteraciones. Se debe señalar que: i) el usuario ejerce más control que el robot en los cinco intentos del experimento; y ii) el proceso de aprendizaje no existe en este caso, ya que TA mantiene un valor aproximadamente constante en las cinco iteraciones. Debido al aprendizaje previo, obtenido en las pruebas más complicadas, el usuario ha aprendido las peculiaridades sobre el robot y el control asistido. De ahí que en este test el usuario haya conseguido un porcentaje de control del sistema mayor al de la máquina.

Finalmente, desde el punto de vista de la eficiencia no se han observado diferencias significativas entre las personas diestras y las zurdas participantes en los experimentos, excepto en el test FW. En este entorno los usuarios debían guiar al robot a una distancia constante de la pared derecha, tarea que parece más sencilla para las personas diestras, entre otras cosas porque el joystick empleado es para diestros. La Figura 3.51 presenta la trayectoria seguida por el robot en el test FW para un usuario zurdo (Figura 3.51a) y uno diestro (Figura 3.51b). Como el joystick está adaptado para personas diestras, la Figura 3.51a muestra los problemas

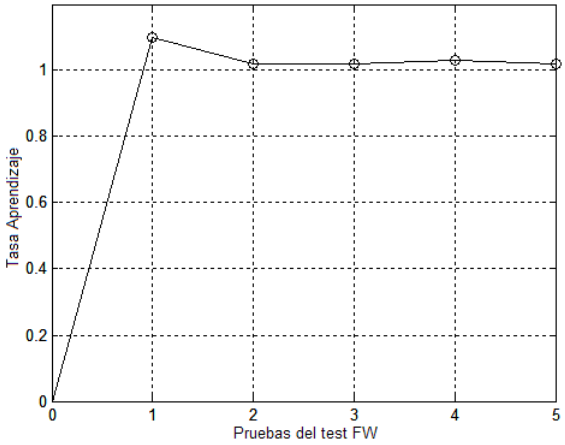

Figura 3.50: Tasa de aprendizaje en el test NC.

con los que se enfrentan las personas zurdas en este experimento. En primer lugar, los usuarios zurdos tienen dificultades para mantener el robot paralelo a la pared derecha, mientras que los diestros no se topan con este inconveniente. En segundo lugar, las personas zurdas no giran a la izquierda en la esquina de una forma tan correcta como las diestras. Por último, según se aprecia en la Figura 3.51a, los usuarios zurdos no dirigen el robot manteniendo una distancia constante a la pared, mientras que los diestros sí lo hacen, incluso después de girar en la esquina (Figura 3.51b). Desde un punto de vista subjetivo, los dos usuarios zurdos no se encontraban cómodos durante este test. Sin embargo, los diestros sí lo estuvieron. Más aún, los usuarios zurdos expresaron cualitativamente los problemas que tuvieron para dirigir el robot, mientras que ninguno de los usuarios diestros expresó dificultad alguna.

Figura 3.51: Trayectoria recorrida por el robot en el test FW: (a) usuario zurdo; (b) usuario diestro.

6 Conclusiones

En este capítulo se han presentado los tres comportamientos de navegación reactiva implementados en nuestro sistema para que el robot se mueva por el entorno. Son los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta. La estrategia seguida es doble. Por un lado se

han desarrollado según un modelo analítico. Por otro, se han diseñado basándose en la filosofía del *CBR*.

El método analítico de implementación de comportamientos está basado en los campos de potencial. La tarea requerida se alcanza mediante una suma de fuerzas. El número de fuerzas sumadas para los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta es de tres, tres y dos, respectivamente. Aunque las pruebas han demostrado la corrección de la operación del robot para cada una de las tres tareas designadas, esta técnica presenta algunos inconvenientes, principalmente que la trayectoria recorrida por el agente presenta oscilaciones y cambios de dirección. Además, hay situaciones en las que no se puede ejecutar la tarea, como es el paso de una puerta de pequeña anchura.

La aproximación basada en el *CBR* que se propone en la presente Tesis es válida para cualquier comportamiento, no solo para los tres usados en el sistema de exploración. El diseño de nuestros tres comportamientos se realiza mediante el ciclo *CBR*, siendo este esquema novedoso una de las aportaciones de la Tesis. En nuestro sistema se emplea la discretización durante la fase de recuperación. En cuanto a la fase reusar, la principal característica es la adaptación basada en el modelo analítico implementado. Uno de los aspectos más importantes de cualquier esquema basado en el *CBR* es la fase de aprendizaje. Esta es una de las ventajas del *CBR* frente al modelo analítico propuesto. Como se puede entrenar el sistema para ejecutar cualquier tarea, aquellas situaciones difíciles de solucionar con el modelo analítico pueden ser aprendidas por el sistema y así evitar los problemas. Además, como existe un proceso de aprendizaje por experiencia, aun en el caso de que alguna situación no haya sido aprendida anteriormente, sí puede ser adquirida y usada en el futuro para resolver problemas similares. Así, por ejemplo, el paso de una puerta estrecha que no se puede realizar con el modelo analítico sí puede ser acometido con la filosofía del *CBR* sin más que entrenar adecuadamente el agente.

Una conclusión común que se extrae de las dos estrategias seguidas es que un único comportamiento no es suficiente para alcanzar el objetivo de exploración completa de un entorno total o parcialmente desconocido. Esto puede ser así porque no se puede explorar de forma íntegra el entorno, como ocurre cuando el robot no para de dar vueltas a un obstáculo en el comportamiento Seguir Pared. O incluso puede ocurrir que, aun cuando el robot va explorando el entorno, su trayectoria no es eficiente, como sucede al seguir un pasillo con forma de L. En cualquier caso, como el entorno está parcialmente desconocido, no tiene sentido limitar la navegación a un comportamiento determinado porque, dependiendo de la configuración del entorno, es posible que no sea el adecuado.

Por último, se ha propuesto un nuevo método de aprendizaje asistido que combina los comandos de movimiento de alguno de los tres comportamientos implementados mediante el modelo analítico con los comandos que un humano introduce a través de un joystick. La clave de la combinación es la medida de la eficiencia de los comandos como una suma de tres factores, suavidad, longitud de la trayectoria y seguridad. La técnica ha sido probada por parte de 13 usuarios voluntarios, los cuales dirigieron la plataforma robótica real *Pioneer P2AT* en cuatro

situaciones diferentes, relacionadas con los tres comportamientos desarrollados. Para analizar las eficiencias de las trayectorias se ha desarrollado una representación visual que permite evaluarlas en cualquier punto de la trayectoria en función de los tres factores involucrados. En casi todos los casos el control asistido ha superado al humano. En muchos casos, incluso, mejora el funcionamiento del robot, especialmente allí donde el agente tiene problemas.

Capítulo 4

Fusión de Comportamientos

Este capítulo describe la técnica diseñada para alcanzar una cooperación eficiente de los comportamientos reactivos presentados en el Capítulo 3, a partir de la caracterización del lugar en el que el robot se encuentra. El algoritmo que se presenta se implementa en el módulo *Fusión* de la capa *FusiónComportamientos*. A partir del mapa local que se genera y actualiza en el módulo *MapaLocal* de la capa *ModeladoEntorno* se calculan los pesos con los que se van a ponderar los comandos de movimiento de cada uno de los tres comportamientos reactivos. El reconocimiento de este mapa constituye lo que se denomina caracterización de lugares.

En primer lugar se efectúa una revisión de la integración de varios comportamientos en un sistema por parte de un agente autónomo móvil en la Sección 1. En segundo lugar se analiza con detalle en la Sección 2 el algoritmo de caracterización de lugares que se ha diseñado y desarrollado, haciendo énfasis en las decisiones de diseño tomadas. Gracias a este algoritmo se consigue reducir el mapa local en un determinado punto del entorno a un pequeño vector de características. Con este vector se muestra la forma en la que se calculan los pesos a aplicar a los comandos de movimiento de los comportamientos. En la Sección 3 se presenta la combinación de los comandos de movimiento que se realiza en el sistema. Las prestaciones temporales del método son analizadas en la Sección 4. En la Sección 5 se presentan los resultados obtenidos con la técnica. Finalmente, en la Sección 6 se presentan las conclusiones que se derivan de este capítulo.

1 Introducción

El objetivo principal de esta Tesis es el desarrollo de un sistema para la exploración completa de entornos total o parcialmente desconocidos. La exploración completa se entiende como una tarea que consiste en hacer que un robot explore de forma íntegra y eficiente el entorno en el que se encuentra [CP97, LV05, GB06]. El problema más importante de la exploración completa radica en la necesidad de planificar sobre entornos desconocidos y, por tanto, de representaciones topológicas dado que en zonas desconocidas la planificación descendente (*top-down*) no es aplicable, por lo que ésta debe ser ascendente (*bottom-up*). En el Capítulo 5 se presenta la

planificación de la ruta de exploración para el robot. Esta ruta se debe seguir mediante los tres comportamientos desarrollados, analizados en el Capítulo 3. El problema es decidir en todo momento el comportamiento de navegación reactivo más adecuado para seguir la ruta de exploración completa planificada.

Existen dos estrategias principales para integrar varios comportamientos de navegación en un sistema, la coordinación de comportamientos y la competición de comportamientos. En la presente Tesis nos planteamos cuál de las dos aproximaciones es la apropiada para nuestro sistema. Al ser total o parcialmente desconocidos los entornos de trabajo del agente, quizás no sea adecuada la estrategia de competición de comportamientos. Como no se conoce de antemano la configuración del entorno, es posible que ninguno de los tres comportamientos de navegación encaje de forma aislada con dicho entorno, sobre todo si éste no es estructurado. Sin embargo, una combinación de los tres comportamientos puede dar lugar a unos comandos de movimiento más plausibles con los diferentes escenarios. Así pues, ésta es la estrategia escogida para nuestro sistema.

La estrategia de fusión de comportamientos necesita de algún mecanismo de decisión de la manera en la que deben ser combinados para navegar de la forma más eficiente en todo momento. En esta Tesis nos basamos en un algoritmo de caracterización de lugares a partir de la información sensorial proporcionada por sensores sonar. Nuestro método proporciona, al finalizar el proceso, los factores con los que hay que ponderar los comandos de movimiento de los tres comportamientos para adaptarse a la configuración instantánea del entorno.

El algoritmo de caracterización de lugares diseñado e implementado se basa en los mapas probabilísticos para alcanzar lecturas sonar más estables. Debido al elevado volumen de datos que estos mapas implican, su tamaño se reduce a un pequeño vector de características por medio del Análisis de Componentes Principales (*Principal Component Analysis*, *PCA*) [Hot33], para su almacenamiento y comparación. El *PCA* reduce la redundancia y devuelve la información relevante minimizando el error de representación. Antes de aplicar el *PCA*, se extrae el mapa de profundidad del contorno del área libre de obstáculos en los alrededores del robot. Este mapa de profundidad se representa mediante el módulo de su Transformada Rápida de Fourier (*Fast Fourier Transform*, *FFT*). El vector de características obtenido gracias al *PCA* permite caracterizar un determinado lugar del entorno en tres categorías: pared, pasillo y puerta. Cada una de estas tres categorías se corresponde con uno de los tres comportamientos implementados en nuestro sistema: Seguir Pared, Seguir Pasillo y Cruzar Puerta, respectivamente. Por lo tanto, se podría seleccionar el comportamiento más adecuado en cada momento a partir de la categorización efectuada, siempre y cuando se pueda descomponer en una combinación de éstos, lo que se prueba a lo largo del capítulo. Según se ha comentado anteriormente, es posible que un comportamiento aislado no sea lo más adecuado para un determinado lugar de un entorno arbitrario. Este hecho no es un problema para nuestro sistema puesto que a partir del vector de características de un lugar se ha implementado un mecanismo que ayuda a la fusión de los tres comportamientos. De este modo, los comportamientos no tienen por qué competir en la navegación reactiva, sino que van a cooperar en todo momento. El método desarrollado

obtiene los factores que ponderan los comandos de movimiento de los tres comportamientos para alcanzar de forma efectiva un comando de movimiento que combina linealmente los tres comandos parciales.

Antes de analizar con detalle el algoritmo de caracterización de lugares diseñado se presenta a continuación una breve revisión de las dos estrategias principales para integrar varios comportamientos de navegación en un sistema.

1.1 Integración de comportamientos

La navegación de un robot se puede alcanzar mediante un control basado en comportamientos, una de las ramas de la Robótica. De hecho, se ha demostrado que los comportamientos pueden servir como bloques funcionales para ejecutar las acciones del robot [Ark98]. Según Mataric [Mat97], un comportamiento es *una ley de control que satisface un conjunto de restricciones para alcanzar y mantener un objetivo determinado.* Las dos aproximaciones más empleadas a la hora de integrar varios comportamientos de navegación son la coordinación de comportamientos y la competición de comportamientos, siendo este último el enfoque empleado por las primeras técnicas [Bro86, Pay86, Fir87].

El mecanismo de coordinación de comportamientos asegura la actividad simultánea de varios comportamientos independientes, obteniéndose un comportamiento emergente que lleva al sistema a la ejecución de la tarea de navegación encomendada [RJ05]. Su principal inconveniente es la dificultad para encontrar la regla para fusionar los comportamientos de una manera eficiente, porque el robot tiende a tener problemas de decisión cuando los comportamientos son excluyentes pero tienen la misma importancia [SDC05]. La aproximación empleada en la presente Tesis se basa en la coordinación o fusión de los comportamientos.

Al usar comportamientos competitivos el sistema tiene que escoger uno de ellos de entre todo el conjunto para llevar a cabo la tarea de navegación. El problema estriba en la correcta elección del comportamiento más apropiado para cada situación dentro del entorno de trabajo. A continuación se presentan sistemas que coordinan los comportamientos así como sistemas que emplean un método competitivo.

1.1.1 Coordinación de comportamientos

Esta estrategia también es conocida como fusión de comandos (*command fusion*). En la literatura nos encontramos con diversas técnicas para conseguir que los comportamientos se coordinen en un sistema. Un primer método es la fusión de señales de control [FBSC04]. El esquema propuesto usa dos filtros de información descentralizada (*DIF*) para alcanzar la fusión. Estos filtros estiman, a partir de la respuesta aislada de cada uno de los comportamientos, las velocidades de rotación y traslación del robot. Además de estos dos filtros, el sistema está supervisado por un módulo encargado de evitar la fusión de comportamientos que no son adecuados en función de la situación particular en cada momento. Por otro lado, hay estudios que tratan de emular el

comportamiento de estímulo-respuesta de un sistema biológico [RKH04], mediante el desarrollo de una máquina con un alto nivel de autonomía y movilidad.

La lógica borrosa (*fuzzy logic*) ha sido ampliamente utilizada como mecanismo de fusión de comportamientos [RJ05, PGS04, SDC05, WL04]. Rahman propone una aproximación para entornos no estructurados basada en un sistema bicapa [RJ05], a partir de la lectura de sensores sonar. La primera capa selecciona un grupo de comportamientos, inhibiendo otros grupos, para así reducir las reglas necesarias del sistema borroso. La segunda incluye el controlador borroso para cada comportamiento. Parasuraman también emplea la lógica borrosa en su método [PGS04], el cual ha demostrado su efectividad para la navegación autónoma en entornos simulados. Selekwa utiliza una modificación de la lógica borrosa, lo que se conoce como lógica borrosa multivaluada [SDC05], consiguiendo un movimiento realista y eficiente con un robot *P2AT*. Wang asigna a cada comportamiento un factor o peso [WL04]. Estos factores se ajustan dinámicamente en función de las reglas del sistema borroso que soporta el método, determinando el peso que cada uno de los comportamientos tiene en el comando de movimiento final. Este ajuste de los pesos es realizado continuamente mientras el robot está en movimiento.

Otros autores prefieren combinar la lógica borrosa con las redes neuronales [KK05, SY04]. Khatoon parte del hecho de que una implementación de comportamientos a partir de la lógica borrosa requiere muchas reglas si la salida debe proporcionar muchas recomendaciones, como es el caso de la fusión de varios comportamientos [KK05]. Por ello, este autor propone emplear una red neuronal para calcular las velocidades de rotación y traslación para el robot, aunque únicamente se demuestra su validez en entornos simulados. En este mismo sentido se encamina la propuesta de Shou [SY04]. La situación difiere respecto de la aproximación de Khatoon en el sentido de que primero se aplica una red neuronal para cada comportamiento, cuyos resultados son llevados a un sistema borroso para decidir las velocidades del robot.

1.1.2 Competición de comportamientos

Las técnicas competitivas son referenciadas algunas veces como de arbritación (*behavior arbitration*). Al contrario que en la fusión de comandos, en este caso el sistema únicamente activa un comportamiento en cada momento. Por lo tanto, la clave está en encontrar la manera de conmutar entre todos los comportamientos existentes. Este punto de vista de los sistemas basados en comportamientos no tiene un tratamiento tan extenso por parte de la comunidad científica.

Un primer método para arbitrar varios comportamientos es el presentado por Zalama [ZGPP02]. Se basa en una red neuronal que controla un robot autónomo móvil en un entorno con obstáculos. El robot es capaz de aprender varios comportamientos como evitar obstáculos, seguir pared y alcanzar un punto partiendo exclusivamente de la información de sensores sonar. Esta información es empleada para elegir el mejor comportamiento para cada situación particular, obteniéndose las velocidades de traslación y rotación que deben ser enviadas al robot.

Al igual que en el caso de la fusión de comandos, también se ha empleado la combinación de las redes neuronales con la lógica borrosa para arbitrar varios comportamientos [RPW$^+$03].

El módulo que conmuta entre los comportamientos se basa en la dirección del robot con respecto al punto de destino. Así mismo, los comportamientos también pueden ser desactivados si sus comandos de movimiento condujeran a un choque o situación de peligro para el agente.

En [DH04] se presenta una técnica basada en el aprendizaje por refuerzo. La selección inicial de comportamientos se realiza de forma aleatoria, puesto que el sistema no tiene conocimiento. A medida que se avanza en el tiempo, el conocimiento adquirido es empleado para escoger el comportamiento más adecuado para cada situación.

2 Algoritmo de caracterización de lugares

Tradicionalmente, los algoritmos de caracterización de lugares por parte de un robot emplean técnicas que se basan en tres tipos de información sensorial:

- Datos visuales capturados con cámaras.
- Lecturas dadas por un láser.
- Lecturas proporcionadas por sensores sonar.

Estos dos últimos se enmarcan en los sensores de distancia y, si bien habitualmente se usaba el sonar por razones de precio y disponibilidad, cada vez se usa más el láser.

Existen muchas aproximaciones para caracterizar lugares visualmente como elemento de ayuda a la navegación de un agente. El método de Martínez [MV01] es una aproximación visual basada en la apariencia. Su sistema se diseña para reconocer la posición actual del robot y, a partir de ella, inferir la dirección que debe tomar. Para operar con el sistema se necesita una fase de entrenamiento previa a la fase de test. La fase de entrenamiento necesita de la supervisión de un humano que identifique las diferentes zonas del entorno que el agente ha aprendido. Así mismo, la captura de imágenes durante esta fase debe ser realizada con cuidado y en diferentes condiciones de iluminación. Durante la fase de test el robot es capaz de reconocer los lugares que previamente ha aprendido, pero en el mismo entorno. Tal y como se establece en el trabajo, el conjunto de datos necesarios es muy grande y tiene que ser significativo. Más grave aún es que el entorno de aprendizaje y el de test han de ser el mismo. Por lo tanto, con este método no se puede entrenar el sistema en un entorno y hacer que el robot trabaje en otro. De ahí que sea una técnica inválida para operar en entornos total o parcialmente desconocidos.

Del mismo modo que en [MV01], en [HLD03] se propone una aproximación para la caracterización de lugares que necesita que el entorno de entrenamiento y el entorno de aprendizaje sean el mismo. A pesar de que el reconocimiento se puede alcanzar desde diferentes perspectivas y en condiciones de visión distintas, no es un procedimiento válido para entornos desconocidos.

En [LAP+04] se presenta una técnica de reconocimiento de lugares cuyo objetivo es la localización de un agente autónomo móvil. Reduce cualquier imagen a un vector de características

de dimensión 40. Estas características están asociadas con las diferentes posiciones en el entorno. Estos vectores son usados para entrenar un clasificador *SVM* (*Support Vector Machine*) [Vap98], a partir del cual se consigue determinar el lugar en el que se encuentra el robot. El reconocimiento de lugares para que el robot se localice se convierte, por tanto, en una clasificación, donde el tamaño del problema se ha reducido. En este sentido, esta técnica es análoga a la que se presenta en la presente Tesis, puesto que a partir de la reducción de un conjunto de datos inicial se consigue un vector de reducidas dimensiones que hay que categorizar, al margen de que el origen de nuestros datos es el sonar y no la información visual. Otra similitud con nuestra propuesta es que la clasificación da como resultado tres clases: puerta, pasillo y espacio libre para el método de [LAP+04], y pared, pasillo y puerta para la técnica propuesta. Sin embargo, la principal diferencia estriba en el hecho de que este método no puede emplearse en un entorno diferente del usado para capturar las imágenes para entrenar el sistema. Así pues, al contrario que nuestra técnica, esta aproximación no es útil en entornos total o parcialmente desconocidos.

Otra aproximación para localizar globalmente en interiores al agente es la de An [ALZT04]. En este caso el robot es capaz de localizarse a partir del reconocimiento de unas placas situadas sobre la puerta con el número de habitación. Mediante un procesamiento de la imagen capturada se puede reconocer si dicha imagen contiene una placa de una puerta o no, determinando de qué puerta se trata. El principal problema, al igual que en las aproximaciones anteriores, es la imposibilidad de operar en entornos desconocidos.

Do [DJ06] también propone emplear el procesamiento visual de imágenes, pero en este caso su objetivo es guiar de forma autónoma el agente. El mecanismo que emplea es la comparación de las imágenes capturas con unos patrones almacenados en una base de datos. Aunque el número de patrones no tiene por qué ser muy elevado y que el sistema puede trabajar en tiempo real, el reconocimiento de lugares solamente puede efectuarse en entornos conocidos de antemano.

Una última propuesta visual es la de Pronobis [PCJC06]. En este trabajo se presenta un algoritmo para reconocer lugares en función de una clasificación basada en la apariencia visual. Es válido para diferentes condiciones de luz, aunque es totalmente supervisado. El principal inconveniente es la necesidad de entrenar el sistema en un entorno y proceder al reconocimiento de lugares en el mismo entorno. Durante la fase de entrenamiento se representa cada una de las habitaciones del entorno mediante un conjunto de imágenes capturadas en diferentes condiciones de iluminación y con distintas orientaciones. Durante la fase de operación el sistema reconoce los lugares a partir de imágenes que deben tener aproximadamente la misma orientación que las del entrenamiento, pero que pueden ser capturadas bajo diferentes condiciones de iluminación.

Todas las técnicas basadas en información visual expuestas son robustas y fiables. Sin embargo, hemos visto que comparten un inconveniente importante, que los entornos de entrenamiento y de test deben ser el mismo, principalmente porque están orientadas hacia la localización del agente. Por lo tanto, estas aproximaciones no son válidas para entornos dinámicos o desconocidos, como es el caso de los escenarios hacia los que va destinado el sistema desarrollado en la presente Tesis. Además, la carga computacional que involucran es muy elevada. En este

sentido, existen métodos basados en las lecturas dadas por un láser o las lecturas proporcionadas por sensores sonar, que permiten al robot operar en entornos distintos del de aprendizaje.

A partir de las lecturas dadas por un láser, Larbi [LA05] propone un clasificador bayesiano para el reconocimiento de lugares en un entorno donde opera un agente autónomo móvil. A pesar de que los resultados parecen prometedores, son simulados. Además, su diseño está muy parametrizado, teniendo que introducir los patrones de comparación manualmente. Cada uno de estos patrones simula una posible situación o clase a la que el robot debe hacer frente. El sistema es capaz de clasificar las situaciones durante la navegación dentro de alguna clase con una tasa de acierto elevada, aunque es quizás un método demasiado supervisado.

Otra técnica que emplea láser es la que se presenta en [MSB05]. En cierto sentido, es una aproximación parecida a la que se propone en esta Tesis, si dejamos a un lado la información sensorial de partida. En [MSB05] se propone una forma para clasificar los lugares de un entorno en cuatro categorías: habitación, vestíbulo, puerta y pasillo. La idea clave es clasificar la posición del robot en función de la lectura actual obtenida del láser, lo que nos permite diferenciarla de nuestra propuesta, puesto que el método que se presenta en este capítulo parte de la integración temporal y espacial de las lecturas obtenidas de sensores sonar. Así mismo, el número de características necesarias en [MSB05] para caracterizar un lugar es muy elevado, siendo precisa una fuerte parametrización, al contrario que para nuestra propuesta. Sin embargo, al igual que nuestra propuesta, el método es útil en entornos diferentes al usado para el entrenamiento del sistema. Una ventaja de la técnica que se propone en la presente Tesis frente al método de [MSB05] es la posibilidad de usar la caracterización para fusionar de forma eficiente comportamientos en un sistema de navegación basado en comportamientos.

Los sensores sonar producen una información relativa a la distancia a los obstáculos alrededor del robot. Gracias a estas lecturas se pueden percibir estructuras como pasillos, esquinas e intersecciones. Su principal inconveniente es que se ven afectados por errores como la inexactitud angular y los ecos múltiples. Para evitar estos problemas, muchas técnicas de corrección de las lecturas sonar se basan en la fusión espacial y temporal de estas lecturas. Algunos métodos se basan en la búsqueda de estructuras simples como paredes, esquinas y aristas en mapas de profundidad [Mat92, LM97], pero necesitan realizar algunas suposiciones sobre la configuración del entorno, como que la geometría 3D es ortogonal al plano horizontal. Otros métodos no incluyen de forma explícita primitivas de búsqueda geométrica [Zim95]. Desafortunadamente, los errores de los sensores sonar hacen que estos sistemas no sean fácilmente predecibles. Por ello, habitualmente se integran en el tiempo y el espacio las lecturas sonar mediante mapas probabilísticos [Mor88]. Este paradigma de representación del entorno ha sido analizado en la Sección 3.3 del Capítulo 2. Estos mapas combinan de forma rápida y eficiente la lectura de cualquier número de sensores sonar, siendo fáciles de construir y actualizar. Sin embargo, implican un elevado volumen de datos. Por lo tanto, la comparación de dos mapas probabilísticos con objeto de decidir su caracterización puede consumir mucho tiempo de proceso. Más aún, esta comparación no es evidente, puesto que los mapas probabilísticos se ven afectados por la orientación y la posición relativa del agente. De ahí que mapas capturados en lugares similares

pueden ser bastante diferentes. Para evitar este problema, algunos métodos prefieren trabajar con el dato de distancia [VMK02], el cual es corregido usando métodos estadísticos como el *PCA*. Otros métodos se basan en técnicas que evalúan la similitud entre los mapas [YB96], con el coste computacional que ello implica.

Por todo lo dicho, se ha escogido la caracterización de lugares mediante mapas probabilísticos construidos a partir de lecturas sonar. Se podría haber usado la información de un láser en vez de la de los sensores sonar. Sin embargo, esta opción es descartada porque la plataforma real *Pioneer P2AT* usada para realizar las pruebas consta únicamente de sensores sonar para adquirir información del entorno. Al mismo tiempo, también debemos comentar que el láser es más caro que el sonar y en interiores no detecta las puertas de cristal. Por las razones arriba descritas se ha descartado emplear una cámara para capturar imágenes y efectuar a continuación un procesamiento de estas imágenes. Además, las aproximaciones visuales suelen requerir que los entornos de entrenamiento y test sean el mismo. Esta suposición implica que, aunque se integre una cámara en la plataforma robótica, no sea una opción adecuada para entornos total o parcialmente desconocidos.

El punto de partida del algoritmo es el mapa local construido a partir de las lecturas sonar que se genera y actualiza en el módulo *MapaLocal* de la capa *ModeladoEntorno*. Tal y como se comentó en la Sección 3.3.2 del Capítulo 2, el mapa local es un mapa probabilístico [Elf87] del entorno más próximo al robot. Partiendo de este mapa local se obtienen los factores de ponderación para cada uno de los tres comportamientos del sistema. Como la cooperación entre los comportamientos se debe hacer para adaptarse al entorno más cercano al agente, con este modelo local es más que suficiente, sin necesidad de tener que procesar el mapa métrico construido por el módulo *MapaMétrico*. Además, como las dimensiones son bastante menores que las del mapa métrico, se consigue una reducción importante en el tiempo de proceso y, por consiguiente, se aumenta la velocidad de reacción y de adaptación a la configuración del entorno. En esta Tesis se trabaja habitualmente con mapas de 113x113 celdas. Considerando que tanto la plataforma robótica real como la simulada tienen una dimensiones de unos $50x50cm^2$, 113 celdas se corresponden con un rango de $2m$ en torno al robot, para un tamaño de celda $Tam\_Celda = 40mm$.

El algoritmo de caracterización de lugares empleado es una evolución del que se presenta en [PUdTS07], donde los vectores de características eran útiles para que los comportamientos compitieran en la navegación. Sin embargo, es más interesante y eficiente el conseguir la cooperación de los comportamientos en todo momento, ya que se obtienen comportamientos nuevos ajustados al entorno en cuestión. La técnica que se presenta en este trabajo mejora nuestros resultados previos en el sentido de que el vector de características propuesto no solamente es válido para que los comportamientos compitan seleccionando el más apropiado, sino que también es válido para alcanzar una cooperación eficiente entre todos ellos.

El primer paso que hay que llevar a cabo para poder caracterizar un lugar es extraer el vector de características de ese lugar. El proceso de extracción del vector de características

consta de cuatro fases, según se muestra en el esquema general de la Figura 4.1:

1. *Cálculo del contorno del mapa local.* Durante esta fase el mapa local es procesado con el objetivo de obtener la Región Libre de Obstáculos de Interés ($RLOI$) en el entorno más próximo al robot. La $RLOI$ se representa por medio de su contorno.

2. *Cálculo del mapa de profundidad.* A partir del contorno de la $RLOI$ del mapa local se extrae el mapa de profundidad asociado en las cercanías del robot.

3. *Obtención de la FFT del mapa de profundidad.* La tercera fase del algoritmo consiste en obtener la FFT del mapa de profundidad conseguido en la etapa anterior. Dicho mapa de profundidad será representado mediante el $|FFT|$, porque éste es invariante ante las rotaciones.

4. *Extracción del vector de características.* Durante la última fase el $|FFF|$ del mapa de profundidad del contorno de la $RLOI$ es proyectado sobre la base del sistema para así determinar el vector de características del lugar actual del robot. La base se calcula mediante el PCA, proceso que hay que realizar *off line* una única vez.

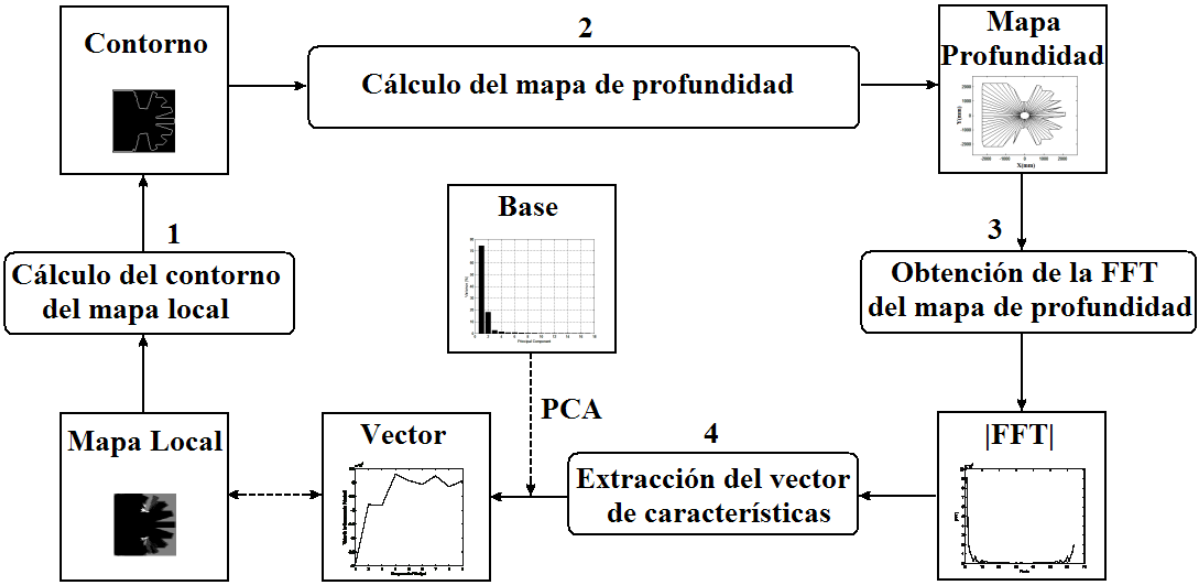

Figura 4.1: Esquema general del algoritmo de extracción del vector de características.

Una vez extraído el vector de características, el sistema podría clasificar el lugar del entorno en una de las tres categorías siguientes: pared, pasillo y puerta. Tal y como se observa, estas tres categorías se corresponden con los tres comportamientos desarrollados para el sistema de exploración, Seguir Pared, Seguir Pasillo y Cruzar Puerta, respectivamente. Por lo tanto, el sistema podría seleccionar el comportamiento más adecuado para navegar en cada momento. Sin embargo, ya sabemos que es posible que un comportamiento aislado no sea la opción recomendable para una determinada configuración de un entorno arbitrario. Con el vector de

características propuesto se soluciona este problema puesto que la técnica presentada en esta Tesis no solo permite clasificar los lugares en pared, pasillo y puerta, sino que va a dotar al sistema con una capacidad adicional, la cooperación entre los comportamientos. A partir de un determinado vector de características obtenido en un lugar el sistema determina los factores de ponderación a aplicar a cada uno de los tres comportamientos reactivos para fusionar sus comandos de movimiento.

A continuación se analizan las cuatro fases del algoritmo que conducen a la extracción del vector de características para pasar, posteriormente, a presentar el procedimiento empleado para la obtención de los factores de ponderación de los tres comportamientos a partir del vector de características.

2.1 Cálculo del contorno del mapa local

El mapa local, punto de partida de todo el proceso, proporcionado por el módulo *MapaLocal* de la capa *ModeladoEntorno*, es una rejilla que constituye una división bidimensional en celdas del espacio alrededor del robot. Cada celda, como sabemos, posee una probabilidad de ocupación del 0% al 100%. Idealmente, un obstáculo presenta una probabilidad de ocupación del 100%, mientras que el espacio libre presenta una probabilidad de ocupación del 0% y las zonas no exploradas del 50%. Así mismo, y como se ha descrito en capítulos anteriores, la forma habitual de mostrar estos mapas probabilísticos es mediante la asociación entre la probabilidad de ocupación de cada celda y un nivel de la escala de grises de 0 a 255. De este modo, las zonas libres se muestran en negro, los obstáculos en blanco y las áreas no exploradas en gris.

En esta sección se analiza el diseño realizado para la primera de las fases del algoritmo de extracción del vector de características, el *Cálculo del contorno del mapa local* (Figura 4.1), cuyo objetivo es representar el mapa local de partida a través del contorno de la región libre de obstáculos en las cercanías del robot. Esta fase se divide, a su vez, en cuatro etapas:

1. *Umbralización.* Durante esta etapa el sistema obtiene la celda obstáculo más cercana al robot en cualquier dirección. Esto permite determinar como regiones a analizar por las siguientes fases como aquéllas formadas por todas las celdas libres a menor distancia que la celda obstáculo más cercana en cualquier dirección.

2. *Filtrado.* Como el mapa local umbralizado es ruidoso, se usa un filtro de la mediana para eliminar este ruido de la representación en la medida de lo posible.

3. *Obtención de la RLOI.* La etapa anterior da lugar a un mapa con un conjunto de regiones de espacio libre no conectadas. Esta fase tiene como objetivo, por tanto, el discriminar la única región de interés dentro del mapa. Esta región se denomina Región Libre de Obstáculos de Interés, *RLOI*.

4. *Obtención del contorno.* Finalmente, la *RLOI* se representa por medio de su contorno $\overrightarrow{C}$.

El proceso de extracción del contorno de la *RLOI* de un mapa local se ilustra en la Figura 4.2, donde se presenta el resultado de la aplicación de los diferentes pasos del algoritmo al mapa local de la Figura 4.2a. A continuación se analizan por separado estas cuatro fases de la obtención del contorno del mapa local.

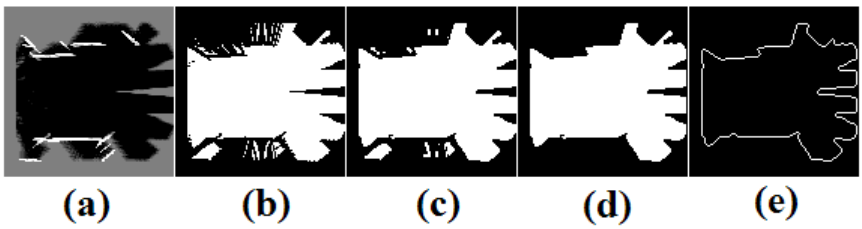

Figura 4.2: Cálculo del contorno del mapa local: (a) mapa local; (b) umbralización, (c) filtrado; (d) *RLOI*; (e) contorno.

2.1.1 Umbralización

El objetivo de esta primera etapa es la determinación de la región de espacio libre más próxima al robot. Para ello habrá que analizar la configuración del mapa local y así obtener el obstáculo más cercano al agente en todas las direcciones. En cualquier dirección, toda celda más alejada que el obstáculo más próximo al agente en esa dirección no pertenece a la región que nos interesa. Por el contrario, todas las celdas en cualquier dirección más cercanas al agente que el obstáculo más próximo en esa dirección sí forman parte de la región a analizar. De este modo, si hay celdas de espacio libre más alejadas que el obstáculo más cercano en una determinada dirección, no pertenecerán a la región a analizar. Este dato tiene todo el sentido, puesto que para una determinado mapa local el campo de visión del robot estará limitado por la configuración de los obstáculos en su proximidad. Aquellas zonas más alejadas que los obstáculos en torno al robot no tienen razón de ser, puesto que en realidad el robot no las ve, más aún cuando quizás están presentes en el mapa local por el propio procedimiento de construcción de este mapa probabilístico.

La umbralización se lleva a cabo en varios pasos consecutivos, considerando que el mapa local es una imagen donde cada celda presenta un nivel de la escala de grises de 0 a 255.

El método para construir el mapa local presentado en la Sección 3.3 del Capítulo 2 implica que la probabilidad de ocupación de una determinada celda no tiene por qué ser exactamente del 0%, 50% o 100%, lo que se corresponde con un nivel de gris 0, 127 o 255, respectivamente. Sin embargo, en el mapa local nos interesa conocer si una determinada celda está ocupada o no. Por ello, el primer paso de esta fase de *Umbralización* consiste en asignarle a cada celda uno de los valores de la escala de gris G_{region}, G_{fondo} o $G_{no\_explore}$, en función de los umbrales de probabilidad $U_{ocupada}$ y $U_{no\_explore}$. A las celdas cuya probabilidad de ocupación sea superior a $U_{ocupada}$ se les asigna el valor G_{fondo}. A las celdas cuya probabilidad de ocupación esté entre $U_{no\_explore}$ y $U_{ocupada}$ se les asigna el valor $G_{no\_explore}$. Para el resto su valor será G_{region}. En la presente Tesis se han empleado unos umbrales $U_{ocupada} = 60\%$ y $U_{no\_explore} = 40\%$, siendo los

valores de la escala de grises $G_{fondo} = 0$, $G_{region} = 255$ y $G_{no\_explore} = 127$. El valor usado para los umbrales $U_{ocupada}$ y $U_{no\_explore}$ son los mismos que se utilizan para la construcción del mapa topológico, según se presenta en la Sección 2 del Capítulo 5.

El agente se encuentra situado siempre en el centro del mapa local, dado por (c_x, c_y). Como normalmente el mapa local presenta 113x113 celdas, su centro en pixeles será $(c_x, c_y) = (57, 57)$. El segundo paso consiste en el almacenamiento del módulo y la fase de la posición en pixeles de cada celda dentro del mapa local, respecto de su centro. Para toda celda (x, y), su módulo $\rho(x, y)$ y su fase $\theta(x, y)$ vendrán dados por las expresiones:

$$\rho(x, y) = \sqrt{(x - c_x)^2 + (y - c_y)^2} \tag{4.1}$$

$$\theta(x, y) = \arctan \frac{y - c_y}{x - c_x} \tag{4.2}$$

siendo $G(x, y)$ el valor asignado durante el paso anterior, G_{fondo}, G_{region} o $G_{no\_explore}$.

Durante esta fase de *Umbralización* hay que obtener el obstáculo más cercano en todas las direcciones. Por lo tanto, es necesario saber cuáles son estas direcciones entre 0^o y 360^o. A partir de la dirección 0^o se han considerado todas aquéllas cuya diferencia con la dirección anterior es de 1^o. Así, nuestro vector de direcciones será:

$$\overrightarrow{\varphi} = [\varphi_1, \varphi_2, \varphi_3, \cdots, \varphi_{360}] = [0^o, 1^o, 2^o, \cdots, 359^o] \tag{4.3}$$

Por lo tanto, la fase de un determinado pixel necesita ser discretizada para que se corresponda con alguna de las direcciones posibles del vector $\overrightarrow{\varphi}$. El almacenamiento en polares del mapa local de partida sirve de ayuda a la hora de buscar, para cualquier dirección φ_i, la distancia al obstáculo más cercano O_{φ_i}. La distancia a este obstáculo más cercano será, para una dirección φi, el valor de distancia mínima a una celda (x, y) de valor $G(x, y) = G_{fondo}$ y con fase $\theta(x, y) = \varphi_i$:

$$0_{\varphi_i} = \min_{\forall \theta(x,y)=\varphi i,\ G(x,y)=G_{fondo}} \{\rho(x, y)\} \tag{4.4}$$

Al finalizar este proceso se consigue el vector de distancias a los obstáculos cercanos en cualquier dirección:

$$\overrightarrow{O_\varphi} = [O_{\varphi_1}, O_{\varphi_2}, \cdots, O_{\varphi_{360}}] \tag{4.5}$$

Finalmente, para toda celda (x, y) se determina si pertenece o no a la zona de interés que deben analizar las siguientes etapas del *Cálculo del contorno del mapa local*:

$$G(x, y) = \begin{cases} G_{region} & \text{si } G(x, y) = G_{region} \text{ y } \rho(x, y) < O_{\theta(x,y)} \\ G_{fondo} & \text{en otro caso} \end{cases} \tag{4.6}$$

con lo que se obtiene el resultado final de la fase de *Umbralización*, consistente en una imagen del mismo tamaño que el mapa local de partida donde la zona de interés tiene un valor G_{region} y el resto de celdas presentan el valor G_{fondo}. Esta zona de interés consiste en un conjunto de regiones no conexas que potencialmente pueden constituir la *RLOI*, muchas de ellas de dimensiones pequeñas o incluso que constan de un único pixel, y todo ello debido al proceso de construcción del mapa local y al propio proceso de umbralización. Como la *RLOI* debe ser única, la imagen resultante de esta fase es procesada por las siguientes etapas del algoritmo.

2.1.2 Filtrado

La imagen obtenida al final de la fase de *Umbralización* todavía no es útil para nuestros propósitos, pues presenta dos inconvenientes principales: i) es muy ruidosa; y ii) no contiene una única región a analizar, sino un grupo de regiones. Para solucionar estos problemas se procesa la imagen mediante un filtro de la mediana [Mar93].

El filtro de la mediana en un filtro espacial paso bajo no lineal. Es extensamente usado en la eliminación de ruido en imágenes debido a sus interesantes propiedades. Es aplicado a cada uno de los pixeles de la imagen a través de una máscara de tamaño $N_{med}xM_{med}$, eliminando los detalles de tamaño menor que dicha máscara. Al aplicar la máscara, cada pixel de la imagen es reemplazado por la mediana de los valores de los pixeles dentro de la máscara. Por lo tanto, a la hora de procesar un determinado pixel es necesario ordenar los valores de los pixeles que caen dentro de la máscara para así saber cuál es la mediana. Esta ordenación se realiza por medio del algoritmo de la burbuja.

Aunque este filtro es muy sencillo, implica una elevada carga computacional que aumenta exponencialmente con el tamaño de la máscara. Como en nuestro sistema deseamos adaptarnos al entorno en torno momento, el vector de características debería poder calcularse en el menor tiempo posible. Por este motivo, en esta Tesis se utiliza una máscara de tamaño $3x3$, no conll-evando un elevado incremento en el tiempo requerido para obtener el vector de características.

Al finalizar esta etapa, la imagen procedente de la fase de *Umbralización* ha visto reducido tanto el ruido de partida como el número de regiones de la misma. Debido a la naturaleza del filtro, aquellas regiones de dimensión muy pequeña, como las formadas por un único pixel, desaparecen. Al mismo tiempo, para el resto de las regiones se suavizan los bordes, aunque se ven preservados. Sin embargo, la imagen procesada sigue presentando un conjunto de R regiones no conectadas, pudiendo cada una de ellas ser la *RLOI*.

2.1.3 Obtención de la *RLOI*

De las R regiones obtenidas al acabar la etapa de *Filtrado* hay que seleccionar la *RLOI*. La *RLOI* es la región de interés en el mapa local, aquélla en la que se encuentra el robot y que delimita el espacio libre en el entorno más próximo al agente.

El primer paso de esta fase consiste en la clasificación de todos los pixeles de la imagen. A

aquellos pixeles que presenten un valor G_{region} se les asigna una clase. A aquéllos cuyo valor sea G_{fondo} no se les asigna clase. Al finalizar este paso todos los pixeles que formen parte de una región tienen asignado un número de clase. El problema que surge es que, debido al proceso de asignación de clases, una misma región puede haber sido dividida en varias y, por lo tanto, pertenecer a varias clases. Así, el siguiente paso es la fusión de las clases que pertenecen a la misma región.

La fusión de clases se realiza en base al estudio de la conectividad entre las mismas. Para ello es necesario el cálculo de la *bounding box* de cada una de las clases. Mediante el análisis de la conectividad entre las clases se sabe cuál o cuáles de las clases deben ser fusionadas. Este estudio es similar al realizado durante la fase *Estudio de Conectividad* del algoritmo de construcción de la estructura jerárquica que implementa la capa *PlanificadorRuta*. Es un proceso no supervisado que analiza la relación de conexión entre todas las parejas posibles de clases en función de la relación espacial definida por sus *bounding boxes* y los pixeles pertenecientes a las mismas. Para que dos clases sean fusionadas sus *bounding boxes* deben intersecar y debe existir, al menos, una celda de cada una de las clases en contacto.

Una vez se conocen todas las regiones del mapa local, hay que seleccionar la *RLOI*. La *RLOI* es la región en al que se encuentra el robot. Como el agente siempre se encuentra en el centro de los mapas locales, es inmediato saber, de entre todas las regiones, cuál es la *RLOI*. En teoría será la región de mayor área, puesto que para llegar hasta su posición actual el robot ha ido explorando el entorno y, por ende, actualizando el espacio libre a su alrededor.

2.1.4 Obtención del contorno

El último paso consiste en la obtención de lo que denominamos contorno del mapa local. Será el contorno interior de la *RLOI* a la que se llega en el paso anterior, es decir, el conjunto de puntos conectados que forman el borde interior de la *RLOI*. El contorno $\overrightarrow{C}$ va a consistir en un vector unidimensional de K puntos. Cada uno de esos puntos, C_k, viene determinado por la posición (x, y) que ocupa en la imagen. Formalmente, el contorno $\overrightarrow{C}$ será:

$$\overrightarrow{C} = [C_1, C_2, \cdots, C_K] = [(x_1, y_1), (x_2, y_2), \cdots, (x_K, y_K)] \tag{4.7}$$

Al finalizar la fase de *Cálculo del contorno del mapa local* el mapa local inicial se ha reducido a un vector de tamaño variable K que es el contorno $\overrightarrow{C}$ de su *RLOI*. Por lo tanto, el volumen de datos del mapa local de partida se ha reducido considerablemente, puesto que el contorno $\overrightarrow{C}$ tiene una dimensión significativamente inferior al mapa probabilístico inicial.

El proceso de extracción del contorno $\overrightarrow{C}$ de la *RLOI* de un mapa local se ilustra en la Figura 4.3, donde se presentan las distintas etapas del algoritmo para tres entorno típicos que se corresponden con una pared (Figura 4.3a), un pasillo (Figura 4.3b), y una puerta (Figura 4.3c). El mapa local inicial de 113x113 celdas se ve reducido a un contorno de 338, 423 y 614 puntos en las Figuras 4.3a, b y c, respectivamente. Esto quiere decir que el gran volumen de datos

inicial se reduce a tan solo un 2.64%, un 3.31% y un 4.80%, respectivamente. Por lo tanto, uno de los objetivos marcados está cumplido, puesto que se ha conseguido reducir drásticamente la información necesaria para representar una determinada situación del robot.

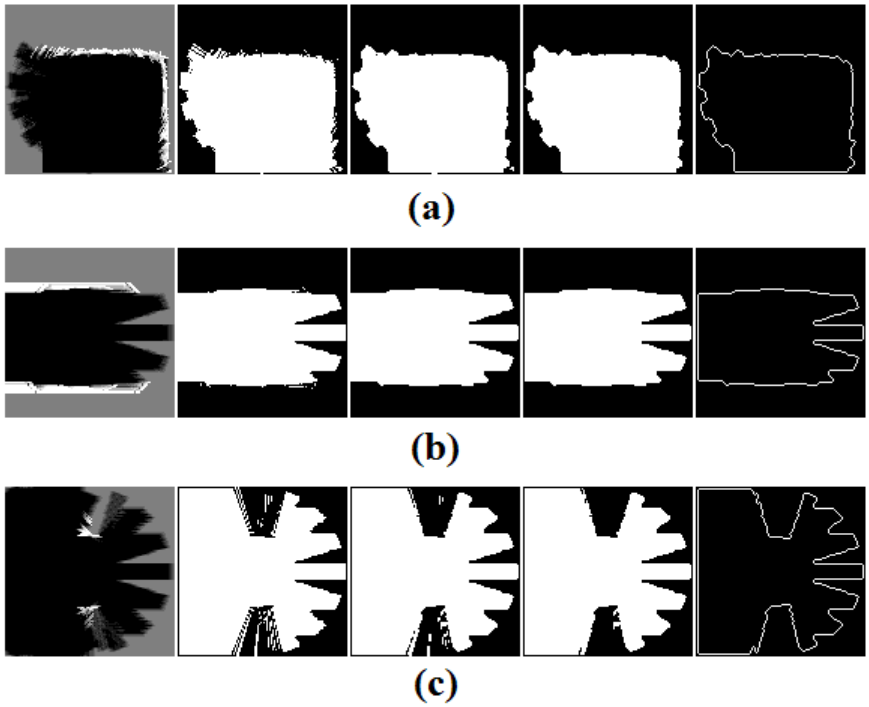

Figura 4.3: Cálculo del contorno del mapa local: mapa local, umbralización, filtrado, *RLOI*, contorno; (a) pared; (b) pasillo; (c) puerta.

Al llevar a cabo la fase de *Umbralización* se obtiene una imagen del mismo tamaño que el mapa local de partida, donde a las zonas de interés en el entorno más cercano al robot se les asigna el valor G_{region}. Como se aprecia en la Figura 4.3, el mapa local umbralizado es ruidoso y presenta varias zonas no conectadas como regiones de interés, según se puede ver en los tres casos, aunque quizás se observa con mayor claridad en la Figura 4.3c. Es en esta figura, la correspondiente a un mapa local adquirido cerca de una puerta, donde se observa una característica del mapa local umbralizado, la aparición habitual de una especie de rayos radiales que confluyen al centro de la imagen, que es la posición del agente en el mapa local. Son debidos al análisis radial efectuado durante la fase de *Umbralización* para determinar si una celda de espacio libre está más cercana o no que el obstáculo más próximo en su dirección. Este hecho dota al mapa local umbralizado de un mayor ruido, e implica la potencial aparición de regiones de dimensiones pequeñas no conectadas.

Con la fase de *Filtrado* se reduce tanto el ruido como el número de regiones presentes. En la Figura 4.3b se ha conseguido reducir el número de regiones a una única región, que conforma la *RLOI*. Sin embargo, y a pesar del filtrado del mapa local umbralizado, tanto en la Figura 4.3a como en la Figura 4.3c aparecen varias regiones no conectadas. Concretamente, surgen 2 regiones para la Figura 4.3a y 10 para la Figura 4.3c. Aunque para un observador humano es obvia la selección de la *RLOI*, para el agente operando de forma autónoma no lo es tanto. El

objetivo de la etapa de *Obtención de la RLOI* es la selección de la región en la que se encuentra el robot, la cual incorpora la configuración del espacio libre en las cercanías del agente, de entre todas las regiones no conectadas que aparecen. Finalmente, se obtiene el contorno $\overrightarrow{C}$ de la *RLOI*, que representa con un menor número de datos el mapa local de partida generado por el robot en un determinado lugar del entorno. Si observamos la Figura 4.3 se aprecia que, cualitativamente, los contornos de la *RLOI* de los distintos mapas locales difieren en gran medida para la pared, el pasillo y la puerta. El sistema, mediante las siguientes fases del algoritmo de caracterización de lugares, va a permitir que a partir del contorno de la *RLOI* del mapa local se pueda caracterizar automáticamente una determinada situación del robot.

2.2 Cálculo del mapa de profundidad

El tamaño del vector con el contorno $\overrightarrow{C}$ de la *RLOI* de un determinado mapa local es variable, dependiendo del mapa local a procesar. Sin embargo, ya que el método de caracterización de lugares usa el *PCA*, todos los vectores usados para obtener el vector de características final deben tener el mismo tamaño. Y además, como el siguiente paso del algoritmo consiste en el cálculo de una *FFT*, es recomendable que el vector usado tenga un tamaño que sea una potencia entera de 2. Así, durante la fase de *Cálculo del mapa de profundidad* se reduce el vector con el contorno de tamaño variable, $\overrightarrow{C}$, a un vector de dimensión fija potencia de 2, $\overrightarrow{MP}$, el mapa de profundidad.

Para nosotros, el mapa de profundidad $\overrightarrow{MP}$ es un vector de tamaño N que modeliza la distancia medida por N sensores equiespaciados en un anillo de sensores en la posición del robot. Es decir, estos sensores virtuales cubren todo el campo de visión posible de 0^o a 360^o, conformando el vector:

$$\overrightarrow{MP} = [MP_1, MP_2, \cdots, MP_N] \tag{4.8}$$

Para calcular $\overrightarrow{MP}$ a partir de $\overrightarrow{C}$ es necesario conocer en primer lugar la orientación de cada uno de los sensores virtuales que contribuyen al mapa de profundidad. Estas orientaciones dan lugar al vector de ángulos $\overrightarrow{\alpha}$:

$$\overrightarrow{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_N] \tag{4.9}$$

siendo la diferencia angular entre dos sensores consecutivos:

$$\triangle\alpha = \frac{360^o}{N} \tag{4.10}$$

El segundo paso consiste en obtener, a partir del contorno original $\overrightarrow{C}$ de tamaño variable K, un vector con los puntos del contorno cuya orientación respecto del sistema de referencia más se acerque a las orientaciones representadas en el vector $\overrightarrow{\alpha}$:

$$\overrightarrow{C}^N = [C_1^N, C_2^N, \cdots, C_N^N] = [(x_1^N, y_1^N), (x_2^N, y_2^N), \cdots, (x_N^N, y_N^N)] \tag{4.11}$$

donde cada una de las componentes C_i^N se evalúa mediante la siguiente expresión:

$$C_i^N = C_k \text{ tal que } |\arctan\frac{y_k - c_y}{x_k - c_x} - \alpha_i| = \min_{1 \leq m \leq K} |\arctan\frac{y_m - c_y}{x_m - c_x} - \alpha_i| \tag{4.12}$$

El vector $\overrightarrow{C}^N$ almacena la posición (x, y) de los pixeles que forman parte del borde interior de la $RLOI$ que cumplen una condición en cuanto a su orientación. Por otro lado, el mapa de profundidad almacena distancias. Parece que una y otra representación difieren en cuanto a la información almacenada. Sin embargo, esto no supone ningún problema, puesto que el mapa local es un mapa probabilístico donde hay una correspondencia geométrica entre el entorno y su modelo a través del tamaño de las celdas ($40mm$ en esta Tesis), y de su posición en el mapa. Por lo tanto, el contorno también almacena implícitamente esta información. Así pues, el último paso que hay que dar es el cálculo de la distancia geométrica que representa cada punto del contorno. Estas distancias constituyen el mapa de profundidad, evaluándose mediante la siguiente expresión:

$$MP_i = \left[\sqrt{(x_i^N - c_x)^2 + (y_i^N - c_y)^2}\right] \times 40mm \tag{4.13}$$

donde MP_i es la distancia medida por el sensor virtual cuya orientación es α_i.

En esta Tesis se emplea un mapa de profundidad de tamaño $N = 64$. Es como si se dispusiera de un robot con un anillo de 64 sensores que cubren de 0^o a 360^o, separados por una diferencia de $\triangle\alpha = 5.625^o$. Se podría pensar que no hace falta realizar todo el procesado de las fases de *Cálculo del contorno del mapa local* y de *Cálculo del mapa de profundidad*, ya que con un robot que disponga de un anillo de 64 sensores es más que suficiente. No obstante, éste no es nuestro escenario. Todas las pruebas realizadas en la presente Tesis se llevan a cabo, bien con el simulador de la plataforma robótica *Nomad 200*, bien con la plataforma robótica real *Pioneer P2AT*. En el primer caso el robot consta de un anillo de 16 sensores equiespaciados que le proporcionan visión de 0^o a 360^o, aunque con una resolución bastante peor que la conseguida con 64 lecturas. En el segundo, el robot únicamente tiene visión delantera de -90^o a 90^o gracias a 8 sensores frontales, con lo que la situación es bastante más problemática, puesto que nunca se adquiere información del entorno en la parte trasera del agente. Aunque el *hardware* disponible justifica el procesado efectuado para tener mayor resolución, mayor campo de visión y mayor número de lecturas, existe otra ventaja fundamental de nuestra aproximación respecto de la captura instantánea de las lecturas de los sensores sonar. Como la lectura de cualquier sensor sonar puede ser errónea, la integración de las lecturas en el tiempo y el espacio que los mapas locales realizan ayuda a corregir estos errores [LA05].

Las Figuras 4.4a, b y c muestran los mapas de profundidad que se obtienen al final de la fase de *Cálculo del mapa de profundidad* a partir de los contornos de la Figuras 4.3a, b y

c, respectivamente. Estos mapas de profundidad no se representan en función de su posición dentro del vector, porque la información gráfica que muestran es muy reducida. En su lugar se ha optado por emplear una representación más clarificadora, puesto que considera tanto la distancia como la orientación de cada punto. Finalmente, es necesario hacer notar que el mapa local inicial de 113x113 celdas se reduce, al concluir esta etapa, a un vector de únicamente 64 componentes, lo que supone una reducción del 99.5% en el volumen de datos. Por supuesto, el mapa de profundidad preserva la forma y la topología tanto del mapa local de partida como del contorno de su *RLOI*.

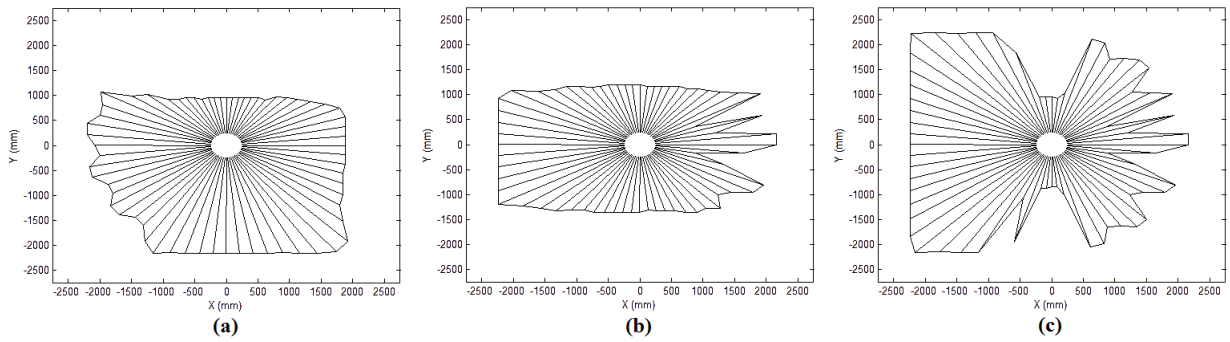

Figura 4.4: Mapas de profundidad del contorno de la *RLOI* de los mapas locales en la Figura 4.3: (a) pared; (b) pasillo; (c) puerta.

2.3 Obtención de la FFT del mapa de profundidad

Los mapas de profundidad presentan un volumen de datos mucho menor que el mapa local inicial. Sin embargo, sería deseable disminuir aun más su tamaño para así conseguir un vector de características de menor número de componentes que ocupara menor cantidad de memoria y que fuera más rápidamente comparable y más fácilmente analizable. El primer paso de esta reducción se realiza en base al cálculo de la Transformada Discreta de Fourier (*Discrete Fourier Transform*, *DFT*) del mapa de profundidad. En nuestro sistema, la *DFT* se calcula mediante el algoritmo de la Transformada Rápida de Fourier (*Fast Fourier Transform*, *FFT*), propuesto por Danielson y Lanczos en 1942 [DL42]. Su complejidad es de tan solo $O(N \log_2 N)$, por lo que se reduce la complejidad aparente $O(N^2)$ de la *DFT*. De este modo se llega al vector $\overrightarrow{FFT}$, que es la *DFT* del mapa de profundidad $\overrightarrow{MP}$ calculada mediante el algoritmo de la *FFT*:

$$\overrightarrow{FFT} = [FFT_1, FFT_2, \cdots, FFT_N] \tag{4.14}$$

donde cada punto del vector $\overrightarrow{FFT}$, FFT_n, se evalúa mediante la siguiente expresión:

$$FFT_n = \sum_{k=1}^{N} MP_k e^{-2\pi j(k-1)(n-1)/N},\ n = 1, 2, \cdots, N \tag{4.15}$$

La orientación del mapa de profundidad no puede afectar en el proceso de caracterización

de lugares. Dos mapas de profundidad similares con un desfasaje entre ellos se corresponden con una misma situación para el agente. Por lo tanto, del vector $\overrightarrow{FFT}$ únicamente nos interesa su módulo porque es invariante a las rotaciones, descartando su fase. A partir del vector $\overrightarrow{FFT}$ se calcula su módulo $|\overrightarrow{FFT}|$ a través de la obtención del módulo $|FFT_i|$ en cada uno de sus puntos FFT_i:

$$|\overrightarrow{FFT}| = [|FFT_1|, |FFT_2|, \cdots, |FFT_N|] \tag{4.16}$$

En las Figuras 4.5a, b y c se presenta el $|FFT|$ al que se llega al final de la fase de *Obtención de la FFT del mapa de profundidad*, a partir de los mapas de profundidad de las Figuras 4.4a, b y c, respectivamente. En los tres casos, la principal característica que se observa es que la información relevante se encuentra concentrada en las bajas frecuencias, siendo la mayor parte de los puntos del vector $|\overrightarrow{FFT}|$ cercanos a cero. Esta afirmación es cierta independientemente de que el mapa local de partida del método sea el correspondiente a una pared, un pasillo o una puerta. Este hecho nos va a servir de ayuda en la reducción, aun más, del tamaño del vector que representa a un determinado lugar. Finalmente, decir que al concluir esta etapa se obtiene un vector del mismo tamaño que el mapa de profundidad, N, pero invariante ante las rotaciones.

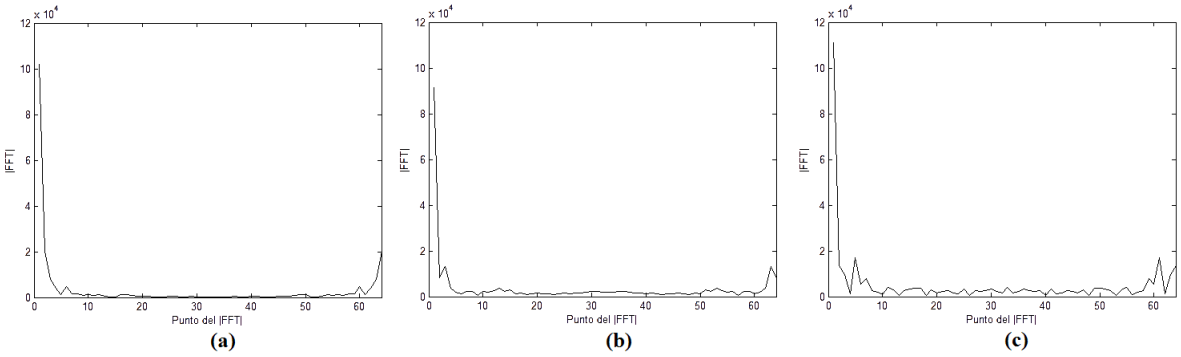

Figura 4.5: $|FFT|$ de los mapas de profundidad de la Figura 4.4: (a) pared; (b) pasillo; (c) puerta.

2.4 Extracción del vector de características

Partiendo del hecho de que la información relevante del $|FFT|$ se encuentra principalmente a baja frecuencia, se puede reducir todavía más el tamaño del vector usado para caracterizar un determinado lugar. Esta reducción se consigue mediante el Análisis de Componentes Principales (*Principal Component Analysis*, *PCA*), también conocido como expansión de Karhunen-Loëve [KS90].

El *PCA* es una técnica muy conocida para reducir la dimensión de un conjunto de datos de variables correladas manteniendo al mismo tiempo la mayoría de su variación original [VK99]. La técnica del *PCA* es debida a Hotelling [Hot33], aunque sus orígenes hay que buscarlos en 1901 en el trabajo de Pearson sobre los ajustes de mínimos cuadrados [Pea01]. Su principal

utilidad es la trasformación de las variables originales, en general correladas, en nuevas variables incorreladas, facilitando la interpretación de los datos.

El método del PCA ha sido usado con anterioridad en Robótica, sobre todo en temas de localización. La información de entrada para el PCA varía dependiendo de la técnica. Así, nos encontramos con aproximaciones visuales [KB99, CP01, VMK02, TZ04], con métodos que usan las lecturas de un láser [CWS98] o información procedente de sensores sonar [VMK00]. En la presente Tesis se emplea el PCA para la caracterización de lugares, no para tareas de localización. La información de entrada del análisis consiste en el vector $\overrightarrow{|FFT|}$, que representa el $|FFT|$ del mapa de profundidad del contorno de la $RLOI$ del mapa local. En cierto sentido, los datos de entrada proceden de los sensores sonar, puesto que sus lecturas van siendo integradas a medida que se genera y actualiza el mapa local.

Todos los vectores $\overrightarrow{|FFT|}$ con el $|FFT|$ del mapa de profundidad del contorno de la $RLOI$ del mapa local tienen una dimensión N. Por lo tanto, el conjunto de todos los vectores $\overrightarrow{|FFT|}$ posibles se puede representar en un espacio vectorial de dimensión N. Sin embargo, tal y como se ha comentado anteriormente, la mayor parte de los puntos del vector $\overrightarrow{|FFT|}$ son cercanos a cero, de ahí que se puede suponer que todos los posibles $\overrightarrow{|FFT|}$ conforman un subespacio vectorial de dimensión P del espacio vectorial de dimensión N, con $P < N$.

Sea $\left\{\overrightarrow{|FFT|}_m\right\}_{m=1}^{M}$ un conjunto de M módulos de la FFT del mapa de profundidad del contorno de la $RLOI$ de M mapas locales. Suponiendo que todos los $\overrightarrow{|FFT|}$ pertenecen a un subespacio vectorial de dimensión P, para aproximar cualquier $\overrightarrow{|FFT|}$ con un vector de únicamente P componentes, se necesita calcular una base de dicho subespacio vectorial de dimensión P, $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$, tal que:

$$\overrightarrow{|FFT|}_i \cong \sum_{k=1}^{P} \beta_{ik} \overrightarrow{\Phi}_k \tag{4.17}$$

donde $\{\beta_{ik}\}_{k=1}^{P}$ es el vector de características de componentes principales del vector $\overrightarrow{|FFT|}_i$, obtenido al proyectar $\overrightarrow{|FFT|}_i$ sobre la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$. Si la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$ es óptima, el error cuadrático medio, ε^2, será mínimo:

$$\varepsilon^2 = \frac{1}{M} \sum_{i=1}^{M} \left| \overrightarrow{|FFT|}_i - \sum_{k=1}^{P} \beta_{ik} \overrightarrow{\Phi}_k \right| \tag{4.18}$$

El error ε^2 es mínimo cuando los P vectores de la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$ se extraen del conjunto de M autovectores de la matriz de covarianza de $\left\{\overrightarrow{|FFT|}_m\right\}_{m=1}^{M}$, C_{FFT}:

$$C_{FFT} = \frac{1}{M} \sum_{i=1}^{M} \overrightarrow{|FFT|}_i \cdot \overrightarrow{|FFT|}_i^T \tag{4.19}$$

La matriz de covarianza C_{FFT} tiene M autovalores. Los autovalores ordenados de mayor a menor valor, $\lambda_1 > \lambda_2 > \cdots > \lambda_M$, tienen asociado su correspondiente autovector $\overrightarrow{\Phi}_1, \overrightarrow{\Phi}_2, \cdots, \overrightarrow{\Phi}_M$. En estas condiciones, el conjunto de los P primeros autovectores conforma la base de P vectores ortogonales del subespacio vectorial del dimensión P, $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$. La proyeccción sobre esta base de cada vector $\overrightarrow{|FFT|}_i$ del conjunto inicial da lugar a su vector de componentes principales $\overrightarrow{\beta}_i$, que constituye su vector de características, $\overrightarrow{VC}_i$:

$$\overrightarrow{VC}_i = \overrightarrow{\beta}_i = [\beta_{i1}, \beta_{i2}, \cdots, \beta_{iP}] \tag{4.20}$$

Cualquier vector $\overrightarrow{|FFT|}_j$ no incluido en el conjunto inicial $\left\{\overrightarrow{|FFT|}_m\right\}_{m=1}^{M}$ se reduce a un vector de características $\overrightarrow{VC}_j$, constituido por las primeras P componentes principales, $\{PC_{jk}\}_{k=1}^{P}$:

$$\overrightarrow{VC}_j = [VC_{j1}, VC_{j2}, \cdots, VC_{jP}] = [PC_{j1}, PC_{j2}, \cdots, PC_{jP}] \tag{4.21}$$

donde PC_{jk} se obtiene al proyectar $\overrightarrow{|FFT|}_j$ sobre el vector $\overrightarrow{\Phi}_k$ de la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$:

$$VC_{jk} = PC_{jk} = \overrightarrow{|FFT|}_j \cdot \overrightarrow{\Phi}_k = \sum_{m=1}^{N} |FFT|_{jm}\Phi_{km} \tag{4.22}$$

Por lo tanto, con la técnica del PCA, cualquier vector $\overrightarrow{|FFT|}_j$ de dimensión N se verá transformado en un vector de características, $\overrightarrow{VC}_j$, de menor dimensión P. No obstante, para poder calcular el vector de características es necesario disponer de la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$. Una vez se está en posesión de la base, ya sí que se pueden proyectar los vectores $\overrightarrow{|FFT|}_j$ para obtener $\overrightarrow{VC}_j$. Por este motivo, se analiza en primer lugar en la siguiente sección el proceso seguido para obtener una base en nuestro sistema, para pasar a continuación a estudiar el procedimiento requerido para calcular el vector de características final.

2.4.1 Obtención de la base

El cálculo de la base es un proceso laborioso desde el punto de vista matemático. Sin embargo, solamente hay que realizarlo una vez y además *off line*. Para demostrar que no es necesario un elevado número de vectores para obtener una base que dé buenos resultados, en nuestro sistema se calcula a partir de únicamente 17 mapas locales arbitrariamente escogidos en entornos simulados (Figura 4.6). Los 17 mapas locales incluyen configuraciones correspondientes a paredes (Base 1, Base 2, Base 3, Base 4 y Base 5), pasillos (Base 11, Base 12, Base 13, Base 14 y Base 15), puertas (Base 7, Base 9 y Base 10), y situaciones que no pertenecen de forma nítida a ninguna de las tres categorías anteriores (Base 6, Base 8, Base 16 y Base 17). Se ha hecho de este modo para dotar de mayor diversidad a la base y conseguir así que se puedan caracterizar lugares en una de las tres categorías de nuestro sistema, pared, pasillo o puerta. Debemos hacer notar que la base que

se calcula en esta sección es la usada en todas las pruebas y experimentos que se han llevado a cabo. Estos tests incluyen entornos simulados y entornos reales, diferentes de aquéllos donde se capturaron los mapas locales usados para calcular la base, para probar que la base puede usarse en entornos no conocidos, ya que en esta Tesis los entornos de prueba son total o parcialmente desconocidos. También es significativo el hecho de que, incluso habiendo obtenido la base a partir de mapas locales adquiridos en entornos simulados, se puede usar satisfactoriamente en entornos reales.

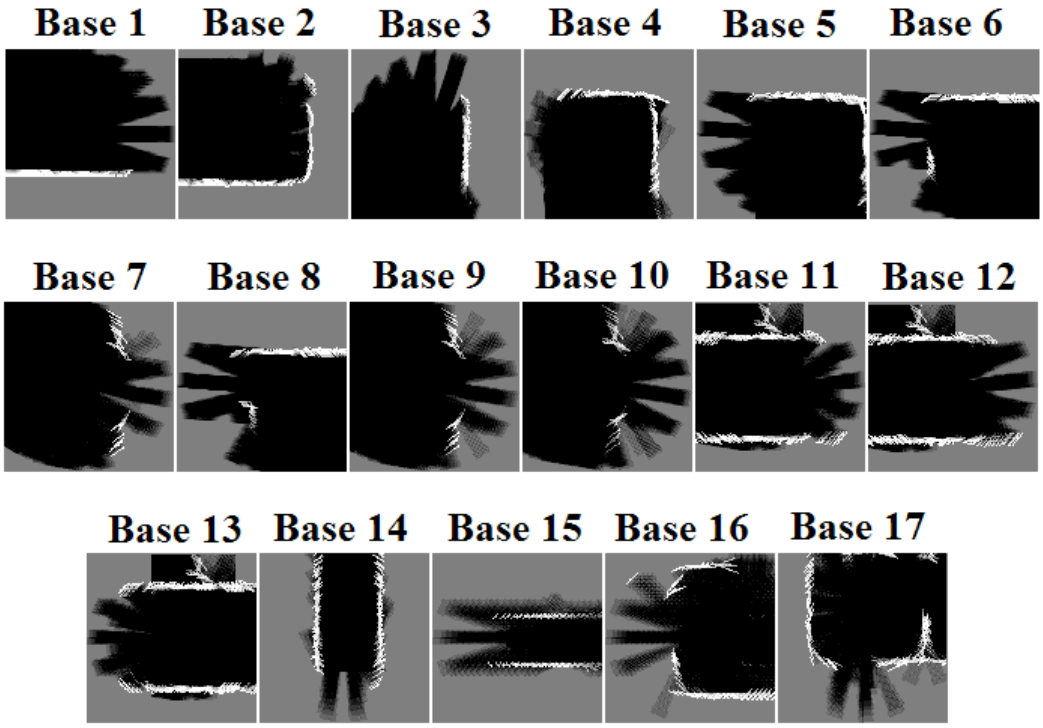

Figura 4.6: Mapas locales a partir de los que se calcula la base del sistema.

Los 17 mapas locales son procesados hasta que se llega al conjunto de 17 vectores de tamaño $N = 64$, $\left\{\overrightarrow{|FFT|}_m\right\}_{m=1}^{17}$ con el $|FFT|$ del mapa de profundidad del contorno de su *RLOI*. Suponiendo que todos los $|FFT|$ pertenecen a un subespacio vectorial de dimensión P ($P < N$), se calcula la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$ mediante el *PCA*, donde el tamaño máximo de la base es 17, puesto que solamente tenemos 17 vectores de entrada. Para ello nos valemos de las utilidades de que dispone *Matlab*, facilitando nuestra tarea. Al final del proceso se obtiene la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{P}$ como el conjunto de los P primeros autovectores de la matriz de covarianza C_{FFT}, que tienen asociados los autovalores $\{\lambda_k\}_{k=1}^{P}$ ordenados de mayor a menor valor.

La selección del tamaño P de la base se realiza en función del análisis del porcentaje de la varianza explicada por cada una de las componentes. Se puede demostrar que el porcentaje de la varianza explicado por una componente principal, PC_i, viene dado por el porcentaje de su autovalor asociado λ_i respecto de la suma de todos los autovalores:

$$Var(PC_i) = \frac{100\lambda_i}{\sum_{k=1}^{P} \lambda_k} \tag{4.23}$$

siendo el porcentaje de la varianza explicada por las m primeras componentes principales:

$$\sum_{k=1}^{m} Var(PC_k) = 100\frac{\sum_{k=1}^{m} \lambda_k}{\sum_{k=1}^{P} \lambda_k} \tag{4.24}$$

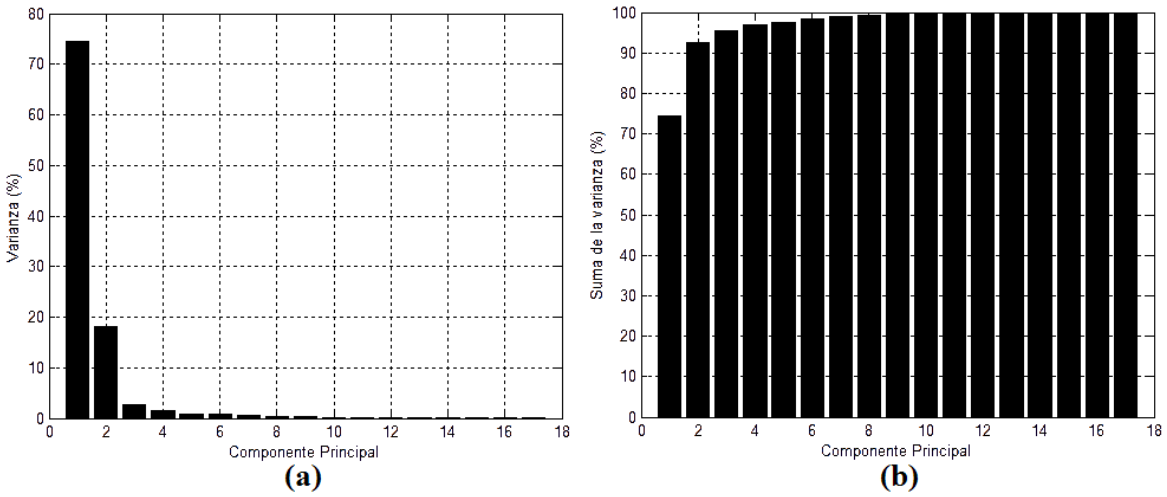

Figura 4.7: (a) Porcentaje de la varianza explicado por cada componente principal; (b) porcentaje de la varianza explicado por las m primeras componentes principales.

En las Figuras 4.7a y b se muestra, respectivamente, el porcentaje de la varianza explicado por cada una de las componentes principales definido por la Ecuación (4.23), y el porcentaje de la varianza explicado por las m primeras componentes principales según la expresión de la Ecuación (4.24). Se aprecia perfectamente que la primera componente explica la mayoría de la información, un 74.5% (Figura 4.7a). La información explicada por la segunda componente es, obligatoriamente, mucho inferior. El porcentaje de la varianza explicada decrece hasta un 18%. Esto implica que, a partir de la tercera componente el porcentaje es significativamente inferior al de las dos primeras. En concreto, a partir de esa tercera componente la información explicada es siempre inferior al 3%. En la Figura 4.7b se observa que a partir de la novena componente, la información explicada es ínfima. De hecho, da la impresión de que a partir de esa novena componente se consigue recoger el 100% de la varianza explicada, debido a la escala de la representación y la resolución de la imagen. En realidad, las nueve primeras componentes explican el 99.5% de la información, es decir, muy próximo al máximo. Como deseamos que el vector de características recoja la máxima información posible para poder discriminar entre lugares distintos, en nuestro sistema se ha optado por una base, $\left\{\vec{\Phi}_k\right\}_{k=1}^{9}$, de dimensión $P = 9$.

Tal y como hemos comentado anteriormente, esta base es la usada en todas las pruebas y experimentos realizados.

Un aspecto sobre el que debemos reflexionar es si, efectivamente, la base contiene la diversidad que se le supone. Por otro lado, también es necesario añadir algún comentario sobre si un vector de características tan reducido es en verdad significativo. En este sentido podría pensarse que el vector con los $|FFT|$ de tamaño $N = 64$ ya es suficiente reducción del mapa local de partida. Sin embargo, en la Figura 4.8a se observa la dificultad para extraer información relevante que nos permita discriminar unas situaciones respecto de otras.

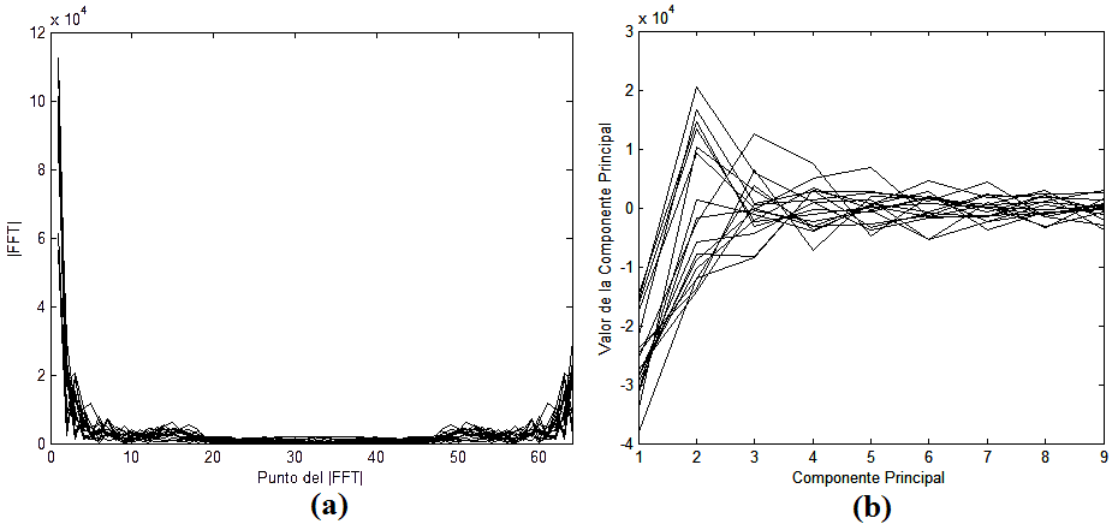

Figura 4.8: (a) $|FFT|$ del mapa de profundidad del contorno de la $RLOI$ de los mapas locales de la base; (b) vectores de características de los mapas locales de la base.

En la Figura 4.8a se muestran los $|FFT|$ del mapa de profundidad del contorno de la $RLOI$ de los mapas locales que participan en el cálculo de la base (Figura 4.6). Tal y como se analizó durante la fase de *Cálculo de la FFT del mapa de profundidad*, la mayoría de la información se encuentra en baja frecuencia, estando muchos de los puntos del $|FFT|$ próximos a cero. Así mismo, no se observa la variación y diversidad que se le presupone a los participantes en el cálculo de la base. Este hecho es potencialmente un problema, puesto que el proceso de caracterización se podría ver complicado de forma sustancial al hacerse difícil la diferenciación entre vectores debido a su parecido. Sin embargo, si analizamos los vectores de características de la Figura 4.8b, el análisis es radicalmente distinto. La Figura 4.8b muestra los vectores de características obtenidos a partir de los $|FFT|$ representados en la Figura 4.8a. Estos vectores de características se pueden obtener proyectando los $\overrightarrow{|FFT|}$ sobre la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{9}$, aunque no hace falta. Al tratarse de los vectores de características de los $|FFT|$ empleados para el cálculo de la base, se obtienen directamente durante dicho proceso. En este caso sí que se aprecia que, aunque existen conjuntos de vectores parecidos entre sí, existe una gran diversidad dentro de los vectores de características. Además, como con nueve componentes principales se explica la mayoría de la información (99.5%), el vector de características es suficiente como representante del correspondiente mapa local. Por lo tanto, podemos concluir que la base incluye situaciones

diversas que se pueden presentar en un entorno arbitrario. Aunque este hecho no asegura la consecución de los objetivos en nuestra sistema, apartados posteriores analizan y demuestran la utilidad, tanto de la base como del vector de características propuesto, para la caracterización de lugares.

2.4.2 El vector de características

Una vez se dispone de la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{9}$, cualquier mapa local capturado en un determinado lugar del entorno es representado por su vector de características $\overrightarrow{VC}$ de $P = 9$ componentes. El vector de características se obtiene proyectando el $|FFT|$ del mapa de profundidad de la $RLOI$ sobre la base $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{9}$. Así pues, un mapa local de 113x113 celdas se ve finalmente reducido a un vector de 9 componentes. Es decir, se ha conseguido reducir el tamaño del problema a tan solo un 0.07% de su tamaño inicial, y a un 14% del tamaño del mapa de profundidad ($N = 64$). El vector de características $\overrightarrow{VC}$ con el que se representa la situación en una determinada posición del entorno encapsula la configuración del entorno en las cercanías de dicha posición. Al ser muy compacto, es eficiente desde el punto de vista del almacenamiento y del tiempo requerido para compararlo con otros vectores de características.

La Figura 4.9 muestra los vectores de características que representan los mapas locales de la Figura 4.3. Se obtienen después de ejecutar las cuatro fases del proceso de obtención de un vector de características (Figura 4.1). Los tres mapas locales de la Figura 4.3 son procesados hasta alcanzar el $|FFT|$ del mapa de profundidad del contorno de su $RLOI$. Los $\overrightarrow{|FFT|}$ son proyectados sobre la base de nuestro sistema, $\left\{\overrightarrow{\Phi}_k\right\}_{k=1}^{9}$, obteniéndose los vectores que se presentan en la Figura 4.9. Se observa que cualitativamente los tres vectores son diferentes entre sí, tal y como esperábamos, puesto que corresponden a tres situaciones distintas, una pared, un pasillo y una puerta. Esta apreciación cualitativa sugiere que el vector de características propuesto es útil para caracterizar un lugar en una de las tres categorías. Sin embargo, una apreciación subjetiva no es suficiente para poder trabajar, más aún cuando el sistema ha de funcionar de forma autónoma sin supervisión humana. Es necesario algún mecanismo que nos permita la caracterización en base a criterios cuantitativos, tema tratado en el siguiente apartado.

El vector de características se emplea en la presente Tesis para la caracterización de lugares. No obstante, al ser tan compacto y representar la situación en una determinada posición del entorno, hacen suponer que es apropiado para la localización global del robot en un entorno conocido, al igual que otras técnicas que emplean el PCA [KB99, VMK00, CP01, TZ04]. El método requeriría de la segmentación del entorno en regiones en función de los vectores de características capturados en distintas posiciones. Una vez segmentado el entorno, se irían obteniendo los vectores de características en la posición actual del agente y, por ejemplo, mediante los modelos ocultos de Markov (HMM), se determinaría la región en la que se encuentra el robot. Sin embargo, la localización es un problema externo que queda fuera de los objetivos de esta Tesis.

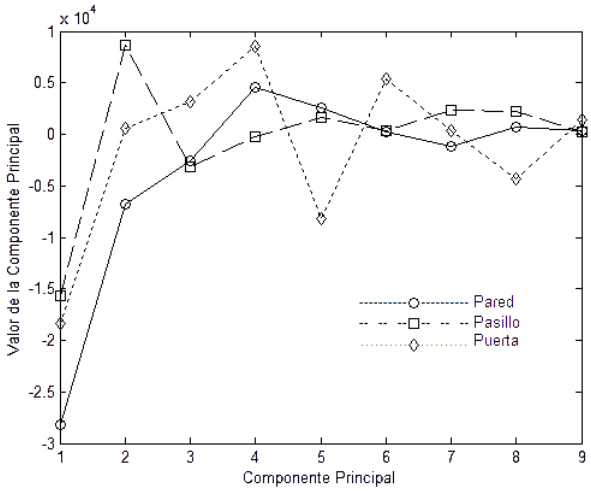

Figura 4.9: Vectores de características de los mapas locales de la Figura 4.3.

2.5 Análisis del vector de características

La caracterización de un lugar del entorno requiere del análisis del vector de características obtenido a partir del correspondiente mapa local. Este análisis permite clasificar el lugar en el que se encuentra el robot en una de las tres categorías de nuestro sistema: pared, pasillo y puerta. La caracterización se realiza en base a la comparación con una serie de patrones conocidos. La clasificación de un lugar en pared, pasillo y puerta permitiría la selección, respectivamente, del comportamiento Seguir Pared, Seguir Pasillo y Cruzar Puerta, como el mejor comportamiento de navegación para adaptarse al entorno en la posición actual del robot. Aunque el vector de características propuesto permite optar por la selección de un único comportamiento para navegación, no es la opción empleada en esta Tesis. Sabemos que para adaptarse apropiadamente a la configuración de un entorno total o parcialmente desconocido es mejor decantarse por la fusión o cooperación de comportamientos. Pues bien, nuestro vector de características también permite trabajar de este modo, contribuyendo al mecanismo de cálculo de los pesos o factores de ponderación que hay que aplicar a cada uno de los tres comportamientos en todo momento para que cooperen en la navegación reactiva. A continuación se presenta en primer lugar el proceso de comparación con los patrones conocidos para, posteriormente, indicar cómo se obtienen los pesos empleados en la fusión de los comportamientos.

2.5.1 Comparación del vector de características

El análisis de un vector de características se basa en su comparación con un conjunto de patrones definidos y conocidos. De este análisis se deduce cuál sería el comportamiento a escoger en el caso de que éstos fueran competitivos. Como en el sistema hay tres comportamientos reactivos, Seguir Pared, Seguir Pasillo y Cruzar Puerta, se han escogido patrones para las tres categorías con las que están relacionados, respectivamente, pared, pasillo y puerta. En principio se pensó en usar un único patrón por categoría. Diferentes pruebas demostraron que esta opción no era

recomendable, puesto que para una misma categoría se pueden presentar situaciones diversas. Por este motivo, se han considerado varios patrones por categoría. De este modo se pueden incluir en los patrones varias situaciones que pertenecen a la misma categoría. El número de patrones se ha fijado experimentalmente a 3 para cada categoría, pues tampoco interesa considerar un número muy elevado para no complicar el proceso de comparación.

Los patrones consisten en los vectores de características extraídos a partir de mapas locales que representan situaciones correspondientes a una pared, un pasillo y una puerta. En la Figura 4.10 se presentan los mapas locales a partir de los cuales se obtienen los vectores de características que se usan como patrones en el sistema. La elección de estos mapas locales se ha realizado heurísticamente, intentando dotar de la diversidad suficiente a cada una de las tres categorías. De los 9 patrones, 3 están asociados con una pared, 3 con un pasillo y 3 con una puerta. Para el caso de una pared se han incluido tres situaciones típicas, mientras que para un pasillo se consideran tres anchuras diferentes. Finalmente, los patrones de una puerta tienen en cuenta tres escenarios típicos que se presentan en las proximidades de una puerta.

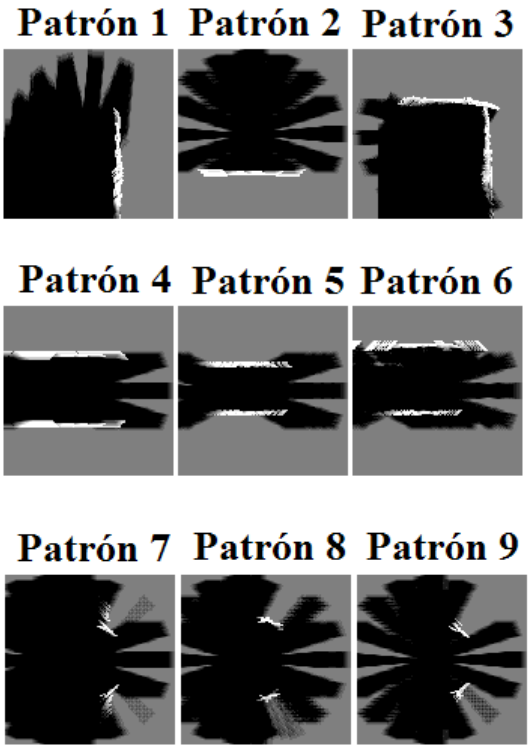

Figura 4.10: Mapas locales de los patrones del sistema. Patrón 1-3: pared; Patrón 4-6: pasillo; Patrón 7-9: puerta.

Los mapas locales de la Figura 4.10 son procesados con el algoritmo propuesto. Los $|FFT|$ del mapa de profundidad del contorno de su $RLOI$ son proyectados sobre la base de dimensión $P = 9$ del sistema, siendo entonces representados por su vector de características. Los 9 vectores de características asociados a los 9 mapas locales de la Figura 4.10 conforman los patrones del sistema. Estos vectores de características se muestran en la Figura 4.11. Como los patrones se corresponden con tres categorías, se muestra en primer lugar la comparación de los vectores de

características de los patrones pared (Figura 4.11a), después para los patrones pasillo (Figura 4.11b), y por último para los patrones puerta (Figura 4.11c). Así mismo, también se presenta la comparativa conjunta de los 9 patrones en la Figura 4.11d.

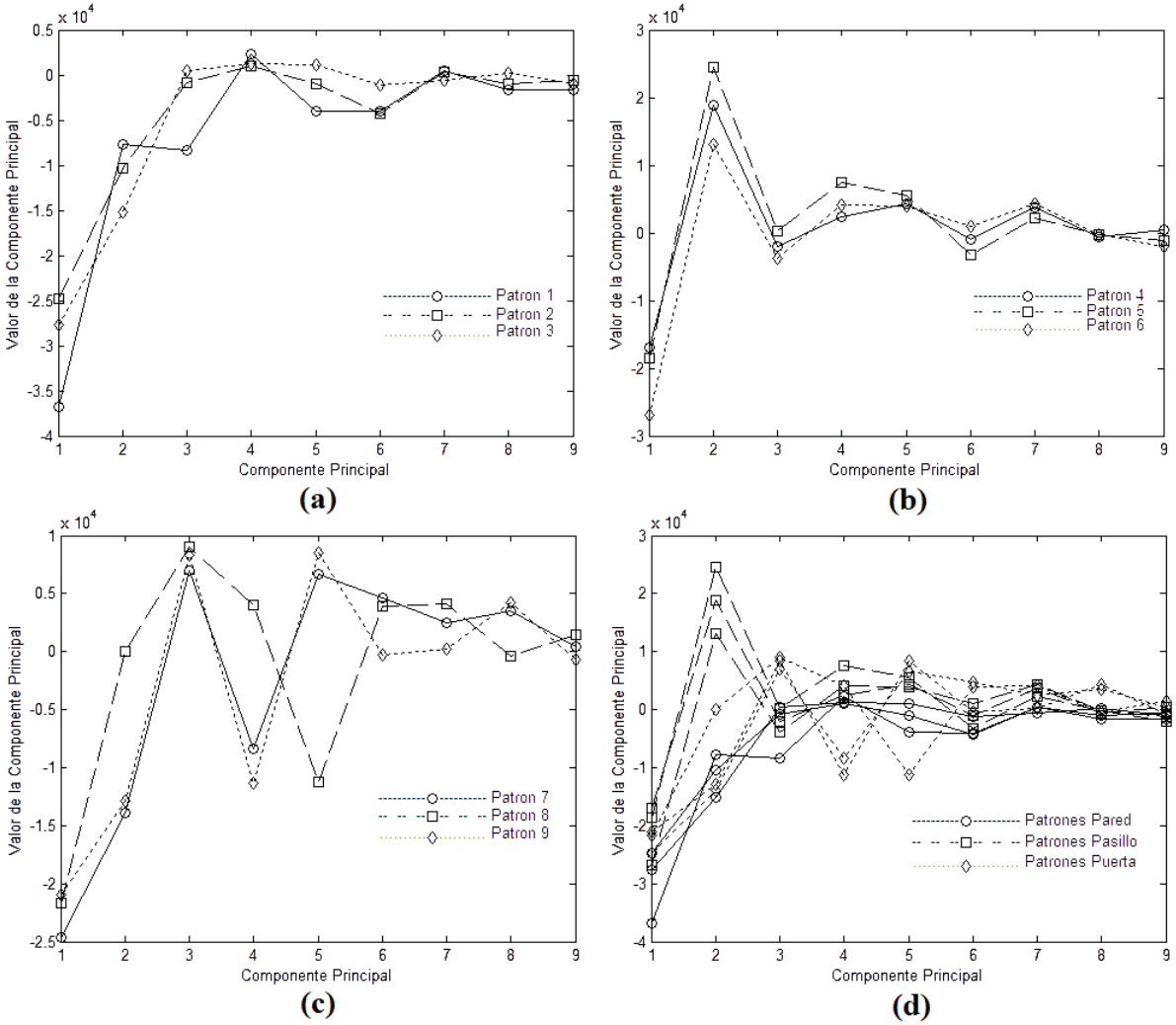

Figura 4.11: Vectores de características que conforman los patrones del sistema: (a) Patrón 1-3: pared; (b) Patrón 4-6: pasillo; (c) Patrón 7-9: puerta; (d) comparación de todos los patrones.

Al analizar cualitativamente las gráficas de la Figura 4.11 se observan algunos hechos interesantes que debemos comentar. En primer lugar, se aprecia que los vectores de características correspondientes a una pared tienen un aspecto de función creciente con el número de la componente principal. Si se extrajera la envolvente del vector de características sería una especie de exponencial creciente, algo así como la tensión medida en los terminales de un condensador en un circuito RC de primer orden con una excitación en forma de pulso. En cuanto a los patrones pasillo, se observa que el máximo está claramente definido por la segunda componente principal, siendo ese máximo mucho mayor en módulo que el resto de las componentes. Por último, para el caso de la puerta se observa que el vector de características presenta varios máximos y mínimos locales, oscilando el vector entre dichos valores. Así, se observan máximos en las componentes

principales tres y/o cinco, mientras que los mínimos los encontramos en las componentes principales cuatro o cinco. Aunque todas estas apreciaciones son cualitativas, sí que se observa la diversidad de los patrones en la Figura 4.11d. Obviamente, no podía ser de otro modo puesto que los mapas locales que dan lugar a los vectores de características que conforman los patrones representan situaciones muy diferentes.

El cálculo de los vectores de características que actúan como patrones del sistema solamente hay que realizarlo una vez y *off line*, siendo el proceso independiente de la operación del robot durante la exploración, al igual que la obtención de la base. Debemos hacer notar que los mapas locales de la Figura 4.10 son distintos de los mapas locales usados para obtener la base (Figura 4.6). Es decir, los dos procesos, obtención de la base y cálculo de los patrones, son independientes entre sí. Este hecho dota de una mayor flexibilidad al sistema, aumentando las posibilidades de generalización. Cuando la base está disponible, se puede seleccionar un conjunto de patrones diferente sin necesidad de recalcularla, incluso aumentando o disminuyendo su número. Igualmente, si la base se modifica, los patrones no tienen por qué hacerlo, salvo por el hecho evidente de que habría que proyectar de nuevo los $|FFT|$ para obtener los vectores de características de los mapas locales que dan lugar a dichos patrones. También tenemos que resaltar que el número de patrones es significativamente bajo, mientras que otros métodos necesitan una base de datos amplia con la que comparar para caracterizar lugares [PCJC06]. Además, todos los patrones fueron obtenidos a partir de mapas locales en entornos simulados. Estos patrones son los usados en todos los experimentos que se han llevado a cabo. En estas pruebas se incluyen entornos reales y simulados, distintos de aquéllos en los que se capturaron los mapas locales usados para obtener la base y, al mismo tiempo, diferentes de los entornos donde se capturaron los mapas locales de la Figura 4.10 que dan lugar a los patrones. Se ha hecho a propósito de este modo para probar que el método se puede aplicar a diferentes plataformas, incluso en entornos reales, a pesar de que los patrones se obtuvieron en entornos simulados.

La comparación de un vector de características con los patrones se debe realizar en base a algún criterio cuantitativo para poder extraer conclusiones que permitan el funcionamiento automático del método durante la operación del robot. La comparación se efectúa en función de la distancia entre dos vectores de características. En la presente Tesis se emplea la distancia de Tanimoto [HSWW03] como métrica para cuantificar el parecido entre vectores. Esta distancia favorece la similitud o correlación entre vectores de características, más que la distancia euclídea entre los mismos [WBD98]. Se ha descartado la distancia euclídea porque no es adecuada para realizar comparaciones absolutas [WBD98], mientras que la distancia de Tanimoto sí lo es. Cuando se trabaja con datos no binarios la distancia de Tanimoto varía entre 0 y $1+\frac{1}{3}$, siendo 0 la máxima similitud entre los vectores de entrada y $1+\frac{1}{3}$ la máxima diferencia [HHW02]. Sean $\overrightarrow{VC}_i = [VC_{i1}, VC_{i2}, \cdots, VC_{iP}]$ y $\overrightarrow{VC}_j = [VC_{j1}, VC_{j2}, \cdots, VC_{jP}]$ dos vectores de características extraídos de dos mapas locales. La distancia de Tanimoto entre $\overrightarrow{VC}_i$ y $\overrightarrow{VC}_j$ se define como:

$$D(\overrightarrow{VC}_i, \overrightarrow{VC}_j) = 1 - \frac{\overrightarrow{VC}_i^T \cdot \overrightarrow{VC}_j}{\left|\overrightarrow{VC}_i\right|^2 + \left|\overrightarrow{VC}_j\right|^2 - \overrightarrow{VC}_i^T \cdot \overrightarrow{VC}_j} \tag{4.25}$$

Con la métrica definida ya sí se pueden extraer conclusiones del análisis de los resultados cuantitativos que se obtienen. Así, en la Tabla 4.1 se presenta la distancia de Tanimoto existente entre los vectores de características que conforman los patrones. Al comentar cualitativamente el aspecto de los vectores de características de la Figura 4.11 se indicó la diversidad de los mismos. A simple vista parece que esos vectores de características son diferentes dependiendo de la categoría a la que pertenecen, pared, pasillo o puerta. Los resultados de distancia de la Tabla 4.1 corroboran este análisis. Los patrones de una pared son, en su forma, muy parecidos entre sí. La distancia entre ellos oscila entre 0.06 y 0.18. Más significativo todavía es el hecho de que la distancia a los patrones del pasillo y de la puerta son muy superiores, sobre todo en el caso del pasillo. Este dato numérico y objetivo se aprecia perfectamente observando la forma de los vectores de características representados en la Figura 4.11.

Patrón	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0.00	0.17	0.18	0.72	0.76	0.42	0.46	0.49	0.56
2	0.17	0.00	0.06	0.80	0.85	0.54	0.35	0.44	0.39
3	0.18	0.06	0.00	0.87	0.91	0.61	0.30	0.48	0.31
4	0.72	0.80	0.87	0.00	0.08	0.16	0.90	0.70	0.93
5	0.76	0.85	0.91	0.08	0.00	0.22	0.95	0.74	0.97
6	0.42	0.54	0.61	0.16	0.22	0.00	0.69	0.53	0.76
7	0.46	0.35	0.30	0.90	0.95	0.69	0.00	0.57	0.06
8	0.49	0.44	0.48	0.70	0.74	0.53	0.57	0.00	0.68
9	0.56	0.39	0.31	0.93	0.97	0.76	0.06	0.68	0.00

Tabla 4.1: Distancia de Tanimoto entre los vectores de características que conforman los patrones.

Para el caso del pasillo el panorama que se deduce de la Tabla 4.1 es similar. La distancia entre los patrones de un pasillo oscila entre 0.08 y 0.22, pues se consideran anchuras de pasillo diferentes. Aun así, esta distancia no muy alta entre ellos es lógica, toda vez si hacemos notar que la forma del vector de características tiene un máximo en la segunda componente principal (Figura 4.11b). La distancia al resto de los patrones (pared y puerta) es significativamente superior. De hecho, se ha comprobado experimentalmente que una configuración de pasillo da lugar a una distancia muy grande con respecto a los patrones de una pared y una puerta.

En último lugar tenemos los patrones de una puerta. Para estos patrones hay que hacer notar el dato que más llama la atención en la Tabla 4.1. El patrón ocho se encuentra muy alejado, en cuanto a su distancia de Tanimoto, de los otros dos patrones para una puerta, 0.57 y 0.68 para los patrones 7 y 9, respectivamente. Sin embargo, esto no es un problema en el sistema. Al contrario, le dota de una mayor diversidad pues, al ser la distancia entre los patrones de una misma categoría elevada, se están considerando diferentes escenarios que el robot puede encontrar en las proximidades de una puerta. Si observamos los mapas locales que dan lugar a los patrones (Figura 4.10), puede parecer que los tres son prácticamente similares, salvo por la anchura de la puerta. Sin embargo, un análisis más cuidadoso nos lleva a apreciar que el mapa local 8 presenta cinco lóbulos debidos a los sensores sonar, mientras que los patrones 7 y 9 solamente presentan 3. La elevada distancia entre los patrones de una misma categoría no se

da, no obstante, para la pared y el pasillo, porque no existen tantas situaciones diferentes que pueden aparecer como en el caso de una puerta.

Los resultados obtenidos no son solamente aplicables a los patrones. Para demostrar que el análisis realizado tiene validez y es aplicable a cualquier mapa local, se presentan en la Figura 4.12 nueve mapas locales de prueba capturados arbitrariamente en entornos reales. Por lo tanto, estos mapas locales son distintos de los usados para obtener tanto la base como los patrones del sistema. La simple inspección de los mapas locales de la Figura 4.12 nos llevaría a pensar que los tres primeros se corresponden con una pared, los mapas 4, 5 y 6 representan a un pasillo y los tres últimos a una puerta. Obviamente, para extraer conclusiones al respecto nos valemos de la métrica definida en vez de hacerlo mediante la observación por parte de un operador humano que supervisa el proceso.

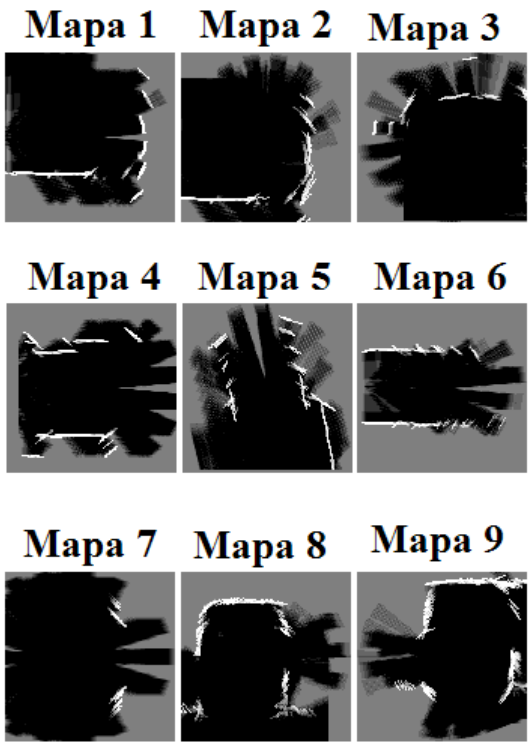

Figura 4.12: Mapas locales de prueba. Mapa 1-3: pared; Mapa 4-6: pasillo; Mapa 7-9: puerta.

En la Figura 4.13 se representan los vectores de características obtenidos al proyectar el $|FFT|$ del mapa de profundidad del contorno de la $RLOI$ de los mapas locales de la Figura 4.12 sobre la base del sistema. En primer lugar se comparan los vectores de características de los mapas locales que se suponen pertenecientes a la misma categoría (Figuras 4.13a, b y c). A continuación se muestran en la Figura 4.13d, superpuestos, los 9 vectores de características obtenidos. Se aprecia cualitativamente que, dentro de una misma categoría, los vectores de características son muy parecidos. Este hecho era esperable, en tanto en cuanto ocurre lo mismo para los patrones (Figura 4.11). Del mismo modo, el aspecto de los vectores de características de una determinada categoría es similar a los patrones de la Figura 4.11. Para la pared la representación tiene el aspecto de una función creciente exponencial, para el pasillo existe un

máximo en la segunda componente principal, y para la puerta hay un máximo en la tercera componente principal y un mínimo en la cuarta. Otro dato interesante es que los vectores de características de una categoría no se parecen mucho, a simple vista, a los de otra categoría distinta, como se aprecia en la Figura 4.13d.

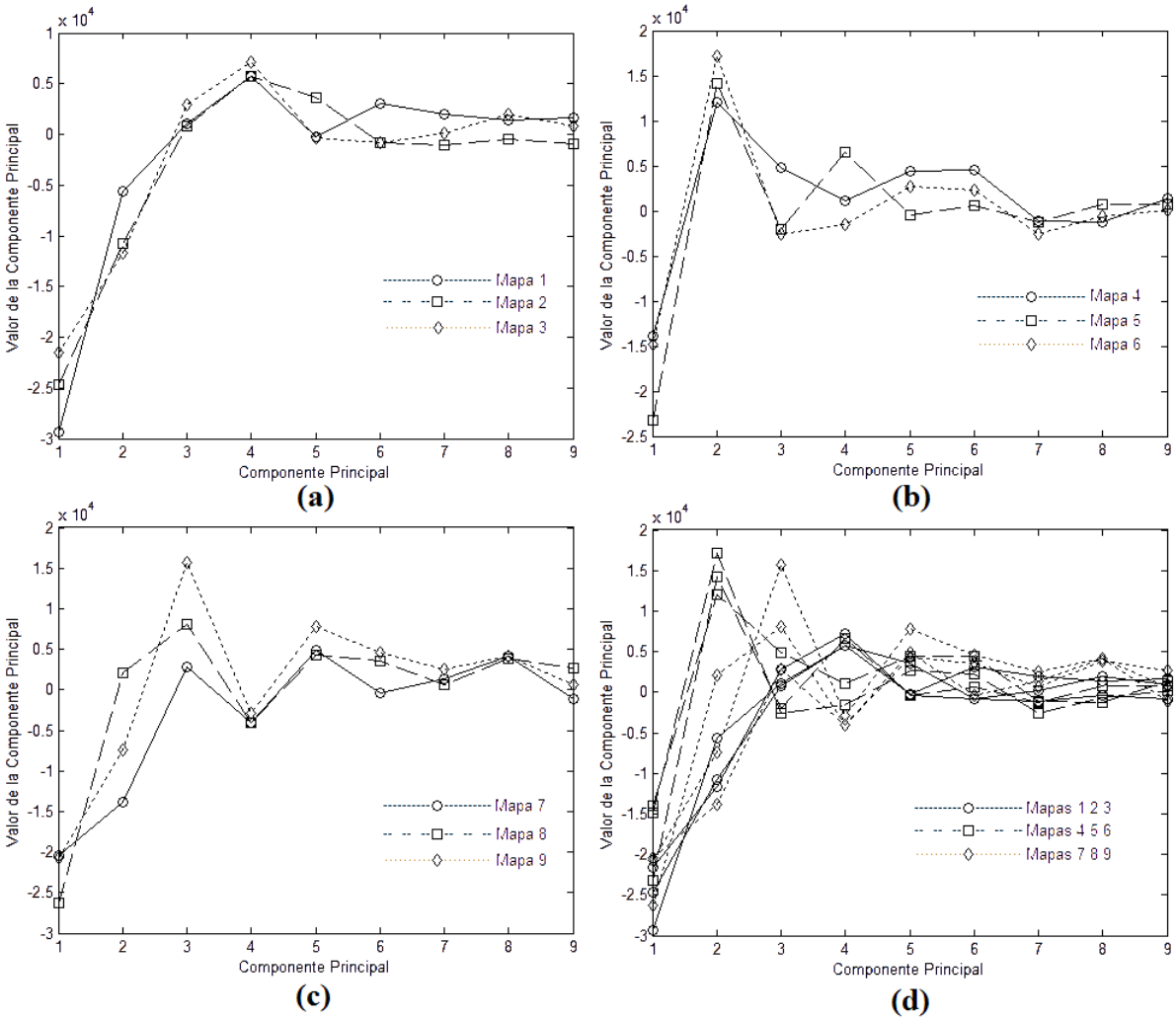

Figura 4.13: Comparación de los vectores de características de los mapas locales de prueba de la Figura 4.12: (a) Mapas 1-3: pared; (b) Mapas 4-6: pasillo; (c) Mapas 7-9: puerta; (d) todos.

Del análisis de los vectores de características de la Figura 4.13 se puede extraer mucha información. Sin embargo, para nosotros es mucho más interesante el estudio de la distancia de Tanimoto entre los vectores de características. Estas distancias se presentan en la Tabla 4.2. La distancia para la categoría pared (Mapas 1, 2 y 3 de la Figura 4.13) oscila entre 0.06 y 0.14. Es decir, tanto por inspección como en lo referente a la distancia de Tanimoto son muy parecidos todos ellos. Además, la distancia con respecto a los vectores de características de los mapas locales de prueba correspondientes a un pasillo es muy elevada, por encima de 0.55 excepto en un caso, que es de 0.41. Sin embargo, con respecto a los tres mapas locales de la puerta, la distancia es inferior. Esto es así porque, si observamos dichos mapas locales, nos damos cuenta

de que, en cierto sentido también parecen una pared. Para el Mapa 7 porque la pared viene marcada por el borde del mapa local y para los Mapas 8 y 9 porque hay una pared claramente definida. Este hecho es clave en nuestro sistema, puesto que será aprovechado para calcular los pesos asignados a cada comportamiento para hacer que todos ellos cooperen.

Mapa	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0.00	0.10	0.14	0.63	0.41	0.72	0.31	0.24	0.40
2	0.10	0.00	0.06	0.74	0.58	0.84	0.30	0.37	0.40
3	0.14	0.06	0.00	0.80	0.65	0.91	0.32	0.43	0.41
4	0.63	0.74	0 80	0.00	0.30	0.19	0.85	0.41	0.65
5	0.41	0.58	0.65	0.30	0.00	0.22	0.80	0.41	0.75
6	0.72	0.84	0.91	0.19	0.22	0.00	0 93	0.55	0.86
7	0.31	0.30	0.32	0.85	0.80	0.93	0.00	0.37	0.27
8	0.24	0.37	0.43	0.41	0.40	0.55	0.37	0.00	0.21
9	0.40	0.40	0.41	0.65	0.75	0.86	0.27	0.21	0.00

Tabla 4.2: Distancia de Tanimoto entre los vectores de características de los mapas locales de la Figura 4.12.

En el caso de los tres mapas locales que tienen aspecto de pasillo, los resultados indican que la distancia entre sus tres vectores de características varía entre 0.19 y 0.30. Aun así, el dato más significativo es que la distancia con respecto al resto de vectores de características es bastante más elevada, excepto en un par de casos. Analizando la Tabla 4.2 destaca que la distancia a los vectores de características de los Mapas 1, 2 y 3 es superior a 0.55 en todos los casos excepto uno. Igualmente, la distancia para los Mapas 7, 8 y 9 también supera el valor 0.55 salvo dos excepciones. Estas excepciones, probablemente, se deben a que el mapa involucrado también tiene un cierto aspecto de pasillo. Por último, los vectores de características de los Mapas 7, 8 y 9 de la Figura 4.12 presentan una distancia de Tanimoto entre ellos que varía entre 0.21 y 0.37. El mínimo valor se obtiene al comparar los Mapas 8 y 9. Este resultado era de esperar, pues cualitativamente se parecen bastante.

La comparación de un vector de características con los patrones del sistema mediante la distancia de Tanimoto permite obtener la información necesaria para caracterizar un lugar en pared, pasillo o puerta. Esta caracterización serviría para seleccionar uno de nuestros tres comportamientos reactivos de navegación, Seguir Pared, Seguir Pasillo o Cruzar Puerta, respectivamente. Para demostrar que el vector de características propuesto en la presente Tesis es útil para este propósito, se presenta un ejemplo con los mapas locales de prueba de la Figura 4.12. La Figura 4.14 presenta la comparación de los vectores de características de los mapas locales de la Figura 4.12 con los patrones. Para que la representación sea más clara se ha optado por representar en un mismo gráfico la comparación de un vector de características con únicamente los patrones de la categoría a la que se supone pertenece. Del análisis cualitativo se deduce que la suposición de la categoría a la que pertenece cada mapa local tiene la impresión de ser cierta, porque el parecido que se observa en la Figura 4.12 entre los patrones y los vectores de características de los mapas locales de prueba es muy alto.

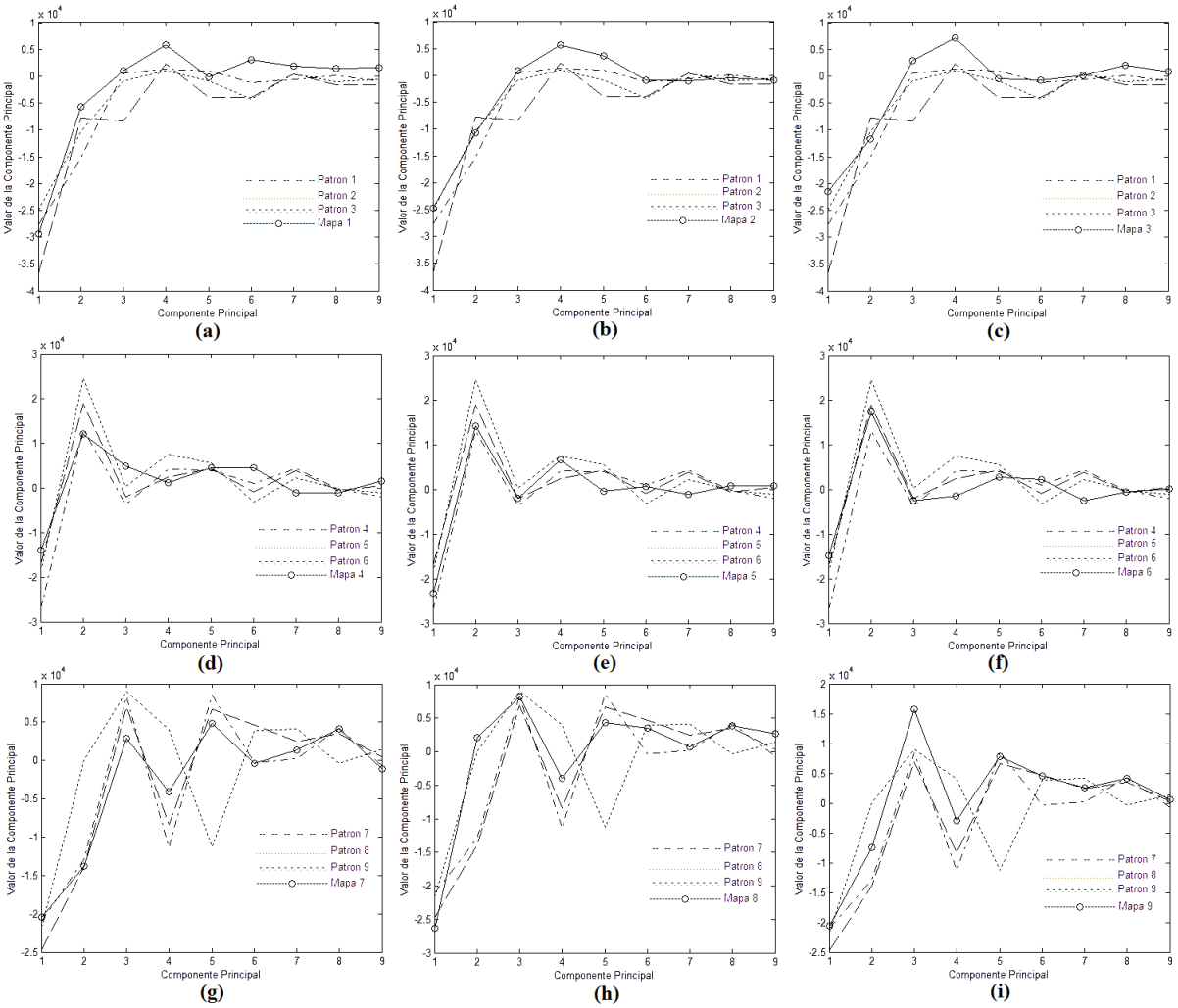

Figura 4.14: Comparación de los vectores de características de los mapas locales de prueba de la Figura 4.12 con los patrones: (a) Mapa 1; (b) Mapa 2; (c) Mapa 3; (d) Mapa 4; (e) Mapa 5; (f) Mapa 6; (g) Mapa 7; (h) Mapa 8; (i) Mapa 9.

Aunque el análisis cualitativo de las gráficas de la Figura 4.14 nos induce a caracterizar cada mapa local en una determinada categoría, esta decisión está basada en la distancia de Tanimoto entre los vectores de características de los mapas locales de prueba y los patrones. La Tabla 4.3 presenta los resultados de la distancia entre los vectores de características de entrada y los patrones. La distancia mínima de un vector de características particular a los patrones se muestra en negrita. Esta distancia mínima es la que permite caracterizar un mapa local dentro de una categoría. La categoría a la que pertenece es la correspondiente al patrón cuya distancia al vector de características es mínima. De este modo, los Mapas 1, 2 y 3 se clasifican como pared, los Mapas 4, 5 y 6 como pasillo, y los Mapas 7, 8 y 9 como puerta. Por lo tanto, el comportamiento seleccionado en cada caso sería, respectivamente, Seguir Pared, Seguir Pasillo y Cruzar Puerta.

Debemos hacer notar que el sistema clasifica correctamente todos los mapas locales de prue-

Mapa \ Patrón	1	2	3	4	5	6	7	8	9
1	0.18	0.15	**0.14**	0.67	0.71	0.36	0.32	0.37	0.47
2	0.24	0.07	**0.06**	0.80	0.83	0.52	0.30	0.47	0.40
3	0.32	**0.10**	0.11	0.86	0.89	0.62	0.37	0.40	0.48
4	0.78	0.79	0.82	**0.25**	0.36	0.36	0.79	0.61	0.82
5	0.50	0.60	0.68	0.16	0.19	**0.08**	0.79	0.48	0.85
6	0.75	0.84	0.90	**0.12**	0.24	0.28	0.90	0.74	0.92
7	0.42	0.38	0.33	0.92	0.97	0.71	**0.09**	0.13	0.11
8	0.45	0.39	0.40	0.54	0.61	0.39	**0.27**	0.37	0.33
9	0.65	0.48	0.40	0.84	0.87	0.68	**0.16**	0.39	0.19

Tabla 4.3: Distancia de Tanimoto de los vectores de características de los mapas locales de prueba de la Figura 4.12 a los patrones del sistema.

ba en la categoría que más se adapta a la situación que modelan. También hay que destacar que las distancias mínimas a los patrones varían en el rango de 0.06 a 0.27. Estas distancias son bajas, haciendo que el proceso de comparación sea bastante exacto. Además, no solamente la distancia mínima es baja. Por ejemplo, la distancia de Tanimoto entre los vectores de características correspondientes a un pasillo (Mapas 4, 5 y 6), y los patrones de un pasillo varía entre 0.08 y 0.36. Estos valores son muy bajos para los tres pasillos de referencia. Se ha verificado que esta situación se repite en el sistema para distintos mapas de prueba.

La técnica propuesta es capaz de caracterizar un lugar en la categoría pared, pasillo o puerta. Esta caracterización solamente permitiría que los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta compitieran. Por lo tanto, no podrían cooperar, aunque el objetivo es desarrollar un mecanismo de cooperación de los comportamientos de navegación reactiva que se adapte a las circunstancias y particularidades del entorno en todo momento. Sin embargo, el vector de características propuesto no solo es adecuado para seleccionar un comportamiento específico, sino también para fusionarlos. Así pues, aunque la selección de un único comportamiento en base a la caracterización de un lugar particular funciona, en la presente Tesis se ha optado por la cooperación de los tres comportamientos, tema que se estudia en las siguientes secciones.

2.5.2 Cálculo de los factores de ponderación

Sea $\overrightarrow{D} = [d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9]$ el vector de las distancias de un vector de características $\overrightarrow{VC}$ que representa un mapa local, a los 9 patrones del sistema. En el vector $\overrightarrow{D}$, los valores d_1, d_2 y d_3 son las distancias a los patrones de la categoría pared, d_4, d_5 y d_6 las distancias a los patrones de la categoría pasillo y d_7, d_8 y d_9 las distancias a los patrones de la categoría puerta. Sea también $\overrightarrow{W}_C$ el vector con los factores de ponderación a aplicar a los comandos de movimiento de los tres comportamientos tal que:

$$\overrightarrow{W}_C = [w_{pared}, w_{pasillo}, w_{puerta}] \tag{4.26}$$

donde w_{pared}, $w_{pasillo}$ y w_{puerta} son los pesos de los comandos de movimiento de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta, respectivamente, verificándose que $0 \leq w_{pared} \leq 1$, $0 \leq w_{pasillo} \leq 1$ y $0 \leq w_{puerta} \leq 1$. La suma de los tres pesos será:

$$w_{pared} + w_{pasillo} + w_{puerta} = 1.0 \tag{4.27}$$

para conseguir unos valores normalizados. Por ejemplo, si el vector con los factores de ponderación es $\overrightarrow{W}_C = [w_{pared}, w_{pasillo}, w_{puerta}] = [0.2, 0.7, 0.1]$, la contribución al comando de movimiento para el robot por parte de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta sería, respectivamente, del 20%, 70% y 10%.

El problema que se plantea es la obtención del vector con los factores de ponderación. Para solucionar este hecho se ha diseñado e implementado un mecanismo para calcular el vector $\overrightarrow{W}_C$ a partir del vector de distancias $\overrightarrow{D}$. Sean d^{min}_{pared}, $d^{min}_{pasillo}$ y d^{min}_{puerta} la distancia mínima a los patrones de las categorías pared, pasillo y puerta, respectivamente, tal que:

$$\begin{aligned} d^{min}_{pared} &= min\{d_1, d_2, d_3\} \\ d^{min}_{pasillo} &= min\{d_4, d_5, d_6\} \\ d^{min}_{puerta} &= min\{d_7, d_8, d_9\} \end{aligned} \tag{4.28}$$

siendo d^{min} el mínimo de estos tres valores, es decir, el mínimo del vector de distancias:

$$d^{min} = min\{d^{min}_{pared}, d^{min}_{pasillo}, d^{min}_{puerta}\} \tag{4.29}$$

Por otro lado, sean $\overline{d_{pared}}$, $\overline{d_{pasillo}}$ y $\overline{d_{puerta}}$ los valores medios de la distancia a los patrones de las categorías pared, pasillo y puerta, respectivamente, obtenidos mediante las siguientes expresiones:

$$\begin{aligned} \overline{d_{pared}} &= \frac{d_1 + d_2 + d_3}{3} \\ \overline{d_{pasillo}} &= \frac{d_4 + d_5 + d_6}{3} \\ \overline{d_{puerta}} &= \frac{d_7 + d_8 + d_9}{3} \end{aligned} \tag{4.30}$$

Una vez calculados todos los valores anteriores se decide el valor de los factores de ponderación correspondientes a cada comportamiento:

$$\overrightarrow{W}_C = [w_{C1}, w_{C2}, w_{C3}] = \begin{cases} [1.0, d^{min}_{pared}/\overline{d_{pasillo}}, d^{min}_{pared}/\overline{d_{puerta}}] & \text{si } d^{min} = d^{min}_{pared} \\ [d^{min}_{pasillo}/\overline{d_{pared}}, 1.0, d^{min}_{pasillo}/\overline{d_{puerta}}] & \text{si } d^{min} = d^{min}_{pasillo} \\ [d^{min}_{puerta}/\overline{d_{pared}}, d^{min}_{puerta}/\overline{d_{pasillo}}, 1.0] & \text{si } d^{min} = d^{min}_{puerta} \end{cases} \tag{4.31}$$

El comportamiento asociado a la categoría cuyo patrón es el más cercano al vector de características se ve favorecido en el sistema, haciendo que su contribución sea incluso mayor que para los otros dos comportamientos, al considerar la distancia mínima en vez de la distancia media. Se ha hecho de este modo para favorecer todavía más al comportamiento que mejor se adapta a la configuración del entorno en todo momento. Sin embargo, los valores obtenidos mediante la Ecuación (4.31) no están normalizados como se indica en la Ecuación (4.27). Si la suma de las componentes del vector $\overrightarrow{W}_C$ de la Ecuación (4.31) es:

$$\Sigma = \sum_{i=1}^{3} w_{Ci} \tag{4.32}$$

los factores de ponderación finalmente obtenidos por el sistema, y que son los que determinan la contribución de los comandos de movimiento de los tres comportamientos, vendrán dados por la expresión:

$$[w_{pared}, w_{pasillo}, w_{puerta}] = [\frac{w_{C1}}{\Sigma}, \frac{w_{C2}}{\Sigma}, \frac{w_{C3}}{\Sigma}] \tag{4.33}$$

3 Cooperación de comportamientos

La cooperación de los tres comportamientos implementados en el sistema, Seguir Pared, Seguir Pasillo y Cruzar Puerta, se alcanza mediante la combinación lineal de sus comandos de movimiento. Los factores que ponderan los comandos de movimiento serán w_{pared}, $w_{pasillo}$ y w_{puerta}, obtenidos mediante el proceso descrito en este capítulo. Recordando que v_{SPt} y v_{SPr} son las velocidades de traslación y de rotación del comportamiento Seguir Pared, que v_{SCt} y v_{SCr} son las velocidades de traslación y de rotación del comportamiento Seguir Pasillo, y que v_{CPt} y v_{CPr} son las velocidades de traslación y de rotación del comportamiento Cruzar Puerta, la combinación lineal de los tres comandos de movimiento da lugar a las velocidades de traslación y de rotación, v_t y v_r:

$$v_t = w_{pared} \cdot v_{SPt} + w_{pasillo} \cdot v_{SCt} + w_{puerta} \cdot v_{CPt} \tag{4.34}$$

$$v_r = w_{pared} \cdot v_{SPr} + w_{pasillo} \cdot v_{SCr} + w_{puerta} \cdot v_{CPr} \tag{4.35}$$

constituyendo el comando de movimiento final que incluye de forma cooperativa a los tres comportamientos del sistema.

La combinación lineal de los comandos de movimiento se realiza en el módulo *ControlRobot*. De este modo el robot consigue navegar por el entorno, adaptándose a su configuración y dirigiéndose a la siguiente zona de la ruta de exploración completa planificada por el módulo *PlanificadorRuta*. Sin embargo, existen situaciones donde la configuración del entorno no pertenece

claramente a ninguna de las tres categorías de nuestro sistema, pared, pasillo y puerta. En este caso, ninguno de los pesos, w_{pared}, $w_{pasillo}$ y w_{puerta} tiene un valor predominante sobre el resto. Cuando esto sucede, el sistema aplica otro comando de movimiento, determinando las velocidades de traslación y de rotación mediante un campo de potencial [Kha86], cuyo atractor es el siguiente punto de destino de la ruta de exploración completa y cuyo repulsor viene dado por la disposición de los obstáculos en el entorno. La estrategia seguida es similar a otras que nos encontramos en la literatura [ZGPP02, FBSC04, RKH04, LAP+04, SDC05], donde el punto de destino es tenido en cuenta en función del punto del entorno por donde navega el agente.

No se ha incluido en el sistema este comportamiento adicional de ir hacia un punto de destino siguiendo un campo de potencial porque no depende de la configuración del entorno y porque la estrategia seguida es similar, como se ha comentado, a la que se propone en los trabajos de otros autores. Con un atractor y un repulsor el comportamiento no puede adaptarse a la configuración del entorno, algo que sí hacen los tres comportamientos reactivos de nuestro sistema. Por otro lado, podría pensarse que únicamente con un comportamiento basado en los atractores y repulsores de Khatib [Kha86] es suficiente en el sistema. Sin embargo, esto no es cierto, puesto que la trayectoria del robot presentaría entonces oscilaciones y podría caer en mínimos locales de manera fácil. Estos problemas se evitan empleando la estrategia del *CBR* de implementación de comportamientos y con la utilización de varios comportamientos coordinados para adaptar la navegación a la configuración del entorno. La técnica de cooperación propuesta presenta otra ventaja importante, el hecho de que tiene potenciales posibilidades para el desarrollo de un método de localización global del robot.

El campo de potencial llevaría al robot hacia el punto de destino cuando los pesos de los comandos de movimiento de los tres comportamientos estén por debajo de un umbral U_{pesos}. Se ha comprobado experimentalmente que un umbral U_{pesos} en torno a 0.4 es un valor adecuado en el sistema.

4 Prestaciones temporales

Una vez concluido el proceso con el cálculo de los factores de ponderación según la expresión de la Ecuación (4.33), se analizan las prestaciones temporales del método que sustenta la fusión de los comandos de movimiento de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta. En la Tabla 4.4 se muestra el tiempo medio empleado para calcular el contorno de la *RLOI* de un mapa local, mientras que en la Tabla 4.5 se muestra el tiempo medio empleado por el algoritmo para obtener los factores de ponderación a partir de un mapa local. Las medidas se han realizado en un Pentium 3 a 550 MHz con 192 MB de memoria RAM. Se han considerado tres mapas locales (Figura 4.3), correspondientes a una pared (Figura 4.3a), un pasillo (Figura 4.3b) y una puerta (Figura 4.3c). Tal y como se ha comentado anteriormente, el tamaño del mapa de profundidad y del módulo de su FFT es $N = 64$, siendo la dimensión del vector de características $P = 9$. Los errores se han calculado sobre un intervalo de confianza del 95%.

Mapa Local	Figura 4.3a	Figura 4.3b	Figura 4.3c
Cálculo del contorno del mapa local			
1. Umbralización	13.120 ± 0.111 ms	12.938 ± 0.109 ms	13.400 ± 0.112 ms
2. Filtrado	6.284 ± 0.083 ms	6.145 ± 0.089 ms	6.724 ± 0.095 ms
3. Obtención de la $RLOI$	12.940 ± 0.130 ms	14.700 ± 0.117 ms	11.436 ± 0.102 ms
4. Obtención del contorno	8.530 ± 0.096 ms	6.020 ± 0.087 ms	9.997 ± 0.095 ms
Total Contorno	40.871 ± 0.172 ms	39.800 ± 0.145 ms	41.559 ± 0.141 ms

Tabla 4.4: Tiempo de cálculo del contorno del mapa local.

Mapa Local	Figura 4.3a	Figura 4.3b	Figura 4.3c
Total Contorno	40.871 ± 0.172 ms	39.800 ± 0.145 ms	41.559 ± 0.141 ms
Mapa de Profundidad	0.665 ± 0.024 ms	0.834 ± 0.036 ms	1.205 ± 0.039 ms
$\|FFT\|$ Mapa Profundidad	46.679 ± 7.986 μs	41.694 ± 0.194 μs	42.443 ± 0.920 μs
Vector de Características	36.204 ± 1.245 μs	41.308 ± 8.255 μs	34.817 ± 0.847 μs
Factores de Ponderación	18.683 ± 0.156 μs	23.443 ± 7.505 μs	19.975 ± 0.865 μs
Total	41.640 ± 0.170 ms	40.740 ± 0.145 ms	42.861 ± 0.141 ms

Tabla 4.5: Tiempo de cálculo de los factores de ponderación a aplicar para conseguir la cooperación de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta.

El primer dato interesante que se observa en las Tablas 4.4 y 4.5 es que el tiempo empleado durante el algoritmo es independiente de la configuración del entorno, es decir, no depende de la categoría final del mapa local, pared, pasillo o puerta. También hay que destacar que el cálculo del contorno de la $RLOI$ del mapa local es el proceso que más tiempo requiere, en torno a los $40ms$. Este resultado es esperable, en tanto en cuanto que durante la extracción del contorno se trabaja con un mapa local de 113x113 celdas. A partir de este punto, el algoritmo emplea un número más reducido de datos, por lo que para las restantes fases del algoritmo el tiempo de proceso es mucho menor.

Para el cálculo del contorno (Tabla 4.4), las tareas que más tiempo consumen son la *Umbralización* y la *Obtención de la RLOI*, aunque la etapa de *Filtrado* y de la obtención final del contorno también tardan un tiempo que no se puede despreciar. Como el contorno $\overrightarrow{C}$ tiene un número de puntos variable, el tiempo requerido para obtener el mapa de profundidad también es variable. Así, si recordamos de la Sección 2.1.4, los contornos de la $RLOI$ de los mapas locales de las Figuras 4.3a, b y c tienen, respectivamente, 338, 423 y 614 puntos. Por este motivo, en la Tabla 4.5 el tiempo para obtener el mapa de profundidad oscila entre $0.665ms$ para la Figura 4.3a y $1.205ms$ para la Figura 4.3c. Al finalizar la fase de *Cálculo del mapa de profundidad* el algoritmo trabaja con vectores que siempre tienen la misma dimensión. Por lo tanto, los tiempos de cálculo deben ser similares, tal y como se muestra en los resultados de la Tabla 4.5.

Las tres últimas etapas del algoritmo de caracterización de lugares implican un tiempo de proceso que se podría considerar despreciable. El cálculo del $|FFT|$ del mapa de profundidad incluye, en primer lugar, la obtención de la FFT y, posteriormente, de su módulo. Debido a que el algoritmo de la FFT es muy rápido y que los vectores involucrados tienen una dimensión

$N = 64$, el tiempo que se emplea es de tan solo unos $45\mu s$. Del mismo modo, para proyectar el $|FFT|$ sobre la base únicamente se necesitan unos $40\mu s$. Obviamente, este tiempo no considera la obtención de la base, puesto que este proceso se efectúa solamente una vez y *off line*. Finalmente, la etapa que da lugar a los factores de ponderación a aplicar para los tres comportamientos necesita unos $20\mu s$. Esta fase final considera tanto el cálculo de la distancia del vector de características a los patrones, como la obtención final del peso para el comando de movimiento de cada comportamiento.

Si se deseara aproximar el tiempo de proceso, con el correspondiente al cálculo del contorno sería más que suficiente, pues supone aproximadamente el 97% del total. Por lo tanto, el tiempo necesario para el resto de las etapas del algoritmo sería perfectamente despreciable. Si, por el contrario, se desea tener un cálculo más exacto, incluiríamos todas las fases del algoritmo. En cualquier caso, podemos concluir que, en una máquina de prestaciones limitadas, el tiempo necesario para obtener los factores de ponderación de los comandos de movimiento de nuestros tres comportamientos a partir de un mapa local en una determinada posición del entorno es solamente de unos $42ms$. Si el robot navega a una velocidad de $200mm/s$, el agente ni siquiera llega a recorrer $1cm$ antes de que se vuelvan a recalcular los factores de ponderación. Por lo tanto, al ser el método tan rápido, se aumenta la velocidad de reacción y de adaptación a la configuración del entorno, independientemente de la misma. De este modo, se consigue una cooperación más eficiente entre los comportamientos en todo momento.

5 Experimentos y resultados

El método desarrollado para obtener los factores que ponderan los comandos de movimiento de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta, a partir de un mapa local en una determinada posición del entorno, ha sido probado tanto en entornos simulados como reales. Las pruebas simuladas se han realizado con el simulador de la plataforma robótica *Nomad 200* de *Nomadics*. Las pruebas en entornos reales se han llevado a cabo empleando la plataforma robótica real *Pioneer P2AT*. En todos los tests un humano guía al robot por medio de un teclado a una velocidad máxima de traslación de $100mm/s$ y una velocidad máxima de rotación de $15^o/s$.

El objetivo de estos tests es analizar la contribución que el método asigna a cada comportamiento en cada punto del entorno por el que navega el robot. Para evaluar la técnica se han desarrollado dos representaciones gráficas, la *Representación de Selección* y la *Representación de Fusión*. En ambas se muestra el mapa métrico construido por el robot durante el guiado con la trayectoria recorrida por el agente superpuesta en color. El mapa métrico únicamente se emplea para mostrar gráficamente el modelo de entorno que el agente adquiere mientras es guiado. La diferencia entre ambas representaciones estriba en la interpretación de los colores de la trayectoria:

1. *Representación de Selección*, *RS*. Usa un color *RGB* puro, rojo, verde o azul. En cada

punto de la trayectoria se determina la categoría a la que pertenece el patrón más cercano al vector de características del mapa local, como si los comportamientos fueran competitivos. A la categoría pared se le asigna el color rojo, a la categoría pasillo el color verde, y a la categoría puerta el color azul.

2. *Representación de Fusión*, *RF*. Se emplea para mostrar la capacidad del método propuesto para hacer que los comportamientos cooperen y el sistema se adapte mejor a la configuración instantánea del entorno. En este caso no se usan colores puros, sino que se hace uso de los pesos w_{pared}, $w_{pasillo}$ y w_{puerta} para determinar el color a aplicar a cada punto de la trayectoria. A cada categoría se le asigna un canal de color *RGB*, rojo para pared, verde para pasillo y azul para puerta. El valor de cada uno de los canales *RGB* vendrá dado por la expresión:

$$[R, G, B] = [255 \cdot w_{pared}, 255 \cdot w_{pasillo}, 255 \cdot w_{puerta}] \tag{4.36}$$

La *RS* es similar en su concepción a los mapas de comportamiento de Dornhege [DK07] y a otras formas de presentar los resultados [PCJC06, MSB05], donde a cada punto de la trayectoria del robot se le asigna una clase. Sin embargo, la *RF* es novedosa en el sentido de que la propia representación muestra de un modo gráfico la fusión que se realizaría de los comportamientos, en vez de seleccionar una única categoría para cada punto de la trayectoria. El análisis de la *RS* muestra claramente el comportamiento que el sistema seleccionaría elitistamente en cada punto de la trayectoria, mientras que la *RF* nos permite, por simple inspección, tener una idea de la forma en la que cooperan los comportamientos. Como en la presente Tesis los comportamientos son cooperativos, la *RF* aporta más información que la *RS*. No obstante, para las pruebas que se describen a continuación se muestran ambas representaciones.

5.1 Entornos simulados

La Figura 4.15 presenta una primera prueba en un entorno simulado. Se trata de una habitación cuadrada con un obstáculo en el centro. El robot fue guiado partiendo desde el sur hacia el este del entorno, después hacia el norte y, finalmente, al oeste. La *RS* de la Figura 4.15b muestra la categoría seleccionada a partir del mapa local que el sistema posee en cada punto de la trayectoria. Se observa que cuando el agente se desplaza entre el obstáculo y una pared lateral del entorno, el sistema lo percibe como un pasillo. Del mismo modo, cuando el agente se aproxima a una pared frontal y, por lo tanto, deja atrás el obstáculo central, la categoría puerta es la seleccionada. Y cuando se está girando, se determina que el vector de características en la posición del robot pertenece a la categoría pared. Aunque esta información es muy útil para seleccionar un único comportamiento de los tres implementados en un sistema competitivo, ya se ha comentado anteriormente que en la presente Tesis nos interesa la fusión de comportamientos. Por ello, la *RF* es la que nos aporta la información necesaria. Se aprecia que la *RF* de la Figura 4.15c tiene la misma distribución de colores que la *RS* de la Figura 4.15b, aunque estos están

suavizados porque no son puros. En cada situación influyen los pesos calculados para los tres comportamientos, w_{pared}, $w_{pasillo}$ y w_{puerta}, participando las tres categorías con un determinado porcentaje en cada punto de la trayectoria. Se siguen detectando perfectamente las situaciones. Así, al estar entre el obstáculo central y una pared lateral la categoría pasillo es la que más influencia ejerce. Sin embargo, en cierto sentido también se tiene una pared y, de algún modo, puede ser visto como una especie de puerta. Esta situación se da, por ejemplo, cuando el agente se desplaza entre el obstáculo y la pared este. En ese caso, el método obtiene una contribución para el comportamiento Seguir Pasillo del 67%, mientras que para los comportamientos Seguir Pared y Cruzar Puerta es del 17% y 16%, respectivamente.

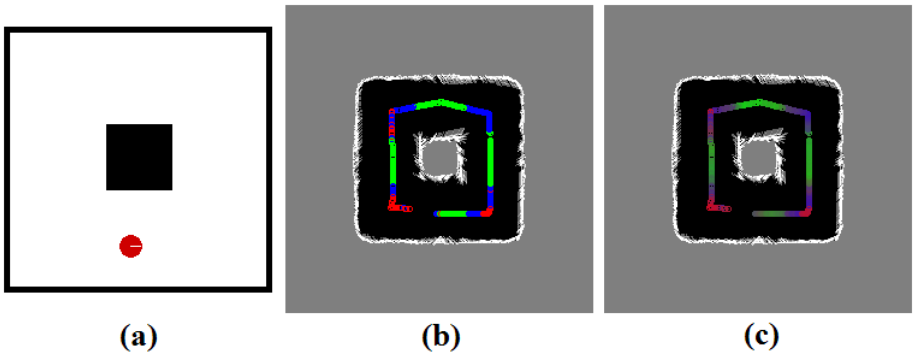

Figura 4.15: Entorno simulado 1: (a) planta; (b) *RS*; (c) *RF*.

Un segundo entorno de test simulado se presenta en la Figura 4.16a. El robot parte del centro del entorno hacia el este. Al llegar a la pared que se encuentra en esa dirección es guiado hacia el norte, a continuación hacia el oeste y, finalmente, hacia el sur. Las Figuras 4.16b y c muestran la *RS* y la *RF*, respectivamente. Aunque la configuración del entorno es diferente del de la Figura 4.15, las conclusiones que se pueden extraer son similares. Examinando en primer lugar la Figura 4.16b se deduce cuál sería el comportamiento competitivo seleccionado en cada punto de la trayectoria. Así, si el robot navega entre dos obstáculos, es como si estuviera en un pasillo. Esta situación sucede (color verde): i) al desplazarse paralelo a la pared central y detectar la pared a su izquierda; ii) cuando se desplaza por el este hacia el norte; iii) cuando navega por el norte hacia el oeste; y iv) al moverse hacia el sur y detectar la pared a su izquierda. En cuanto a la categoría puerta, sería la seleccionada durante el primer y el segundo giro de 90º. En el tercero no se seleccionaría esta categoría, sino la categoría pared, porque la situación es diferente. En ese caso, el giro desde el norte hacia el oeste, el robot no detecta obstáculo a su izquierda, sino espacio libre. Por lo tanto, es lógico que la categoría pared sea la seleccionada en esa situación. La *RF* de la Figura 4.16c sigue estas mismas directrices. No obstante, se puede hacer un análisis más profundo en relación a la fusión de los comportamientos. Gracias al vector de características propuesto y al mecanismo de cálculo de los factores de ponderación, se puede determinar la contribución que cada comportamiento tendría a la navegación en cada punto de la trayectoria. Cuando el robot se encuentra en el pasillo norte, la contribución de los comandos de movimiento de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta sería, respectivamente, del 11%, 77% y 12%. Por otro lado, al girar hacia el norte y hacia el

oeste el color dominante es el azul, por lo que la categoría principal es la puerta. En ambos casos la contribución del comportamiento Cruzar Puerta sería superior al 60%, mientras que el comportamiento Seguir Parte contribuiría con un 27.5% al comando de movimiento final. En esas dos situaciones el robot percibe el entorno como una puerta, aunque también tiene un cierto aspecto de pared, de ahí que w_{pared} sea bastante superior que $w_{pasillo}$.

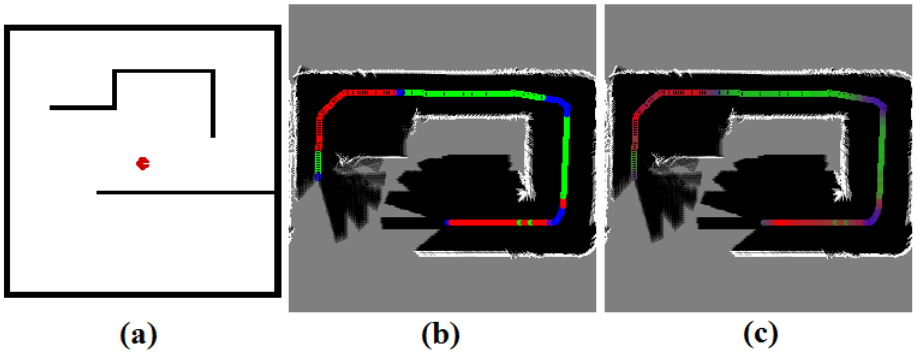

Figura 4.16: Entorno simulado 2: (a) planta; (b) *RS*; (c) *RF*.

En una tercera prueba se pretende demostrar que mapas locales capturados en puntos cercanos del entorno presentan vectores de características parecidos, es decir, la distancia de Tanimoto entre ellos es pequeña. Para analizar este hecho se presenta el entorno simulado de la Figura 4.17a. El operador humano guía al robot por el pasillo de oeste a este, evitando el obstáculo que aparece en el camino. Durante este guiado se extraen 64 vectores de características, correspondientes a 64 puntos de la trayectoria. Cada vector de características extraído se compara, mediante la distancia de Tanimoto, con el vector de características obtenido en la posición inmediatamente anterior, excepto para el primer vector, puesto que no hay valor anterior. Esta distancia es representada en la Figura 4.17b. De los 63 valores obtenidos, solamente 6 distancias son superiores a 0.05. De estos 6 valores, 2 son mayores que 0.10, y solamente uno es superior a 0.20 (0.2046). Incluso en este último caso, que es el peor, la distancia entre vectores de características consecutivos es muy pequeña. De hecho, al ser en la mayoría de los puntos inferior a 0.05, se puede decir, sin temor a equivocarnos, que dos posiciones espacialmente cercanas del entorno presentan vectores de características cercanos en cuanto a su distancia de Tanimoto. Este resultado era esperable, puesto que al estar cerca en el espacio, la configuración del entorno presente en el mapa local es similar. Por lo tanto, los vectores de características también tienen que ser parecidos. En cualquier entorno, tanto simulado como real, la cercanía espacial va a dar lugar a las mismas conclusiones.

Para el entorno de la Figura 4.17a también se presenta la *RS* (Figura 4.17c), y la *RF* (Figura 4.17d). Ambas representaciones son coherentes con la topología del entorno. El color principal a lo largo de la trayectoria es el verde, puesto que el robot se desplaza a lo largo de un pasillo. En los puntos donde la categoría principal es la de pasillo la contribución del comando de movimiento del comportamiento Seguir Pasillo nunca es inferior al 70%, de ahí que incluso en la *RF* el color verde siga siendo intenso. Al final del movimiento se detecta la pared frontal. Por

lo tanto, en la *RS* aparece una pequeña zona roja final, debida a que la categoría principal es la pared. Sin embargo, la contribución del comando de movimiento del comportamiento Seguir Pared no es tan elevada como en el caso anterior para el pasillo. De hecho, el valor de esta contribución es del 51%, mientras que para los comportamientos Seguir Pasillo y Cruzar Puerta sería del 22% y 27%, respectivamente. Es así porque la situación que el robot detecta no está claramente definida, con una especie de pasillo, y con obstáculos todavía no definidos al frente como si fuera una puerta.

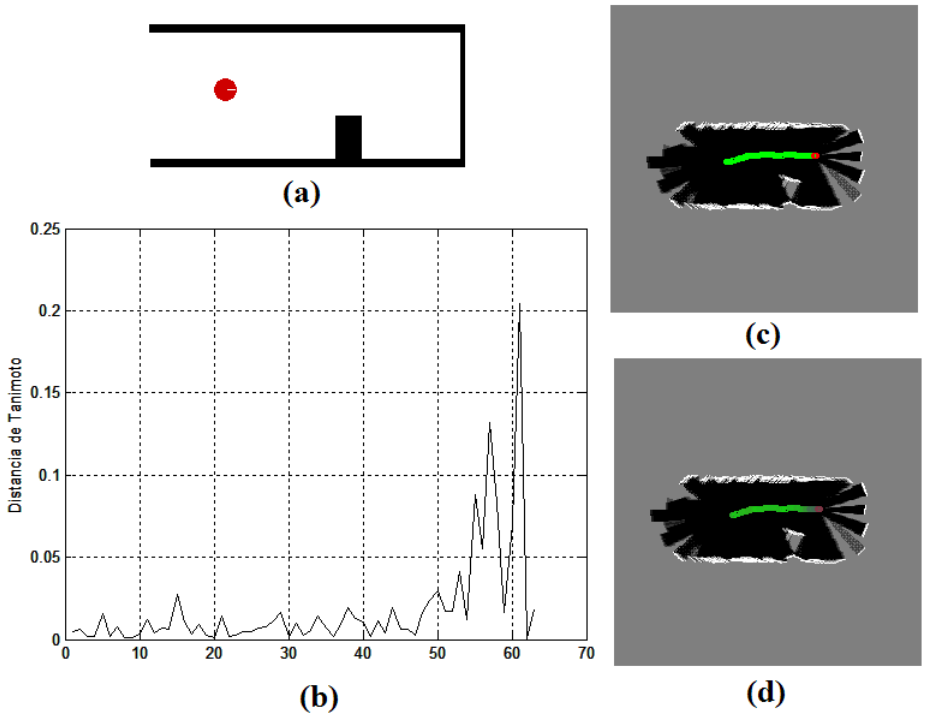

Figura 4.17: Entorno simulado 3: (a) planta; (b) distancia de Tanimoto entre vectores de características consecutivos de la trayectoria; (c) *RS*; (d) *RF*.

5.2 Entornos reales

Una vez descritas las pruebas en entornos simulados, pasamos a analizar los resultados obtenidos en entornos reales. El principal problema con el que nos encontramos a la hora de realizar las pruebas en escenarios reales es el de los errores de las lecturas sonar. El método de generación y actualización del mapa local, implementado en el módulo *MapaLocal*, intenta corregir estos errores mediante la integración temporal de las lecturas. Aun así, los mapas locales obtenidos en entornos reales no están tan definidos como aquéllos de los entornos simulados. Sin embargo, nuestra técnica puede tratar satisfactoriamente con este problema, como se demuestra a continuación con varios experimentos en entornos reales.

El robot fue guiado en primer lugar en escenarios simples. En las Figuras 4.18 y 4.19 se presentan dos de estos escenarios, junto con la *RS* y la *RF* resultantes tras el guiado. En el entorno de la Figura 4.18a el robot fue guiado de forma paralela a la pared en una habitación

de $4.8x4.4m^2$, mientras que en el entorno de la Figura 4.19a fue guiado por el centro a lo largo del pasillo de $6m$ de longitud y $1.8m$ ó $2.2m$ de anchura. Una característica que se aprecia tanto en la *RF* de la Figura 4.18c como en la *RF* de la Figura 4.19c, es la suavidad existente en los cambios de color de la trayectoria. En la práctica esto implicaría transiciones suaves, consiguiendo un movimiento más suave durante la navegación.

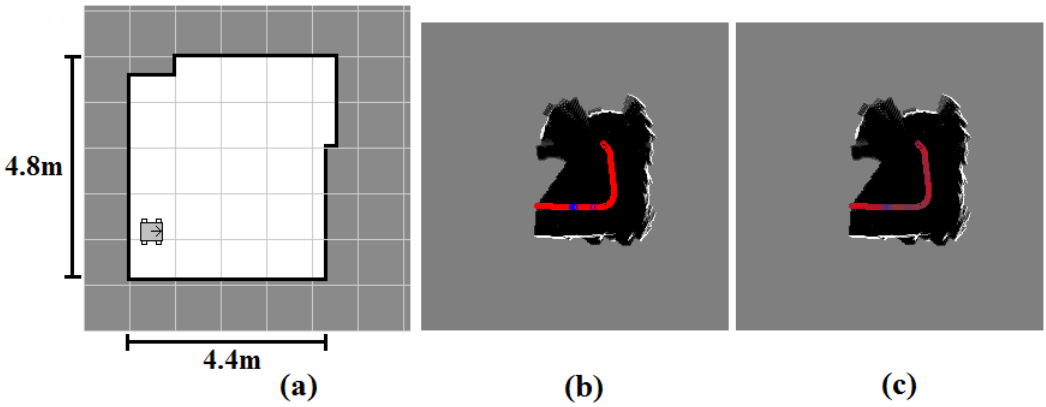

Figura 4.18: Entorno real 1: (a) planta; (b) *RS*; (c) *RF*.

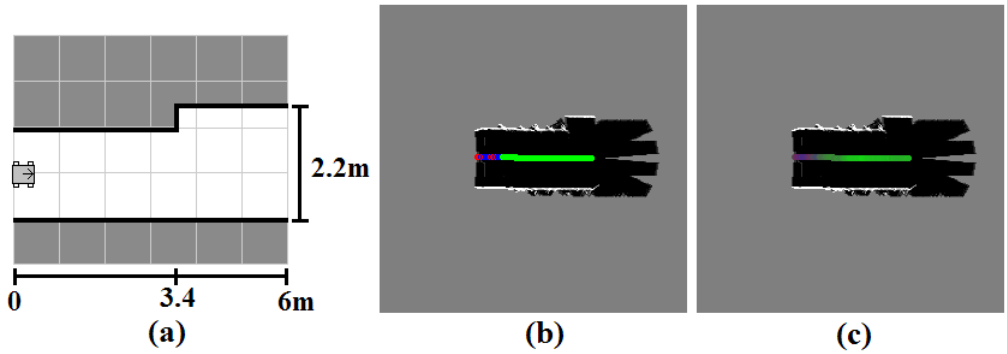

Figura 4.19: Entorno real 2: (a) planta; (b) *RS*; (c) *RF*.

En la *RS* de la Figura 4.18b se observa que la categoría seleccionada en la mayoría de las ocasiones es la de pared. Aparecen algunos puntos espúreos que no deberían correspondientes a la categoría puerta (color azul), debidos a los errores en las lecturas sonar. La contribución que el sistema asignaría al comando de movimiento del comportamiento Seguir Pared variaría entre un 63% y un 88%, salvo en los puntos indicados. Por otro lado, el peso del comportamiento Seguir Pasillo a lo largo de toda la trayectoria es siempre inferior al 15%. En cuanto al peso asignado al comando de movimiento del comportamiento Cruzar Puerta, éste no supera el 20%. Es así excepto en los puntos indicados, donde está en torno al 48%, si bien el peso de la categoría pared es, en este caso, del 35%.

En cuanto al entorno de la Figura 4.19a, las conclusiones son similares. Antes de analizarlas, debemos comentar un aspecto no introducido anteriormente. Como al comenzar la navegación el mapa local no está todavía conformado, al principio de las trayectorias el sistema no detecta

la situación correctamente. Esto ocurre tanto en los entornos reales, como los de las Figuras 4.19a y 4.18a, como simulados (ver Figuras 4.15 y 4.16). También debemos añadir que no es un problema del método, sino que simplemente al principio el mapa local no dispone de suficiente información. Recordemos que este mapa local, construido por el módulo *MapaLocal*, se va generando y actualizando progresivamente a medida que el robot se mueve por el entorno. Este efecto también se aprecia en la RS de la Figura 4.19b. Al iniciar el movimiento, el sistema clasificaría la posición del robot como si fuera una pared o una puerta, hasta que dispone de un mapa local totalmente conformado. Entonces, la categoría seleccionada sería la de pasillo (zona verde), siendo coherente con la configuración del entorno. La zona verde en la RF sigue teniendo un color todavía intenso, lo que significa que, incluso cuando los comportamientos cooperaran, el de Seguir Pasillo tendría una contribución significativamente superior al Seguir Pared y Cruzar Puerta. De hecho, el valor de esa contribución varía entre el 70% y el 88% a lo largo de la trayectoria, excepto en la parte inicial por el motivo mencionado.

Los resultados en los entornos simples de las Figuras 4.18a y 4.19b demuestran que los factores de ponderación a aplicar a los comandos de movimiento de nuestros tres comportamientos que el método obtiene son coherentes con el entorno por el que el robot se mueve. No obstante, para poder asegurar la validez de la técnica se han realizado otros experimentos, esta vez en entornos más complejos.

En la Figura 4.20a se muestra un escenario consistente en una habitación de $5.5x4.8m^2$ donde el robot es guiado, en primer lugar, de forma paralela a una pared. Posteriormente recorre un pasillo hasta que detecta una puerta y es orientado como si fuera a atravesarla. En las Figuras 4.20b y c se presentan la RS y la RF obtenidas durante el guiado. En la RS inicialmente se selecciona la categoría pared, puesto que esa es la configuración detectada en el mapa local. Cuando aparece la pared a la izquierda del robot, éste lo percibe como si hubiera una puerta. Cuando la pared ya está conformada, es como si el agente navegara a lo largo de un pasillo (zona verde al este del mapa métrico). Cuando esta pared izquierda se deja atrás, la situación es nuevamente la de una pared, para pasar después a comenzar a detectar una pared frontal. En este momento, otra vez, se percibe el entorno como una puerta, seguida de una pared. Al seguir avanzando, el agente recorre el pasillo norte, siendo la categoría pasillo la seleccionada por la técnica. Finalmente, se detecta la puerta de $1.2m$. Salvo algunas situaciones espúreas que aparecen porque las lecturas de los sensores sonar presentan errores, los resultados casan adecuadamente con la configuración del entorno que el robot se encuentra en su entorno más inmediato a lo largo de toda la trayectoria.

La distribución de colores de la RF de la Figura 4.20c es similar a la de la RS de la Figura 4.20b, con la diferencia de que ya no son puros, sino que se ven suavizados para reflejar la contribución de las tres categorías, pared, pasillo y puerta. Si examinamos la contribución de cada categoría en el pasillo norte, se obtiene un peso para el comportamiento Seguir Pasillo entre el 55% y el 72%. En ese mismo área, el comando de movimiento del comportamiento Seguir Pared tendría un peso del 13% al 25%. Por otro lado, cerca de la puerta se aprecia una zona azul, donde la contribución del comportamiento Cruzar Puerta sería del 50%, mientras que para

las categorías pared y pasillo sería, respectivamente del 30% y del 20%. En ese último caso es así porque al detectar la puerta, la configuración presente en el mapa local también tiene un aspecto de pared y de pasillo que no es despreciable. Finalmente, cuando se está siguiendo la pared al principio del guiado, la contribución de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta sería, respectivamente, del 65%, 10% y 25%. Aunque todos los valores obtenidos demuestran la efectividad de nuestra aproximación para conseguir los factores de ponderación, debemos hacer notar que los resultados no son tan claros como en el caso de los entornos simples de las Figuras 4.18a y 4.19a. Sin embargo, esta situación es razonable, ya que el entorno de la Figura 4.20a es más complicado. Por lo tanto, los mapas locales en cualquier punto de la trayectoria difícilmente pertenecerán de forma pura a una de las tres categorías. Por otro lado, al tratarse de un entorno real, los colores están más mezclados que en las pruebas simuladas debido al ruido que introduce el mundo real en la lectura de los sensores sonar.

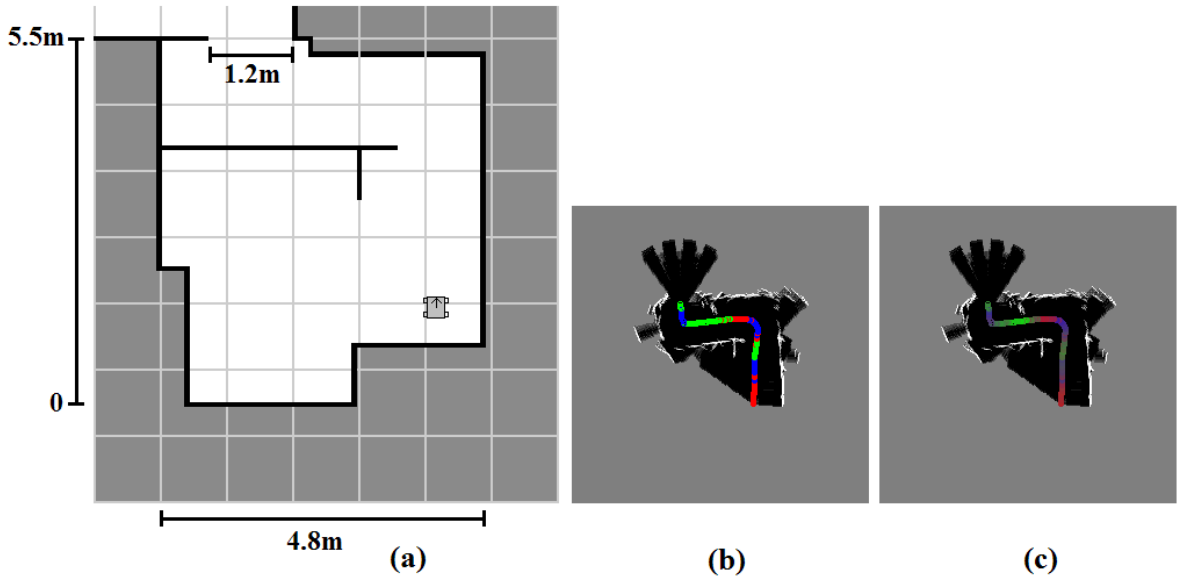

Figura 4.20: Entorno real 3: (a) planta; (b) *RS*; (c) *RF*.

El último entorno real de prueba que se presenta es el de la Figura 4.21a. Consiste en un pasillo y una habitación con un obstáculo pequeño de $60x60cm^2$ en su centro. En la *RS* de la Figura 4.21b se observa que inicialmente se seleccionaría el comportamiento Seguir Pared y después el Cruzar Puerta. Esto es así porque el mapa local todavía no está conformado. Cuando esto ocurre, se detecta el pasillo (zona verde). Al seguir avanzando, el robot gira hacia su derecha para seguir la pared, por lo que la categoría seleccionada pasa a ser la de pared. Después de esto, el robot entra en la habitación. En este momento percibe el entorno como un pasillo, porque el mapa local presenta una pared a ambos lados del agente. Finalmente, cuando el obstáculo central aparece, el sistema lo detecta como una puerta. Debemos hacer notar que este último caso es distinto de la situación que se presentaba en el entorno simulado de la Figura 4.15a. Allí, cuando el robot navegaba entre el obstáculo y la pared lateral se seleccionaba la categoría pasillo. Sin embargo, el obstáculo era bastante más grande que el de la Figura 4.21a. Tanto es así, que no es suficiente para considerar que el robot se encuentra en un pasillo.

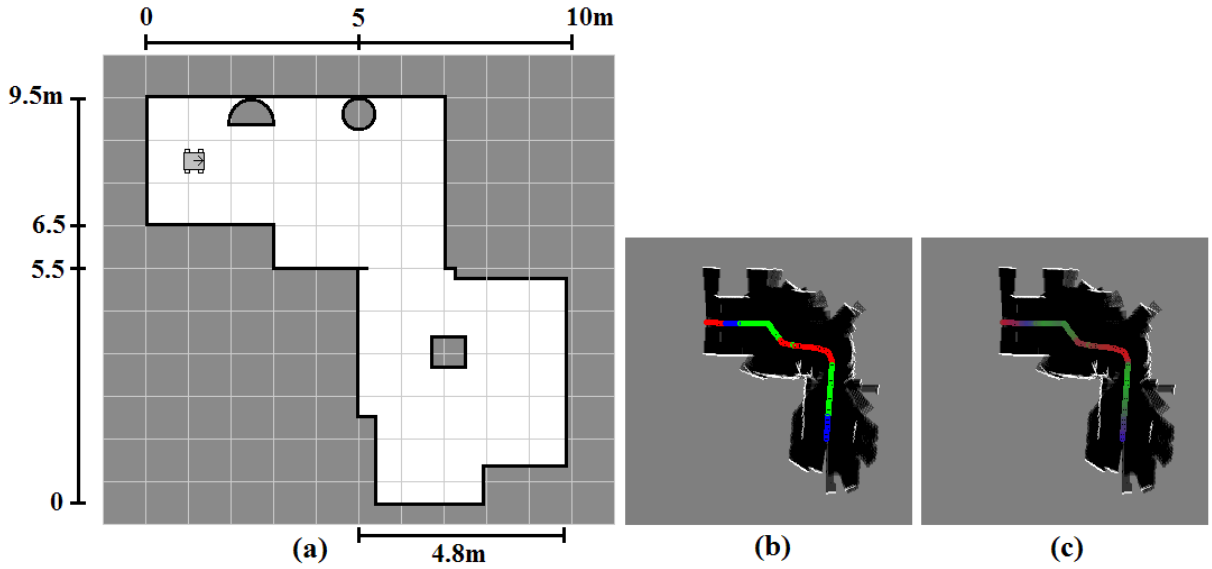

Figura 4.21: Entorno real 4: (a) planta; (b) *RS*; (c) *RF*.

En cuanto a la *RF* de la Figura 4.21b, presenta la misma distribución de colores en cada punto (y por lo tanto de la categoría principal) que la *RS* de la Figura 4.21c. Obviamente, los colores están suavizados porque en cada punto contribuyen los pesos calculados para los comandos de movimiento asociados a las tres categorías. Así, cuando el robot se desplaza por el pasillo norte y gira hacia la derecha para seguir la pared, el porcentaje con el que contribuiría el comportamiento Seguir Pared variaría entre el 55% y el 78%. Del mismo modo, cuando el obstáculo en el centro de la habitación aparece, el robot piensa que se está acercando a una puerta. De hecho, esta situación daría lugar a una contribución del comportamiento Cruzar Puerta entre el 45% al principio de la zona azul y el 65% al final de esa misma zona.

Las pruebas reales analizadas nos permiten extraer las mismas conclusiones presentadas para los entornos simulados. Como los mapas locales de partida no están tan bien definidos por el hecho de que las lecturas sonar poseen errores, quizás se puede decir que la caracterización que se presenta en las *RS* no es tan nítida como en el caso simulado, aunque exhibe las mismas características. Cuando se considera que los comportamientos cooperan en todos los puntos de la trayectoria del agente, los resultados simulados y reales son cualitativamente similares.

6 Conclusiones

En este capítulo se ha presentado un algoritmo de caracterización de lugares que permite el cálculo de los factores de ponderación de tres comportamientos de navegación reactiva (Seguir Pared, Seguir Pasillo y Cruzar Puerta), que en la literatura se consideran suficientes para navegar en interiores, para que cooperen en todo momento en la exploración completa de un entorno total o parcialmente desconocido.

La técnica propuesta reduce un mapa local inicial, construido a partir de la lectura de

sensores sonar, a un vector de características eficiente de únicamente 9 componentes que explica el 99.5% de la información. La reducción del tamaño del problema se realiza mediante un algoritmo diseñado para la extracción del vector de características a partir del mapa local. El algoritmo calcula en primer lugar el contorno de la zona libre de obstáculos presente en el mapa local en las proximidades del robot. El contorno es representado mediante su mapa de profundidad y éste, a su vez, es transformado mediante la aplicación de la FFT. Como nos interesa que dos mapas locales con diferente orientación produzcan el mismo resultado, se trabaja con el $|FFT|$. Para obtener el vector de características se proyecta sobre una base del sistema, obtenida mediante el Análisis de Componentes Principales.

La caracterización de un lugar se lleva a cabo por comparación de su vector de características con unos vectores de características definidos y conocidos que conforman los patrones del sistema. Cada patrón está asociado con una categoría, pared, pasillo o puerta. Una de las ventajas del método es que tanto el cálculo de la base como la obtención de los patrones son dos procesos independientes entre sí que se realizan una vez y *off line*, al margen de la operación del robot. Tanto la base como los patrones se obtienen a partir de mapas capturados en entornos simulados, siendo su uso satisfactorio en otros entornos distintos, tanto reales como simulados. Todos estos hechos dotan de una gran flexibilidad y generalidad a la técnica propuesta. Además, la técnica es lo suficientemente rápida para trabajar *on line*, tardando tan solo unos $40ms$ en una máquina de prestaciones bajas, no dependiendo de la configuración del mapa local. Este tiempo supone una ventaja para el sistema, pues permite recalcular los factores de ponderación para los comandos de movimiento de los tres comportamientos antes de que el agente haya recorrido mucha distancia, aumentando la velocidad de reacción y de adaptación al entorno.

Aunque se han extraído conclusiones cualitativas en relación al aspecto de los vectores de características de una pared, un pasillo y una puerta, para poder caracterizar un lugar de forma eficiente y automática se hace uso de la distancia de Tanimoto como métrica. La distancia de un vector de características a los patrones es la información útil para el sistema. Su utilidad es doble. Permite caracterizar cualquier lugar en la categoría pared, pasillo y puerta. Si bien el método propuesto es válido para conseguir que los comportamientos compitan en la navegación reactiva, no es el propósito de la presente Tesis, pues van a cooperar para adaptarse mejor a la situación del entorno en todo momento. El método, en segundo lugar, es capaz de obtener los factores de ponderación a aplicar a los comandos de movimiento de los tres comportamientos del sistema para que cooperen de forma eficiente, en base a un mecanismo diseñado para su cálculo a partir de las medidas de distancia a los patrones. El método propuesto ha sido probado tanto en entornos simulados como reales. En ambos casos ha demostrado su validez, pues los resultados son coherentes con la configuración del entorno de prueba. Para la presentación de estos resultados se han desarrollado dos representaciones gráficas, la RS y la novedosa RF, donde se observa que las trayectorias recorridas por el robot no presentan transiciones bruscas de color, sino que se consigue una evolución suave.

Por último, decir que aunque el vector de características es útil para caracterizar lugares, también tiene unas propiedades que lo dotan de una gran potencialidad para realizar tareas de lo-

calización global. Estas tareas requerirían del diseño y desarrollo de mecanismos de segmentación y del proceso de localización. Sin embargo, una gran parte del trabajo, la representación eficiente de un lugar, ya estaría diseñada gracias al vector de características propuesto.

Capítulo 5

Exploración Completa de Entornos

Este capítulo presenta un algoritmo eficiente para el cálculo de una ruta que permita la exploración completa de todas las zonas no exploradas de un entorno total o parcialmente desconocido, es decir, la planificación deliberada de caminos en el nivel más alto de la Jerarquía de Navegación. Este problema entraña una doble dificultad interrelacionada, la representación a nivel topológico del terreno desconocido y el hecho de que la planificación de una ruta de este tipo es un problema NP-completo. El algoritmo que se propone se implementa en el módulo *PlanificadorRuta* de la capa *Planificación*. La navegación entre las distintas zonas no exploradas se realizaría, en el orden calculado por el algoritmo, mediante los comportamientos reactivos del Capítulo 3, adaptándose así a las características desconocidas del entorno que está en proceso de exploración.

El capítulo comienza con una introducción al problema de la exploración completa en la Sección 1. En la Sección 2 se presenta la estructura jerárquica de integración métrico-topológica que sustenta el algoritmo. En la Sección 3 se muestra el algoritmo que calcula de forma eficiente la ruta para visitar todas las zonas inexploradas. La Sección 4 presenta los resultados obtenidos con la técnica propuesta. Finalmente, la Sección 5 presenta las conclusiones extraídas.

1 El problema de la exploración completa

La exploración de un entorno total o parcialmente desconocido es vital en la Robótica moderna en campos como la robótica de rescate, la robótica aplicada a la vigilancia, la exploración espacial o en entornos de riesgo como volcanes. Su objetivo es adquirir el máximo grado de conocimiento del espacio por el que el robot se mueve para que éste pueda realizar diversas tareas, como planificación de caminos o localización. Es habitual el buscar un conocimiento de todo el entorno, lo que se conoce como exploración completa. Consiste en hacer que el robot explore de forma íntegra todo el entorno, bien sea para la adquisición de un modelo completo de éste (exploración) o para recorrerlo de forma sistemática (vigilancia). La exploración se debe realizar de manera eficiente, siendo deseable no visitar dos veces una misma zona inexplorada, así como que el algoritmo empleado sea lo más rápido posible.

En esta Tesis se da respuesta al problema de la exploración completa. Para ello se propone un método para el cálculo eficiente de rutas con el objetivo de visitar todas las zonas no exploradas de un entorno total o parcialmente desconocido. La técnica está soportada por una estructura jerárquica que explícitamente representa las regiones no exploradas. Esta estructura integra los paradigmas métrico y topológico de modelado de entornos. Gracias a ella se pueden planificar rutas a través de zonas no exploradas de modo deliberado. El robot seguiría la ruta de un modo reactivo mediante los comportamientos del Capítulo 3.

La técnica propuesta se encuadra dentro del nivel de Navegación por Inspección, el superior de la Jerarquía de Navegación [FM00], puesto que va a permitir planificar rutas a través de zonas no exploradas. El método que se propone en esta Tesis queda enmarcado dentro de este nivel porque el mapa métrico-topológico jerárquico empleado representa las zonas inexploradas, así como las relaciones geométricas entre ellas. Además, el mapa métrico-topológico se puede construir *on line*. Por lo tanto, este mapa se puede actualizar fácilmente mientras el robot navega de forma reactiva entre las regiones inexploradas.

Obviamente, la técnica también soporta los dos niveles inferiores del grupo de comportamientos de Búsqueda de Caminos, la Respuesta Disparada por Reconocimiento y la Navegación Topológica, puesto que en la Jerarquía de Navegación un determinado nivel dentro de un grupo posee todas las capacidades de los niveles inferiores dentro de ese grupo. Así mismo, conviene recordar que la Navegación por Inspección, como todos los niveles dentro del grupo de comportamientos de Búsqueda de Caminos, necesita de los comportamientos de Navegación Local para desplazarse de un lugar a otro. Esto permite al agente encontrar lugares que no se podrían hallar empleando únicamente los Comportamientos de Navegación Local.

El problema de la exploración completa también se conoce como cobertura completa. Las aplicaciones de la cobertura completa son muy variadas, incluyendo aplicaciones domésticas destinadas a limpiar, barrer o aspirar [NVVR97], de detección de minas [AZC+01], de inspección [GB05], para robots exploradores [LV05], e incluso para navegación de plataformas móviles [GB06, CP97], como en la presente Tesis.

En general, los métodos para explorar de forma completa un entorno se pueden clasificar en dos grupos, analizados en los dos siguientes apartados:

- Aproximaciones que tienen un conocimiento *a priori* del entorno. Se basan en el cálculo de la ruta a seguir *off line*. Por lo tanto, la ruta puede ser estimada de forma óptima, resolviendo de forma eficiente el tema de la cobertura completa.

- Aproximaciones en las que no hay un conocimiento *a priori* del entorno. Consisten en planificar rutas en tiempo real, permitiendo la respuesta del sistema ante entornos dinámicos en el que pueden aparecer obstáculos móviles. Al no disponer de información sobre el entorno es posible que la ruta no sea tan óptima como en el caso anterior.

1.1 Con conocimiento *a priori* del entorno

Existen varios métodos para resolver el problema de la cobertura completa cuando el entorno en conocido *a priori*. Algunas estrategias se basan en programar la trayectoria que el robot debe seguir de forma autónoma [CB94]. No es necesario, por tanto, el empleo de un algoritmo para generar un camino. Su principal inconveniente estriba en la imposibilidad del tratamiento de entornos dinámicos con obstáculos móviles.

Zelinsky sugiere un método basado en una extensión de la transformada de la distancia para realizar una planificación de los caminos a seguir [ZJ93]. Para ello propaga una onda de distancia a través de todas las celdas de espacio libre de un mapa métrico. Para conseguir el camino de cobertura completa el robot sigue, desde el punto de partida, el camino de gradiente descendente. Sin embargo, existen algunas desventajas inherentes al método: i) no se garantiza la optimalidad del camino; ii) el camino resultante presenta muchos giros, por lo que es difícil de seguir; y iii) no es eficiente en el tratamiento de los obstáculos móviles, pues el robot se pararía hasta que desaparecieran.

En [CP97] se propone una aproximación basada en la descomposición en celdas. Consiste en descomponer el área a cubrir en regiones libres de obstáculos. El sistema autónomo lleva a cabo una búsqueda en estas regiones usando una secuencia de movimientos calculados para unir las regiones de forma óptima.

Hofner y Schmidt [HS95] presentan un método basado en plantillas que tiene en cuenta un mapa de dos dimensiones conocido con antelación. En base a este mapa se genera una secuencia de trayectorias predefinidas que hay que ejecutar, consiguiendo, tal y como los autores indican, la cobertura de la mayoría del entorno. En este mismo orden de cosas, en [NVVR97] también se presenta una técnica basada en el cálculo de una secuencia de trayectorias predefinidas para dar solución al problema de la cobertura completa. La aportación más significativa respecto del método de Hofner y Schmidt es la posibilidad de tratar los obstáculos dinámicos no representados en el mapa. Sin embargo, presenta algunos inconvenientes: i) el robot se puede quedar atrapado dentro de un obstáculo sin poder salir; y ii) el área cubierta presenta un alto grado de redundancia.

Las redes neuronales también han sido empleadas para dar solución al problema que nos ocupa. Así, en [GB06] se presenta un método basado en las redes neuronales para la planificación de caminos de cobertura completa en un entorno. Una vez que el camino de exploración completa ha sido obtenido, la técnica calcula trayectorias suaves mediante polinomios parametrizados, aunque todavía no ha sido probada en robots reales.

1.2 Sin conocimiento *a priori* del entorno

Una primera solución para la exploración completa de un entorno no conocido *a priori* se presenta en [VR95]. Este método emplea las lecturas de sensores sonar e infrarrojos, aunque tiene bastantes limitaciones, dado el carácter preliminar del mismo: i) el entorno debe tener

paredes rectas y esquinas cuadradas; ii) solamente puede haber un obstáculo presente, y además estacionario; y iii) el obstáculo debe ser lo suficientemente grande para poder ser detectado.

Al igual que cuando hay un conocimiento *a priori* del entorno también hay métodos basados en la transformada de la distancia [OPC01], la descomposición en celdas [AC01] y las redes neuronales [YL04, QLY04]. En el caso del método de Oh [OPC01], el algoritmo se desarrolla en tres fases. En una primera el robot navega siguiendo la pared para obtener el contorno del entorno. En la segunda, se realiza una cobertura completa del entorno por medio de algunas plantillas de comandos de movimiento muy simples. Finalmente, se efectúa una búsqueda para recorrer las regiones no cubiertas. Acar y Choset [AC01] sugieren una aproximación basada en la descomposición en celdas. La descomposición en celdas del entorno se construye de forma incremental a partir de la lectura de sensores sonar. Esta aproximación también incluye mecanismos de recuperación frente a medidas erróneas.

En cuanto a los métodos basados en redes neuronales [YL04, QLY04], en ambos casos se trata de redes neuronales inspiradas en la biología. Ambos métodos se basan en guiar el robot a través de una serie de objetivos locales, moviéndose a lo largo de caminos óptimos locales que permitan evitar los obstáculos. Aunque se ha demostrado su validez, únicamente se presentan resultados simulados.

2 Construcción del mapa topológico

La exploración completa de un entorno necesita de una representación del mismo que permita mantener un conocimiento del espacio por el que se mueve el robot. Las dos aproximaciones más empleadas a la hora de representar y modelar un entorno son los paradigmas métrico y topológico [Thr98].

Según se comentó en el Capítulo 2, el paradigma métrico obtiene una representación del entorno en la que el espacio se modela de acuerdo a relaciones geométricas absolutas [Elf87, Mor88]. Los mapas métricos representan el entorno de forma fiable, siendo fáciles de construir, representar y mantener. Así mismo, el reconocimiento de lugares no es ambiguo ni dependiente del punto de vista, ya que está basado únicamente en la información geométrica. Sin embargo, la planificación es ineficiente, tanto en el tiempo de proceso como en el espacio requerido para el almacenamiento de la información geométrica, ya que la resolución no depende de la complejidad del entorno. Además, debido a los errores sensoriales, habitualmente es difícil conseguir una representación exacta en entornos grandes.

El paradigma topológico, por otro lado, modela el entorno mediante un grafo o representación simbólica, el cual consta de nodos y arcos de conexión entre ellos [KB91]. Los nodos representan las distintas zonas o regiones en las que se divide el entorno. Los arcos de conexión indican las relaciones espaciales entre dichas regiones. Los mapas topológicos presentan un volumen de información mucho menor que los mapas métricos. Por lo tanto, el procesado de alto nivel es más óptimo en tiempo y en recursos. Debido a que no existe representación física exacta

del entorno, la representación topológica es mucho menos vulnerable a errores de medida. Sin embargo, se ha demostrado que pueden dar lugar a caminos subóptimos [TBB+98]. Su principal desventaja es la obtención, a partir de la información sensorial, del conjunto de nodos y arcos que conforman el modelo del entorno. También pueden tener dificultades para distinguir adecuadamente lugares distintos dentro del entorno.

La estructura jerárquica empleada en esta Tesis integra los paradigmas métrico y topológico para modelar entornos. Dicha estructura es una evolución de la que el autor presenta en [PPU+02a], la cual consistía en un polígono foveal [SB90], basada en algoritmos de visión artificial. A partir de un mapa métrico se construye un mapa topológico sobre él, dividiendo el entorno en regiones. Aunque se demostró que la estructura propuesta en [PPU+02a] es válida para representar el entorno y planificar rutas de exploración completa, presenta algunos problemas:

- La generación del mapa topológico se realiza de forma supervisada. Al tener que escoger un nivel de trabajo dentro de la estructura, el número de nodos del mapa topológico está definido de antemano. Por lo tanto, el mapa topológico generado no depende de la complejidad ni de la configuración del entorno.
- La conectividad entre las regiones no está garantizada. Para intentar forzar esta conectividad la estructura jerárquica necesita de una etapa de reenlazado de celdas aisladas. Esta etapa puede llegar a ser excesivamente costosa en tiempo. Además, no se asegura que no existan regiones no conectadas.

La estructura que se emplea en esta Tesis se presenta en [PPB+02]. Intenta solventar los problemas conocidos de la primera aproximación jerárquica [PPU+02a], así como mejorar el método de generación, almacenamiento de la información y selección de nodos del mapa topológico. Para ello:

- Usa un único nivel de resolución, simplificando el algoritmo de generación de la estructura.
- No se incluyen en el mapa topológico los nodos representados como obstáculos (espacio ocupado), puesto que el robot jamás podrá navegar por zonas ocupadas.
- Define el conjunto de nodos del mapa topológico de forma no supervisada a lo largo de distintos niveles de la estructura. Por lo tanto, el número de regiones en las que se divide el entorno viene determinado por su complejidad.

Antes de pasar a detallar los aspectos formales de la técnica propuesta para la construcción de un mapa topológico, se presentan a continuación distintos métodos de modelado del entorno basados tanto en el paradigma topológico como en la integración de los paradigmas métrico y topológico. No se incluye un análisis del paradigma métrico, puesto que éste ya se llevó a cabo al presentar la capa *ModeladoEntorno* en el Capítulo 2.

2.1 Estado del arte

2.1.1 Paradigma topológico

Tal y como se ha comentado anteriormente, los mapas topológicos intentan dar respuesta a los inconvenientes que presentan los mapas métricos. Para ello, emplean una representación simbólica que consta de un grafo con nodos y arcos de conexión entre los mismos. Existen varios métodos para la obtención de este grafo. Los primeros métodos sugerían modelar los objetos por medio de poliedros. Estos poliedros eran usados para dividir el espacio en un número limitado de regiones [CL85]. Esquemas más recientes insertan nodos en un grafo cada vez que el robot se mueve una distancia prefijada [DN99], o cada vez que se detecta un nuevo patrón sensorial [KB91, Mat91]. Dos nodos estarán conectados cuando sea posible encontrar un camino libre de obstáculos entre ellos. Si se insertan esos dos nodos de forma consecutiva en el grafo, estarán implícitamente conectados por un arco.

Otros métodos para la construcción de un mapa topológico se basan en modelos de Markov ocultos (*Hidden Markov Models (HMM)*) [SK97], en el escalado multidimensional (*Multi-Dimensional Scaling (MDS)*) [YHH01], o en procesos de Markov parcialmente observables (*Partially Observable Markov Decision Process (POMDP)*) [YTH02]. En el primero de estos tres métodos, Shatkay y Kaelbling proponen un mapa topológico basado en modelos ocultos de Markov donde los estados están asociados con determinados puntos de una representación geométrica del entorno. Sin embargo, este método no es válido para trabajar *on line*, puesto que el algoritmo para el cálculo del mapa topológico es preliminar. Por otro lado, Yairi emplea dos formas para obtener un modelo topológico del entorno. En [YHH01] se presenta una técnica heurística basada en el análisis multivariante de la distancia empírica calculada entre dos objetos arbitrarios. Aun así, únicamente se presentan resultados simulados, que requieren de algún conocimiento *a priori* del entorno. En el caso de los mapas basados en *POMDP* [YTH02], la técnica realiza una serie de observaciones y movimientos por parte del robot, almacenando información sobre ellos. La información almacenada incluye el instante temporal en el que las observaciones fueron capturadas. A partir de los datos almacenados se realiza un análisis para generar el mapa topológico. El gran inconveniente es la complejidad inherente del método, lo que hace que la obtención del mapa topológico no sea evidente.

Algunos autores han desarrollado otros métodos para la obtención de mapas topológicos en base a la observación directa del entorno. Así, Dedeoglu presenta un mapa topológico basado en un comportamiento que va construyendo incrementalmente el mapa sin conocimiento *a priori* del entorno, aunque carece de información de la posición [DMS99]. Por otro lado, Lankenau propone un método topológico para la navegación en entornos grandes que sí incluye información de la posición y ha sido probado en entornos reales [LR02]. Sin embargo, el entorno debe ser conocido *a priori* y no puede cambiar. Por lo tanto, este método no sería válido, por ejemplo, en presencia de obstáculos móviles.

2.1.2 Integración de los paradigmas métrico y topológico

Muchos autores [CL85, KB91, TBB+98], se han dado cuenta de que modelar el entorno, bien mediante el paradigma métrico, bien mediante el paradigma topológico, presenta inconvenientes. La tendencia actual consiste en combinar ambos paradigmas con el objetivo de aprovechar las ventajas que posee cada uno de ellos. Así, integrando ambos esquemas es posible resolver el problema de la ambigüedad que presentan los mapas topológicos a la hora de distinguir las regiones entre sí partiendo únicamente de la información sensorial. La integración se puede hacer de dos formas, siendo su enfoque completamente distinto:

- Anotando el mapa topológico con información métrica [KB91, SD98, TNS03].
- Extrayendo el mapa topológico de un mapa métrico [Zel92, Thr98, AMF99, FS00, KSEP00]. Esta es la opción por la que se ha optado en la presente Tesis.

El primer grupo de técnicas consiste en construir un mapa topológico donde cada nodo incluye información sobre la geometría del lugar donde el nodo fue insertado. Gracias a esta información métrica se evita la ambigüedad del mapa topológico, permitiendo la obtención de relaciones geométricas entre los nodos. Sin embargo, estas operaciones suelen tener que realizarse *off line*, porque son lentas y costosas computacionalmente. A pesar de estos inconvenientes, la literatura nos muestra diversos ejemplos de métodos que se basan en este enfoque. Así, Kuipers y Byun [KB91] desarrollaron la Jerarquía Espacial Semántica (*Spatial Semantic Hierarchy (SSH)*), donde uno de los niveles de su propuesta es el topológico. Sin embargo, con este método el mapa topológico es difícil de construir cuando no hay información *a priori*. Por otro lado, la información que relaciona los nodos entre sí suele ser vaga. En [SD98] se presenta otra posibilidad. Simhon y Dudek construyen un mapa global constituido por un conjunto de mapas locales organizados como un mapa topológico. Esta estructura se emplea para representar entornos estáticos grandes, almacenándose esta información para futuras tareas, por ejemplo, de localización. Por lo tanto, es obvio que no es válida para entornos dinámicos donde hay obstáculos móviles. El método de Tomatis [TNS03] presenta los mismos problemas que los dos anteriores, ya que para realizar tareas en un entorno determinado se necesita tener un conocimiento *a priori* del mismo. Por lo tanto, a pesar de que este método ha demostrado su validez en la integración de los paradigmas métricos y topológicos, no es útil cuando el entorno está total o parcialmente inexplorado.

El segundo grupo de técnicas se basa en dividir el mapa métrico en regiones significativas del entorno. La representación geométrica se basa, habitualmente, en mapas probabilísticos [Elf87]. Dentro de este tipo de métodos destacan los trabajos de Zelinsky, Thrun, Arleo, Fabrizi y Kraetzschmar.

Zelinsky [Zel92] lleva a cabo la división del entorno por medio de los denominados *quadtrees*. Consiste en dividir toda región no uniforme en cuatro cuadrantes iguales, repitiendo este proceso de división de forma recursiva hasta que todas las regiones que representan el entorno son

uniformes o tienen un tamaño mínimo. Sin embargo, el grado de optimalidad de la partición resultante depende en gran medida de la distribución de los obstáculos [CSU97]. En la Figura 5.1 se muestra la manera en la que se modela un entorno con un obstáculo por medio de los *quadtrees*.

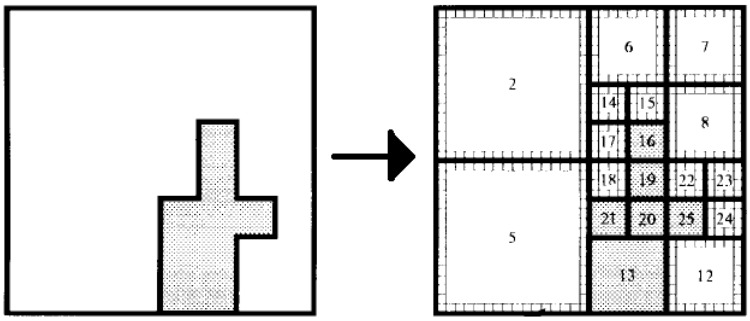

Figura 5.1: Descomposición de un entorno en *quadtrees* propuesta por Zelinsky.

Uno de los métodos más conocidos es el de Thrun [Thr98], el cual divide el espacio libre en regiones homogéneas de acuerdo a la forma de dichas regiones. El algoritmo de Thrun se resume en la Figura 5.2, partiendo de un entorno que debe estar completamente explorado. Consta de los siguientes pasos:

- *Umbralización.* La primera etapa convierte el mapa métrico probabilístico en un mapa binario donde solamente se distinguen dos niveles, espacio libre (denominado E), y espacio ocupado ($\overline{E}$). Esta fase limita ya de partida la posibilidad de incluir entornos total o parcialmente inexplorados.

- *Diagrama de Voronoi.* Para cada punto del espacio libre $(x, y) \in E$, hay uno o más puntos cercanos en el espacio ocupado $\overline{E}$, denominados puntos base. La segunda etapa del algoritmo calcula el diagrama de Voronoi [OY82], que es el conjunto de puntos del espacio libre E que tienen al menos dos puntos base diferentes y equidistantes. El resultado de esta etapa da lugar a un esqueleto del entorno, según se muestra en la Figura 5.2a.

- *Puntos Críticos.* A partir del diagrama de Voronoi hay que obtener los puntos pertenecientes al mismo que tienen una distancia mínima a sus puntos base en un entorno ϵ (Figura 5.2b).

- *Líneas Críticas.* Se obtienen conectando los puntos críticos con sus puntos base, descomponiendo el espacio libre en regiones (Figura 5.2c).

- *Grafo Topológico.* Finalmente, cada región se corresponde con uno de los nodos del mapa topológico (Figura 5.2d).

A pesar de que es una de las técnicas de integración métrico-topológica más conocida y referenciada, el método de Thrun requiere de una construcción *off line* del mapa topológico, y

únicamente cuando se dispone de un mapa métrico completamente explorado, lo que no hace viable su utilización si pueden aparecer móviles en cualquier momento. Este método ha sido implementado y estudiado en [Mar02], un Proyecto Fin de Carrera dirigido por el autor de esta Tesis. Las conclusiones de este estudio muestran la validez de la propuesta de integración métrico-topológica para la navegación en entornos completamente explorados, a pesar de que el mapa topológico no se calcula *on line*. Se demuestra también que, generalmente, suelen aparecer nodos del grafo en zonas conflictivas, como en puertas abiertas. Este hecho permitiría, a partir de la detección de las puertas, la adecuación del comportamiento de navegación a esa situación particular. Aun así, la conclusión más clara es que el método de Thrun no es válido para entornos total o parcialmente inexplorados, lo que requeriría de una modificación del mismo. Por lo tanto, no es aplicable al problema tratado en esta Tesis.

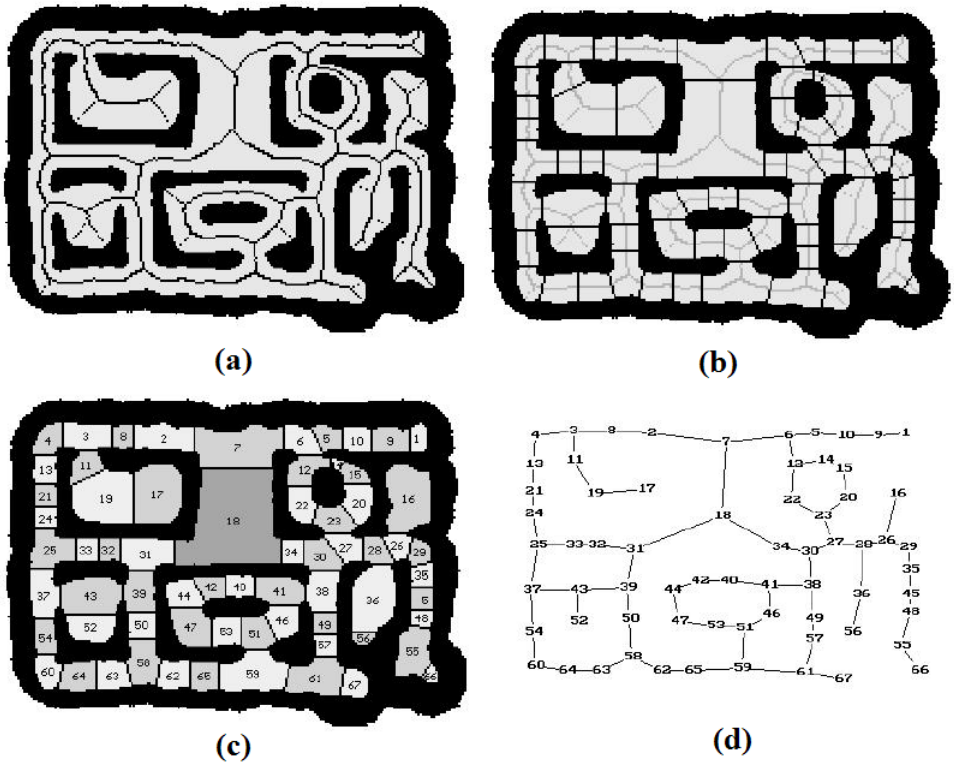

Figura 5.2: Método de Thrun para la obtención de un mapa topológico: (a) diagrama de Voronoi; (b) líneas críticas; (c) regiones; (d) mapa topológico.

Kraetzschmar también hace referencia a las desventajas que presenta la aproximación de Thrun [KSEP00]. Su propuesta se basa es la extracción de un mapa topológico de un mapa métrico usando los histogramas de las paredes. Este método supone que todas las regiones son rectangulares, extrayendo después de procesar el mapa métrico los bordes de los obstáculos, lo cual permite dividir el entorno en regiones (Figura 5.3a). Sin embargo, tal y como se reconoce en [KSEP00], su método no proporciona un mapa topológico que sea útil para la navegación.

Fabrizi y Saffiotti [FS00] extraen el mapa topológico del mapa probabilístico analizando la configuración del espacio libre en base a una herramienta de procesamiento de la morfología

matemática de imágenes. Su estudio se centra en espacios abiertos conectados por pasillos, ya que estas configuraciones son útiles para realizar una navegación basada en comportamientos y para la autolocalización del robot. La Figura 5.3b muestra un ejemplo de la descomposición de un entorno en regiones mediante esta técnica. Al igual que en el método de Thrun, el mapa topológico no incluye las regiones no explorados sino únicamente el espacio libre.

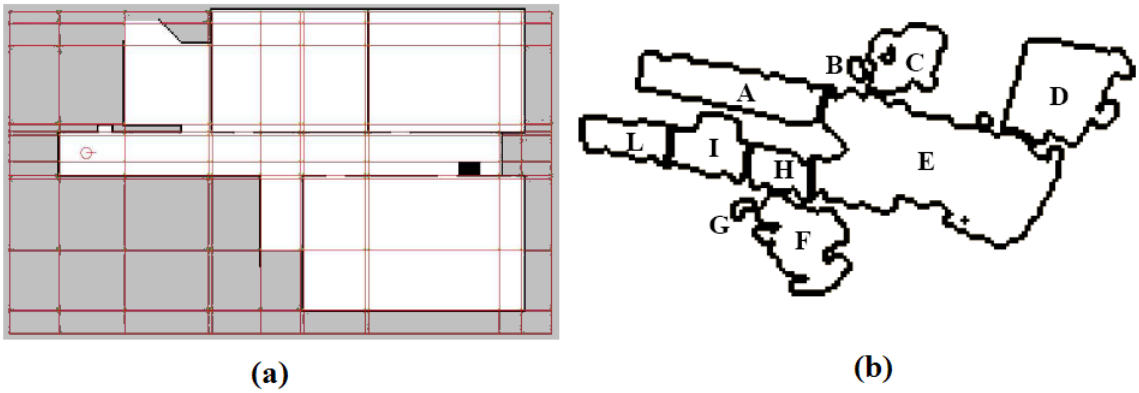

Figura 5.3: (a) Descomposición en regiones propuesta por Kraetzschmar; (b) descomposición en regiones según el método de Fabrizi.

Finalmente, el método de Arleo modela los bordes los obstáculos mediante líneas rectas [AMF99]. Se basa en la técnica de la partición de resolución variable (*variable-resolution partitioning*), que realiza una interpretación neuronal de la lectura de los sensores para obtener un mapa métrico local. Este mapa métrico se emplea a continuación para el modelado de los bordes de los obstáculos, lo cual permite segmentar el entorno en regiones. Su principal inconveniente es que no puede tratar con regiones irregulares ni con paredes que no sean paralelas o perpendiculares entre sí. En la Figura 5.4 se presenta un ejemplo de construcción de un mapa topológico por este método a partir de un entorno real (Figura 5.4a).

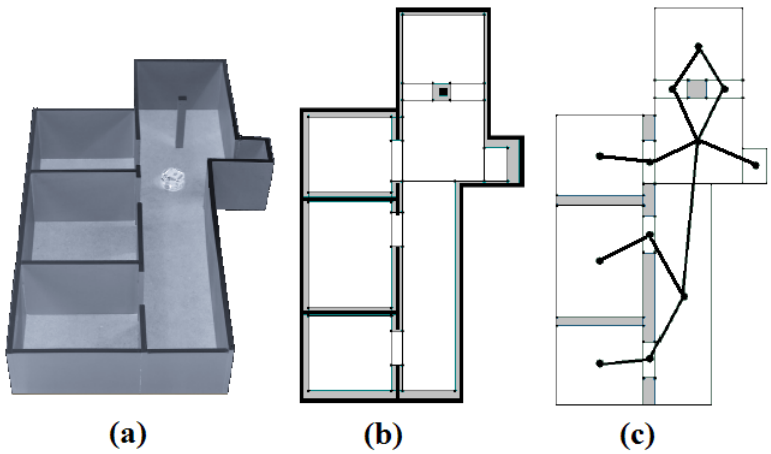

Figura 5.4: Método de Arleo para calcular mapas topológicos: (a) entorno de prueba; (b) partición de resolución variable; (c) mapa topológico.

Todos los métodos comentados que extraen el mapa topológico de una representación métrica comparten un grave inconveniente. Necesitan tener completamente explorado un entorno para poder construir su mapa topológico. Por lo tanto, no son apropiados para la exploración de entornos total o parcialmente desconocidos. De ahí que no se puedan emplear para trabajar en el nivel más alto de la Jerarquía de Navegación [FM00]. En la presente Tesis se emplea un método de integración métrico-topológica capaz de trabajar en el nivel superior de la Jerarquía de Navegación, ya que representa explícitamente tanto las zonas exploradas como las que no lo están. Será posible, por tanto, el cálculo de rutas para la exploración eficiente de un entorno a través de zonas no exploradas.

2.2 Modelado del entorno

2.2.1 Preprocesado del mapa métrico

El punto de partida para la construcción de la estructura jerárquica es el mapa métrico del entorno, proporcionado por el módulo *MapaMétrico* de la capa *ModeladoEntorno*. En esta Tesis se emplea un mapa métrico probabilístico o rejilla de ocupación [Elf87, Mor88]. Estas rejillas constituyen una división bidimensional del espacio en celdas. Cada celda representa una probabilidad del 0% al 100% de estar ocupada. Idealmente, las zonas ocupadas (obstáculos) presentan una probabilidad de ocupación del 100%, mientras que las zonas de espacio libre tienen una probabilidad de ocupación del 0%. Del mismo modo, el espacio inexplorado presentaría una probabilidad de ocupación del 50%. Para representar estos mapas métricos se le asocia al nivel de probabilidad de ocupación de cada celda un nivel de la escala de grises de 0 a 255. Así, las zonas libres se muestran en negro, los obstáculos en blanco y las áreas no exploradas en gris.

Antes de construir la estructura jerárquica de integración métrico-topológica se efectúa un preprocesado del mapa métrico. Consta de dos fases:

1. *Umbralización.* Debido al método empleado para construir el mapa métrico, la probabilidad de ocupación de una determinada celda no tiene por qué ser exactamente el 0%, 50% ó 100%. Sin embargo, la estructura jerárquica tiene como uno de sus objetivos el dividir el entorno en tres tipos de zonas, ocupadas, libres e inexploradas. Su probabilidad de ocupación es el 100%, 0% y 50%, respectivamente. Por lo tanto, el mapa métrico se umbraliza para que todas las celdas presenten alguno de estos tres valores de probabilidad. Así, a las celdas cuya probabilidad de ocupación esté por encima de un umbral U_{obst} se les asigna la probabilidad de ocupación P_{obst}, considerándose espacio ocupado. A las celdas con probabilidad de ocupación inferior a un umbral U_{libre} se las considera espacio libre, asignándoles una probabilidad de ocupación P_{libre}. Por último, las celdas cuya probabilidad de ocupación esté entre U_{libre} y U_{obst} serán consideradas como no exploradas, tomando el valor de probabilidad de ocupación P_{inex}, que suele fijarse al 50%, indicando que no se sabe si la celda está ocupada o no. En esta Tesis se han empleado unos umbrales $U_{obst} = 60\%$ y $U_{libre} = 40\%$, así como unas probabilidades de ocupación $P_{obst} = 100\%$, $P_{libre} = 0\%$ y

$P_{inex} = 50\%$.

2. *Crecimiento de Obstáculos.* La segunda fase consiste en aumentar el tamaño de los obstáculos en *Obst_Inc* celdas. El crecimiento se basa en la propagación de ondas desde el contorno de los obstáculos del entorno [BLL92]. Su objetivo es: i) conseguir unos obstáculos más definidos; y ii) tener en cuenta el tamaño del robot para evitar una hipotética colisión con cualquier obstáculo. Si se eliminara esta fase de realzado de obstáculos el mapa topológico y, por añadidura, la ruta resultante, podría conectar dos regiones que no lo están. También puede ocurrir que al crecer los obstáculos el robot quede dentro de una zona ocupada, cuando esto es imposible. Esta situación ocurre cuando el robot está muy cerca de un obstáculo. Para evitar esta situación se elimina el crecimiento de obstáculos en la posición del robot.

La Figura 5.5 muestra un ejemplo de aplicación de esta etapa de preprocesado a un mapa métrico de 256x256 celdas. El mapa métrico de partida es el de la Figura 5.5a. El entorno con el que se corresponde está totalmente explorado, salvo por cuatro zonas a las que el robot no puede acceder. En primer lugar, la zona oeste y norte del mapa están fuera del entorno. Así mismo, existen dos zonas en gris completamente rodeadas de espacio ocupado. Se corresponden con dos obstáculos. Lógicamente, el interior de los mismos no se puede explorar. Por último, la esquina inferior derecha del mapa también se corresponde con un obstáculo y tampoco se puede explorar. Después de aplicar la fase de *Umbralización* se obtiene el mapa de la Figura 5.5b. Los umbrales empleados son $P_{obst} = 60\%$ y $P_{libre} = 40\%$, tal y como se ha comentado previamente. Al mapa umbralizado se le aplica la fase de *Crecimiento de Obstáculos* de $Obst\_Inc = 1$ celda, resultando el mapa de la Figura 5.5c, convirtiéndose en la base de estructura jerárquica que se analiza en el siguiente apartado.

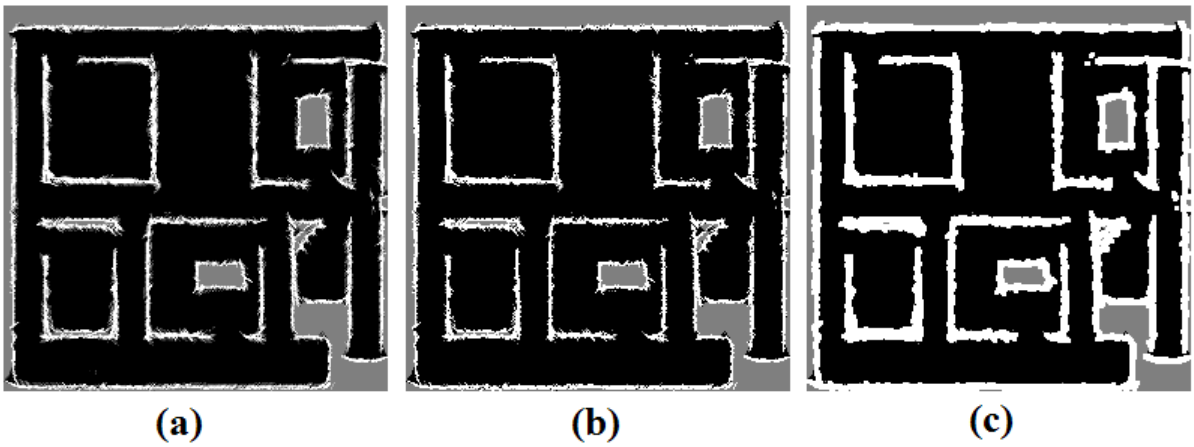

Figura 5.5: Preprocesado de un mapa métrico de 256x256 celdas: (a) mapa métrico de partida; (b) umbralización, $U_{obst} = 60\%$ y $U_{libre} = 40\%$; (c) crecimiento de obstáculos, $Obst\_Inc = 1$.

2.2.2 La estructura jerárquica

La estructura jerárquica es una pirámide cuya base es el mapa métrico umbralizado y con los obstáculos crecidos. Constituye el nivel $l = 0$ de la estructura. La pirámide se construye sobre este mapa métrico. El proceso consta de tres fases:

1. *Creación e Inicialización.* Los niveles que están por encima de la base se construyen siguiendo una estructura $4 - a - 1$. Así, un nivel l de la pirámide es una versión reducida del mapa, presentando $1/4$ de las celdas o nodos del nivel $l - 1$ inmediatamente inferior. La celda del nivel l resultante del promediado de las $2x2$ celdas del nivel $l - 1$ se denomina nodo padre, siendo los 4 nodos del nivel $l - 1$ los nodos hijo. La terna (x, y, l) identifica la celda en la posición (x, y) del nivel l de la pirámide. Tiene varios parámetros asociados:

   - *Homogeneidad*, $H(x, y, l)$. Será 1 si los cuatro nodos hijo del nivel $l-1$ son homogéneos y tienen la misma probabilidad de ocupación. En cualquier otro caso su valor es 0. Obviamente, el valor de la homogeneidad de las celdas del nivel 0 es $H(x, y, 0) = 1$.
   - *Área*, $A(x, y, l)$. Es igual a la suma de las áreas de los nodos hijo del nivel $l - 1$. En el nivel 0 la celdas cuya probabilidad de ocupación sea P_{inex} o P_{libre} tienen área 1. Las celdas con probabilidad de ocupación P_{obst} tienen área 0. Este parámetro hace referencia al número de nodos de la base que pertenecen a la región asociada a la celda (x, y, l) del nivel l.
   - *Probabilidad de ocupación*, $P(x, y, l)$. Si las 4 celdas hijo del nivel $l - 1$ tienen una probabilidad de ocupación P_{obst}, $P(x, y, l)$ será P_{obst}. En cualquier otro caso será una suma de la probabilidad de ocupación ponderada por el área de las cuatro celdas hijo del nivel $l - 1$, y normalizada por $A(x, y, l)$, en la forma:

$$\begin{aligned} P(x,y,l) = & \frac{A(2x,2y,l-1)\cdot P(2x,2y,l-1)}{A(x,y,l)} + \\ & + \frac{A(2x,2y+1,l-1)\cdot P(2x,2y+1,l-1)}{A(x,y,l)} + \\ & + \frac{A(2x+1,2y,l-1)\cdot P(2x+1,2y,l-1)}{A(x,y,l)} + \\ & + \frac{A(2x+1,2y+1,l-1)\cdot P(2x+1,2y+1,l-1)}{A(x,y,l)} \end{aligned} \tag{5.1}$$

     La probabilidad de ocupación $P(x, y, l)$ según la Ecuación (5.1) se umbraliza tal y como se hizo durante el preprocesado del mapa métrico. Así, todas las celdas de cualquier nivel de la estructura presentarán únicamente tres valores, P_{obst}, P_{libre} o P_{inex}. Se corresponden, respectivamente, con zonas ocupadas, libres y no exploradas.
   - *Padre*, $(X, Y)_{(x,y,l)}$. Si $H(x, y, l) = 1$, esto es, si los cuatro nodos hijo del nivel inferior son homogéneos y tienen la misma probabilidad de ocupación, el valor de este parámetro para las cuatro celdas hijo es (x, y). En otro caso los nodos hijo, denominados huérfanos, no se enlazarán con el padre, $(X, Y)_{(x,y,l)} = NULL$.

- *Clase*, $Cl(x, y, l)$. El objetivo de la estructura es dividir el entorno en regiones. Cada región tendrá una clase asignada. Este parámetro hace referencia a la clase asignada al nodo (x, y) del nivel l. Existe la posibilidad de que una celda no tenga clase asignada, esto es, que no pertenezca a ninguna región. Inicialmente ninguna celda de ningún nivel tiene clase asignada, $Cl(x, y, l) = NULL$. La clase $Cl(x, y, l)$ a la que pertenece una celda (x, y) del nivel l, se asigna durante la etapa de *Clasificación*. Las celdas cuya probabilidad de ocupación es $P(x, y, l) = P_{obst}$ nunca tendrán clase asignada puesto que las regiones ocupadas no están representadas en el mapa topológico.
- *Centroide*, $Ct(x, y, l)$. Es el centro de masas de la región en la base asociada a la celda (x, y, l).

Cuando esta fase finaliza las celdas de niveles por encima de la base ($l = 0$) cuya homogeneidad es la unidad están relacionadas con un conjunto de celdas de la base. Sin embargo, al igual que en los *quadtrees* [Zel92], la descomposición depende de la posición de los obstáculos. La estructura de enlaces sirve para distinguir regiones homogéneas, presentando una forma regular. Además, no reflejan la verdadera distribución que nos encontramos en el entorno. Para solucionar estos problemas se emplean las dos etapas siguientes de *Enlazado* y *Clasificación*.

2. *Enlazado*. Su objetivo es enlazar las celdas del nivel l que no tienen padre con alguna celda del nivel $l + 1$. Para ello, los nodos huérfanos del nivel l buscan un nodo vecino en el mismo nivel cuyo padre sea $(x_p, y_p, l + 1)$, enlazándose a este padre. Una celda huérfana (x, y, l) se enlaza con el padre de una celda vecina, $(x_p, y_p, l + 1)$, cuando se cumplen las siguientes condiciones:

   - Si (x, y, l) y $(x_p, y_p, l + 1)$ son homogéneos, $H(x, y, l) = 1$ y $H(x_p, y_p, l + 1) = 1$.
   - Si el nodo huérfano y el padre del vecino tienen la misma probabilidad de ocupación, $P(x, y, l) = P(x_p, y_p, l + 1)$.
   - Si la distancia euclídea entre los centroides de las regiones asociadas en la base es menor que una constante $Dist\_Max$, $\|Ct(x, y, l) - Ct(x_p, y_p, l + 1)\|_2 < Dist\_Max$

   Durante esta fase se fusionan zonas adyacentes que son parecidas, disminuyendo el número final de regiones y eliminando la dependencia con la distribución de los obstáculos en el entorno. El proceso se efectúa de un modo descendente, comenzando por la cúspide de la pirámide, y terminando por su base. El parámetro $Dist\_Max$ es un umbral que controla el tamaño máximo de las regiones, es decir, la dispersión de las mismas. Tras finalizar esta etapa las regiones de la base tendrán forma irregular.

3. *Clasificación*. Durante esta etapa se obtiene la clase a la que pertenece cada celda. Para generar una nueva clase, $Cl(x, y, l) \neq NULL$, un nodo (x, y, l) del nivel l debe cumplir que:

   - No está enlazado a ningún padre, $(X, Y)_{(x,y,l)} = NULL$.

- Es homogéneo, $H(x, y, l) = 1$.
- La celda no se corresponde con una zona ocupada, $P(x, y, l) \neq P_{obst}$, siendo $P(x, y, l)$ la probabilidad de ocupación de la clase generada. Tal y como hemos comentado, $P(x, y, l)$ solamente podrá tomar dos valores, P_{libre} o P_{inex}.
- El área de la región asociada en la base es mayor que A_{min}, $A(x, y, l) > A_{min}$.
- No puede fusionarse con ningún nodo vecino.

A_{min} es una constante que establece el mínimo área de una región para poder ser considerada como tal y formar parte del mapa topológico. Este valor se introduce para evitar la aparición de demasiadas regiones en el mapa topológico con un área muy pequeña.

El número de clases se obtiene de un modo no supervisado, pudiendo generarse esas clases a partir de celdas de cualquier nivel excepto la cúspide de la pirámide. Así mismo, el número de clases se reduce puesto que el nodo (x_1, y_1, l) se fusionará con el nodo vecino (x_2, y_2, l), es decir, formará parte de la misma clase $Cl(x_2, y_2, l)$, cuando se cumplan las siguientes condiciones:

- Si ninguno de los dos nodos implicados tiene padre, $(X, Y)_{(x_1,y_1,l)} = NULL$ y $(X, Y)_{(x_2,y_2,l)} = NULL$.
- Si ambos nodos son homogéneos, $H(x_1, y_1, l) = 1$ y $H(x_2, y_2, l) = 1$
- Si la probabilidad de ocupación es la misma para ambos nodos, no pudiendo ser la correspondiente a una zona ocupada, $P(x_1, y_1, l) = P(x_2, y_2, l)$.
- Si el nodo vecino (x_2, y_2, l) pertenece a alguna clase, $Cl(x_2, y_2, l) \neq NULL$
- Si la distancia euclídea entre los centroides de las regiones asociadas en la base es menor que $Dist\_Max$, $\|Ct(x_1, y_1, l) - Ct(x_2, y_2, l)\|_2 < Dist\_Max$. Solamente se pueden fusionar regiones que no están muy separadas entre sí.
- Por último, si el área de la región en la base a la que pertenece (x_1, y_1, l) es mayor que A_{min}, $A(x_1, y_1, l) > A_{min}$

Al igual que en la fase de *Enlazado* el proceso se realiza de un modo descendente. Cualquier celda (x, y, l) del nivel l cuyo padre sea $(X, Y)_{(x,y,l)} = (x_p, y_p, l+1)$, pertenecerá a la misma clase que su padre, $Cl(x, y, l) = Cl(x_p, y_p, l + 1)$. De este modo se propagan los enlaces hasta llegar a la base de la pirámide.

Al finalizar esta etapa se completa la estructura de enlaces, estando las celdas de la base de la pirámide clasificadas en regiones (siempre y cuando el área de la región sea mayor que A_{min}). Puede ocurrir que el robot se encuentre en una región que no ha sido clasificada porque su área es muy pequeña. Sin embargo, el robot debe pertenecer a una región del mapa topológico porque será el comienzo de la ruta que se obtenga para la exploración completa. En este caso, se le asigna al robot una nueva clase. También debemos comentar que los obstáculos no pertenecen a ninguna clase. Únicamente estarán representadas en el mapa topológico las regiones de espacio libre o inexploradas.

La Figura 5.6 ilustra la generación de la estructura jerárquica a partir de un mapa métrico simple de 8x8 celdas. La Figura 5.6a muestra la fase de *Creación e Inicialización*, donde se aprecian los primeros enlaces creados entre el nivel 0 y el 1. Los nodos homogéneos se representan con el mismo nivel de la escala de grises. Los nodos blancos representan nodos no homogéneos, debido a que la probabilidad de ocupación de sus 4 hijos no es la misma. En la Figura 5.6b se muestra el resultado tras aplicar la fase de *Enlazado* a la estructura de la Figura 5.6a. Se aprecia que los nodos homogéneos del nivel 1 que no tienen padre se enlazan a los padres de los nodos vecinos, pasando a formar parte de la misma región en la base. Finalmente, la Figura 5.6c muestra el estado de la estructura tras la fase de *Clasificación*. Al terminar esta etapa la estructura de enlaces se ha completado. En este caso se fusionan los nodos vecinos del mismo nivel que son homogéneos y tienen la misma probabilidad de ocupación.

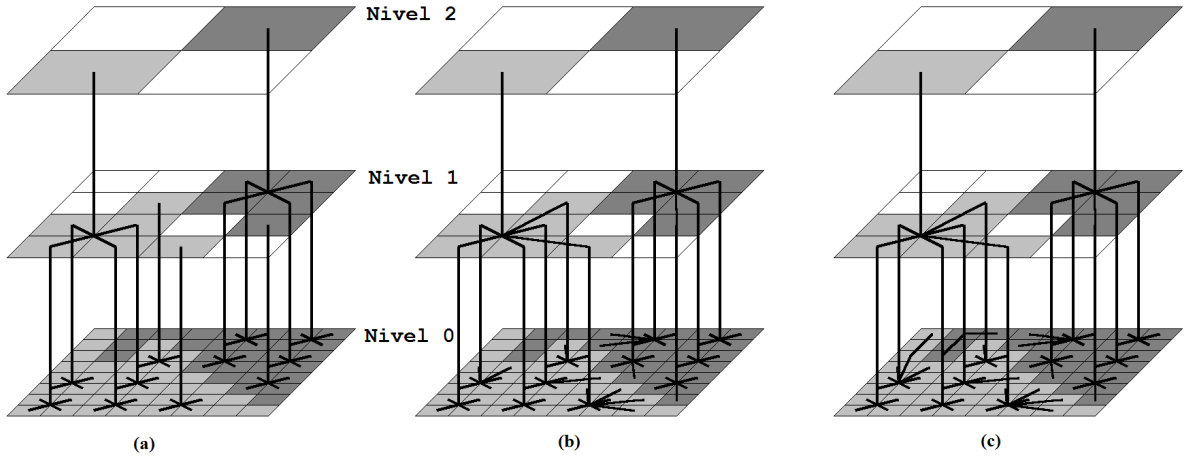

Figura 5.6: Estructura jerárquica: (a) *Creación e Inicialización*; (b) *Enlazado*; (c) *Clasificación*.

En la Figura 5.7 se presentan los distintos niveles de la pirámide una vez construida toda la estructura jerárquica. La base de la pirámide está constituida por el mapa métrico umbralizado y crecido de la Figura 5.5c. Se muestran, a lo largo de los niveles, las distintas clases que van apareciendo, observándose al final del proceso una descomposición no supervisada del entorno en 27 regiones. Como hemos indicando anteriormente, los obstáculos no están incluidos en la clasificación. El robot se supone en la posición $(128, 128)$ del mapa métrico. Se han empleado unos valores $Dist\_Max = 40$ y $A_{min} = 16$. Las regiones en las que se divide el entorno se presentan en diferentes niveles de la escala de grises. No hay ningún nodo que genere clase en los niveles de 2x2 y 4x4 celdas (Figuras 5.7a y b). La primera clase se genera a partir del nivel 8x8 (Figura 5.7c). El nivel 4 es origen de otras 9 clases más, según se aprecia en la Figura 5.7d. Las 17 clases restantes parten del nivel 3 de 32x32 celdas, tal y como se observa en la Figura 5.7e. Los niveles inferiores no generan ningún nodo del mapa topológico. Las clases que se van generando a lo largo de la pirámide se propagan hacia la base (nivel 0), gracias a la estructura de enlaces establecida. Finalmente, se tiene en la base una descomposición en regiones del entorno.

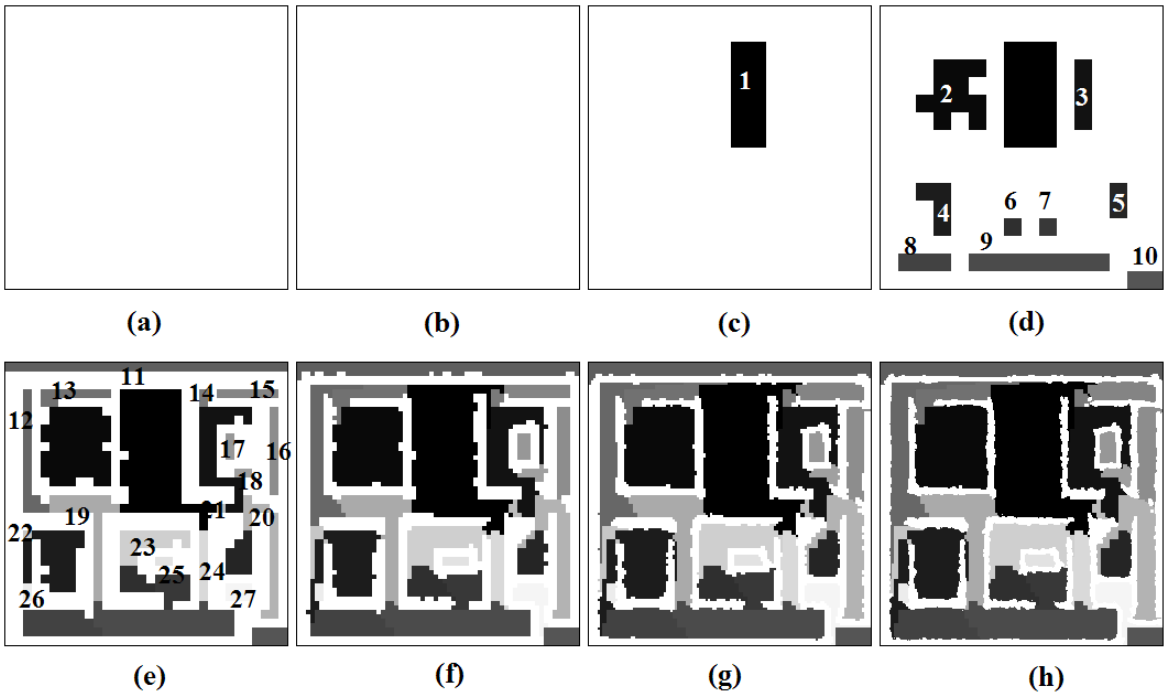

Figura 5.7: Estructura jerárquica: (a) nivel 7, 2x2 celdas; (b) nivel 6, 4x4 celdas; (c) nivel 5, 8x8 celdas; (d) nivel 4, 16x16 celdas; (e) nivel 3, 32x32 celdas; (f) nivel 2, 64x64 celdas; (g) nivel 1, 128x128 celdas; (h) nivel 0, 256x256 celdas.

2.2.3 El mapa topológico

Cuando la estructura está generada, cualquier celda cuya homogeneidad sea $H(x, y, l) = 1$ en cualquier nivel de la estructura está enlazada a una región homogénea de la base. Las regiones se constituyen a partir de celdas homogéneas que no tienen padre, tal y como se comentó en la fase de *Clasificación* del apartado anterior. Estas celdas dan lugar a los nodos n_i del mapa topológico, los cuales están asociados a las regiones en la base. Es importante hacer notar que los nodos resultantes del mapa topológico se generan a partir de celdas distribuidas a lo largo de todos los niveles de la estructura. Cada región está representada por su centroide, siendo éste el centro de masas de la región en la base. Sin embargo, aún no se han establecido las relaciones entre los nodos del mapa topológico. Para ello se siguen los siguientes pasos:

1. *Cálculo de las Bounding Boxes.* La *bounding box* de una región es la caja o rectángulo de dimensiones mínimas que engloba a la región en la base. Esta etapa es necesaria para poder estudiar posteriormente la conectividad entre las regiones y establecer los arcos de conexión del grafo topológico.

2. *Estudio de Conectividad.* Es el último paso que hay que dar para conseguir el mapa topológico del entorno. Analiza de forma no supervisada la relación de conexión entre todas las parejas posibles de regiones. El análisis se lleva a cabo en función de la relación espacial definida por sus *bounding boxes* y las celdas pertenecientes a ellas en la base de la

pirámide. Al finalizar esta fase se dispone de una matriz de conexión bidimensional, M_C. Si dos nodos del mapa topológico n_i y n_j están conectados, $M_C(n_i, n_j) = 1$. En otro caso, $M_C(n_i, n_j) = 0$. El proceso para comprobar si dos nodos n_i y n_j están conectados es el siguiente:

- Si las *bounding boxes* de ambas regiones no intersecan, no estarán conectadas, $M_C(n_i, n_j) = 0$.
- Si las *bounding boxes* de las regiones intersecan pero no hay ninguna celda de la clase Cl_i vecina a alguna celda de la clase Cl_j, no estarán conectadas, $M_C(n_i, n_j) = 0$.
- Si las *bounding boxes* de las dos regiones intersecan y alguna celda de la clase Cl_i es vecina de alguna celda de la clase Cl_j, estarán conectadas, $M_C(n_i, n_j) = 1$.

Al finalizar este proceso conocemos, además de los nodos del grafo, las relaciones de conexión entre ellos, por lo que el mapa topológico está completamente definido. El peso de los arcos de conexión entre regiones es igual a la distancia euclídea entre los centroides de las regiones con las que están relacionados. En este mapa topológico no están representadas las zonas ocupadas (obstáculos), puesto que el robot nunca podrá acceder a zonas de este tipo. Sin embargo, las zonas no exploradas sí que están explícitamente representadas, al contrario que en otros algoritmos clásicos de integración métrico-topológica [Thr98]. Este hecho permite planificar rutas de alto nivel para la exploración completa de las zonas inexploradas del entorno.

En la Figura 5.8 se observa el mapa topológico que se obtiene al finalizar el proceso. En este caso se han empleado unos valores $Obst\_Inc = 1$, $Dist\_Max = 40$ y $A_{min} = 16$. El robot se supone en la posición $(128, 128)$ del mapa métrico. La generación parte del mapa de 256x256 celdas de la Figura 5.8a. Después de llevar a cabo todas las fases comentadas el obtiene el mapa topológico de la Figura 5.8c, donde aparecen los nodos y los arcos de conexión de la representación. Los nodos se corresponden con cada una de las 27 regiones en las que se descompone el entorno (Figura 5.8b). Para obtener los arcos de conexión se realiza la fase de *Estudio de Conectividad*. Se aprecia, por ejemplo, que la zona inferior derecha no explorada consta de dos nodos. Es lógico que estos dos nodos estén conectados con un arco de conexión, puesto que son adyacentes. Sin embargo, ninguno de estos dos nodos está conectado con cualquiera de los adyacentes que representa una zona explorada, pues en medio hay zonas ocupadas, reforzadas durante la fase de *Crecimiento de Obstáculos*.

En el caso de que no se hubiera aplicado un realzado de obstáculos ($Obst\_Inc = 0$), al mapa métrico de la Figura 5.8a, el mapa topológico resultante sería el de la Figura 5.9. Esta descomposición del entorno en regiones no es correcta, puesto que aparecen arcos de conexión donde no deberían. Así, cuando $Obst\_Inc = 1$, la zona inferior derecha del mapa presenta dos nodos conectados entre sí, pero sin conexión alguna a ningún otro nodo del mapa topológico (Figura 5.8c). Al eliminar el realzado de obstáculos aparecen conexiones desde esta zona del entorno con áreas libres del mismo. Sin embargo, esto no puede ocurrir puesto que entre ambas zonas existen obstáculos. Debido al proceso de construcción del mapa métrico, los obstáculos no

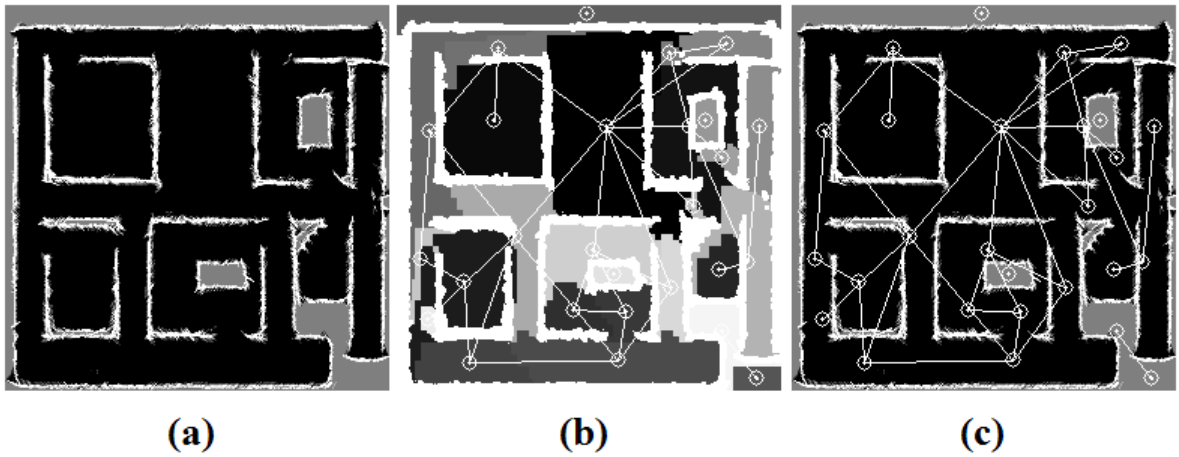

Figura 5.8: Mapa topológico: (a) mapa métrico de partida; (b) descomposición en regiones; (c) mapa topológico.

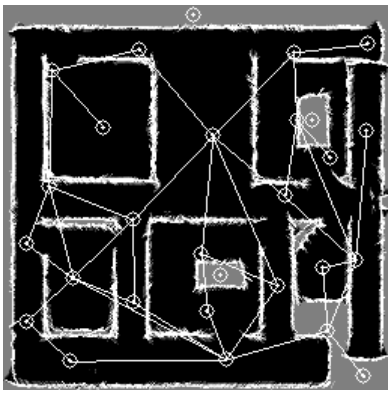

Figura 5.9: Eliminación del crecimiento de obstáculos en la construcción del mapa topológico.

están completamente definidos. Por lo tanto, la fase de *Crecimiento de Obstáculos* es necesaria para eliminar posibles situaciones como la representada en la Figura 5.9.

La Figura 5.10 muestra la manera en la que el umbral $Dist\_Max$ influye en el mapa topológico resultante, tomando $Obst\_Inc = 1$ y $A_{min} = 16$. Se puede apreciar que un valor bajo de $Dist\_Max$ provoca muchos nodos en el mapa topológico. Así, si $Dist\_Max = 10$ aparecen 76 regiones distintas en el entorno (Figura 5.10a). Un valor demasiado bajo de $Dist\_Max$ como en este caso da lugar a una descomposición del entorno donde aparecen muchas regiones pequeñas. La habitación superior izquierda en la Figura 5.10a presenta, por ejemplo, 15 nodos del mapa topológico. Al aumentar un poco el valor de $Dist\_Max$ (20 en la Figura 5.10b) el número de clases decrece sustancialmente hasta 31. Sin embargo, la topología del grafo es muy parecida al caso anterior. Por lo tanto, aunque los mapas topológicos pueden presentar más o menos nodos dependiendo del valor de $Dist\_Max$, los grafos resultantes son muy similares, ventaja clara sobre mapas topológicos obtenidos con otros algoritmos basados en el paradigma de division y mezclado como los conocidos *quadtrees* [Zel92]. De hecho, la representación tiende a depender únicamente de la configuración del entorno cuando $Dist\_Max$ supera un cierto valor.

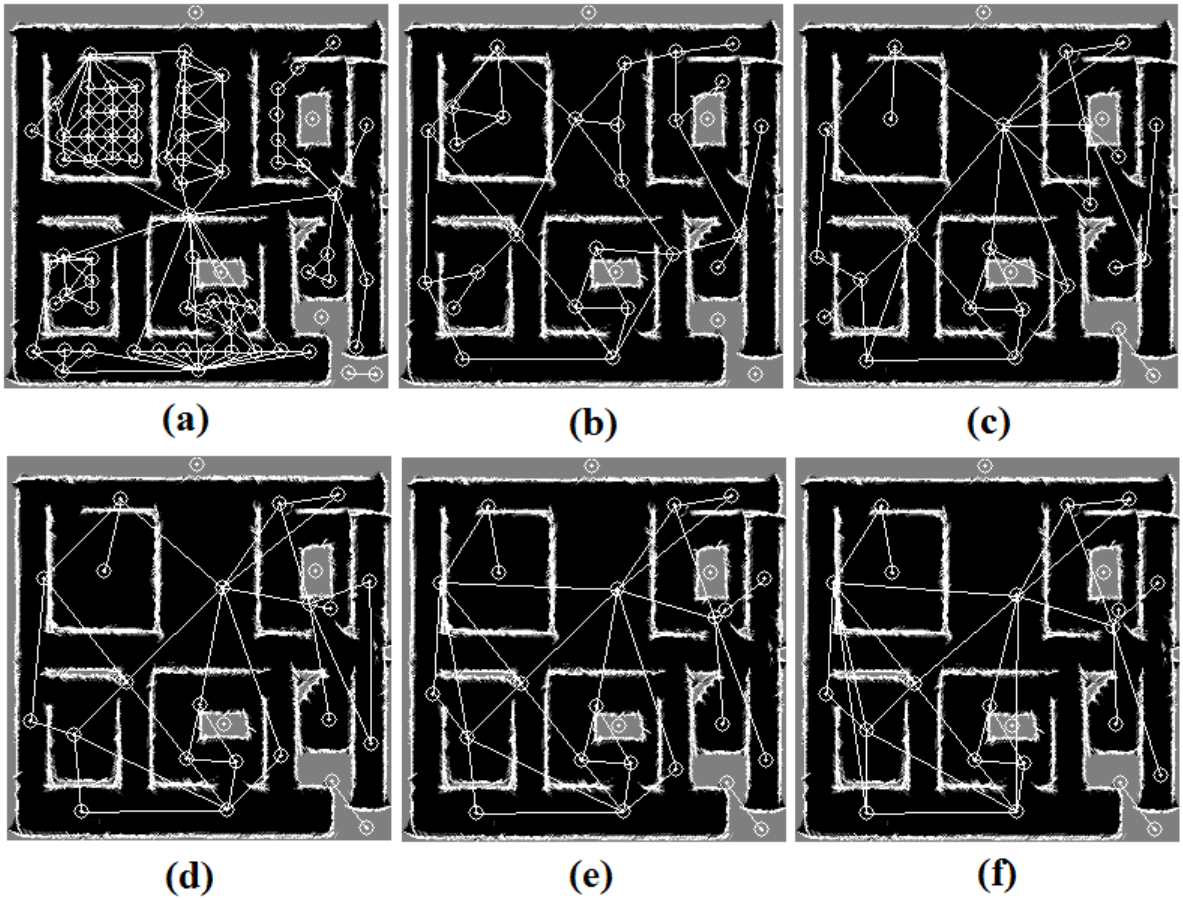

Figura 5.10: Mapa topológico para diferentes valores de $Dist\_Max$: (a) $Dist\_Max = 10$; (b) $Dist\_Max = 20$; (c) $Dist\_Max = 40$; (d) $Dist\_Max = 60$; (e) $Dist\_Max = 80$; (f) $Dist\_Max = 100$.

En la Figura 5.10 se muestra empíricamente que cuando $Dist\_Max \geq 40$, los mapas topológicos resultantes son muy parecidos (Figuras 5.10c, d, e y f). El número de clases obtenidas para $Dist\_Max = 40$, $Dist\_Max = 60$, $Dist\_Max = 80$ y $Dist\_Max = 100$ es, respectivamente, de 27, 25, 25 y 24. Desde el punto de vista de la planificación de rutas para exploración completa debemos decir que un valor muy bajo de $Dist\_Max$ como en la Figura 5.10a da lugar a la división de un misma zona es muchas regiones pequeñas. Aunque la representación del entorno es correcta, esta situación no es tan deseable como la que sucede cuando $Dist\_Max = 40$ (Figura 5.10c), donde la región superior izquierda presenta un único nodo del mapa topológico.

Otro factor que influye en nuestra representación métrico-topológica es A_{min}. En la Figura 5.11 se muestran los mapas topológicos obtenidos para distintos valores de A_{min} cuando $Obst\_Inc = 1$ y $Dist\_Max = 40$. Cuando $A_{min} = 2$, $A_{min} = 8$, $A_{min} = 16$, y $A_{min} = 64$, aparecen 52, 36, 27 y 10 clases, respectivamente (ver Figuras 5.11a, b, c y d, respectivamente). Si A_{min} es muy pequeño, el número de clases resultantes es muy grande, no siendo la representación topológica del entorno tan útil. Así, en la Figura 5.11a aparecen nodos con un área ínfima en la zona oeste del entorno, pero fuera del mismo. A medida que A_{min} crece el número

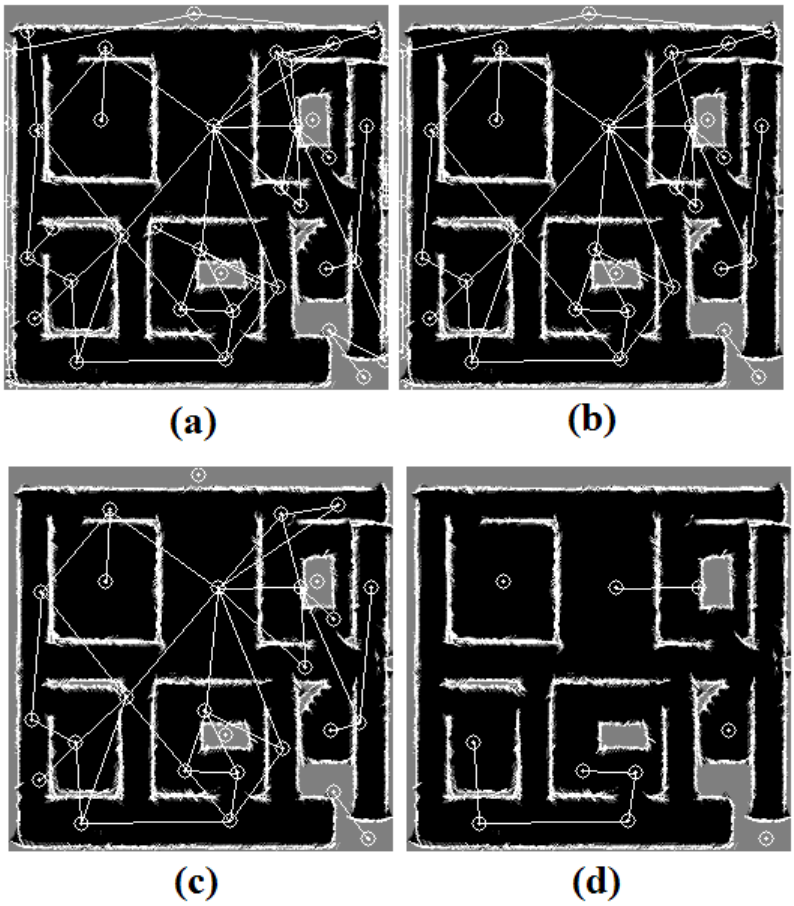

Figura 5.11: Mapa topológico para diferentes valores de A_{min}: (a) $A_{min} = 2$; (b) $A_{min} = 8$; (c) $A_{min} = 16$; (d) $A_{min} = 64$.

de clases resultantes se ajusta mucho mejor a la topología del entorno, siendo muy fiel en el entorno representado cuando $A_{min} = 16$. Sin embargo, cuando A_{min} es demasiado grande el grafo resultante no se corresponde con la topología del entorno (Figura 5.11d). En este último caso, $A_{min} = 64$, surgen únicamente 10 clases. Peor aún que el hecho de que hayan aparecido tan solo 10 clases es que el grafo no sea útil, pues no presenta una conexión coherente entre sus nodos en relación al entorno que pretende representar. Se ha comprobado que un valor $A_{min} = 16$ suele ser adecuado para obtener una representación topológica que represente fielmente el entorno y sea adecuada para la planificación de rutas.

Para evaluar la posibilidad de emplear el mapa topológico propuesto *on line* mientras el agente navega por el entorno se presenta en la Tabla 5.1 un resumen del tiempo medio que se tarda en su construcción. Las medidas se han realizado en un Pentium 3 a 550 MHz con 192 MB de memoria RAM. Se han realizado medidas temporales para tres entornos cuyo mapa métrico de partida es de distinto tamaño: 256x256 celdas, 512x512 celdas y 1024x1024. Obviamente, cuanto mayor sea el tamaño del mapa métrico, mayor será el tiempo empleado en la construcción del mapa topológico. Los parámetros empleados para la construcción de la estructura son $Obst\_Inc = 1$, $A_{min} = 16$ y $Dist\_Max = 40$. Los errores se han calculado sobre un intervalo de

confianza del 95%. La construcción de la estructura jerárquica es la que más tiempo de proceso consume, en torno al 75% o el 80% del total. Además, a medida que crece el tamaño del mapa el tiempo de proceso se incrementa drásticamente. Para un mapa de 1024x1024 celdas se tarda más de 2 segundos. Si el robot navega con una velocidad de traslación de $200mm/s$, recorrería $0.4m$ antes de volver a calcular el mapa topológico, lo que podría suponer un problema si aparece un obstáculo móvil durante ese tiempo. Sin embargo, en esta Tesis se trabaja normalmente con mapas de 256x256 celdas. El tiempo de construcción del mapa topológico es de tan solo 122 ms en este caso. Esto supondría que el robot recorrería $3.5cm$ antes de recalcular el grafo. Debido a la rapidez del método se observa que la propuesta es válida para la planificación de rutas de exploración completa, ya que incluso en el caso de la aparición de obstáculos móviles inesperados el robot puede recalcular una nueva ruta antes de colisionar con el obstáculo. Hay que comentar que la máquina en la que se han realizado las pruebas no es muy moderna. Si se empleara una máquina de mejores prestaciones los tiempos se reducirían bastante, con lo que se refrendaría todavía más la validez del método.

	256x256 celdas	512x512 celdas	1024x1024 celdas
Preprocesado			
1. Umbralización	1.510 ± 0.035 ms	6.936 ± 0.081 ms	115 ± 1 ms
2. Crecimiento Obstáculos	1.546 ± 0.040 ms	2.085 ± 0.057 ms	280.9 ± 1.8 ms
Estructura Jerárquica			
1. Creación e Inicialización	53.74 ± 0.12 ms	219.45 ± 0.21 ms	918.29 ± 7.62 ms
2. Enlazado	25.55 ± 0.10 ms	62.40 ± 0.12 ms	598.16 ± 3.65 ms
3. Clasificación	23.81 ± 0.10 ms	109.26 ± 0.15 ms	201.78 ± 1.61 ms
Mapa Topológico			
1. Bounding Boxes	11.077 ± 0.080 ms	45.25 ± 0.12 ms	115.9 ± 1.1 ms
2. Estudio Conectividad	5.01 ± 0.05 ms	62.78 ± 0.10 ms	2.3 ± 0.4 ms
Total Mapa Topológico	122.26 ± 0.15 ms	508 ± 3 ms	2.234 ± 0.014 s

Tabla 5.1: Tiempo de construcción del mapa topológico.

Aunque en los ejemplos que se han mostrado hasta el momento los mapas métricos presentan un grado de exploración máximo, el algoritmo de construcción del mapa métrico funciona de la misma manera cuando el grado de exploración es mucho menor. Al estar las zonas no exploradas representadas en el mapa métrico también lo estarán en el mapa topológico, tanto en entornos simulados como reales. En la Figura 5.12 se presenta la representación métrico-topológico obtenida en un entorno simulado y en un entorno real.

3 Algoritmo de exploración completa

La técnica propuesta para planificar rutas que exploren de forma íntegra el entorno se basa en el mapa métrico-topológico presentado en la Sección 2. El método es válido para entornos total o parcialmente desconocidos, y se puede resumir con las siguientes consideraciones:

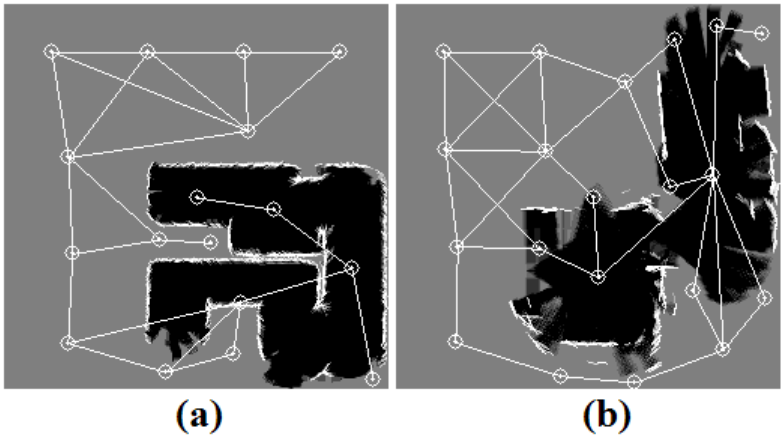

Figura 5.12: Representación métrico-topológico de un entorno parcialmente explorado: (a) entorno simulado; (b) entorno real.

- El mapa topológico presenta un conjunto de nodos cuya probabilidad de ocupación es P_{inex}. Estos nodos se corresponden con las regiones inexploradas del entorno.

- Hay que tener en cuenta a la hora de planificar la ruta el coste para ir de un nodo a cualquier otro del mapa topológico. Recordemos que el peso de los arcos de conexión entre nodos es la distancia euclídea entre los centroides de las regiones a las que representan. Estas distancias deben ser consideradas para optimizar el cálculo realizado.

- La ruta debe incluir todas las regiones inexploradas, y una única vez.

Una vez está disponible la representación métrico-topológica, el algoritmo de exploración completa se ejecuta en tres fases diferenciadas, tal y como se ilustra en la Figura 5.13:

1. *Obtención de las regiones no exploradas.* El primer elemento necesario es una lista de todos los nodos no explorados del mapa topológico. Esta fase tiene como objetivo la obtención de este conjunto de nodos, eliminando aquéllos relacionados con regiones a las que el robot no tiene acceso.

2. *Cálculo de la matriz de distancias.* Los nodos por sí mismos no son suficientes para poder planificar la mejor ruta de exploración completa. Se necesita un segundo elemento, que consiste en la matriz de distancias entre los nodos no explorados accesibles por el robot.

3. *Cálculo de la ruta.* Cuando se dispone de los nodos y de la matriz de distancias entre ellos, ya se puede planificar la ruta óptima que explora de forma completa el entorno.

A continuación se analizan las tres fases del algoritmo de exploración completa.

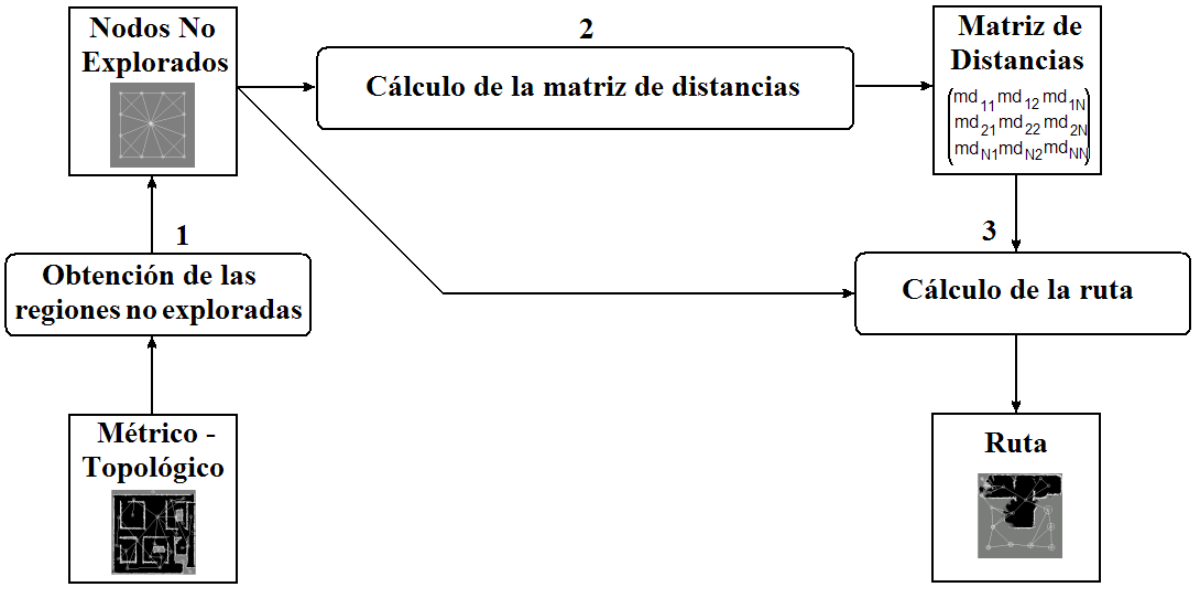

Figura 5.13: Fases del algoritmo de exploración completa a partir de la representación métrico-topológica.

3.1 Obtención de las regiones no exploradas

El objetivo de esta fase es obtener una lista con todos los nodos no explorados accesibles desde la posición del robot. Aunque la región en la que está situado el robot siempre estará explorada, deberá ser incluida en esta lista, puesto que es el inicio de la ruta de exploración completa. El mapa topológico presentará N nodos no explorados, los cuales pertenecen a la lista de nodos inexplorados $L_N = \{n_1, n_2, \ldots, n_N\}$. De los N nodos inexplorados, solamente N_a serán accesibles desde la posición del robot, con $N_a \leq N$. La lista $L_{N_a} = \{n_1, n_2, \ldots, n_{N_a}\}$ de los N_a nodos accesibles se obtiene en dos pasos:

- En primer lugar se obtienen los nodos n_i pertenecientes al mapa topológico cuya probabilidad de ocupación, P_{n_i} cumple que $P_{n_i} < U_{obst}$ y $P_{n_i} > U_{libre}$. Es decir, se escogen los nodos en los que $P_{n_i} = P_{inex}$. Todos estos nodos, junto con el nodo correspondiente a la región del robot, se incluyen en la lista L_N.

- En una segunda etapa se obtiene la lista $L_{N_a} \in L_N$ con un subconjunto de los nodos no explorados. Contiene únicamente los N_a nodos no explorados a los que hipotéticamente el robot podría llegar desde su posición actual. El eliminar de L_N aquéllos nodos a los que el robot no tiene acceso es un paso necesario que hay que ejecutar. Este hecho se pone de manifiesto si observamos el entorno de la Figura 5.14. Se trata del mismo entorno de la Figura 5.5a, en un estadio previo de la exploración. Para llegar a la representación métrico-topológica final se han empleado unos valores $Obst\_Inc = 1$, $Dist\_Max = 40$ y $A_{min} = 16$, estando el robot situado en la posición $(128, 128)$ del mapa. El mapa topológico consta de 29 nodos, de los cuales 10 están inexplorados. Sin embargo, en la Figura 5.14 se aprecian 5 nodos que no son accesibles, los nodos 6, 7, 8, 9 y 10. El nodo 6 se corresponde con una

zona fuera del entorno, mientras que los nodos 7, 8, 9 y 10 son obstáculos. Obviamente, al no poder explorar el interior de estos obstáculos aparece un nodo del mapa topológico con una probabilidad de ocupación P_{inex}. Por lo tanto, la lista de nodos inexplorados consta de 6 nodos, los 5 inexplorados accesibles más el correspondiente a la región en la que se encuentra el robot.

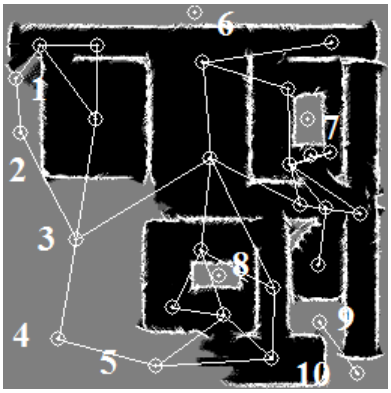

Figura 5.14: Obtención de las regiones no exploradas de un entorno.

3.2 Cálculo de la matriz de distancias

Una vez que se dispone de todos los nodos no explorados accesibles desde la posición del robot, se necesita conocer el coste para ir desde un nodo hacia otro, de modo que se puedan planificar rutas para la exploración completa del entorno. El coste para ir desde un nodo n_i hasta otro nodo n_j será evaluado en función de la distancia entre los centroides asociados a las regiones a las que representan, $D(n_i, n_j)$. Este valor debe ser conocido para todas las posibles parejas de nodos que están en la lista de nodos no explorados accesibles. El resultado es una matriz de distancias entre nodos, M_D.

La matriz de distancias M_D es simétrica, puesto que la distancia para ir del nodo n_i al nodo n_j es la misma que para ir del nodo n_j al nodo n_i, $M_D(n_i, n_j) = M_D(n_j, n_i)$. A la hora de calcular M_D se pueden presentar dos situaciones. La primera situación se produce cuando dos nodos n_i y n_j están conectados de forma directa, es decir, que $M_C(n_i, n_j) = 1$. En este caso, la distancia entre esos dos nodos será la distancia euclídea entre los centroides de las regiones a las que representan, $M_D(n_i, n_j) = \|Ct(n_i) - Ct(n_j)\|_2$. Es la situación que se da, por ejemplo, entre los nodos 2 y 3 de la Figura 5.14.

Por el contrario, puede ocurrir que dos nodos n_i y n_j no estén conectados de forma directa, es decir, que $M_C(n_i, n_j) = 0$. Sin embargo, sí existe algún camino que atraviesa varios nodos del mapa topológico para llegar de n_i a n_j. En este caso hay que buscar el camino para llegar desde el nodo origen hacia el nodo destino.

Los métodos de búsqueda de caminos son muy variados. Se ha escogido el algoritmo de Dijkstra [Dij59], puesto que calcula el camino óptimo, es decir, el camino de distancia mínima

entre los nodos origen y destino. La complejidad de este algoritmo es $O(n^2)$, siendo n el número de nodos del mapa topológico. A pesar de que la solución es la óptima, se podría pensar que no es la adecuada debido al tiempo de proceso necesario para obtener el camino. Otra posible elección sería el algoritmo heurístico A* [Pea84, Sto00], cuya solución no tiene por qué ser óptima, pero la complejidad es mucho menor que el algoritmo de Dijkstra, $O(n \cdot log(n))$. Sin embargo, para nuestra aplicación el algoritmo de Dijkstra es una opción acertada. El mapa topológico abstrae la configuración del entorno en unas pocas regiones, normalmente en un número inferior a 25 ó 30. En estas condiciones el tiempo que el algoritmo de Dijkstra emplea en calcular los caminos no es crítico en el sistema. Es más, ese tiempo es prácticamente despreciable frente a otras operaciones. De ahí que frente a otros métodos de búsqueda de caminos se haya escogido el que obtiene la solución óptima. Al final, para cada pareja de nodos, n_i y n_j, se almacena una lista ordenada, $L_C(n_i, n_j) = \{n_C(1), n_C(2), \ldots n_C(C)\}$, con el camino que atraviesa C nodos, incluyendo el origen. Si $L_C(n_i, n_j, k)$ es el nodo almacenado en la posición k de la lista $L_C(n_i, n_j)$, la distancia entre los nodos origen y destino será:

$$M_D(n_i, nj) = \sum_{k=1}^{C-1} \|Ct(L_C(n_i, n_j, k+1)) - Ct(L_C(n_i, n_j, k))\|_2 \tag{5.2}$$

Por ejemplo, en la Figura 5.14 los nodos 3 y 5 no están conectados directamente. Para ir del nodo 3 al 5, o viceversa, el camino tiene que pasar por el nodo 4. La lista almacenada es $L_3(3, 5) = \{3, 4, 5\}$. Por lo tanto, a la hora de calcular la distancia entre los nodos 3 y 5 habrá que sumar la distancia entre los nodos 3 y 4, así como la distancia entre los nodos 4 y 5, es decir, $M_D(3, 5) = M_D(3, 4) + M_D(4, 5)$.

3.3 Cálculo de la ruta

Llegados a este punto, se dispone de la lista con los nodos N_a no explorados accesibles desde la posición del robot, $L_{N_a} = \{n_1, n_2, \ldots, n_{N_a}\}$, y la matriz de distancias entre cualquier pareja de nodos, M_D. El paso final consiste en el cálculo de la ruta óptima de exploración completa que pase por todos los nodos una única vez, minimizando la distancia. Esta situación es equivalente al problema del viajante de comercio (*Traveling Salesman Problem*, *TSP*), un problema NP-completo clásico [JP85], cuya formulación data de 1954 [DFJ54].

A continuación se presenta una revisión de los principales algoritmos que se pueden utilizar para resolver el *TSP*, así como la técnica de planificación de rutas que se emplea en esta Tesis.

3.3.1 Algoritmos para la optimización de rutas

Existen numerosas técnicas para la resolución del *TSP*. En función del resultado obtenido se pueden clasificar en dos grandes grupos [Pér04]:

1. Técnicas exactas. Obtienen la ruta óptima, lo que suele requerir un elevado tiempo de

cálculo. Por lo tanto, esta opción está limitada a problemas pequeños.

2. Técnicas heurísticas. Consiguen una solución subóptima o aproximada, que puede ser la ruta óptima requerida. Según Díaz [Día96], un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. El objetivo de estas técnicas es encontrar una solución factible al problema en un tiempo razonable. Estos métodos están avalados por la gran cantidad de bibliografía existente, donde se emplean para resolver problemas de gran dificultad con gran rapidez.

En cuanto a las técnicas exactas, se analiza a continuación lo que hemos denominado búsqueda exhaustiva de la mejor ruta. Respecto a las técnicas heurísticas, se efectúa un repaso de las más comunes: redes neuronales, *simulated annealing*, colonias de hormigas, búsqueda tabú y algoritmos genéticos.

3.3.1.1 La búsqueda exhaustiva El algoritmo de búsqueda exhaustiva de la ruta óptima explora todas las posibles rutas, proporcionando como solución la mejor de todas ellas. Para un conjunto de n nodos, la complejidad es $O(n!)$. Como el nodo de partida es siempre el mismo (el nodo en el que se encuentra el robot), existen $(n-1)!$ rutas posibles. No obstante, de las $(n-1)!$ rutas posibles, solamente $\frac{(n-1)!}{2}$ son distintas, puesto que la otra mitad son rutas con el orden invertido [FS93].

Este algoritmo garantiza la solución exacta, es decir, la ruta que minimiza la distancia. Sin embargo, el tiempo necesario para obtener dicha ruta crece exponencialmente con el número de nodos. Si el número de nodos se hace muy grande, el tiempo necesario para calcular la ruta óptima hace inviable el empleo de la búsqueda exhaustiva para la resolución del problema. Así, por ejemplo, según Medina [Med98], para 37 nodos se tienen $\frac{36!}{2} = 1.86 \cdot 10^{41}$ soluciones. En estas condiciones, considerando que la atmósfera terrestre tiene $100km$ de espesor y 500 millones de km^2 de superficie, sería más fácil encontrar una mota de polvo de $1\mu m$ en la atmósfera de la Tierra que la solución óptima al *TSP* de 37 nodos. Si $n = 100$ existen más soluciones posibles al *TSP* que motas de polvo de $1\mu m$ en el Universo, suponiendo éste como una esfera de 15000 millones de años luz de radio. Por lo tanto, se hace inviable la utilización de la búsqueda exhaustiva para la resolución del *TSP*.

3.3.1.2 Redes neuronales La aplicación de las redes neuronales para resolver el *TSP* se basa, fundamentalmente, en dos aproximaciones diferenciadas [Pér04]:

1. Soluciones basadas en redes de Hopfield [Hop82]. Para una ruta con n nodos, la red de Hopfield consta de n^2 neuronas, organizadas en una tabla de nxn. Se hacen corresponder los puntos fijos de la red con las soluciones óptimas al problema, definiendo una función objetivo que encuadre como una función de energía. Así, la dinámica de la red convergerá a unos valores de los pesos en cada punto fijo de la misma, siendo la solución al *TSP*.

Aunque las redes de Hopfield son una solución para resolver el *TSP*, no son la aproximación más idónea [Smi96, GP95]. Se ha demostrado que la calidad de las soluciones halladas usando la red de Hopfield no se puede comparar con las soluciones proporcionadas con otros métodos heurísticos, en términos del número de mínimos locales en que puede quedar atrapado. Esto es debido a que la red de Hopfield emplea una formulación cuadrática del *TSP*, en vez de una formulación lineal [Smi96].

2. Soluciones basadas en mapas autoorganizativos de Kohonen [Koh82]. Los mapas autoorganizativos (*Self Organizing Maps, SOM*), fueron propuestos por Teuvo Kohonen en 1982. Estos mapas han sido usados para resolver el *TSP* [For88]. La red se especifica con un número de neuronas fijo, igual al número de nodos n a visitar. Los nodos se sitúan en un anillo, donde el orden indica el orden en el que deben ser visitados. Los nodos se colocan inicialmente de un modo aleatorio, presentándose en este orden a la red neuronal durante varias épocas. Sin embargo, esta aproximación tiene algunos inconvenientes. El principal es el riesgo de no convergencia a una solución. Además, incluso en el caso de converger, la solución puede ser bastante pobre. Por estos motivos se han desarrollado otros esquemas basados en los *SOM*, como el que se presenta en [AOA99]. Aun así, se comenta la lentitud del método, algo indeseable si se desea una solución rápida.

Además de estas dos aproximaciones clásicas, existen otros algoritmos basados en redes neuronales para solucionar el *TSP*. Una técnica muy conocida es el *ENA* (*Elastic Net Algorithm*) [AMAM98, vdBG96]. Aunque se adapta mejor al problema que la red de Hopfield, no es la opción adecuada para resolver cualquier formulación del *TSP*. Así, el algoritmo no funciona si debe partir de la matriz de distancias, como sucede en esta Tesis, puesto que es fundamentalmente geométrico.

3.3.1.3 *Simulated annealing* El método del *Simulated annealing* (Recocido simulado) [Kir84], es una técnica empleada para la optimización de problemas combinatorios con mínimos locales, como el *TSP*. En Física, el *annealing* es un proceso termodinámico para obtener estados de baja energía de un sólido en un baño caliente [Hro03]. El proceso de *annealing* se lleva a cabo en dos pasos: i) se aumenta la temperatura del baño hasta que el sólido se funde; ii) se disminuye lentamente la temperatura del baño hasta conseguir un estado de baja energía en el sólido. Este proceso se puede modelar usando métodos de simulación informáticos [MRR$^+$53], siendo el *TSP* uno de los ejemplos típicos de aplicación de la técnica.

Para implementar este algoritmo hay que incluir necesariamente algunos componentes:

- Un criterio de enfriamiento. Consta de: i) una temperatura inicial T_0, que debe ser lo suficientemente alta para poder originar nuevas rutas que minimicen la longitud inicial; ii) una temperatura de parada, T_{fin}; y iii) un criterio de reducción de la temperatura.
- Un operador de intercambio, que permita seleccionar dos nodos de forma aleatoria para ser permutados.

- Un número de intercambios por nivel de temperatura.
- Una función de energía, que es la distancia total de la ruta obtenida como consecuencia de la permutación realizada.

3.3.1.4 Colonias de hormigas Las heurísticas de colonias de hormigas (*Ant Colony Optimization*, *ACO*), se inspiran en el comportamiento de las hormigas naturales. Su utilización parte de la tesis de Marco Dorigo [Dor92]. Una hormiga real puede encontrar el camino más corto entre su nido y una fuente de comida, incluso rodeando un obstáculo, sin usar mecanismos visuales [APV00]. Esta capacidad se debe a la comunicación mediante el rastro de feromona, una sustancia química producida por un animal que afecta a la conducta de otros animales. Al caminar, las hormigas depositan feromonas en el suelo y siguen, en probabilidad, las feromonas depositadas por otras hormigas.

Este esquema se puede aplicar a la resolución del *TSP* [DG97]. Para ello se emplea un conjunto de hormigas artificiales para optimizar la solución. Cada hormiga busca una solución al problema, comunicándose con las demás mediante la feromona que va depositando en la ruta generada. Esta feromona permite a las hormigas seleccionar soluciones mejores en las sucesivas iteraciones del algoritmo, pues eligen con mayor probabilidad las rutas con mayor cantidad de feromonas.

Dentro de los algoritmos de colonias de hormigas se pueden distinguir tres tipos fundamentales: *ant cycle*, *ant density* y *ant quantity*. *Ant cycle* usa información global, es decir, las hormigas dejan una cantidad de feromona proporcional a la calidad de la solución encontrada. De hecho, las hormigas que consiguen rutas más cortas contribuyen con una mayor cantidad de feromona que aquéllas cuya ruta es peor. Por otro lado, tanto *ant density* como *ant quantity* usan información local. Por lo tanto, el algoritmo que mejores resultados proporciona es el *ant cycle* [DMC97].

Comentar también que existen aproximaciones que combinan la técnica de *ACO* con el concepto de temperatura definido en el método *Simulated annealing* [APV00].

3.3.1.5 Búsqueda tabú Los orígenes de la búsqueda tabú (*Tabu search*), se encuentran a finales de los años 70 [Glo77]. La metodología y la definición formal de esta técnica de optimización combinatoria fueron establecidos una década después, a finales de los años 80 [Glo89]. La búsqueda tabú es una técnica para resolver problemas combinatorios de gran dificultad que está basada en principios generales de Inteligencia Artificial [Mar03]. El campo de aplicación de este método es muy amplio, estando el *TSP* incluido dentro de su radio de acción [GL93, HTW95]. En este orden de cosas, se ha demostrado que la búsqueda tabú es una aproximación válida para resolver el *TSP* [Mis04].

La característica distintiva de la búsqueda tabú es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas [GM03]. Su estrategia consiste en evitar que la búsqueda quede atrapada en un mínimo local que no sea global. Para ello toma de la Inteligencia

Artificial el concepto de memoria y lo implementa mediante estructuras simples con el objetivo de dirigir la búsqueda teniendo en cuenta la historia de ésta. El método trata de extraer información de lo sucedido y actúa en consecuencia. Bajo esta perspectiva puede decirse que hay un cierto grado de aprendizaje, siendo la búsqueda inteligente. Normalmente se resume el principio de la búsqueda tabú con la siguiente afirmación [Mar03]:

"Es mejor una mala decisión basada en información que una buena decisión al azar, ya que, en un sistema que emplea memoria, una mala elección basada en una estrategia proporcionará claves útiles para continuar la búsqueda. Una buena elección fruto del azar no proporcionará ninguna información para posteriores acciones."

3.3.1.6 Algoritmos genéticos Los algoritmos genéticos son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Se inspiran en el proceso observado de la evolución natural en los seres vivos. Fueron introducidos por Holland en 1975 [Hol75], el cual afirma que *"se pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional mediante un proceso de evolución simulada."*. Estos algoritmos no garantizan que se encuentre la solución óptima del problema, si bien se ha demostrado empíricamente que las soluciones que se hallan son muy buenas, siendo su coste computacional competitivo en comparación con otros algoritmos de optimización combinatoria.

Charles Darwin formuló en 1859 la teoría de la evolución, según la cual las poblaciones evolucionan en la naturaleza a lo largo de las generaciones de acuerdo con los principios de la selección natural [Dar59]. En la naturaleza, los individuos de una población compiten entre sí en la búsqueda de recursos para sobrevivir. Aquellos individuos con mayor éxito a la hora de sobrevivir y de atraer compañeros/as, tienen mayores probabilidades de generar un gran número de descendientes. Por el contrario, los que menos se adapten, generarían menos descendientes. Por lo tanto, los genes de los individuos mejor adaptados se propagan cada vez más hacia los individuos de generaciones venideras. Esto hace que los descendientes sean superiores genéticamente respecto de sus antecesores, evolucionando hacia características cada vez mejor adaptadas.

Los algoritmos genéticos emplean una analogía directa con el comportamiento de la naturaleza para crear soluciones cada vez mejores a problemas reales, cuya convergencia tiende al óptimo. Así, trabajan con una población de individuos, cada uno de los cuales representa una solución factible al problema. A cada individuo se le asigna un valor relacionado con la calidad de dicha solución. Las mejores soluciones tienen mayor probabilidad de reproducirse. Por ello, a medida que se evoluciona las áreas más prometedoras del espacio de búsqueda son exploradas. De este modo se consigue que generación tras generación los individuos y, por ende, el algoritmo, evolucione hacia la solución óptima.

Un algoritmo genético genérico consta de los siguientes componentes:

- Población inicial de individuos. Se suele escoger de forma aleatoria [Med98].

- Función objetivo (*fitness function*). Es la función que asigna un número real a cada individuo para evaluar la calidad de la solución que representa. Para el caso que nos ocupa, el *TSP*, es la longitud total de la ruta que visita todas las regiones inexploradas que se pueden acceder desde la posición del robot.
- Criterio de selección. Es aquél que se tiene en cuenta a la hora de elegir los individuos padre que van a generar los hijos de la siguiente generación.
- Operador de cruce (*crossover*). Define el proceso de creación de los hijos a partir de los padres, de manera que se transfieran las características deseables de generación en generación.
- Operador de mutación (*mutation*). Introduce un cierto grado de aleatoriedad en la evolución de los individuos para diversificar las soluciones.
- Criterio de reducción. Mantiene una población del mismo tamaño en todas las generaciones.

Los algoritmos genéticos han sido ampliamente usados con éxito para resolver el *TSP*, demostrando la validez de la técnica que proponen [GGRG85, WSF89, Med98, LKM$^+$99, TYTK04]. Incluso se ha hecho notar que son una de las mejores aproximaciones para solucionar el *TSP* [Pér04, TYTK04].

3.3.2 Planificación de rutas de exploración completa

Al hablar de la técnica exacta o búsqueda exhaustiva se comentó que esta solución es inviable cuando el número de nodos crece por encima de un valor no muy grande. Aun así, se ha implementado este método para justificar de manera cuantitativa este dato y tener una medida temporal que pueda compararse con otras técnicas. El algoritmo implementado es recursivo, el cual explora todo el espacio posible de soluciones, $\frac{(n-1)!}{2}$ para n nodos. Tras su finalización se obtiene la ruta óptima de exploración completa. Esta técnica se ha probado en los mapas de 256x256 celdas de la Figura 5.15. Se muestran 3 entornos con 6, 10 y 13 nodos inexplorados accesibles desde la posición del robot, incluyendo aquél donde se encuentra el agente (Figuras 5.15a, b y c, respectivamente). El tiempo aproximado de cómputo para obtener la ruta óptima de exploración completa se puede apreciar en la Tabla 5.2. Las medidas han sido tomadas en un Pentium 3 a 550 MHz con 192 MB de memoria RAM.

Nº Nodos No Explorados	6 nodos	10 nodos	13 nodos
Algoritmo Búsqueda Exhaustiva	100 μs	218 ms	6 m 10 s

Tabla 5.2: Tiempo aproximado de ejecución del algoritmo de búsqueda exhaustiva para obtener la ruta óptima de exploración completa.

A pesar de que la búsqueda exhaustiva obtiene siempre la mejor ruta, de la Tabla 5.2 se deduce su inviabilidad. Cuando se trabaja *on line* es necesaria la obtención de un resultado en

el menor tiempo posible. El algoritmo de búsqueda exhaustiva, por tanto, supeditaría un rápido resultado a la existencia de un número de nodos no muy grande. Así, para una situación real con 13 nodos no explorados (Figura 5.15c), el método emplea más de 6 minutos en calcular la solución. Se ha probado con entornos de 14 nodos no explorados, incrementándose el tiempo de cálculo hasta más de 1 hora. Este hecho descarta cualquier posibilidad de emplear la técnica exacta para resolver nuestro problema. Además, esta decisión se encuentra respaldada porque:

- En un entorno total o parcialmente desconocido no se conoce *a priori* el número de nodos no explorados con los que se va a trabajar.
- En muchas ocasiones no es tan crítico el hecho de conseguir la solución exacta como el tenerla en el menor tiempo posible. Como el algoritmo va a funcionar *on line*, se va a tener que recalcular la ruta a medida que el robot vaya explorando el entorno, puesto que al estar parcialmente inexplorado, el agente deberá ser capaz de reaccionar ante los cambios que se van detectando.

Por todo ello se ha optado por emplear una técnica heurística para resolver el problema. De todas las técnicas que se presentan en la Sección 3.3.1, de partida se descartan las redes neuronales [Smi96, GP95]. Del resto se ha escogido el algoritmo genético, puesto que en la literatura se presenta como una de las mejores aproximaciones al *TSP* [Pér04, TYTK04]. Para corroborar la decisión tomada se ha comparado el algoritmo genético implementado con otras tres técnicas heurísticas, el *Simulated annealing*, las colonias de hormigas y la búsqueda tabú [1]. La comparación se realiza en base a:

- Estándares de comparación, tomados de la librería $TSPLIB$ [Rei91], que incluye *TSP*s de diferente número de ciudades, constituyendo un marco de referencia empleado por los investigadores del problema del viajante. Normalmente el número de nodos inexplorados que aparecen en un entorno no suele ser superior a unos 25 ó 30. De hecho, a lo largo de esta Tesis se suele trabajar con entornos representados mediante mapas métricos de 256x256 celdas. En estas condiciones el número de nodos no explorados es incluso inferior a 20. Por este motivo, se han tomado las rutas *ulysses16* y *ulysses22*, que cuentan con 16 y 22 nodos inexplorados, respectivamente. Aun así, también se han probado los algoritmos con rutas de mayor número de nodos, *dantzig42*, *eilon50* y *berlin52*.
- Entornos simulados y reales en los que navega el robot, con diferente número de nodos no explorados.

La conclusión a la que se llega es que el algoritmo genético es el que mejores resultados proporciona desde el punto de vista de la calidad de la solución cuando el tamaño del problema es pequeño, como es nuestro caso. Este hecho se manifiesta teniendo en consideración tanto los

[1] La implementación de los algoritmos *Simulated annealing*, colonias de hormigas y búsqueda tabú se realiza en [Cañ04], Proyecto Fin de Carrera dirigido por el autor de la presente Tesis.

estándares de comparación como entornos simulados y reales del robot. La técnica del *Simulated annealing* obtiene unos resultados comparables, aunque ligeramente inferiores, al algoritmo genético en cuanto a esa calidad de la solución encontrada. Sin embargo, el método genético implica un tiempo de procesado menor.

El algoritmo genético implementado para encontrar la mejor ruta de exploración completa que se emplea en esta Tesis ha sido aplicado con éxito a nuestro problema [PPB+02, PPU+02a]. A la hora de desarrollar este algoritmo se han tomado las siguientes decisiones:

- Codificación de las regiones. Las dos aproximaciones más utilizadas al trabajar con algoritmos genéticos son la *representación binaria* y la *representación matricial*. En el segundo caso puede ocurrir que los operadores de cruce y mutación generen rutas no válidas [LKM+99]. Por lo tanto, se emplea la *representación por camino*, donde las n regiones que deben ser visitadas se colocan en una lista de n elementos. Así, si la región n_i es el elemento j de la lista, dicha región será visitada en el lugar j.

- Población inicial de individuos. Generada de forma aleatoria [Med98], prestando especial atención a que todos los individuos generados representen una ruta válida. Se ha tomado el número de nodos del problema como el tamaño de la población inicial y un número de generaciones igual a 10000 [TYTK04].

- Función objetivo. Tal y como se ha comentado previamente, es la longitud total de la ruta representada por un individuo que visita todas las regiones inexploradas que se pueden acceder desde la posición del robot.

- Criterio de selección. Se ha optado por escoger el método de selección por torneo de los individuos que se van a cruzar para generar un individuo de la siguiente generación. Consiste en seleccionar al azar un conjunto potencial de individuos padre, de los cuales se toma el mejor de todos. De esta forma se utiliza una versión probabilística que permite la selección de los padres sin que necesariamente sean los mejores. Se ha descartado la selección elitista porque tiende a converger rápidamente hacia óptimos locales.

- Operador de cruce. Debido a que se emplea la *representación por camino*, los operadores de cruce clásicos no son válidos. Se usa el operador de cruce denominado *order crossover* [Dav85], al ser el que mejores resultados consigue en nuestro caso. Una vez seleccionados el primer padre, $Padre_i$, y el segundo, $Padre_j$, el operador funciona del siguiente modo para obtener los individuos hijo, $Hijo_k$ e $Hijo_l$:

  - Escoger dos posiciones aleatorias, $pos_i(1)$ y $pos_i(2)$ en $Padre_i$ y $pos_j(1)$ y $pos_j(2)$ en $Padre_j$.
  - Copiar en $Hijo_k$ los valores de $Padre_i$ entre $pos_i(1)$ y $pos_i(2)$, ambas inclusive.
  - Seleccionar y almacenar en $Hijo_k$, a partir de $pos_i(2)$, los valores de $Padre_j$ que todavía no están almacenados en $Hijo_k$.

- $Hijo_l$ se genera del mismo modo, considerando $Padre_j$ como el primer padre y $Padre_i$ como el segundo.

- Operador de mutación. Al igual que en el caso de los operadores de cruce, la *representación por camino* fuerza el uso de operadores de mutación que no son los clásicos. En nuestro caso no se utiliza operador de mutación [SMM+91].

- Criterio de reducción. Una vez obtenidos los individuos descendientes de una determinada población, la reducción consiste en seleccionar de manera elitista la población de los mejores individuos cuyo tamaño sea igual al de la población inicial.

El resultado de la aplicación del algoritmo genético a nuestro *TSP* es una lista ordenada de nodos, $OL_{N_a} = \{n_{o1}, n_{o2}, \ldots, n_{oN_a}\}$, que indica el orden en el que deben ser visitadas las zonas inexploradas del entorno. Esta ruta se envía a la capa *InterfazRobot* para que el módulo *ControlRobot* sepa en todo momento el siguiente punto de destino. Tal y como se ha comentado previamente, el robot seguiría la ruta de un modo reactivo mediante los comportamientos del Capítulo 3.

La Figura 5.15 muestra el resultado obtenido tras la aplicación del método propuesto a tres entornos distintos modelados mediante un mapa métrico de 256x256 celdas. El robot se supone situado en la posición $(128, 128)$, siendo los parámetros empleados $Obst\_Inc = 1$, $A_{min} = 16$ y $Dist\_Max = 40$. Para los tres entornos se presenta el mapa topológico resultante sobre el mapa métrico de partida. También se puede observar el orden en el que deberían visitarse las regiones según la solución proporcionada por el algoritmo. Siempre se parte de la zona en la que está el robot, la región etiquetada con el número 1. El tamaño del *TSP* en las Figuras 5.15a, b y c es de 6, 10 y 13 nodos, respectivamente. En los tres casos el algoritmo propuesto obtiene la solución óptima, la misma que la técnica de búsqueda exhaustiva. Sin embargo, el tiempo necesario es mucho menor. Por ejemplo, la Figura 5.15c muestra un entorno prácticamente inexplorado. El mapa topológico consta de 15 nodos, 13 de los cuales (incluyendo la región en la que se encuentra el agente), están inexplorados. Partiendo de la posición del robot, la ruta óptima debería explorar la parte sur del mapa métrico, a continuación la parte oeste y, finalmente, la parte norte.

3.4 Prestaciones temporales

En cuanto a las prestaciones temporales del algoritmo propuesto, la Tabla 5.3 presenta un resumen del tiempo medio empleado. Las medidas, nuevamente, han sido realizadas en un Pentium 3 a 550 MHz con 192 MB de memoria RAM. Los entornos considerados son los de la Figuras 5.15a, b y c, cuyo mapa topológico consta de 29, 24 y 15 nodos, respectivamente. Así mismo, el número de nodos inexplorados es de 6, 10 y 13, respectivamente. Los parámetros empleados por el método son $Obst\_Inc = 1$, $A_{min} = 16$ y $Dist\_Max = 40$. El robot está situado, en los tres casos, en la posición $(128, 128)$. Los errores se han calculado sobre un intervalo de confianza del 95%.

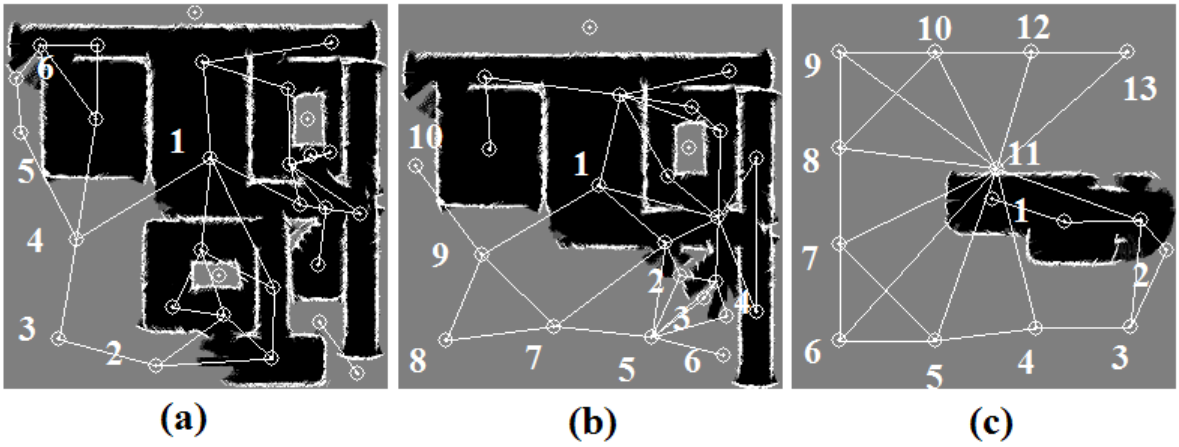

Figura 5.15: Ruta de exploración completa en mapas de 256x256 celdas: (a) 6 nodos inexplorados; (b) 10 nodos inexplorados; (c) 13 nodos inexplorados.

El mapa métrico de partida presenta un grado diferente de exploración para los tres entornos. Para todos ellos, la construcción del mapa topológico tarda un tiempo similar, alrededor de $120ms$. Este tiempo es similar al que se presenta en la Tabla 5.1. Esto nos sugiere que la construcción de la representación topológica es independiente de la complejidad del entorno y del grado de exploración del mismo. En cuanto al tiempo necesario para obtener los nodos no explorados accesibles, decir que es ínfimo, pudiendo ser despreciado. Sin embargo, el tiempo para calcular la matriz de distancias ya es bastante superior, del orden de varios milisegundos. Aun así, a pesar de usar el algoritmo de Dijkstra para el cálculo de la matriz, este tiempo es mucho menor que el que se necesita para construir el mapa topológico y para obtener la ruta óptima. A este respecto, comentar que el tiempo consumido en conseguir la ruta óptima aumenta con el número de nodos del *TSP*. No obstante, incluso para un mapa de partida prácticamente inexplorado (Figura 5.15c, con 13 nodos no explorados), el algoritmo encuentra la mejor solución en tan solo $133ms$. Además, esta solución es exactamente la misma que la proporcionada por la técnica exacta, pero en un tiempo muy inferior (ver Tabla 5.2). El tiempo total empleado en esta situación es de $259ms$, lo que supondría que el robot recorrería $5.2cm$ a una velocidad de $200mm/s$ antes de recalcular una nueva ruta. Por lo tanto, la propuesta es válida incluso en el caso de que aparezcan obstáculos inesperados o móviles, pues el robot podría calcular una nueva ruta antes de impactar con el obstáculo.

Nº Nodos No Explorados	6 nodos	10 nodos	13 nodos
Total Mapa Topológico	$124.7 \pm 1.1\ ms$	$125.0 \pm 1.2\ ms$	$120.9 \pm 1.7\ ms$
Nodos No Explorados	$6.89 \pm 0.12\ \mu s$	$10.29 \pm 1.64\ \mu s$	$5.21 \pm 0.19\ \mu s$
Matriz de Distancias	$2.67 \pm 0.47\ ms$	$2.31 \pm 0.04\ ms$	$4.57 \pm 1.25\ ms$
TSP	$58.97 \pm 0.79\ ms$	$97.57 \pm 6.24\ ms$	$133.54 \pm 2.03\ ms$
Total	$186.39 \pm 1.40\ ms$	$224.93 \pm 6.30\ ms$	$259.00 \pm 2.82\ ms$

Tabla 5.3: Tiempo de ejecución del algoritmo de exploración completa.

4 Experimentos y resultados

En este apartado se presentan los resultados obtenidos que demuestran la validez del método para la planificación de rutas de exploración completa. Todas las pruebas se han efectuado en la misma máquina, un Pentium 3 a 550 MHz con 192 MB de memoria RAM. En los mapas métricos que sirven de base al método, las zonas libres se muestran en negro, los obstáculos en blanco y las áreas no exploradas en gris.

El algoritmo se ha aplicado tanto a entornos simulados como a entornos reales. Las pruebas simuladas se han realizado con el simulador de la plataforma robótica *Nomad 200* de *Nomadics*. Las pruebas en entornos reales se han llevado a cabo empleando la plataforma robótica real *Pioneer P2AT*.

La técnica propuesta en esta Tesis sirve, además de para planificar rutas de exploración completa para visitar las regiones no exploradas de un entorno total o parcialmente explorado, para planificar cualquier otra ruta que cumpla determinados criterios. Lo único que haría falta es seleccionar los nodos de acuerdo a dichos criterios, que incluso pueden ser muy variados. Así, se podrían seleccionar todos los nodos del modelo para visitar todo el entorno en tareas de vigilancia. También se podrían escoger una serie de nodos explorados, de manera que el agente se dirigiera hacia unos lugares predeterminados del entorno. Igualmente, el método que se propone también se puede aplicar a la planificación de tareas en entornos multiagente que implicaran la navegación de forma eficiente.

A continuación se presentan los resultados de las pruebas de planificación de rutas de exploración completa tanto en entornos simulados como reales. Posteriormente también se demuestra la validez del método para planificar rutas de exploración que dirijan el robot hacia un conjunto determinado de zonas ya exploradas, mediante un ejemplo de planificación multiagente en un entorno hospitalario.

4.1 Entornos simulados

La Figura 5.16 presenta diferentes etapas del proceso de exploración de un entorno simulado. El objetivo de esta prueba es comprobar la validez del método para la generación de rutas de exploración completa de un entorno durante distintos instantes del proceso de exploración. No es un objetivo el presentar el seguimiento de la ruta, que se efectuaría con los comportamientos reactivos del Capítulo 3. La planta del entorno se observa en la Figura 5.16a. Se modela mediante un mapa métrico de 256x256 celdas. En las Figuras 5.16b, c, d y e se puede apreciar la ruta de exploración completa propuesta en distintos instantes de la exploración. Para las 4 etapas que se presentan se muestra el mapa topológico que se construye sobre el mapa métrico. Así mismo, se indica el orden en el que se deberían visitar las regiones no exploradas del entorno, partiendo desde la posición del robot (región 1).

En la Figura 5.16b se encuentra explorada la parte sureste del entorno. Con este grado de

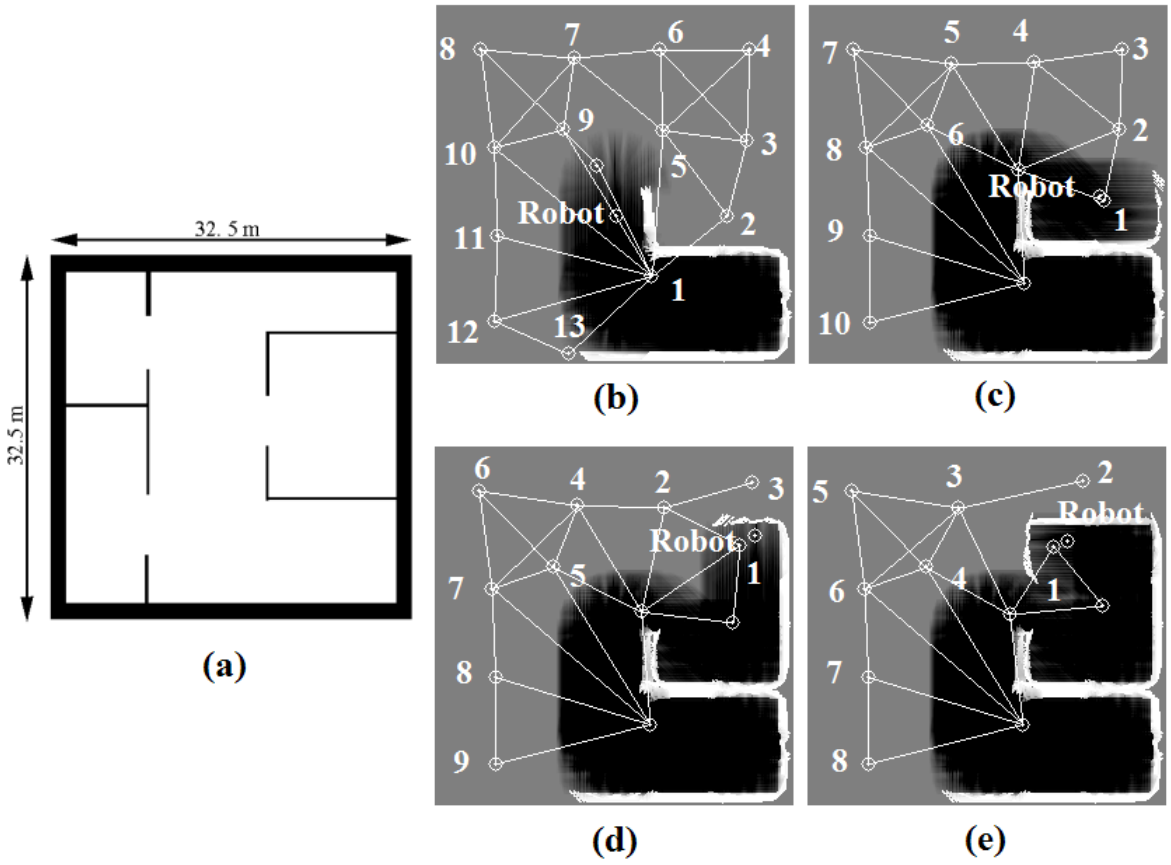

Figura 5.16: Cálculo de la ruta de exploración completa en un entorno simulado: (a) planta; (b) etapa 1 de la exploración; (c) etapa 2 de la exploración; (d) etapa 3 de la exploración; (e) etapa 4 de la exploración.

exploración, la ruta de exploración completa propuesta debería explorar la parte este, la parte norte y la parte oeste del entorno, siguiendo este orden. Así, el robot se dirige hacia la zona este del entorno, comenzando por la siguiente región proporcionada por el algoritmo (región 2). Al alcanzar la zona este del entorno, el robot detecta la pared derecha del mismo, por lo que se recalcula una nueva ruta (Figura 5.16c). En realidad, se construye un nuevo mapa topológico y, por ende, se obtiene una nueva ruta, cada vez que el mapa métrico cambia significativamente, cuando el número de celdas del mapa que varía supera el umbral $U_{metrico}$ (ver Sección 3.3.1 del Capítulo 2). La nueva ruta de exploración completa que se propone en esta etapa del proceso de exploración tiende a finalizar la exploración de la zona este, continuando por la zona norte y la zona oeste. Como podemos observar, este resultado es coherente con la ruta propuesta en la anterior etapa de exploración (Figura 5.16b), apreciándose también la constancia del mapa topológico a pesar de la exploración progresiva. Por lo tanto, el robot enfilaría hacia la región 2 de la ruta presentada en la Figura 5.16c. Nuevamente se detecta un cambio significativo en el mapa métrico al detectar la pared norte de la habitación central de la zona este (Figura 5.16d), superando el número de celdas que ha variado el umbral $U_{metrico}$. Se obtiene una nueva ruta de exploración, la cual sigue manteniendo el mismo esquema que las anteriores. Se propone

continuar el proceso por la zona este, a continuación la zona norte y, finalmente, la zona oeste. Por último, en la Figura 5.16e se presenta una nueva ruta de exploración cuando la última pared de la habitación que se estaba explorando es detectada. La ruta obtenida por el algoritmo corrobora el mismo esquema, primero la zona norte y después la zona oeste. La exploración continuaría hasta que se tuviera conocimiento de todo el entorno.

Es necesario remarcar que el seguimiento de las rutas calculadas se llevaría a cabo con los comportamientos reactivos implementados (ver Capítulo 3). Además, en todos los casos el algoritmo genético implementado obtiene la ruta óptima, pero en un tiempo enormemente inferior al algoritmo de búsqueda exhaustiva cuando el número de nodos crece por encima de unos 10. De hecho, en la Figura 5.16b habría sido imposible el cálculo de la ruta óptima con la técnica exacta, puesto que al ser el *TSP* de 13 nodos, se necesitarían más de 6 minutos (Tabla 5.2).

4.2 Entornos reales

Al igual que cuando se trabaja con entornos simulados, el objetivo de las pruebas con entornos reales es comprobar la efectividad del método propuesto para la obtención de rutas de exploración completa en distintos instantes del proceso de exploración. Tal y como se ha comentado previamente, el seguimiento de la ruta se realizaría con los comportamientos reactivos del Capítulo 3.

La Figura 5.17 presenta la planta de dos entornos reales de prueba. El entorno real 1 (Figura5.17a), consta de una habitación de aproximadamente $5.5x5.5m^2$ y un pasillo de $3m$ de ancho con varias columnas. La habitación y el pasillo están conectados a través de una puerta de $1.5m$ de ancho. Por otro lado, el entorno real 2 de la Figura 5.17b consta de varios pasillos conectados de diferentes longitudes y anchuras. Ambos entornos se modelan con mapas métricos de 256x256 celdas.

En la Figura 5.18 se pueden observar las rutas de exploración completa calculadas en distintos instantes de la exploración del entorno de la Figura 5.17a. En este caso se parte de un conocimiento preciso de la habitación de este entorno, generándose la primera ruta de exploración completa (Figura 5.18a). En esta primera etapa se aprecia que la habitación se modela con un único nodo del mapa topológico, cubriendo el resto de nodos el exterior de esta habitación. Todos los nodos del mapa topológico excepto el de la habitación están inexplorados. Sin embargo, el robot se encuentra en la habitación. De este modo, la ruta de exploración completa incluye todas las regiones del entorno. La ruta sugerida exploraría el exterior de la habitación, partiendo hacia el norte, después el oeste y finalmente el sur del entorno. Al comenzar la exploración del pasillo se genera una nueva ruta, la cual concuerda con las premisas establecidas en la etapa anterior (Figura 5.18b). En este caso ya existen nodos explorados en el mapa topológico, por lo que la ruta de exploración completa no incluye todas las regiones, sino únicamente las no exploradas. El robot debe dirigirse hacia el norte, posteriormente hacia el oeste y finalmente al sur para completar la exploración de las zonas no exploradas del entorno.

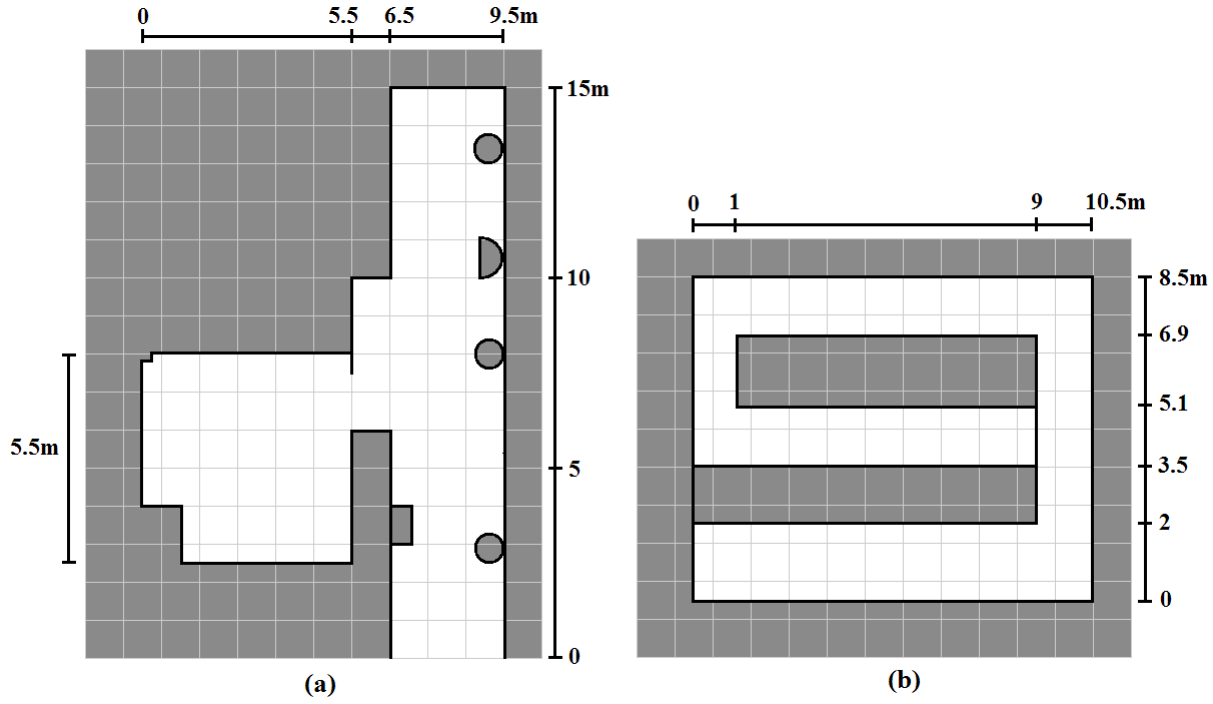

Figura 5.17: Planta de los entornos reales de prueba: (a) entorno real 1; (b) entorno real 2.

Al llegar el robot a la posición de la Figura 5.18c el mapa métrico ha cambiado notablemente, pues el número de celdas que ha variado supera el umbral $U_{metrico}$ (ver Sección 3.3.1 del Capítulo 2). Por lo tanto, se obtiene una nueva ruta. No obstante, este nueva ruta no varía la secuencia de exploración propuesta en las etapas anteriores. La ruta calculada se dirige hacia el norte, el oeste y, por último, el sur. En la Figura 5.18d el agente ha detectado la pared norte del pasillo, por lo que ya no puede continuar hacia el norte, recalculando la ruta de exploración completa. Ya no se sugiere continuar hacia el norte, puesto que está completamente explorado. Por el contrario, la ruta sugiere ir hacia el oeste (región 2), continuando por el sur. Sin embargo, como se observa en la Figura 5.17a, la zona oeste del mapa métrico no tiene su correspondencia física real en el entorno. Debido a que el mapa métrico no está suficientemente conformado, no se detecta esta situación. Al continuar el proceso de exploración, el mapa métrico se define por completo, obteniéndose una nueva ruta (5.18e). Con este grado de exploración se observa que el mapa topológico refleja la realidad del entorno. La zona oeste ya no es accesible desde el norte del pasillo, pues no existe esta conexión en la representación topológica. Por lo tanto, la ruta propuesta dirige el agente hacia el sur, para finalizar la exploración del pasillo.

El algoritmo genético es capaz de obtener para este primer entorno real la ruta óptima en todos los casos. Nuevamente debemos destacar que no habría sido posible emplear la técnica de búsqueda exhaustiva pues el número de nodos del *TSP* es de 13 (Figuras 5.18a y b), ó 12 (Figuras 5.18c, d y e), lo cual hace que el tiempo de proceso se dispare, imposibilitando su utilización práctica.

Para el entorno real 2 se presentan dos etapas del proceso de exploración (Figura 5.19). Se

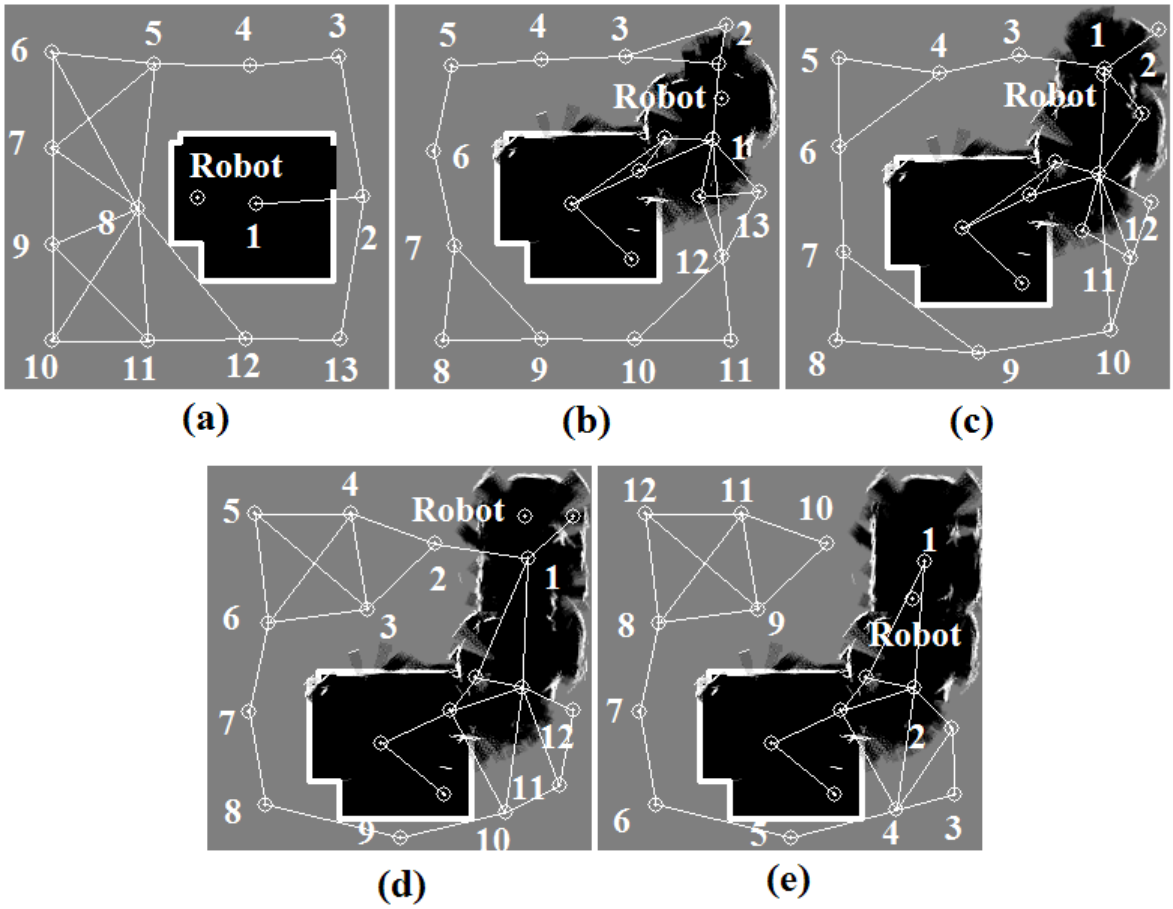

Figura 5.18: Cálculo de la ruta de exploración completa en el entorno real 1: (a) etapa 1 de la exploración; (b) etapa 2 de la exploración; (c) etapa 3 de la exploración; (d) etapa 4 de la exploración; (e) etapa 5 de la exploración.

parte de un mapa en el que los pasillos norte y este están completamente explorados, y el pasillo sur en su mayoría. El robot se encuentra en el centro del pasillo oeste, que es más corto que el resto. La ruta que se propone pretende explorar el pasillo central y después la zona suroeste y sur del entorno, aunque estas dos últimas áreas del mapa métrico no representan zonas reales del mismo (Figura 5.17b). Al continuar el proceso se llega a la situación de la Figura 5.19b, donde el mapa métrico ha variado profundamente (se supera el umbral $U_{metrico}$), calculándose otra nueva ruta. El algoritmo propone finalizar la exploración del pasillo central. Sin embargo, para dirigirse hacia la parte sur y oeste del entorno debe atravesar los pasillos oeste, norte, este y sur. Esto ocurre porque, debido a que el proceso de exploración ha avanzado, no existe conexión directa entre el pasillo central y la parte suroeste del entorno a través del mapa topológico.

Nuevamente, el método heurístico genético calcula la ruta óptima. En este caso, hay únicamente 10 y 6 nodos no explorados en las Figuras 5.19a y b, respectivamente. Este hecho permitiría emplear el algoritmo de búsqueda exhaustiva, pues su tiempo de proceso sería bajo (Tabla 5.2). Sin embargo, como *a priori* no se conoce el número de nodos no explorados, esta solución debe ser descartada en previsión de que aparezcan más nodos de los que la técnica

exacta puede tratar de forma rápida. Más aún, cuando se ha demostrado que el algoritmo genético empleado obtiene una ruta óptima o subóptima de forma rápida.

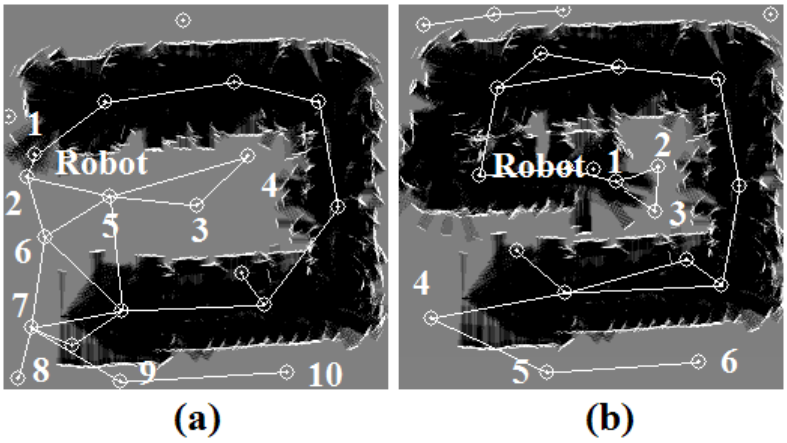

Figura 5.19: Cálculo de la ruta de exploración completa en el entorno real 2: (a) etapa 1 de la exploración; (b) etapa 2 de la exploración.

4.3 Un caso de aplicación: planificación en un entorno hospitalario

La programación de las tareas de médicos y enfermeras en un hospital es una tarea compleja. En un entorno de esta índole los miembros del personal deben llevar a cabo varias tareas en lugares diferentes. Un problema típico es el orden en el que estos lugares deben ser visitados para trabajar eficientemente. Aunque la experiencia del personal proporciona un conocimiento valioso para organizar el trabajo, las situaciones inesperadas o urgentes hacen que no siempre se pueda seguir el plan establecido. En estos casos, sería útil y deseable tener algún mecanismo de programación de tareas capaz de proporcionar la mejor planificación de las mismas de manera que se visitaran todos los lugares establecidos de la forma más rápida posible.

El problema anterior se puede resumir como, dado un conjunto de lugares, así como la distancia entre estos lugares, encontrar el orden en el que deben ser visitados para conseguir la máxima eficiencia. Este planteamiento es similar al realizado en la Sección 3.3, equivalente al *TSP*. Por lo tanto, la técnica presentada en este capítulo es aplicable a la resolución del problema. Los resultados de su aplicación se presentan en [UPA+03a], donde se planifican tareas en la tercera planta de la *Fondazione IRCCS Santa Lucia* de Roma.

Para poder aplicar el método necesitamos un modelo del entorno donde se va a trabajar. El punto de partida de este modelo es el mapa de la tercera planta del hospital [2] que se muestra en la Figura 5.20a. El mapa original no es válido por varias razones: i) las habitaciones están etiquetadas con su número, de la 301 a la 327; y ii) el espacio libre está representado en color blanco, lo que se correspondería con una probabilidad de ocupación de una zona ocupada. Por lo tanto, este mapa necesita ser procesado antes de poder aplicar nuestro método de planificación de

[2] Proporcionado por la *Fondazione IRCCS Santa Lucia* de Roma.

rutas. En primer lugar, eliminamos manualmente la etiqueta numérica de todas las habitaciones. A continuación aplicamos un procesado de la imagen hasta tener una representación útil (Figura 5.20b), basado en tres fases encadenadas: i) un filtro gaussiano de definición; ii) un aumento del contraste por estiramiento del histograma; y iii) una inversión del color de la imagen. Al finalizar este proceso se dispone de un mapa métrico útil de 1024x1024 celdas donde el espacio libre está representado en negro, correspondiéndose físicamente con los pasillos y las dependencias de la tercera planta del hospital.

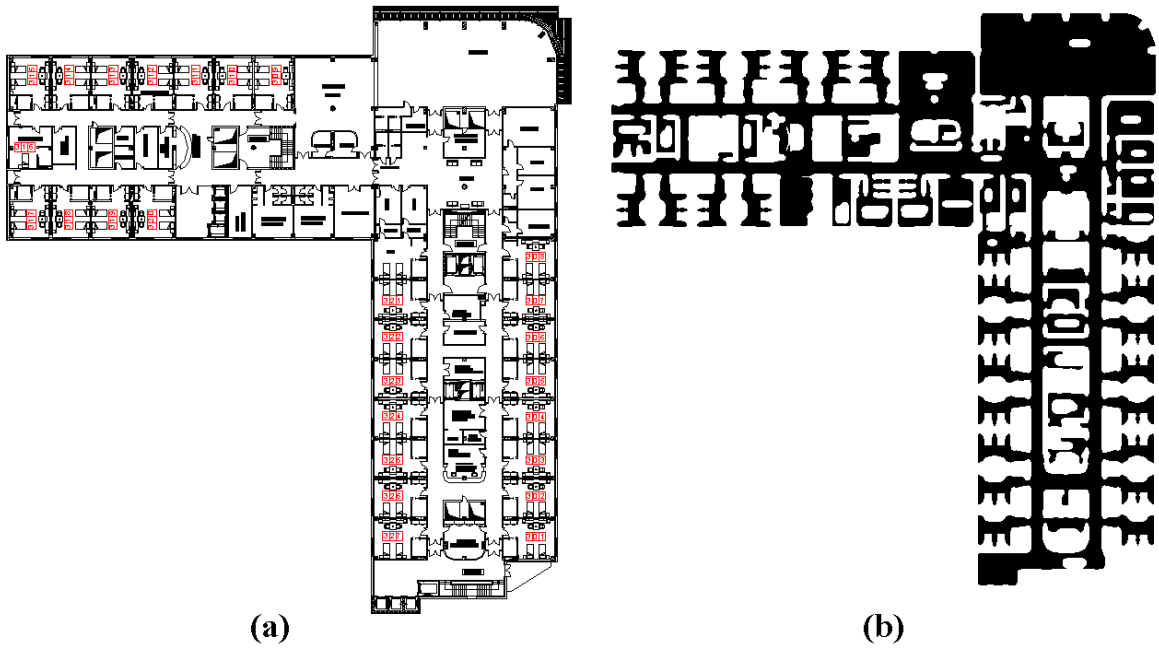

Figura 5.20: (a) Mapa original del hospital; (b) mapa métrico obtenido al procesar el mapa original del hospital.

El mapa topológico se construye a partir de un mapa métrico completamente explorado de 1024x1024 celdas. El resultado de la construcción de la representación topológica se muestra sobre el mapa de la tercera planta del hospital en la Figura 5.21. Se observa que aparece un nodo en cada una de las 27 habitaciones de la planta, de la *H301* a la *H327*. Así mismo, aparecen nodos en o muy cerca de otras dependencias significativas de la planta: Habitaciones Botiquín 1 y 2 (*HB1* y *HB2*), Habitaciones de las Enfermeras 1 y 2 (*HE1* y *HE2*), y la Habitación de los Médicos (*HM*). También aparecen nodos del mapa topológico en los pasillos, que sirven de unión entre las habitaciones. A la vista de la Figura 5.21 se aprecia la validez de la representación métrico-topológica, puesto que está implícitamente adaptada a la geometría del entorno. En este caso la entrada del *TSP* no consistirá en los nodos inexplorados del entorno, sino en un conjunto de nodos asociados a determinadas estancias de la tercera planta del hospital. En cuanto a la matriz de distancias, se calcula exactamente igual que en la Sección 3.2, pero considerando todos y cada uno de los nodos del mapa topológico. De este modo se conoce la relación de distancia entre cualquier pareja de nodos. Este hecho añadirá versatilidad, puesto que se podrán planificar rutas de cualquier tamaño que visiten cualquier conjunto de habitaciones del hospital.

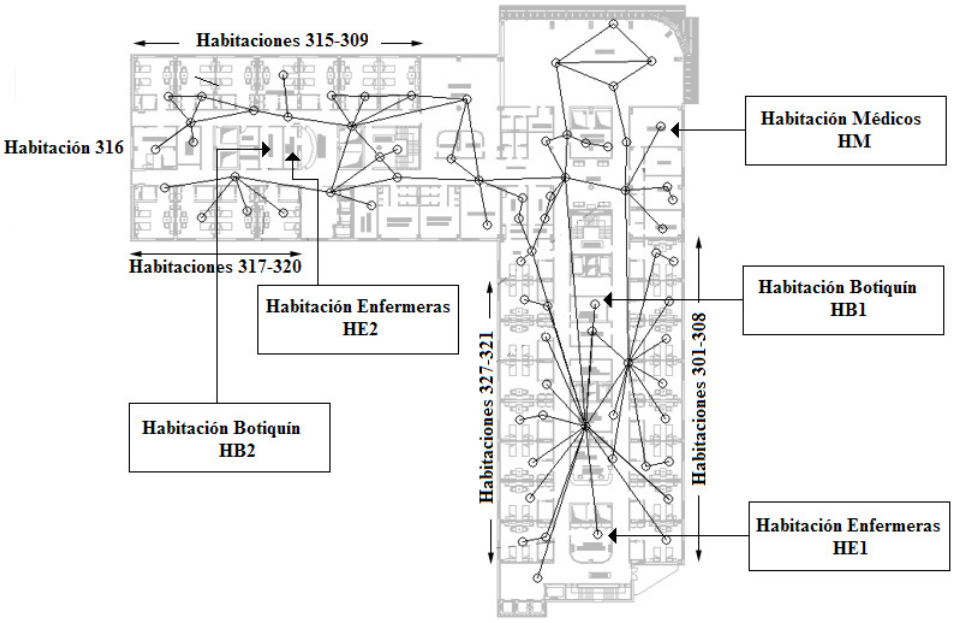

Figura 5.21: Mapa topológico superpuesto al mapa original del hospital, junto con la situación de varias habitaciones importantes.

En los experimentos llevados a cabo el agente móvil es el personal del hospital. En las situaciones de urgencia, el sistema debe tratar con más de un agente. En esos casos el algoritmo planifica las mejores rutas para cada uno de ellos de manera que se visiten los lugares prefijados una única vez. Inicialmente el *TSP* es resuelto para todos los agentes implicados de manera que todos ellos visiten todos los lugares. Con estas soluciones se determina qué agente alcanza alguno de los lugares antes que nadie, por ejemplo el agente A_1. Se preserva, por tanto, la ruta para este agente. Sin embargo, el *TSP* es resuelto nuevamente para todos los agentes tras eliminar el destino alcanzado por A_1. Hay que resaltar que todo este procesado se realiza *a priori*, cuando ninguno de los agentes ha comenzado a seguir la ruta. Esta operación se efectúa tantas veces como sea necesario hasta que todos los agentes alcanzan el destino final.

Supongamos un escenario en el que sucede una urgencia nocturna cuando hay únicamente dos enfermeras y un médico. La primera enfermera, *Enfermera1* se encuentra en la habitación *H307*, mientras que la segunda, *Enfermera2* está en la habitación *H324*. El médico, *Doctor*, está en esos momentos en la habitación *H314*. La urgencia es detectada en la habitación *H312*. Todo el equipo (desfibrilador y equipo de emergencia), se supone situado en *HB1*. Los expedientes de los pacientes se encuentran en *HE2* por la noche. Por lo tanto, una ruta completa para una única persona incluiría su posición, la Habitación Botiquín 1 (*HB1*), la Habitación de Enfermeras 2 (*HE2*) y la habitación 312. En la Tabla 5.4 se presenta el resultado de la planificación de la ruta, junto con el coste asociado. Para cada una de las ejecuciones del algoritmo, los lugares que finalmente deben ser visitados por algún miembro del personal están en negrilla. La *Enfermera1* es la persona más cercana a *HB1* y llegaría en primer lugar (Ejecución 1 de la Tabla 5.4), por lo que ni el *Doctor* ni la *Enfermera2* van a esta estancia. Cuando se recalculan las rutas teniendo

en cuenta este dato (Ejecución 2), el *Doctor* está más cerca de *HE2*, donde se encuentran los expedientes. Por lo tanto, ni la *Enfermera1* ni la *Enfermera2* visitan *HE2* y van directamente a la habitación 312. Las rutas finales para los miembros del personal son *HE2*, *H312*, *HB1*, *H312* y *H312* para el *Doctor*, la *Enfermera1* y la *Enfermera2*, respectivamente. El algoritmo optimiza la ruta rápidamente, consiguiendo: i) que todas las estancias se visiten una única vez; y ii) que al final el *Doctor*, la *Enfermera1* y la *Enfermera2* estén en la habitación 312.

	Ejecución 1	Coste Total
Doctor	HB1(675), HE2(496), H312(113)	1284
Enfermera1	**HB1(86)**, HE2(496), H312(113)	695
Enfermera 2	HB1(120), HE2(496), H312(113)	729
	Ejecución 2	Coste Total
Doctor	**HE2(180)**, H312(113)	293
Enfermera1	**HB1(86)**, HE2(496), H312(113)	695
Enfermera 2	HE2(522), H312(113)	634
	Ejecución 3	Coste Total
Doctor	**HE2(180) H312(113)**	293
Enfermera1	**HB1(86) H312(608)**	694
Enfermera 2	**H312(634)**	634

Tabla 5.4: Lugares visitados por el personal, así como el coste durante la situación de emergencia.

La situación planteada consiste en un *TSP* de pocos nodos, aunque se resolvería del mismo modo si hubiera que visitar más estancias. Debido a que el algoritmo implementado es muy rápido, la ruta óptima para cada uno de los agentes se calcula en muy poco tiempo, incluso en el caso de que hubiera que ejecutar varias iteraciones hasta alcanzar la solución final. De este modo, se consigue una gestión eficiente y coordinada de las urgencias en el hospital.

5 Conclusiones

En este capítulo se ha presentado un algoritmo para la planificación eficiente de rutas de exploración completa de un entorno total o parcialmente desconocido. La técnica se encuadra dentro del nivel de Navegación por Inspección, el más alto de la Jerarquía de Navegación.

El método está soportado por una estructura jerárquica que integra las dos paradigmas más utilizados a la hora de representar y modelar un entorno, el métrico y el topológico. De este modo se aprovechan las ventajas de cada uno de ellos. El método de integración métrico-topológico se basa en una estructura piramidal que divide el entorno en regiones, representando explícitamente las zonas no exploradas. Por lo tanto, se puede emplear para planificar rutas a través de áreas desconocidas.

La estructura piramidal se construye a partir de un mapa métrico probabilístico. En una primera etapa el mapa métrico es umbralizado y se realiza un crecimiento de obstáculos para que la pirámide tenga una base apropiada. A partir de este mapa preprocesado se construyen los distintos niveles de la jerarquía, consiguiendo al final un mapa topológico que abstrae la

información del entorno en regiones y las conexiones entre ellas. Una de las grandes ventajas del mapa topológico presentado es que contiene a las zonas no exploradas, por lo que es posible planificar rutas incluyendo estas regiones. Así mismo, el tiempo empleado en su construcción no depende de la complejidad ni del grado de exploración del entorno, sino del tamaño del mapa. Se ha comprobado que el tiempo necesario para su construcción con mapas métricos del tamaño que normalmente se usan para modelar el entorno es muy pequeño.

La planificación de las rutas de exploración completa se basa en la estructura presentada. Un paso previo necesario es la obtención de las regiones desconocidas accesibles desde la posición del robot, así como la distancia entre estas regiones. Con estos datos de entrada el problema es similar al *TSP*. Debido a que el número de nodos en un entorno arbitrario no es conocido *a priori*, el problema se ha resuelto mediante una técnica heurística. Entre todas las opciones se ha escogido un algoritmo genético, pues es el que mejores resultados ha dado en comparación con otros métodos. Este algoritmo obtiene una ruta subóptima de exploración completa en un tiempo razonable, lo que permitiría recalcularla en el caso de que aparecieran obstáculos inesperados. Con el objetivo de realizar una comparación temporal también se ha implementado una técnica exacta de búsqueda de la ruta óptima. Se ha demostrado que este método no es viable cuando el número de nodos del *TSP* se incrementa por encima de un valor no muy elevado.

La técnica ha sido probada tanto en entornos simulados como reales. En ambos casos se ha demostrado la validez de la propuesta, ya que las rutas planificadas cubren de forma óptima todas las zonas inexploradas de los entornos de prueba. Además de verificar este hecho, se ha mostrado que son coherentes para un mismo entorno con distintos grados de exploración. Por último, también se presenta una aplicación de la técnica propuesta en un entorno hospitalario. Además de planificar rutas que visiten varios lugares de una forma rápida, también se ha demostrado su aplicación a la gestión eficiente y coordinada de urgencias en un hospital.

Capítulo 6

Resultados

En este capítulo se presentan los resultados que se obtienen tras la integración de todos los módulos que forman parte del sistema para la exploración completa basada en comportamientos de entornos total o parcialmente desconocidos, tanto sobre un simulador como sobre una plataforma física, en este caso un *Pioneer P2AT*.

Los resultados en entornos simulados se presentan en la Sección 2, mientras que los resultados en escenarios reales se describen en la Sección 3. Antes de revisar estos resultados se analiza en la Sección 1 el entorno de pruebas y los preparativos necesarios para que el funcionamiento sea adecuado. Por último, en la Sección 4 se comentan las conclusiones que se extraen.

1 Entorno de pruebas

En la Figura 6.1 se presenta de forma esquemática el entorno de pruebas utilizado para el sistema de exploración completa. Todas las transferencias de información se realizan a través de *IP* en una red de área local. Sin embargo, existen variaciones en el esquema según se trabaje con el simulador de la plataforma robótica *Nomad 200* de *Nomadics* o con la plataforma robótica real *Pioneer P2AT*. En el primer caso, el simulador trabaja en la máquina *PC_Simulador*, estando conectada por cable a la red (Figura 6.1a). En el segundo, el robot real se conecta a la red a través de una conexión inhalámbrica (*WIFI*), para así permitir que el agente se mueva libremente durante la exploración sin tener una conexión física que limite su movilidad (Figura 6.1b). En ambos casos la máxima velocidad de traslación se fija a $v_{tMAX} = 100mm/s$ y la máxima velocidad de rotación a $v_{rMAX} = 15^o/s$. Así mismo, y como ya se ha comentado anteriormente, las plataformas robóticas constan únicamente de sensores sonar y de un sistema de posicionamiento basado en odometría. En concreto, la plataforma simulada consta de un anillo de 16 sensores equiespaciados y la plataforma robótica real consta de 8 sensores frontales. Es en este último caso donde se presentan los efectos reales, debidos fundamentalmente a los errores de los sensores sonar. El uso de las simulaciones permite estudiar el comportamiento del sistema en ausencia de errores mecánicos y sensoriales. Por lo tanto, constituye una base para los experimentos reales.

Tanto cuando se realizan pruebas simuladas como cuando éstas se efectúan con el robot real, todos los módulos del sistema (ver Figura 2.1), se ejecutan en la misma máquina, *PC_Módulos*. Esta máquina es un Pentium 3 a 550 MHz con 192 MB de memoria RAM, escogida a propósito para demostrar que el sistema no es computacionalmente muy exigente. A pesar de que es una máquina de bajas prestaciones, los módulos del sistema funcionan adecuadamente en ella, compartiendo sus recursos. Todo ello es gracias a las bondades del esquema local síncrono de la arquitectura *DLA*, pues el uso de los buzones permite optimizar los recursos disponibles para que la operación del sistema sea eficiente. En cualquier caso, si el funcionamiento no hubiera sido adecuado con este esquema se podría haber hecho uso del esquema distribuido síncrono, distribuyendo los módulos en varias máquinas. De los módulos del sistema, los principales son: i) los módulos *SeguirPared*, *SeguirPasillo* y *CruzarPuerta*, descritos en el Capítulo 3; ii) el módulo *Fusión*, descrito en el Capítulo 4; y iii) el módulo *PlanificadorRuta*, descrito en el Capítulo 5. Estos tres capítulos (3, 4 y 5) concentran las aportaciones realizadas con la presente Tesis y se combinan en el presente capítulo de resultados.

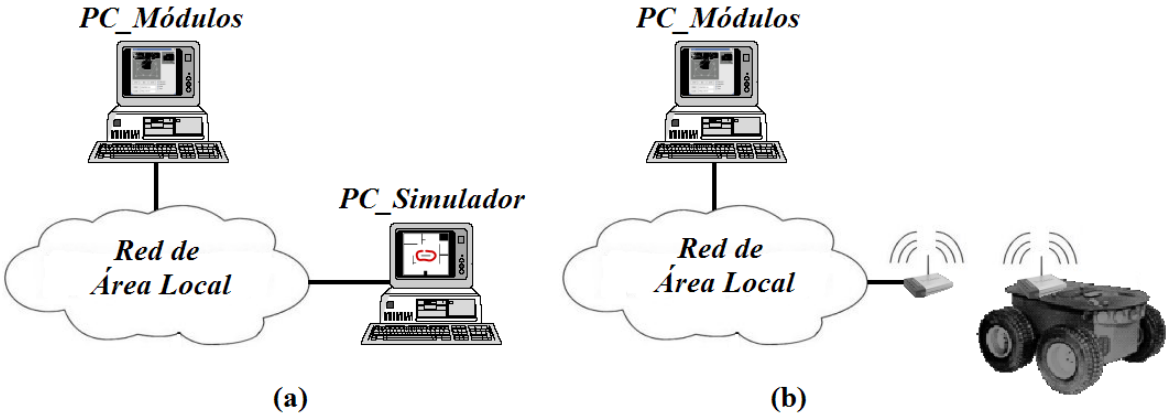

Figura 6.1: Entorno de pruebas: (a) simulador de la plataforma robótica *Nomad 200* de *Nomadics*; (b) plataforma robótica real *Pioneer P2AT*.

En cuanto a la capa reactiva, *Comportamientos*, decir que en las pruebas se emplea el esquema implementado siguiendo la filosofía del *CBR* para los tres comportamientos. Esto es así porque dota de una mayor flexibilidad al sistema y de una mejor operación, permitiendo tanto el aprendizaje por observación de un humano como el aprendizaje por experiencia, adaptándose de un forma más eficiente a la configuración del entorno. No obstante, debido a la implementación de esta capa reactiva, se podría haber seguido otro esquema. Como cada comportamiento se implementa en un módulo independiente, una opción válida sería la de combinar comportamientos desarrollados según el modelo analítico con otros implementados con el *CBR*. Sin embargo, se ha preferido usar para los tres, Seguir Pared, Seguir Pasillo y Cruzar Puerta, la técnica basada en el *CBR*.

Para el cálculo de los factores de ponderación en el módulo *Fusión* se utiliza la base que se presenta en la Sección 2.4.1 del Capítulo 4, calculada según el procedimiento analizado. Del mismo modo, los patrones del sistema para comparar el vector de características obtenido a

partir de un mapa local son los presentados en la Sección 2.5.1 del Capítulo 4.

En la Tabla 6.1 se presentan los valores empleados durante las pruebas para los parámetros del sistema. Estos parámetros se han agrupado según el capítulo en el que fueron descritos, para así facilitar su búsqueda en el texto y su interpretación en caso de duda.

Otros (Capítulo 2)		
Parámetro	**Valor**	**Descripción**
$P_{espacio}$	10%	Decremento de probabilidad de las celdas libres
$P_{ocupado}$	20%	Incremento de probabilidad de las celdas ocupadas
β	10^{o}	Anchura del lóbulo principal del sensor sonar
$U_{metrico}$	200	Mínimo número de celdas que cambia para planificar
Tam_Celda	$40mm$	Tamaño de la celda de los mapas probabilísticos
$Tam\_MapaMetrico$	$256x256$	Número de celdas del mapa métrico
$Tam\_MapaLocal$	$113x113$	Número de celdas del mapa local
Comportamientos (Capítulo 3)		
Parámetro	**Valor**	**Descripción**
d_D	$700mm$	Distancia a la que se sigue la pared en Seguir Pared
d_{SPSEC}	$1000mm$	Distancia de seguridad en Seguir Pared
d_{SCSEC}	$400mm$	Distancia de seguridad en Seguir Pasillo
d_{CPSEC}	$300mm$	Distancia de seguridad en Cruzar Puerta
U_{reusar}	0.1	Umbral de adaptación de casos en el ciclo *CBR*
$Dist\_MaxMin$	1000	Distancia mínima entre clases del algoritmo *MaxMin*
Fusión (Capítulo 4)		
Parámetro	**Valor**	**Descripción**
G_{region}	255	Nivel de gris de la región de interés
$G_{no_explore}$	127	Nivel de gris de las zonas no exploradas
G_{fondo}	0	Nivel de gris de las zonas fuera de la región de interés
$U_{ocupada}$	60%	Umbral de probabilidad para las zonas ocupadas
$U_{no\_explore}$	40%	Umbral de probabilidad para las celdas libres
N	64	Tamaño del mapa de profundidad y de su $\|FFT\|$
P	9	Tamaño de la base de componentes principales
U_{pesos}	0.4	Umbral de activación de campo de potencial
Planificación (Capítulo 5)		
Parámetro	**Valor**	**Descripción**
P_{obst}	100%	Probabilidad de ocupación de las celdas obstáculo
P_{libre}	0%	Probabilidad de ocupación de las celdas libres
P_{inex}	50%	Probabilidad de ocupación de las celdas no exploradas
U_{obst}	60%	Umbral de probabilidad para las celdas ocupadas
U_{libre}	40%	Umbral de probabilidad para las celdas libres
$Obst\_Inc$	2	Número de celdas en que se crecen los obstáculos
$Dist\_Max$	40	Distancia máxima entre las regiones de un entorno
A_min	16	Área mínima de una región

Tabla 6.1: Parámetros empleados durante las pruebas del sistema de exploración.

Por último, debemos recordar que el módulo *Interacción* es el usado para iniciar y parar

el proceso de exploración, siendo su finalización automática gracias al sistema desarrollado. También es de utilidad para visualizar los resultados durante la exploración. Una parte muy importante de esta visualización corresponde al mapa métrico que se va generando y actualizando en el módulo *MapaMétrico*, así como el mapa local disponible en cada punto del entorno, obtenido mediante el módulo *MapaLocal*. En estos mapas probabilísticos, las zonas libres se muestran en negro, los obstáculos en blanco y las áreas no exploradas en gris. Para visualizar los resultados también son de ayuda las dos representaciones gráficas que se han desarrollado en la presente Tesis, introducidas en el Capítulo 4. Se trata de la *Representación de Selección* (*RS*), y la *Representación de Fusión* (*RF*), siendo esta última la que aporta la información más significativa en relación a la cooperación de los tres comportamientos del sistema.

2 Entornos simulados

Todas las pruebas en entornos simulados usan las bases de casos clasificadas obtenidas tras el aprendizaje por observación realizado en los entornos de la Figura 6.2. El conocimiento utilizado por el comportamiento Seguir Pared (Figura 6.2a) es el mismo de la Figura 3.13 del Capítulo 3, consistente en 393 casos clasificados en 37 clases. En cuanto al segundo comportamiento, Seguir Pasillo, decir que el aprendizaje por observación usado (Figura 6.2b) también es el mismo que uno previo empleado en las pruebas del Capítulo 3, concretamente el de la Figura 3.17. Este entrenamiento almacena 145 casos que se clasifican en tan solo 6 clases. Por último, para el comportamiento Cruzar Puerta se utiliza el aprendizaje por observación que se muestra en la Figura 6.2c, el mismo de la Figura 3.23, donde los 103 casos adquiridos se dividen en 13 clases.

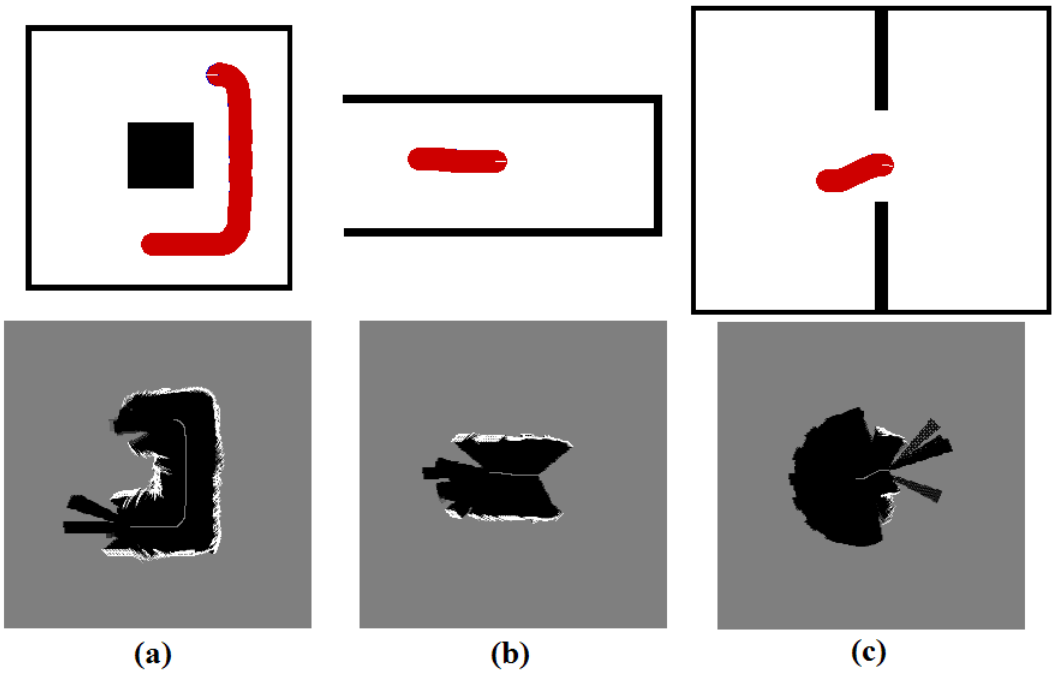

Figura 6.2: Aprendizaje por observación: (a) Seguir Pared; (b) Seguir Pasillo; (c) Cruzar Puerta.

2.1 Entorno simulado de prueba 1

En una primera prueba el agente autónomo móvil se encuentra en el centro de un entorno de $10x10m^2$ delimitado por paredes, pero completamente vacío en su interior (Figura 6.3). Por supuesto, este hecho es desconocido inicialmente porque el entorno está totalmente inexplorado al dar la orden de exploración. La trayectoria recorrida por el agente se muestra en la Figura 6.3a. Esta trayectoria determina la distribución de colores que se presenta en la *RF* de la Figura 6.3b tras finalizar la exploración. En ésta se observa que inicialmente ninguno de los tres comportamientos es el predominante, puesto que el agente no se encuentra ni en una pared, ni en un pasillo, ni cerca de una puerta. La ruta de exploración propuesta hace que el robot se dirija hacia la esquina superior derecha, siendo la contribución de los tres comportamientos proporcional al valor de los pesos calculados en el módulo *Fusión*.

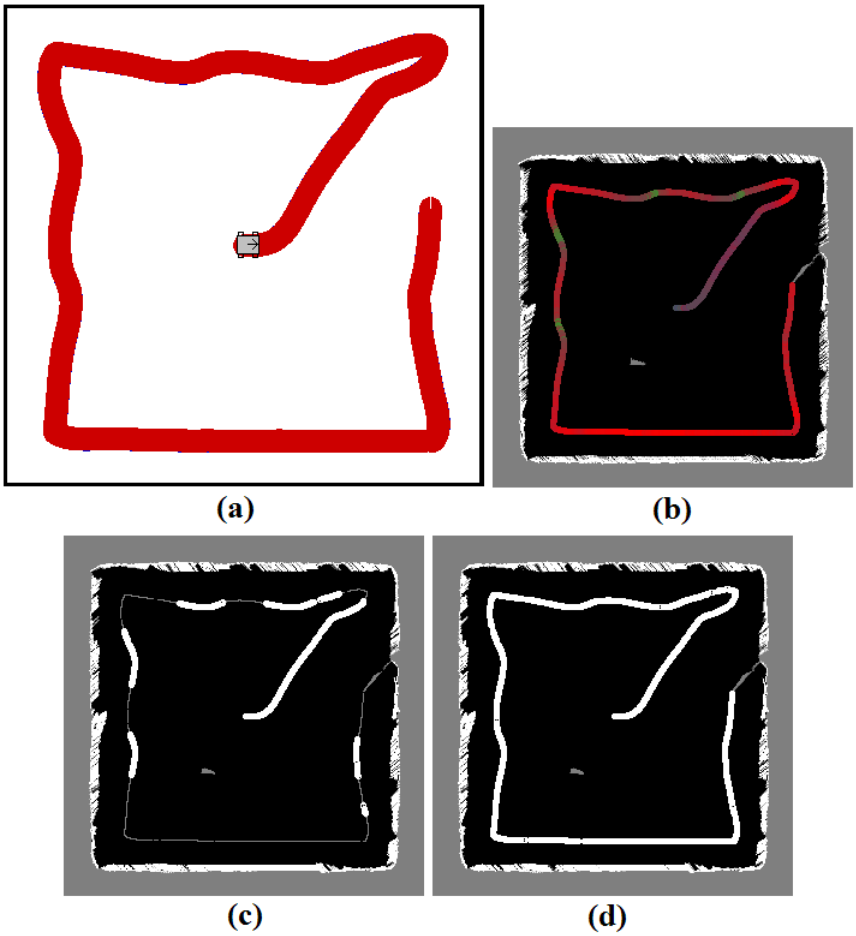

Figura 6.3: Exploración de entorno simulado: (a) trayectoria; (b) *RF*; (c) comportamiento Seguir Pared; (d) comportamiento Seguir Pasillo.

Una vez el robot se encuentra en la parte superior derecha, el comportamiento principal será el Seguir Pared, tal y como se observa perfectamente a través del color rojo de la trayectoria de la *RF* de la Figura 6.3b. A partir de ese momento, la trayectoria presenta un color rojo intenso en la mayoría de sus puntos porque el robot percibe el entorno como una pared. Sin

embargo, en la parte superior y oeste del entorno, la trayectoria presenta algunas oscilaciones, apareciendo algunos puntos donde el comportamiento Seguir Pared no es el que más contribuye al movimiento. Esto ocurre porque existen zonas que aún no están exploradas algo más alejadas de la pared que aparecen en la ruta de exploración y el robot también debe dirigirse hacia ellas. Finalmente, cuando solamente falta por explorar la zona inferior del entorno y un trozo de la parte este, la trayectoria seguida por el robot es prácticamente rectilínea (con la ventaja que esto supone), con una marcada contribución del comportamiento Seguir Pared hasta que todo el entorno está explorado. Una característica importante que se aprecia en la RF es la suavidad en las transiciones de color, es decir, en la adaptación de los factores de ponderación de los comportamientos a la configuración del entorno. Este hecho implica una navegación con movimientos y cambios de dirección más suaves por parte del agente. Si bien esta propiedad ya fue comentada en las pruebas de la Sección 5 del Capítulo 4, en aquéllas pruebas el robot era guiado manualmente, mientras que en las que se presentan en este capítulo navega de forma autónoma. Por lo tanto, la suavidad en los cambios de color de la trayectoria es una característica que depende de la configuración del entorno, inherente al método de cálculo de los factores de ponderación y, al mismo tiempo, es independiente de la forma en la que se controla el robot.

Durante la navegación, cada uno de los tres comportamientos hace uso de la base de casos clasificada que se obtiene tras el aprendizaje por observación en los entornos de la Figura 6.2. A modo de ejemplo se presenta en las Figuras 6.3c y d la trayectoria recorrida por el agente sobre el modelo de entorno construido. Al mismo tiempo, la Figura 6.3c muestra con un círculo blanco la posición de los casos adaptados para el comportamiento Seguir Pared, mientras que en la Figura 6.3d se muestra la misma información para el comportamiento Seguir Pasillo. El primer dato significativo que se extrae es que en ningún momento el agente percibe el entorno como un pasillo, puesto que en la Figura 6.3d los casos se adaptan en todos los puntos de la trayectoria para el comportamiento Seguir Pasillo. Sin embargo, para el comportamiento Seguir Pared la situación es diferente. Se observa en la Figura 6.3c que el sistema adapta casos únicamente cuando la configuración del entorno no se corresponde con una pared. Esto sucede al principio de la navegación, cuando el robot se dirige hacia la esquina superior izquierda del entorno, y cuando el agente se aleja un poco de la pared. En los giros hacia la izquierda en las esquinas no se adaptan los casos porque el aprendizaje por observación de la Figura 6.2a almacena este conocimiento en la base de casos.

En la Figura 6.4 se presenta el estado de la exploración del entorno de la Figura 6.3a en distintos instantes de la misma. A tal efecto se muestra para cada etapa el mapa topológico con la ruta de exploración planificada superpuesta al modelo de entorno almacenado en ese momento. La primera etapa mostrada corresponde a un instante del comienzo de la exploración (Figura 6.4a). La ruta de exploración completa consta de 14 nodos, dirigiendo al robot hacia el este y posteriormente hacia el norte, es decir, hacia la esquina superior derecha del entorno. En las siguientes etapas que se presentan la ruta indica que debe explorarse la zona norte del entorno (Figuras 6.4b y c), la cual consta de 13 nodos. Cuando esa zona ya está explorada, el robot se dirige hacia el sur, siendo entonces la ruta propuesta la que se muestra en la Figura 6.4d, con 11

regiones. Obviamente, a medida que avanza la exploración, el número de regiones sin explorar que aparecen es mucho menor, algo que se aprecia en las Figuras 6.4e, f, g y h. Así, en la etapa que se representa mediante la Figura 6.4e, el número de zonas sin explorar que aparecen en el mapa topológico es de tan solo 6, por lo que el tamaño del *TSP* en ese instante sería de 7. El agente sigue avanzando en la exploración hasta que no quede ninguna zona desconocida en el entorno, momento en el que el sistema detiene la exploración de forma automática (Figura 6.4i).

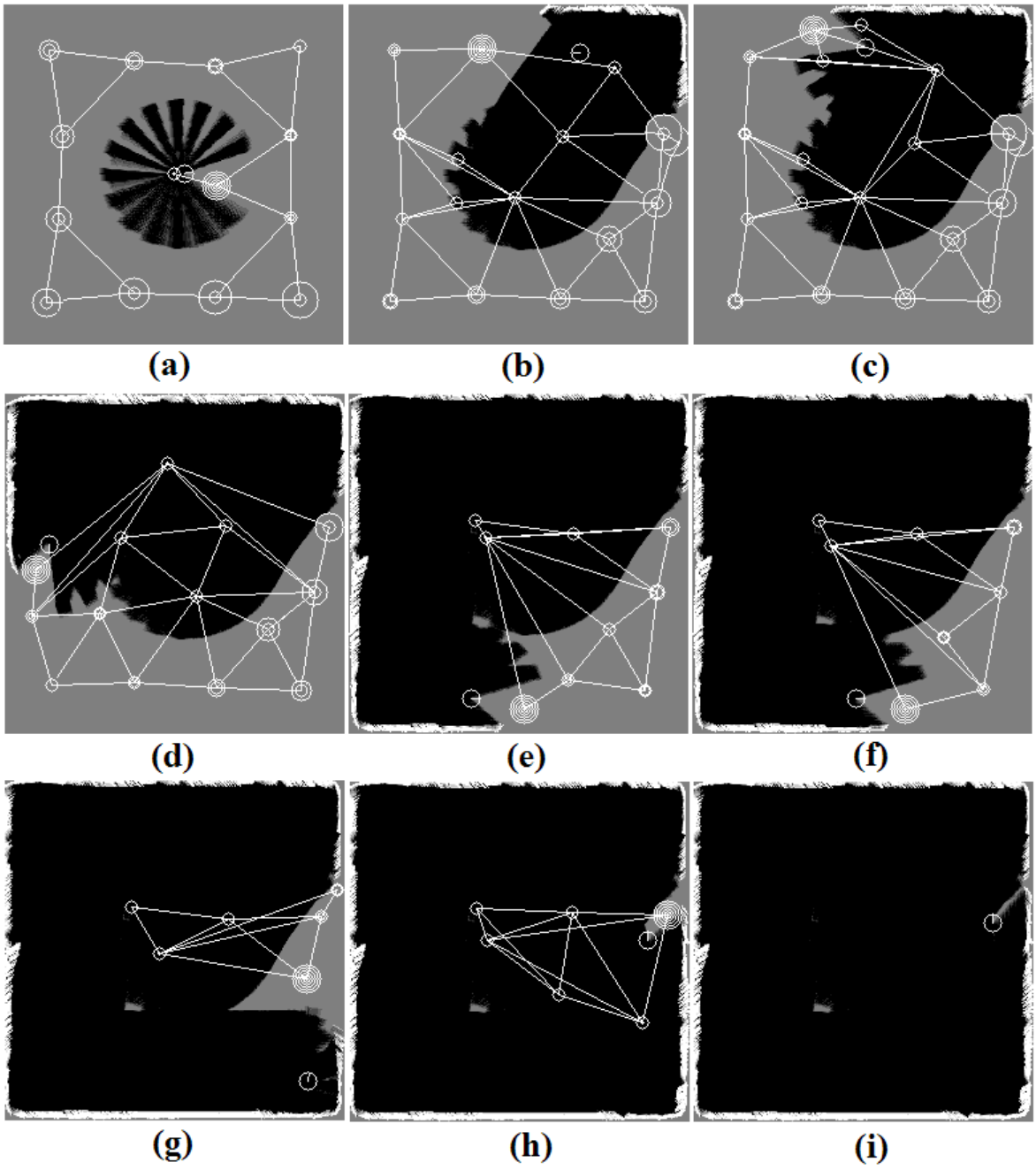

Figura 6.4: (a)-(i) Mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno de la Figura 6.3a.

2.2 Entorno simulado de prueba 2

Un segundo entorno de test es el que se presenta en la Figura 6.5a, el cual tiene unas dimensiones mayores que el entorno vacío de la Figura 6.3a. Debido al tamaño de celda y del mapa métrico escogidos, $Tam\_Celda = 40mm$ y $Tam\_MapaMetrico = 256x256$ respectivamente, no todo el entorno tiene cabida en el mapa métrico usado para planificar. Desde el punto de vista de la operación del sistema no es un problema puesto que, tal y como se comentó en la Sección 3.3.1 del Capítulo 2, el módulo *MapaMétrico* se implementa con un esquema de almacenamiento circular que asegura que la zona del entorno representada en el modelo es en todo momento la más recientemente explorada. Sin embargo, ésto puede suponer un problema desde el punto de vista de la planificación de la ruta de exploración completa. Debido a que el mapa métrico descarta la zona más antigua para poder representar la que el robot está explorando actualmente, es posible que zonas ya exploradas vuelvan a aparecer en el mapa métrico como no exploradas. Esto supone que, dependiendo del entorno de test, puede que el sistema detecte el final de la exploración o puede que no lo haga incluso en el caso de que ésta haya finalizado. Así, para el entorno de la Figura 6.5a el sistema sí detecta la finalización de la exploración, aunque es debido a la configuración del entorno y no a que todo el modelo del mismo esté representado al finalizar el proceso.

Al iniciar la exploración del entorno de la Figura 6.5a el robot se dirige hacia la derecha porque así se lo marca la ruta de exploración completa. Al detectar la pared frontal gira 180º, sigue la pared y se centra en el pasillo que detecta. Si bien al principio de la trayectoria el comportamiento principal es el Seguir Pared (ver color rojo inicial en la *RF* de la Figura 6.5b), a partir de ese momento la mayoría de la trayectoria presenta un marcado color verde, debido a que el robot navega prácticamente siempre a través de un pasillo. El comportamiento Seguir Pasillo es el principal excepto en los giros que el robot debe llevar a cabo. Excepto cuando gira hacia el este en la parte norte del pasillo central, en los giros se dan dos situaciones diferentes. Por un lado, si el robot detecta espacio libre a su izquierda, el entorno se percibe como una pared hasta que de nuevo sigue la ruta planificada centrándose en el pasillo, como sucede al girar hacia el norte en el pasillo sur para tomar el pasillo central del entorno. Por otro, en el resto de los giros siempre hay una pequeña zona de la trayectoria donde el comportamiento más importante es el Cruzar Puerta, debido a que el agente percibe una abertura, el nuevo pasillo, que todavía no detecta como tal. Así mismo, también aparece otra pequeña zona donde el comportamiento principal debe ser el Seguir Pared. Al igual que para el entorno de la Figura 6.3a, la *RF* representada en la Figura 6.5b presenta la misma suavidad en las transiciones de color, consiguiendo así un movimiento más suave durante la navegación.

Durante la navegación, los comportamientos hacen uso de la base de casos clasificada que se obtiene tras el aprendizaje por observación en los entornos de la Figura 6.2. A modo de ejemplo se presenta en la Figura 6.5c la trayectoria recorrida por el agente en gris sobre el modelo de entorno construido para el comportamiento Cruzar Puerta. Los casos adaptados se muestran con un círculo blanco en la posición que fueron capturados. El agente no percibe prácticamente

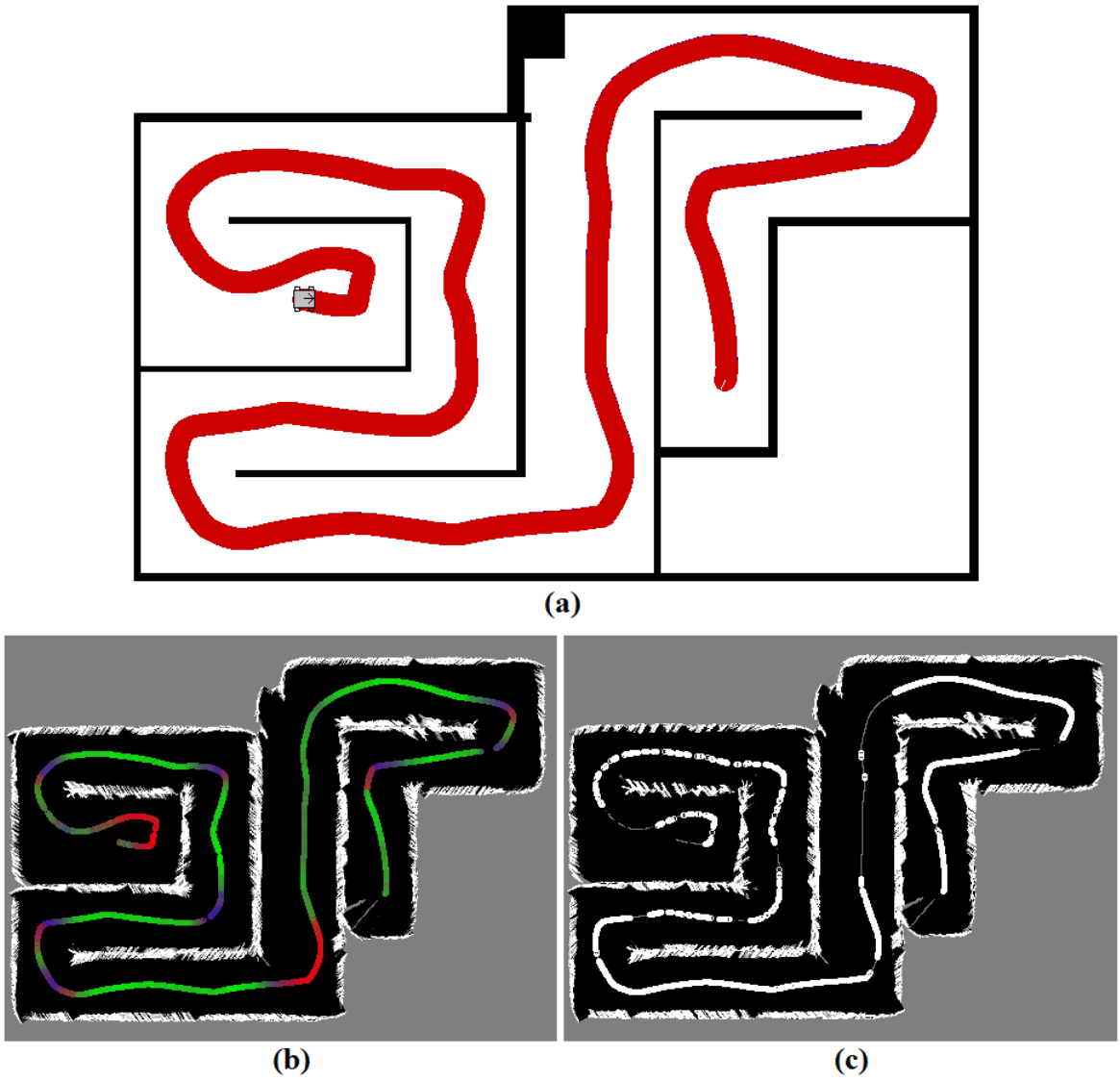

Figura 6.5: Exploración de entorno simulado: (a) trayectoria; (b) RF; (c) comportamiento Cruzar Puerta.

nunca a lo largo de la trayectoria que se encuentra en o cerca de una puerta. Por lo tanto, los casos deben ser adaptados en la mayoría de la trayectoria.

La Figura 6.6 presenta el mapa topológico con la ruta de exploración planificada superpuesta al modelo de entorno almacenado en diferentes instantes de la exploración del entorno de la Figura 6.5a. Las rutas planificadas en los instantes que se muestran en la Figura 6.6 constan todas de un número igual o inferior a 10 regiones, excepto para la primera etapa. En este caso (Figura 6.6a), la ruta está constituida por 13 nodos, dirigiéndose en primer lugar el robot hacia el oeste para explorar el primer pasillo que se encuentra. En algunas de las etapas aparecen nodos del mapa topológico que no se corresponden físicamente con una zona del espacio, como se aprecia en las Figuras 6.6c, d, e, f, j y k. Se dan dos situaciones diferenciadas. En primer lugar, las regiones sin correspondencia física pueden aparecer como un nodo que hay que visitar

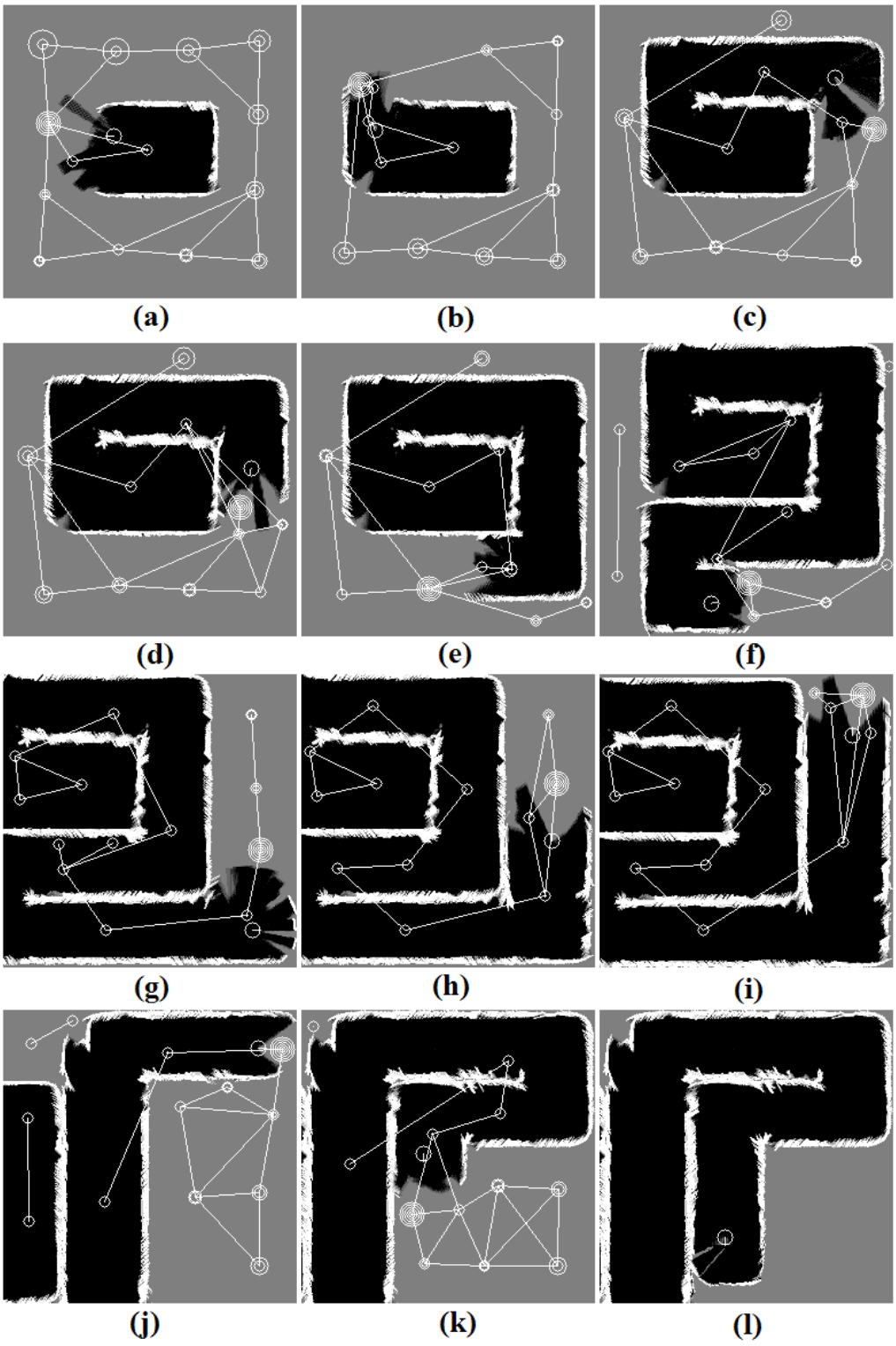

Figura 6.6: (a)-(l) Mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno de la Figura 6.5a.

tras planificar la ruta de exploración, según se muestra en las Figuras 6.6c, d y e. En los tres casos hay nodos del mapa topológico que forman parte de la ruta en la parte superior y oeste

del entorno porque, debido al propio proceso de generación y actualización del mapa métrico durante la exploración, se puede llegar a esas zonas no exploradas desde la posición del robot. En segundo lugar, surgen zonas no exploradas que no son accesibles desde la posición del agente (Figuras 6.6f, j y k). Esto es lógico, porque se corresponden con zonas amplias donde debe aparecer un nodo del grafo, aunque la progresiva exploración que el agente lleva a cabo hace que no tengan conexión con los nodos accesibles. Esto no es un problema para el método propuesto en esta Tesis, ya que en ningún caso se incluyen en la ruta de exploración los nodos no accesibles.

A partir de la etapa de la Figura 6.6g se pierde la información que se tenía almacenada en el modelo de entorno sobre su zona oeste, para dar cabida en la representación a la zona que actualmente explora el agente. Así, el robot sigue avanzando hacia el norte por el pasillo central hasta que parece que el mapa métrico está completamente explorado (Figura 6.6i), excepto por un par de nodos que representan dos regiones de pequeño tamaño. Sin embargo, debido al almacenamiento circular que se lleva a cabo en el módulo *MapaMétrico*, se descarta la zona sur ya explorada para permitir explorar la última parte del entorno. Así, se explora el pasillo noreste (Figura 6.6j), hasta llegar a la última zona inexplorada. Finalmente, cuando el robot se encuentra en la posición indicada en la Figura 6.6l, el sistema determina que la exploración ha finalizado. Aunque el modelo de entorno usado para planificar en la Figura 6.6l no abarca el entorno entero, no es posible alcanzar ninguna de las regiones no exploradas que aparecen desde la posición del robot. Casualmente, al observar la *RF* de la Figura 6.5c, se observa que el entorno de la Figura 6.5b ha sido explorado íntegramente.

2.3 Entorno simulado de prueba 3

La Figura 6.7a muestra la trayectoria seguida por el agente autónomo móvil en un tercer entorno de prueba simulado. Esta trayectoria viene marcada por la ruta que el sistema planifica en todo instante de manera adaptativa en función del modelo de entorno que tenga almacenado en cada momento. Gracias a esta trayectoria se consigue explorar de forma íntegra el entorno, obteniéndose al finalizar la exploración el modelo de entorno representado en la *RF* de la Figura 6.7b.

La trayectoria presenta relativamente pocas oscilaciones y pocos cambios de dirección. Se aprecia, por ejemplo, cuando el agente está recorriendo la zona sur del entorno, donde después de una trayectoria recta alejándose brevemente de la pared sur, gira y se orienta de forma paralela a la misma. Posteriormente, al detectar la pared frontal el agente lleva a cabo un giro casi perfecto de 90^o y sigue navegando de forma paralela a dicha pared. Esta minimización de los cambios de dirección se debe a la metodología de diseño basada en el ciclo *CBR*, donde el aprendizaje por observación usado para los tres comportamientos (Figura 6.2), tiene una importancia vital. Del mismo modo que para los dos entornos de test presentados anteriormente, la suavidad en las transiciones de color es una característica de la *RF* representada en la Figura 6.7b. Esto permite una evolución suave del robot durante la navegación sin cambios bruscos. La distribución de colores que se muestra en la Figura 6.7b indica que el sistema adapta la navegación del agente

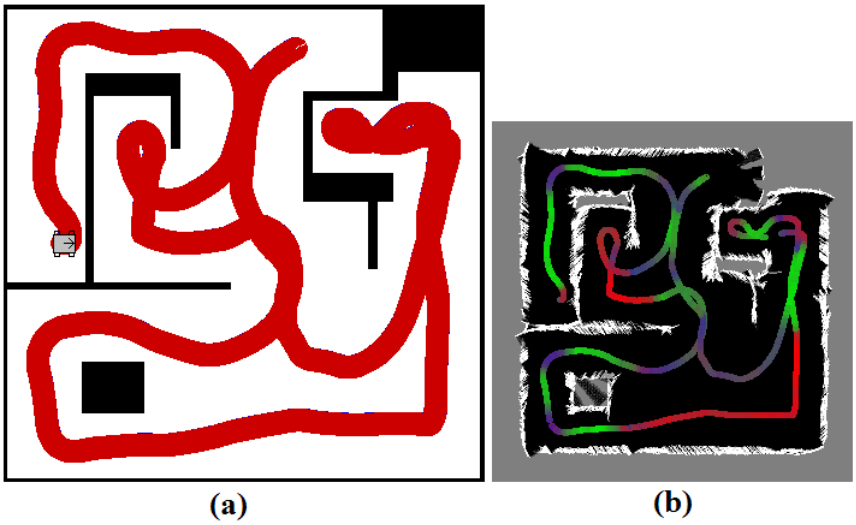

Figura 6.7: Exploración de entorno simulado: (a) trayectoria; (b) *RF*.

a la configuración del entorno detectada en cada momento, gracias al método propuesto para el cálculo de los factores de ponderación de los tres comportamientos.

En la trayectoria de la *RF* en la Figura 6.7b se aprecia que el comportamiento principal en cada punto de la misma es coherente con la configuración del entorno más cercano al robot. El comportamiento Seguir Pared es el que predomina en cinco zonas bien diferenciadas. La primera, junto al comienzo de la navegación, cuando el robot se acerca a la pared girando hacia el norte y así se centra en el pasillo. La segunda, al comenzar a explorar la zona central del entorno, donde debe girar hacia el norte y posteriormente hacia el sur. Quizás la zona más clara para este comportamiento sea la de color rojo intenso cerca de la pared sur y este, donde el robot sigue la pared porque es lo más adecuado para esa situación. Las dos últimas áreas de color rojo aparecen al finalizar de explorar la pared este, donde el agente se debe acercar a las paredes para conseguir explorar toda la zona.

El comportamiento Seguir Pasillo es el principal (color verde de la trayectoria), siempre que el agente se encuentra entre dos paredes, una a su izquierda y la otra a su derecha. Esta situación se da en distintos puntos de la trayectoria, como es al comenzar a navegar por el pasillo oeste, al virar a la derecha hacia el pasillo norte o al bordear el obstáculo en la zona suroeste por su norte, oeste y sur. En cuanto al comportamiento Cruzar Puerta, aparece sobre todo en los giros, como sucedía también en el entorno de la Figura 6.5a. Así, nos encontramos con que este tercer comportamiento es el que domina la trayectoria al girar para dirigir hacia el este el robot por el pasillo en la parte superior del entorno, o cuando se está bordeando el obstáculo en la esquina inferior izquierda del escenario. Por último, decir que hay dos zonas de la trayectoria donde el sistema no tiene una contribución determinante por parte de ninguno de los tres comportamientos. Son las zonas en las que el agente gira en la zona central del entorno. En un caso, para dirigirse a explorar el área suroeste. En el otro, cuando gira hacia el norte para navegar con dirección hacia la última de las zonas no exploradas, la noreste.

2.4 Entorno simulado de prueba 4

La Figura 6.8 muestra la trayectoria y la *RF* que se obtiene al explorar otro entorno simulado, en las Figuras 6.8a y 6.8b, respectivamente. La ruta planificada hace que el robot avance en primer lugar hacia la región que se encuentra al norte de su posición inicial, donde sigue la pared hacia el oeste y después hacia el sur. En ese momento, detecta la configuración del entorno como un pasillo, pues aparecen obstáculos a su izquierda y a su derecha. La ruta de exploración hace que el agente navegue por el sur del entorno. Durante dicha navegación, el comportamiento que más influencia tiene en los comandos de movimiento varía dependiendo del punto por el que el agente autónomo se mueva. Así, al iniciar a recorrer la zona sur, el comportamiento Seguir Pared es el principal, puesto que el sistema detecta una pared a la derecha y espacio libre a la izquierda. Sin embargo, al detectar el obstáculo izquierdo, el comportamiento Seguir Pasillo es el que contribuye con un mayor peso. Una vez dejado atrás el obstáculo, nuevamente es el comportamiento Seguir Pared el que mayor contribución tiene, como se aprecia en la zona rojo intenso de la *RF* de la Figura 6.8b, al explorar la parte inferior derecha del entorno. Debemos hacer notar que los cambios de dirección durante la exploración de la zona sur son mínimos, con la ventaja que esto implica para la navegación del agente. Por supuesto, las transiciones de color son suaves, redundando en una mayor suavidad en los movimientos del robot.

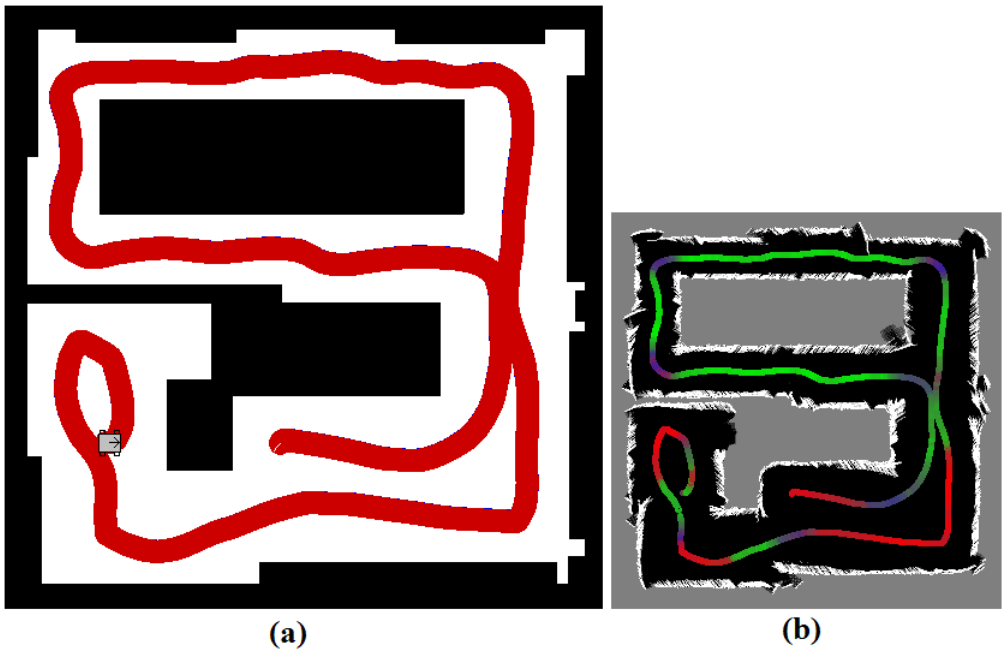

Figura 6.8: Exploración de entorno simulado: (a) trayectoria; (b) *RF*.

Después de explorar la zona sur, la ruta planificada dirige al agente hacia el norte, siguiendo en primer lugar la pared y después centrándose en el pasillo que se encuentra. Este comportamiento, Seguir Pasillo, es el principal hasta que se detecta la pared norte del entorno, momento en el cual la ruta planificada hace que gire hacia la izquierda y se centre en el pasillo norte. Es destacable el hecho de que el robot no tiene ningún problema para entrar en el pasillo norte, a

pesar de que su anchura es en la parte más estrecha de tan solo $1.2m$. En la parte más ancha mide $1.5m$, la misma anchura que el pasillo de norte a sur en la parte noroeste del entorno, y el pasillo que recorre el entorno de oeste a este en la parte central. Durante la navegación por estos pasillos la trayectoria tiene un marcado color verde intenso, lo que nos indica que, con diferencia, el comportamiento más importante es el Seguir Pasillo. Esto ocurre excepto en los dos giros que se producen en el pasillo al oeste del entorno, donde el agente ve el escenario como una puerta, ya que en el mapa local hay una especie de abertura. Es significativa también la oscilación del agente hasta que alcanza la parte más ancha del pasillo central. Esto ocurre porque se adaptan muchos casos durante esa parte de la trayectoria, debido a que el aprendizaje por observación en el entorno de la Figura 6.2b se realiza en un pasillo de bastante mayor anchura. Por lo tanto, al adaptar tantos casos con la aproximación analítica, el agente oscila porque dicho modelo está basado en los campos de potencial. Finalmente, la planificación de la ruta de exploración completa provoca que el agente atraviese de nuevo el corredor central en el este del entorno para incluir en el modelo final que se presenta la única zona que falta por explorar.

2.5 Entorno simulado de prueba 5

El último entorno de test simulado es el de la Figura 6.9a. Con el tamaño de celda empleado, $Tam\_Celda = 40mm$, y el tamaño del mapa métrico usado para la planificación, $Tam\_MapaMetrico = 256x256$, no todo el modelo de entorno está representado en el mapa métrico. Por lo tanto, al ir descartando las regiones más antiguamente exploradas para dar cabida a las zonas que se están explorando, se pierde la información en relación a las áreas ya exploradas, como se comentó para el entorno de la Figura 6.5. Aun así, en aquél entorno el sistema daba por finalizada la exploración porque en un instante determinado no existía ya ninguna región accesible por explorar. Casualmente, esta situación coincidía con el momento en el cual todo el entorno quedaba completamente explorado. Sin embargo, para el entorno que se muestra en la Figura 6.9 no se tiene tanta suerte. El mapa métrico usado por el módulo *PlanificadorRuta* para planificar la ruta en un instante determinado se muestra en la Figura 6.9b. En ese instante, el modelo de entorno generado, representado en un mapa métrico de mayores dimensiones, se presenta en la *RF* de la Figura 6.9c. Quizás el hecho más significativo es que el entorno está completamente explorado pero, debido al proceso de generación y actualización del mapa métrico, el sistema detecta que todavía existe alguna zona no explorada.

Una posible solución al problema descrito es emplear un tamaño de celda mayor, para que todo el entorno esté representado en el mapa métrico y así se planifique la ruta de forma adecuada. A tal efecto, se ha repetido la prueba empleando un tamaño de celda mayor, $Tam\_Celda = 80mm$. En estas condiciones, el sistema sí que detecta correctamente y de forma automática cuándo se ha explorado la totalidad del entorno. En ese momento el modelo de entorno almacenado es el que se presenta en la Figura 6.10b. Si se superpone el mapa topológico al mapa métrico cuando finaliza la exploración se observa que el grafo tiene dos partes diferenciadas que no están conectadas entre sí (Figura 6.10c). Una parte se corresponde con las zonas no ocupadas exploradas anteriormente por el robot. La otra está constituida por nueve regiones no

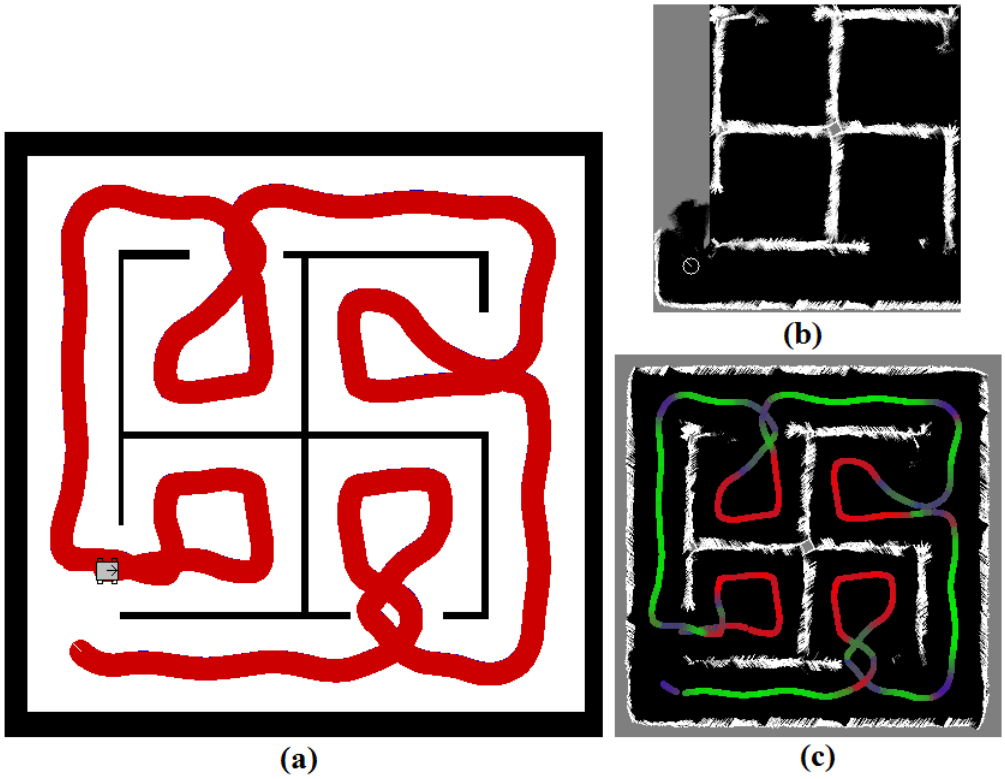

Figura 6.9: Exploración de entorno simulado: (a) trayectoria; (b) mapa métrico cuando todo el entorno está explorado; (c) *RF*.

exploradas no accesibles desde la posición del robot. Este resultado es coherente con el entorno de trabajo, puesto que la parte del grafo correspondiente a las regiones no exploradas no tiene su correspondencia física en el mundo simulado.

Las trayectorias seguidas en ambos casos, $Tam\_Celda = 40mm$ y $Tam\_Celda = 80mm$, se representan en las Figuras 6.9a y 6.10a, respectivamente. En estos dos casos se ha usado el modelo analítico para los tres comportamientos, para demostrar la validez de los mismos. Por este motivo el robot oscila ligeramente durante todo el tiempo que dura la navegación. La trayectoria seguida por el agente es similar en ambos casos, independientemente del tamaño de celda empleado. Así, la ruta de exploración permite explorar las cuatro habitaciones y los pasillos anejos correspondientes de forma consecutiva. Se empieza por la habitación suroeste, para pasar a continuación a la habitación noroeste a través del pasillo oeste del entorno. Después le toca el turno al pasillo norte, a la habitación noreste y al pasillo este. Finalmente, el agente explora la habitación sureste y el pasillo sur. La ruta de exploración que se va planificando en todo momento dirige el robot hacia una habitación. Cuando finaliza su exploración lo dirige hacia la siguiente hasta que todo el entorno aparece en el modelo.

Del mismo modo que para las trayectorias, las *RF*s presentan una distribución de colores similar, independientemente del tamaño de celda empleado, $Tam\_Celda = 40mm$ en la Figura 6.9c y $Tam\_Celda = 80mm$ en la Figura 6.10d. Durante la exploración de una habitación es el comportamiento Seguir Pared el que mayor contribución al comando de movimiento final tiene.

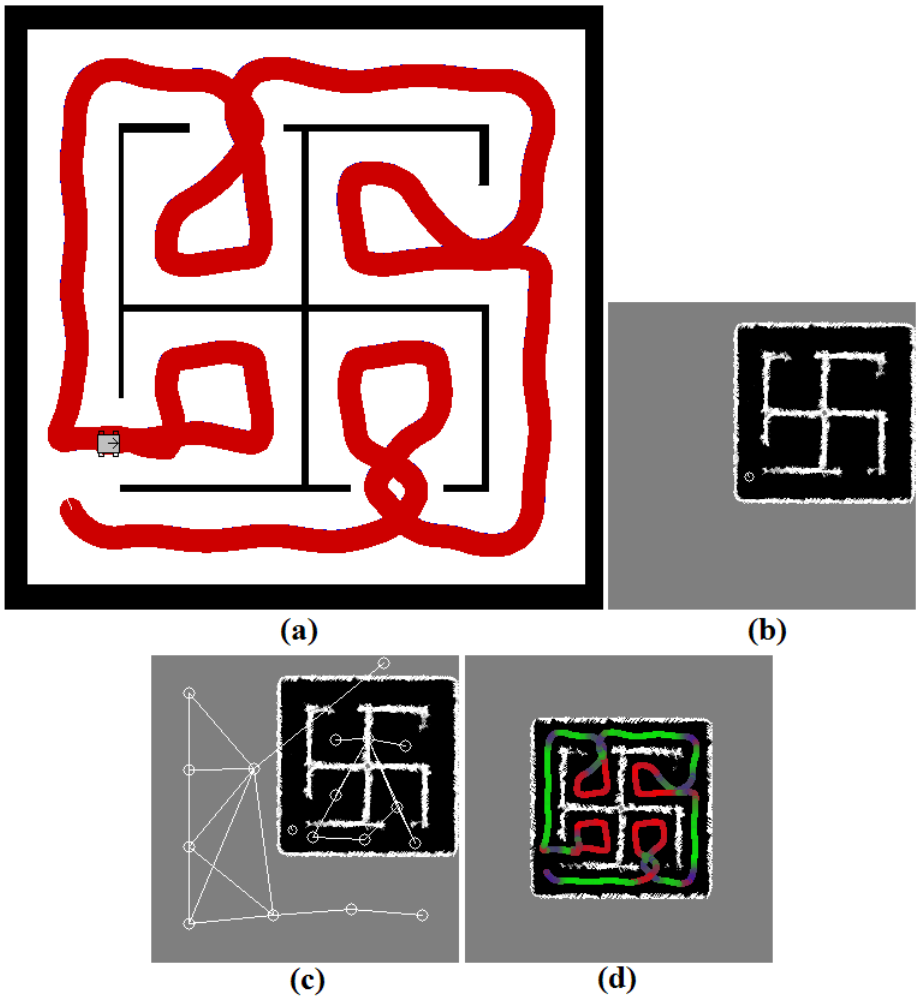

Figura 6.10: Exploración de entorno simulado con $Tam\_Celda = 80mm$: (a) trayectoria; (b) mapa métrico cuando todo el entorno está explorado; (c) mapa topológico sobre el mapa métrico en (b); (d) *RF*.

Durante la navegación por un pasillo es, lógicamente, el comportamiento Seguir Pasillo el que ostenta el mayor peso de un comando de movimiento. Y durante el giro en las cuatro esquinas o cuando el robot se orienta para entrar en una habitación es el comportamiento Cruzar Puerta el que influye en mayor medida en el comando de movimiento aplicado al robot. Estos hechos son coherentes con la configuración del entorno, puesto que se dan en las situaciones apropiadas de las trayectorias recorridas por el agente.

A pesar de que aumentando el tamaño de celda se obtienen unos resultados acordes con el entorno que se está explorando (Figura 6.10), quizás no sea la solución adecuada para cualquier entorno, puesto que su tamaño no es conocido de antemano. Tampoco es recomendable aumentar el tamaño del mapa métrico porque entonces el proceso de planificación tardaría un tiempo bastante mayor, haciendo más lenta la adaptación del sistema a la configuración del entorno. Por lo tanto, sería deseable plantear alguna posibilidad para aumentar el campo de visión del mapa métrico y, por ende, del mapa topológico, para poder abarcar entornos de trabajo mayores.

No obstante, este tema queda fuera de los objetivos de esta Tesis.

3 Entornos reales

Para todas las pruebas en entornos reales se usan las bases de casos clasificadas obtenidas tras el aprendizaje por observación realizado en los entornos de la Figura 6.11. El comportamiento Seguir Pared emplea como aprendizaje por observación (Figura 6.11a), el mismo de la Figura 3.30a del Capítulo 3, el cual consiste en una base de casos de 543 casos clasificados en 62 clases. Del mismo modo, el entrenamiento del comportamiento Seguir Pasillo (Figura 6.11b), también es el mismo que uno previo, el de la Figura 3.33b. Este entrenamiento almacena 152 casos en 18 clases. Por último, para el comportamiento Cruzar Puerta se utiliza el aprendizaje por observación que se muestra en la Figura 6.11c, el mismo de la Figura 3.38, donde los 174 casos adquiridos se dividen en 23 clases.

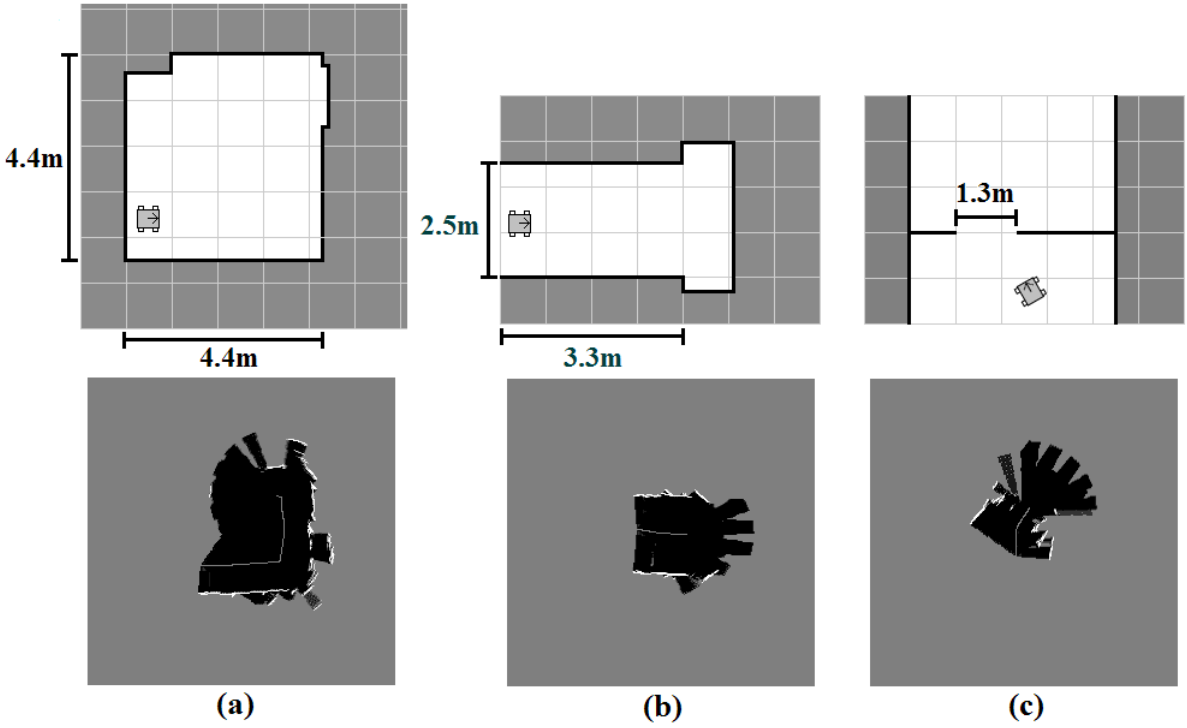

Figura 6.11: Aprendizaje por observación: (a) Seguir Pared; (b) Seguir Pasillo; (c) Cruzar Puerta.

3.1 Entorno real de prueba 1

En una primera prueba se coloca al agente en un entorno real cuadrado de una dimensión aproximada de $4.4x4.4m^2$ (Figura 6.12a). Tanto en este entorno de test como en los siguientes que se presentan, aparecen diferentes materiales como son paredes, metal, madera y cartón. Este hecho influye en las lecturas proporcionadas por los sensores sonar, pues éstas dependen del material sobre el que incide su haz. El robot va explorando el entorno hasta que el sistema

detecta que ya no existe ninguna zona no explorada accesible por el agente en el mapa topológico. Cuando esto sucede, el modelo de entorno almacenado es el que se muestra en la Figura 6.12c, donde se ha superpuesto el mapa topológico para demostrar la razón por la que la exploración finaliza. Dicho mapa topológico consta de 14 nodos. La habitación que el agente explora se concentra en uno de los nodos del mapa topológico mientras que los 13 restantes están inexplorados. Sin embargo, esos 13 nodos no se corresponden físicamente con el espacio real. Esta circunstancia queda perfectamente contemplada puesto que no existe conexión entre el nodo de la habitación y los otros 13.

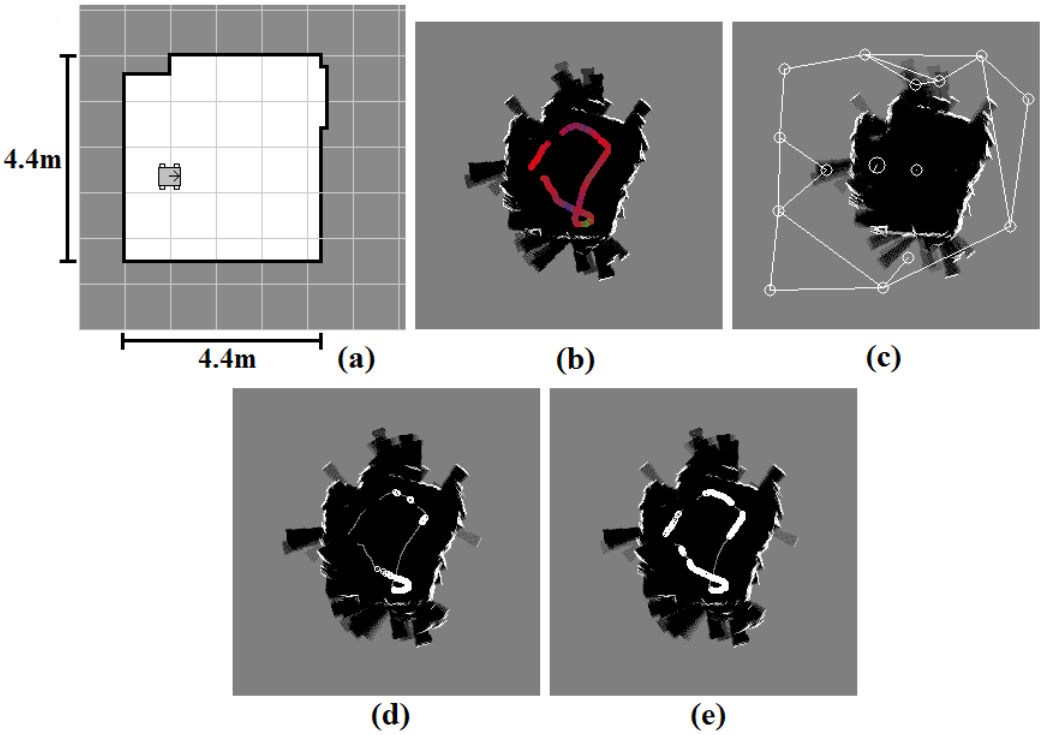

Figura 6.12: Exploración de entorno real: (a) planta; (b) *RF*; (c) mapa topológico sobre el mapa métrico cuando todo el entorno está explorado; (d) comportamiento Seguir Pared; (e) comportamiento Seguir Pasillo.

En la *RF* de la Figura 6.12b se observa que el comportamiento principal a lo largo de toda la trayectoria es el Seguir Pared. Esto es así porque el entorno es percibido por el robot como si fuera una pared mientras va explorando de acuerdo a la ruta planificada. Es destacable también la suavidad en las transiciones de color a lo largo de la trayectoria. Esta característica es independiente de que el entorno de trabajo sea real o simulado, e independiente al mismo tiempo del método empleado para controlar el robot, automático o guiado.

Durante la exploración cada uno de los tres comportamientos hace uso de la base de casos clasificada que se obtiene tras el aprendizaje por observación en los entornos de la Figura 6.11. En las Figuras 6.12d y e se presenta, como ejemplo, la trayectoria recorrida por el agente sobre el modelo de entorno que se construye. Al mismo tiempo, la Figura 6.12d muestra con un círculo blanco la posición de los casos adaptados para el comportamiento Seguir Pared, mientras que

en la Figura 6.12e se muestra la misma información para el comportamiento Seguir Pasillo. El primer dato significativo que se extrae es que el entorno no es percibido como un pasillo en casi ningún momento. Por otro lado, una vez el robot ha avanzado hacia la derecha y se ha acercado a la pared, no se adaptan apenas casos para el comportamiento Seguir Pared, puesto que el aprendizaje por observación de la Figura 6.11a considera las situaciones que aparecen. Sin embargo, durante la trayectoria de aproximación sí que se adaptan los casos. No obstante, los resultados de la Figura 6.12d son coherentes con la RF de la Figura 6.12b, donde el color rojo indica que el comportamiento Seguir Pared es el que mayor influencia tiene.

3.2 Entorno real de prueba 2

Un segundo entorno de test es el que se presenta en la Figura 6.13a. El modelo de entorno obtenido al finalizar la exploración es el de la Figura 6.13c. El mapa topológico en ese instante final de la exploración se superpone al mapa métrico en la Figura 6.13d. El grafo representado demuestra que el sistema detecta automáticamente la exploración completa del entorno. En dicho grafo se distinguen dos subgrafos que no están conectados. Uno de ellos se corresponde con la zona explorada, consistente en 6 nodos conectados entre sí. Debido a que hay dos obstáculos en medio del entorno, la habitación no se representa con un único nodo, al contrario de lo que ocurría en el grafo de la Figura 6.12c. El otro subgrafo no tiene su equivalente en el mundo real, puesto que sirve para representan con 10 nodos zonas no exploradas del mapa métrico que no tienen correspondencia física.

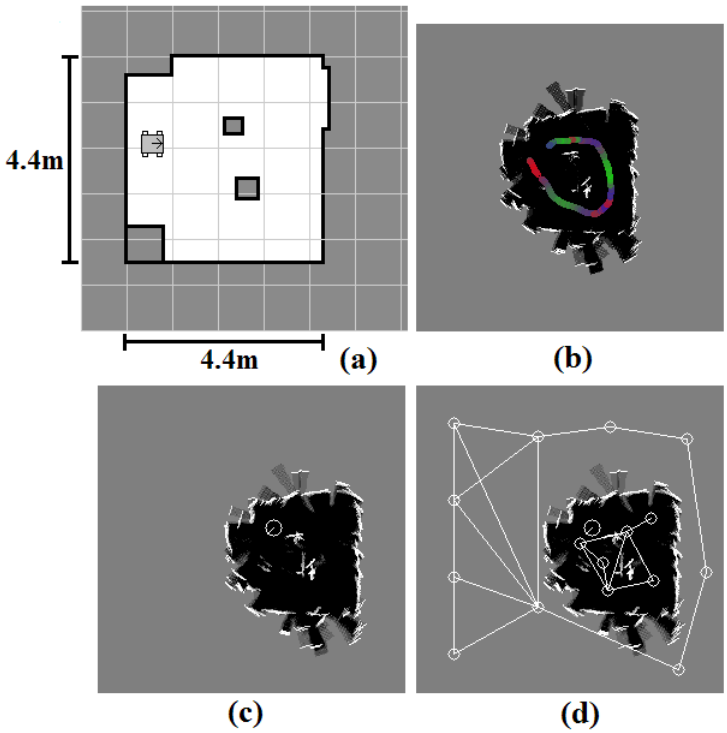

Figura 6.13: Exploración de entorno real: (a) planta; (b)RF; (c) mapa métrico cuando todo el entorno está explorado; (d) mapa topológico sobre el mapa métrico en (c).

La RF de la Figura 6.13b es coherente con la configuración del entorno que el agente detecta a lo largo de su trayectoria. Al iniciar la navegación es el comportamiento Seguir Pared el que lleva la delantera a los otros dos. Cuando el robot detecta un obstáculo a su izquierda, es el Seguir Pasillo el que mayor peso ostenta. Así, a largo de toda la trayectoria se suceden zonas con contribuciones principales de los tres comportamientos. Sin embargo, los colores están más mezclados y no tienen la misma intensidad que en las pruebas en entornos simulados debido al ruido que introduce el mundo real en la lectura de los sensores sonar. Esto hace que los mapas probabilísticos obtenidos en el mundo real no estén tan bien definidos como en el mundo simulado. Por lo tanto, el cálculo de los factores de ponderación se verá dificultado por el hecho de trabajar con un mapa local más ruidoso. No obstante, sí que se cumple la característica anteriormente comentada en relación a la suavidad en las transiciones de color, la cual implica una navegación con movimientos y cambios de dirección más suaves.

3.3 Entorno real de prueba 3

El siguiente entorno real de prueba es el de la Figura 6.14a. El mapa métrico usado para la planificación al finalizar la exploración se muestra en la Figura 6.14b. Al ser progresivo el proceso de actualización de este modelo del entorno, el mapa topológico va cambiando a la vez que lo hace el modelo, teniendo en cuenta el umbral $U_{metrico}$. A tal efecto se presenta en las Figuras 6.14f-h, el mapa topológico con la ruta planificada, ambos superpuestos al mapa métrico, en distintos instantes de la exploración del entorno de la Figura 6.14a. El número de regiones con las que trabaja el algoritmo de planificación en cada uno de los instantes de la exploración es de 12, 13 y 11 para las etapas de las Figuras 6.14f-h, respectivamente. Como se observa, muchas de las regiones no exploradas que se incluyen en la planificación no tienen su equivalente en el mundo real. No obstante, estas regiones deben ser incluidas en el problema hasta que el propio sistema determine que ya no hay conexión entre la parte del grafo correspondiente a la zona ocupada rodeada de obstáculos y la parte del grafo que nunca podría ser explorada por no representar a ninguna región real.

Para el entorno de la Figura 6.14a se muestra tanto la RS como la RF en las Figuras 6.14c y d, respectivamente, una vez el robot ha detenido su exploración. La RS no aporta la información útil que deseamos en relación a la ponderación que los comandos de movimiento de los tres comportamientos sufren en cada punto de la trayectoria. Sin embargo, se presenta aquí para ilustrar con claridad cuál es el comportamiento predominante en cada punto, puesto que al estar trabajando en entornos reales los colores no son tan intensos como en los entornos simulados. Es decir, no existe tanta diferencia entre el peso del comportamiento principal y el de los otros dos como en las pruebas simuladas. La RS de la Figura 6.14c muestra con claridad que mientras el agente está explorando la habitación, el comportamiento que más aporta al comando de movimiento final es el Seguir Pared. Al entrar en el pasillo, como era de esperar, es el comportamiento Seguir Pasillo el que más contribuye. Finalmente, unos instantes antes de que la exploración finalice el robot detecta una pared frontal, por lo que es de nuevo el comportamiento Seguir Pared el que ostenta el mayor peso de la navegación. Estas guías cualitativas se mantienen

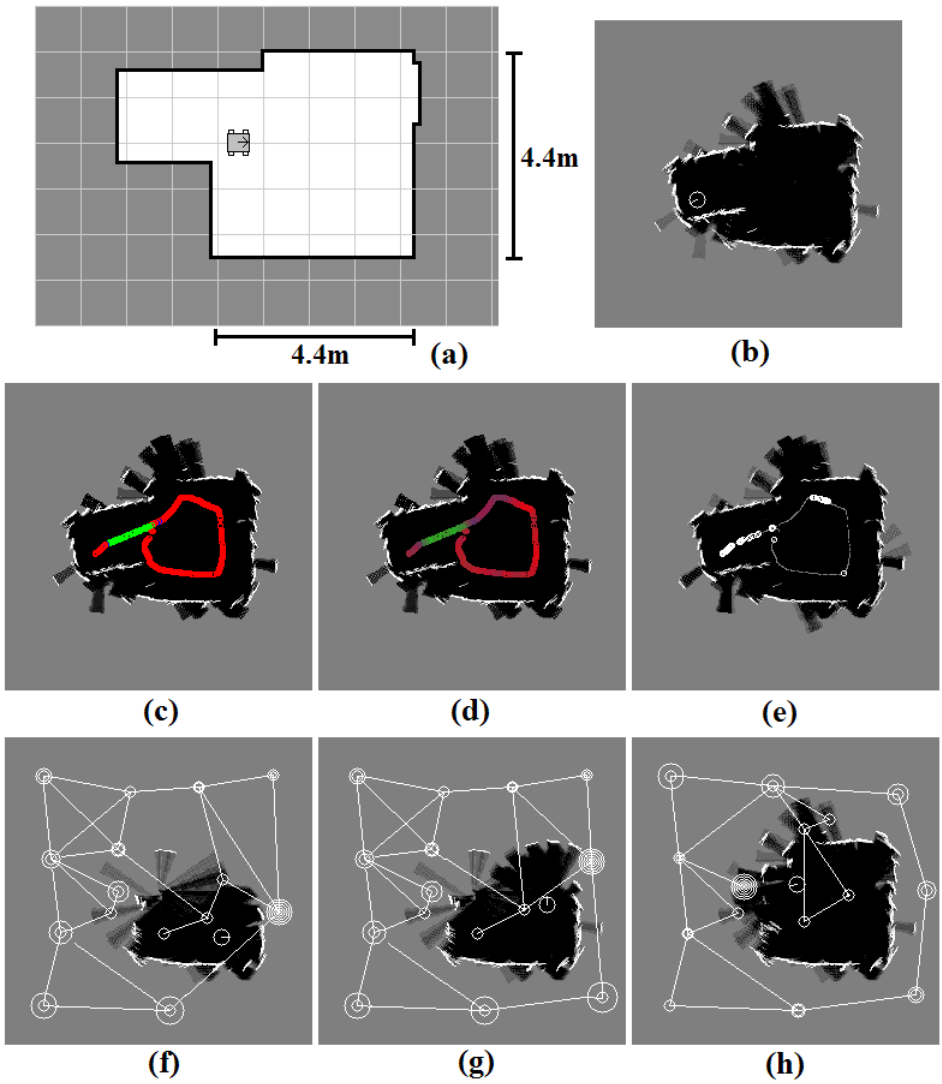

Figura 6.14: Exploración de entorno real: (a) planta; (b) mapa métrico cuando todo el entorno está explorado; (c) *RS*; (d) *RF*; (e) comportamiento Seguir Pared; (f)-(h) mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno en (a).

en la *RF* de la Figura 6.14d, aunque los colores están suavizados porque en esta representación se tienen en cuenta los factores de ponderación asignados a los comandos de movimiento de cada comportamiento en todos los puntos de la trayectoria seguida por el agente. Todo esto se corrobora observando los pocos casos adaptados para el comportamiento Seguir Pared en la Figura 6.14e.

3.4 Entorno real de prueba 4

El entorno de la Figura 6.14a es modificado colocando un obstáculo en medio de la habitación, dando lugar a un nuevo escenario, el de la Figura 6.15a. Tras proceder a su exploración completa se obtiene el modelo de entorno representado en la Figura 6.15b. Cuando el sistema tiene almacenado ese mapa métrico no existe ninguna zona sin explorar, tal y como se ilustra en

el mapa topológico de la Figura 6.15c. Al igual que para el entorno de la Figura 6.13a, el grafo topológico consta de dos subgrafos que no están conectados entre sí. El primero modela topológicamente el espacio libre del entorno con 4 nodos y sus conexiones. El segundo representa el espacio no explorado que no tiene equivalente en el mundo real, consistente en 12 nodos y sus correspondientes conexiones. Este resultado es coherente con el grado de exploración del entorno y con su configuración puesto que, a pesar de que hay regiones sin explorar en el grafo, éstas no son accesibles desde la posición del robot, según se ve en la Figura 6.15c.

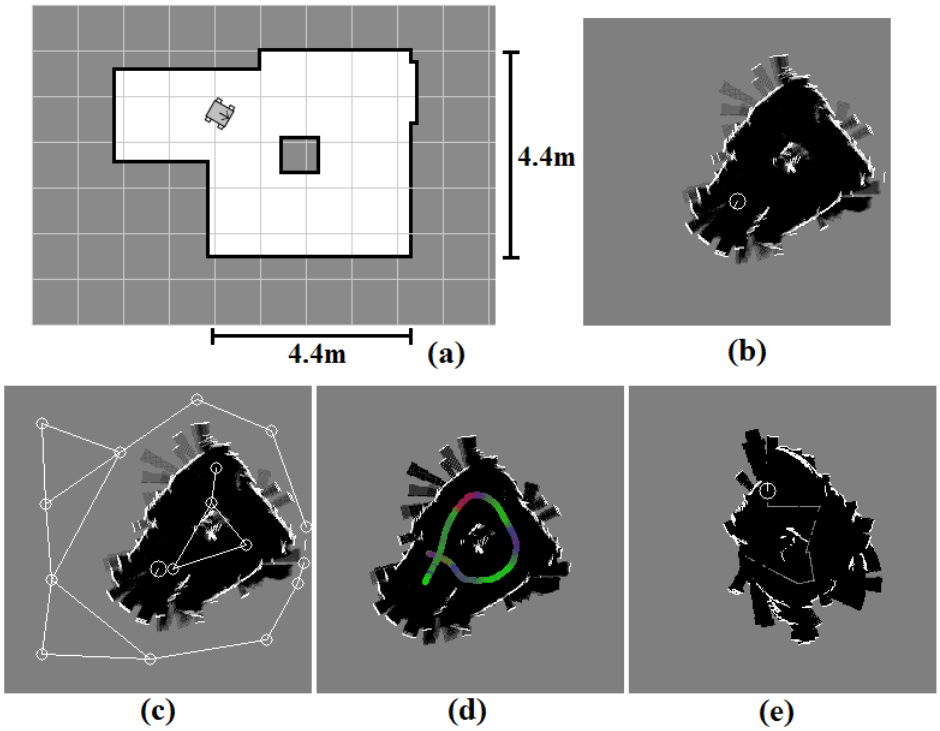

Figura 6.15: Exploración de entorno real: (a) planta; (b) mapa métrico cuando todo el entorno está explorado; (c) mapa topológico sobre el mapa métrico en (b); (d) *RF*; (e) modelo de entorno y trayectoria de exploración aleatoria.

La trayectoria seguida por el agente da lugar a la *RF* de la Figura 6.15d. Aunque la trayectoria recorrida por el agente, gracias a la ruta planificada, es similar a la de la Figura 6.14d, las diferencias cualitativas son notables. En este caso, el comportamiento Seguir Pared solamente es el principal en una breve zona de la trayectoria, al contrario de lo que ocurría para el entorno de la Figura 6.14a. En vez de ese comportamiento, el principal en muchos puntos de la trayectoria es el Seguir Pasillo, porque el agente navega por una zona que detecta como un pasillo. Esto se da porque el robot se mueve por el pasillo, o entre el obstáculo y una pared. También debemos hacer notar que cuando va a girar en las esquinas es el comportamiento Cruzar Puerta el más importante de los tres. Se debe a que el agente detecta la pared derecha y también una parte del obstáculo a su izquierda, lo que hace que vea el entorno como una puerta. Este resultado es coherente con los obtenidos en los entornos simulados, puesto que esta situación también se aprecia en las *RF*s de las Figuras 6.5b, 6.7b, 6.8b y 6.9c.

Para evaluar la eficiencia de la técnica de exploración propuesta se ha repetido la prueba en el entorno de la Figura 6.15a siguiendo una estrategia de exploración errante (*Wander*). El algoritmo usado es simple. El agente avanza en una dirección aleatoria hasta que detecta un obstáculo a una distancia menor que, en nuestro caso, $300mm$. En ese momento se escoge otra dirección aleatoria y se repite el proceso. El modelo de entorno adquirido tras un tiempo de 4 minutos se presenta en la Figura 6.15e, donde la trayectoria recorrida por el agente está superpuesta en gris al mapa métrico. Transcurridos esos 4 minutos, el agente ha sido incapaz de explorar completamente el entorno. De hecho, solamente ha recorrido la habitación, y no de forma completa. Además, la trayectoria seguida es caótica con un sinfín de cambios de dirección, siendo éstos muy bruscos en la mayoría de las ocasiones. En la Figura 6.15d se observa también que la orientación actual del agente hace que éste se dirija de nuevo hacia una zona explorada, no hacia el pasillo que le falta por modelar. Por otro lado, al avanzar aleatoriamente en una dirección el agente a veces se acerca más de lo recomendable a algún obstáculo, con el riesgo que esto tiene para su seguridad. Por lo tanto, esta estrategia no es eficiente para la exploración, ya que introduce un grado de aleatoriedad en la navegación que ni siguiera asegura que se explore por completo el entorno. Con la técnica propuesta en esta Tesis se mejora sustancialmente la exploración errante pues en un tiempo significativamente menor, 2 minutos y 30 segundos, se alcanza el objetivo de la exploración completa del entorno inicialmente desconocido, recorriendo el escenario de una manera ordenada y eficiente.

3.5 Entorno real de prueba 5

El último entorno de prueba real que se presenta es el de la Figura 6.16a. Consiste en una habitación de aproximadamente $4.4x4.4m^2$ de área unida a través de una puerta de $1.1m$ a un pasillo de $3.1m$ de anchura y $5.1m$ de longitud.

La *RF* de la Figura 6.16c muestra la adaptación que la navegación del agente sufre en función de los factores de ponderación calculados en cada punto de la trayectoria. Al igual que para el entorno de la Figura 6.14a, se presenta también la *RS* en la Figura 6.16b, aunque no aporte la información cualitativa que deseamos. Como el entorno de trabajo es real, los colores en la *RF* no son tan intensos como los obtenidos en entornos simulados, de ahí que se haya añadido la *RS* para aportar claridad a los resultados. Al iniciar el avance el mapa local no está conformado completamente, por lo que el comportamiento más influyente no es el esperado Seguir Pasillo. Cuando el mapa local sí que aporta la suficiente información, el agente se encuentra ya cercano a la pared frontal, por lo que el mayor peso lo aporta el comando de movimiento del comportamiento Seguir Pared. La pared se sigue hasta que el robot se acerca a la puerta para entrar en la habitación inferior del entorno. A partir de ese momento se percibe el escenario como un pasillo, porque el agente tiene paredes a ambos lados, aunque esté a punto de cruzar la puerta. En la habitación se pueden hacer los mismos comentarios cualitativos ya realizados para otros entornos. Hay una alternancia de los comportamientos, siendo destacable de nuevo la influencia del comportamiento Cruzar Puerta en los giros. Este resultados es análogo al obtenido para el entorno de la Figura 6.15a, ya que el agente percibe el entorno como una

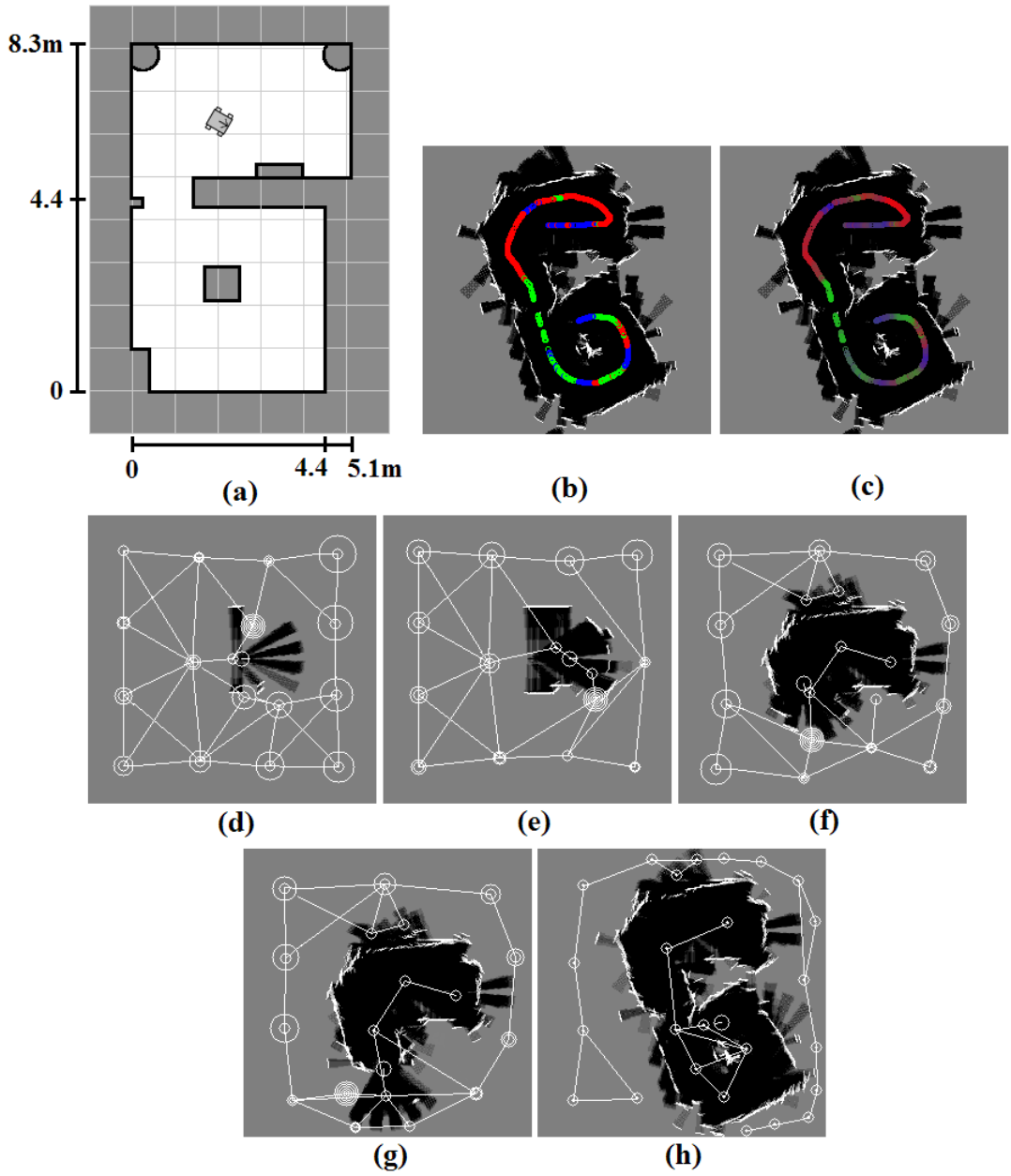

Figura 6.16: Exploración de entorno real: (a) planta; (b) *RS*; (c) *RF*; (d)-(g) mapa métrico, mapa topológico y ruta de exploración en diferentes instantes de la exploración del entorno en (a); (h) mapa topológico sobre el mapa métrico cuando todo el entorno está explorado.

puerta cuando se está dejando atrás el obstáculo a su izquierda y se debe girar porque hay una pared frontal.

La exploración llevada a cabo por el robot finaliza cuando el sistema determina que no existe ninguna región inexplorada a la que se puede llegar desde su posición actual. En ese momento, el modelo de entorno generado es el de la Figura 6.16h, donde se ha superpuesto el mapa topológico correspondiente. La principal característica del grafo es que existen dos subgrafos independientes sin conexión entre sí, al igual que ocurría en las Figuras 6.12c, 6.13d y 6.15c. De

los dos subgrafos, uno es el que modela la zona de espacio libre explorada, mediante 7 nodos y sus conexiones. Debemos hacer notar que en este subgrafo queda plasmada perfectamente la topología del entorno, con dos nodos para representar el pasillo, unidos a la habitación a través de la puerta, existiendo un nodo muy cerca de ella. El otro subgrafo modela las zonas desconocidas no exploradas que, aunque no tienen equivalente en el mundo real, aparecen en la representación. Esto no supone ningún problema para el método propuesto, puesto que desde la posición final del robot no se puede llegar a esas zonas.

Las Figuras 6.16d, e, f y g muestran el estado de la exploración en distintos instantes de la misma. Para cada uno de esos instantes se presenta el modelo de entorno generado hasta el momento, el mapa topológico y la ruta de exploración planificada. En este sentido, destacar que las rutas planificadas constan de 17, 14, 14 y 13 nodos en las Figuras 6.16d, e, f y g, respectivamente. Esos nodos incluyen aquél en el que se encuentra el robot, puesto que es el inicio de la ruta. Por lo tanto, y tal como se analizó en el Capítulo 5, no sería posible emplear la búsqueda exhaustiva para planificar esta ruta de forma rápida para adaptarse a la configuración del entorno que se va explorando. Este hecho demuestra y da significado al empleo del algoritmo heurístico propuesto en esta Tesis para planificar. Además, las rutas obtenidas son coherentes con la configuración y grado de exploración del entorno.

4 Conclusiones

En este capítulo se han presentado las pruebas, tanto en entornos simulados como reales, del sistema de exploración completa cuando todos sus módulos están integrados. Los resultados se obtienen tras hacer que el robot explore entornos inicialmente desconocidos lanzando todos los módulos del sistema en una misma máquina. La interacción de los módulos entre sí se realiza mediante el esquema local síncrono de la arquitectura *DLA*.

Los resultados en entornos simulados demuestran que la navegación del agente se adapta a la configuración instantánea del entorno, haciendo así que el movimiento sea más eficiente. Al mismo tiempo, la contribución de cada comportamiento a lo largo de las trayectorias descritas es coherente con la zona por la que el robot va explorando, al igual que la ruta de exploración completa planificada.

Al trabajar en entorno reales también se ha demostrado, al igual que en los entornos simulados, que la navegación del robot se adapta a la configuración del entorno, gracias a la coherencia en el cálculo de los factores de ponderación de los comportamientos y de la ruta de exploración que se planifica. No obstante, el peso asignado al comportamiento principal en cada punto de la trayectoria no es tan diferente de los otros dos como en el caso simulado. Esto es debido a que la técnica propuesta para su cálculo se basa en la lectura de varios sensores sonar, cuyos errores son más apreciables en entornos reales.

Por todo lo dicho se puede concluir que el sistema de exploración propuesto da cumplimiento al objetivo principal de esta Tesis, alcanzar una exploración completa eficiente de entornos total

o parcialmente desconocidos. Así mismo, el sistema funciona de manera autónoma sin necesidad de supervisión humana.

Capítulo 7

Conclusiones y Líneas Futuras

En este último capítulo se presentan las conclusiones a las que se llega tras la realización de la presente Tesis. También se presentan posibles líneas de investigación futuras, así como las publicaciones que se derivan del trabajo realizado.

A lo largo de los distintos capítulos de este trabajo se han descrito las principales conclusiones relacionadas con el tema tratado en cada uno de ellos. Por lo tanto, en primer lugar se presenta en la Sección 1 un resumen de las principales conclusiones derivadas de la realización de esta Tesis, destacando las aportaciones efectuadas. A continuación se proponen, en la Sección 2, diversas líneas de investigación abiertas que surgen al finalizar este trabajo. Finalmente, se describen en la Sección 3 las contribuciones que están relacionadas con la Tesis.

1 Conclusiones

En esta Tesis se ha presentado un sistema eficiente de navegación basado en comportamientos para la exploración completa de entornos total o parcialmente desconocidos por parte de un agente autónomo móvil. El sistema está basado en tres comportamientos habitualmente utilizados cuando el robot está explorando un área desconocida: Seguir Pared, Seguir Pasillo y Cruzar Puerta. Los entornos hacia los que va dirigido el sistema son de interior, pudiendo ser dinámicos. Las características del sistema lo dotan de la funcionalidad más elevada dentro de la Jerarquía de Navegación, el nivel de Navegación por Inspección. Esas mismas características hacen que el sistema sea eficiente en el uso de los recursos existentes en cuanto al tiempo y la memoria, permitiendo la adaptación a los cambios que sufre el entorno.

El sistema propuesto está basado en una arquitectura híbrida que combina las ventajas de los sistemas reactivos y los deliberados. Se ha optado por esta elección porque actualmente está admitida como la mejor alternativa a la hora de trabajar con agentes autónomos móviles.

Como estilo de la arquitectura se emplea el de la arquitectura *DLA* (*Distributed and Layered Architecture*). De todas las facilidades que proporciona, se emplea el esquema local síncrono, usando los buzones siempre que ha sido posible. Este esquema es el más apropiado en nuestro

caso, puesto que todos los módulos del sistema se ejecutan en la misma máquina. Los resultados han demostrado que la elección ha sido correcta, gracias a que uno de los criterios de diseño para los distintos módulos ha sido el uso eficiente de los recursos temporales disponibles.

La estructura de la arquitectura es novedosa, y se corresponde con la de un sistema híbrido. En ella se distinguen las tres capas típicas de todo sistema híbrido: la capa reactiva, la capa deliberada y la capa intermedia que actúa de interfaz entre el nivel reactivo y el nivel deliberado. Al margen de estas tres capas, la estructura propuesta también cuenta con otras capas y módulos necesarias para la correcta operación de todo el sistema de exploración. Su misión es variada: i) actuar de interfaz entre el robot y el resto del sistema; ii) obtener una representación geométrica global del entorno, útil para realizar tareas de planificación; iii) obtener una representación local del entorno para conseguir una adaptación eficiente de la navegación reactiva; y iv) actuar de interfaz entre el usuario de la aplicación y el sistema.

Las capas reactiva, intermedia y deliberada constituyen el núcleo del sistema desarrollado, descritas en los Capítulos 3, 4 y 5, respectivamente. Concentran las principales aportaciones realizadas con la presente Tesis. Se han analizado y justificado con profundidad a lo largo de los capítulos de este trabajo. A continuación se presenta un resumen de estas aportaciones.

El nivel reactivo del sistema de exploración consta de tres comportamientos, Seguir Pared, Seguir Pasillo y Cruzar Puerta. Las principales aportaciones realizadas a la navegación reactiva con la presente Tesis son:

- Una nueva implementación de los tres comportamientos siguiendo un modelo analítico basado en los campos de potencial. La implementación consiste en la suma ponderada de varias fuerzas que varían dependiendo de la tarea asignada.

- Una nueva metodología de diseño de comportamientos reactivos basada en el ciclo *CBR*. En el diseño se propone usar una adaptación derivada tomando como solución adaptada del caso la proporcionada por el correspondiente modelo analítico. Al emplear una filosofía basada en el *CBR*, el sistema es capaz de aprender por observación y aprender de su propia experiencia. Este aprendizaje es almacenado en una base de casos organizada según una estructura plana. Ambos tipos de aprendizaje dotan de una gran flexibilidad al sistema, al poder éste ser entrenado para solventar situaciones complicadas y para ejecutar la tarea deseada. Al mismo tiempo, si aparecen situaciones nuevas el sistema es capaz de aprenderlas. De este modo, el conocimiento adquirido se puede emplear para resolver situaciones similares en el futuro. La nueva metodología incluye también una fase de clasificación de la base de casos usada durante la operación del robot.

- Un nuevo método de aprendizaje asistido que permite combinar los comandos de movimiento que el robot calcula de forma autónoma con las órdenes que un humano intenta aplicar mediante un joystick. Esta técnica mejora los métodos existentes, puesto que en éstos el control lo ostenta, bien la máquina, bien el humano, provocando cambios bruscos al pasar el control de uno a otro. En este sentido se aporta:

  - Un mecanismo para aumentar o disminuir el peso de los comandos del sistema autónomo y el peso de los comandos del humano en función de la eficiencia de la conducción por parte del usuario. El mecanismo está basado en una nueva métrica, la eficiencia de un comando de movimiento.
  - Una representación visual 3D que asigna un canal de color RGB a cada uno de los tres factores involucrados en la eficiencia. Sirve para analizar las características de la trayectoria del robot desde un punto de vista global y para evaluar la eficiencia en cada punto de dicha trayectoria en función de sus tres factores.

Las principales aportaciones realizadas a la cooperación o fusión de comportamientos con la presente Tesis son:

- Un nuevo vector de características eficiente de dimensiones reducidas de tan solo 9 componentes que sirve para caracterizar un lugar. Además de por su eficiencia, el vector propuesto destaca por el poco tiempo necesario para su cálculo, aproximadamente $40ms$.
- Un nuevo algoritmo de extracción de un vector de características y de caracterización de lugares. Es útil para seleccionar de forma competitiva uno de nuestros comportamientos o para obtener los factores de ponderación de sus comandos de movimiento y hacer que cooperen. Esta segunda es la utilidad dada en la presente Tesis. El método propuesto es original, siendo su punto de partida un mapa local en las cercanías del robot construido a partir de la lectura dada por sensores sonar.
- Una nueva aplicación del Análisis de Componentes Principales (PCA) a un sistema de exploración de entornos total o parcialmente desconocidos. De este modo se consigue la reducción del problema inicial, el mapa local, a un vector de 9 componentes en base a criterios objetivos y cuantitativos, en concreto, al análisis de la varianza explicada por cada componente principal. Se han considerado las primeras 9 componentes porque explican el 99.5% de la información.
- Por el propio diseño del algoritmo de extracción del vector de características, otra aportación es la metodología de caracterización de lugares. En el sistema analizado en la presente Tesis se trabaja con tres comportamientos concretos. Sin embargo, el proceso es aplicable a cualquier otra combinación de comportamientos. Se pueden sustituir los que están presentes actualmente por otros o añadir algún otro comportamiento nuevo. Como tanto la obtención de la base como el cálculo de los patrones son independientes entre sí y, a su vez, independientes de la operación del robot, se puede cambiar la base y/o los patrones para satisfacer las necesidades de otro comportamiento reactivo. Por lo tanto, se puede decir que el sistema es flexible en este sentido.
- Un método original para el cálculo de los factores de ponderación de los comandos de movimiento de los tres comportamientos reactivos del sistema de exploración para hacer que cooperen en todo momento durante la navegación. La técnica se basa en una métrica

definida que permite establecer la relación entre cualquier vector de características y los patrones del sistema.

- Dos nuevas representaciones gráficas, la *Representación de Selección* (*RS*) y la *Representación de Fusión* (*RF*). Aunque ambas son originales, ya se han desarrollado anteriormente representaciones parecidas a la *RS*. Sin embargo, la *RF* sí que es novedosa en su concepción y en la información que aporta. Ambas representaciones permiten analizar los resultados alcanzados por el algoritmo de caracterización. El análisis que proporcionan es doble. Por simple inspección, se pueden obtener resultados cualitativos. Pero también proporcionan resultados cuantitativos analizando el color representado para cada punto de la trayectoria del robot.

Las principales aportaciones realizadas a la planificación de rutas de exploración completa con la presente Tesis son:

- Un nuevo método eficiente para la planificación de una ruta de alto nivel que permita la exploración completa de todas las zonas no exploradas de un entorno total o parcialmente desconocido.
- Una estructura jerárquica que integra los paradigmas métrico y topológico de modelado de entornos que representa explícitamente las regiones no exploradas del entorno, por lo que se pueden planificar rutas a través de áreas no exploradas. La estructura piramidal propuesta soporta el algoritmo de planificación, permitiendo que se alcance la funcionalidad más elevada de la Jerarquía de Navegación, la Navegación por Inspección. El mapa topológico que se obtiene se construye de forma rápida, no supervisada, poco parametrizada e independientemente de la configuración del entorno.
- Un estudio de diferentes algoritmos para la optimización de rutas, que conduce al diseño del algoritmo heurístico que se describe en la siguiente aportación.
- El diseño de un nuevo algoritmo genético para resolver el problema del viajante (*TSP*), aplicado en esta Tesis para la optimización de la distancia recorrida por el agente cuando debe visitar todas las regiones no exploradas del entorno una única vez. Este algoritmo es heurístico y proporciona una ruta subóptima aunque, dependiendo del problema de entrada, puede obtener también la ruta óptima. Los datos de entrada del algoritmo, regiones no exploradas accesibles desde la posición del robot y matriz de distancias, se calculan a partir del mapa topológico propuesto. En el diseño se ha primado el evitar caer en mínimos locales y la obtención de una solución aceptable de forma rápida.

Además de las aportaciones principales realizadas a las capas reactiva, intermedia y deliberada, una aportación fundamental de esta Tesis es la integración de todas las capas y módulos desarrollados en un único sistema útil de exploración completa de entornos total o parcialmente desconocidos. Todos los módulos del sistema se ejecutan en la misma máquina, un Pentium 3 a

550 MHz con 192 MB de memoria RAM. A pesar de que es un máquina de bajas prestaciones, los resultados obtenidos han sido satisfactorios, tanto desde el punto de vista de la operación del robot como del tiempo requerido para la ejecución de los diferentes algoritmos involucrados en el sistema.

Una vez integrado todo el sistema, se han efectuado distintas pruebas en entornos simulados y reales con dos plataformas distintas que constan únicamente de sensores sonar y un sistema de posicionamiento basado en odometría. En el primer caso se emplea el simulador de la plataforma robótica *Nomad 200* de *Nomadics*, con un anillo de 16 sensores sonar equiespaciados. En el segundo se usa la plataforma robótica real *Pioneer P2AT*, equipada con 8 sensores sonar *Polaroid* frontales. De las pruebas efectuadas se extraen las siguientes conclusiones:

- El sistema de exploración propuesto, diseñado y desarrollado da cumplimiento al objetivo principal de esta Tesis, puesto que se alcanza, de manera eficiente, la exploración completa de entornos total o parcialmente desconocidos.

- Se ha comprobado que los comportamientos reactivos implementados incorporan mecanismos de razonamiento y aprendizaje, gracias a la metodología basada en el *CBR*.

- El sistema es independiente de la plataforma empleada, permitiendo su adaptación a una plataforma distinta con pequeños cambios. De nuevo es gracias a una de las dos filosofías seguidas para implementar los comportamientos reactivos, el *CBR*. Este hecho también implica que el sistema se adapta dinámicamente a los cambios o situaciones inesperadas, pudiendo aprender las peculiaridades del entorno. Al mismo tiempo, se elimina la necesidad de tener que considerar la cinemática del robot, pues ésta se tiene en cuenta de manera implícita.

- Se han comparado las dos estrategias seguidas en la implementación de los comportamientos reactivos, el modelo analítico y el *CBR*. Ambas han demostrado que los resultados de la navegación son satisfactorios, tanto en entornos simulados como reales. Sin embargo, el *CBR* supera en las características de operación al modelo analítico. Así, se ha comprobado que al usar el modelo analítico el robot oscila y tiene dificultades para navegar entre obstáculos cercanos. Este resultado era esperable, sobre todo cuando el modelo está basado en los campos de potencial. Sin embargo, las oscilaciones se reducen cuando se trabaja con el *CBR*. Así mismo, se pueden resolver situaciones que con el modelo analítico no era posible, como el paso de una puerta estrecha. Y, por último, al poder aprender de la propia experiencia se pueden resolver situaciones futuras a partir del conocimiento adquirido en el pasado, algo que no es posible al operar con el modelo analítico.

- En cuanto al aprendizaje asistido, se ha realizado un estudio con voluntarios con distintas características que permite extraer como principal conclusión que el control asistido supera al humano. En muchos casos, incluso mejora el funcionamiento del robot, especialmente allí donde el agente tiene problemas. Esto permite absorber las características más positivas

de la conducción de cualquier usuario incluso cuando carece de experiencia previa con los robots y, por tanto, mejora el rendimiento del aprendizaje asistido.

- Un sistema de exploración para entornos total o parcialmente desconocidos basado en comportamientos no puede utilizar un único comportamiento. Las resultados obtenidos demuestran que puede dar lugar a una exploración parcial del entorno y/o a una trayectoria ineficiente que no se adapta adecuadamente a la configuración del entorno. Esto es cierto más aún cuando el robot va navegando por zonas cuya configuración es desconocida.

- Se ha diseñado, implementado y alcanzado un mecanismo eficiente de cooperación de los tres comportamientos reactivos de navegación, Seguir Pared, Seguir Pasillo y Cruzar Puerta, que se adapta a las circunstancias y particularidades del entorno en cada momento.

- La velocidad de reacción y de adaptación a la configuración del entorno es muy rápida, puesto que los factores de ponderación de los tres comportamientos se calculan en aproximadamente $40ms$. Aun cuando el procesado para la obtención del vector de características es intenso y que la máquina usada es de bajas prestaciones, el tiempo requerido es muy pequeño. La principal ventaja de esta conclusión es que la cooperación de los tres comportamientos se va a ajustar de manera fiel a las características de la zona del entorno por la que navega el robot.

- Se ha verificado, a través de diversas pruebas en entornos simulados y reales, que la ponderación de los comportamientos en un escenario arbitrario es coherente con la configuración del entorno detectada. Y todo ello, a pesar de que la base de componentes principales se obtiene a partir de únicamente 17 mapas locales y de que solamente se usan 3 patrones por comportamiento, habiendo sido realizados ambos procesos en entornos simulados. Los resultados han sido satisfactorios, tanto desde un punto de vista cualitativo como cuantitativo.

- Se ha demostrado que el algoritmo de búsqueda exhaustiva es una solución inviable para resolver el *TSP* cuando el número de regiones no exploradas crece por encima de un valor no muy alto como es 10. Ello implica que se debe emplear un algoritmo heurístico para resolver este problema, tal y como se hace en la presente Tesis.

- De todos los algoritmos heurísticos para resolver el *TSP* que se han analizado, el que mejores resultados da es el algoritmo genético. Por este motivo es el empleado en la presente Tesis. Para llegar a esta conclusión se ha comparado la solución proporcionada por diferentes algoritmos ante estándares de comparación (librería *TSPLIB*), y ante problemas en entornos simulados y reales capturados con el robot.

- Los resultados temporales alcanzados para el algoritmo de planificación de rutas de exploración completa a partir de un mapa métrico total o parcialmente inexplorado demuestran su viabilidad para operar mientras el robot navega. Igual de importante es el hecho de que este tiempo es independiente de la configuración del entorno y de su grado de exploración.

- Las rutas obtenidas con el algoritmo de exploración completa cubren de forma íntegra todas las zonas representadas en el mapa métrico de partida, sea éste el correspondiente a un entorno real o simulado. Se ha demostrado también que las rutas son coherentes para un mismo entorno con distinto grado de exploración.

- Las pruebas globales de exploración completa nos permiten extraer una conclusión vital. El sistema funciona de forma autónoma sin supervisión humana mientras el robot navega y explora las zonas que inicialmente eran desconocidas. Estas pruebas globales de exploración se han llevado a cabo con los comportamientos reactivos implementados siguiendo el esquema del *CBR*. Ello implica que el operador humano únicamente tiene que entrenar, si se desea, los tres comportamientos reactivos y dar la orden para iniciar la exploración. La finalización de dicha exploración es detectada automáticamente por el sistema.

2 Líneas futuras

El sistema que se presenta en esta Tesis se centra en la exploración basada en comportamientos de entornos total o parcialmente desconocidos. Este sistema admite posibilidades de mejora y ampliación. A continuación se proponen futuras líneas de investigación que se abren tras el trabajo desarrollado:

- Incorporar nuevos sensores a la plataforma robótica usada en las pruebas. Quizás la opción más inmediata es el láser. Con este dispositivo se conseguiría una mayor exactitud en las lecturas y una mayor resolución, reduciéndose el error en las medidas. Con la incorporación de este sensor se podría mejorar la operación de los comportamientos reactivos, tanto con el modelo analítico como con el *CBR*, y también se facilitaría el proceso de caracterización necesario para ponderar los comandos de movimiento de los comportamientos. También sería posible la inclusión de sensores basados en visión, cámaras, que mejoraran la caracterización de lugares.

- Estudiar la posibilidad de mejorar los comportamientos implementados con el modelo analítico. En este sentido, sería interesante el poder seguir una pared que se encuentra a la izquierda o a la derecha del agente, puesto que en el sistema solamente se puede seguir la que se encuentra a la derecha. De este modo se podría conseguir una mejor adaptación del robot a la configuración del entorno.

- Realizar un estudio más profundo del diseño de comportamientos con la metodología de diseño propuesta que se basa en el *CBR*. En relación a este tema, situaciones como el seguir una pared que se encuentra a la izquierda del robot no suponen un problema, al contrario de lo que sucede al emplear el modelo analítico. Es así porque una de las ventajas del esquema propuesto es que, mediante la fase de aprendizaje por observación, un operador humano puede entrenar al agente para ejecutar la tarea deseada. Sin embargo, sí que se pueden investigar otros aspectos interesantes:

  - Sustituir el guiado de la fase de aprendizaje por observación por parte de un humano por un aprendizaje automático con algún modelo analítico. El humano únicamente supervisaría el comienzo y la finalización del entrenamiento.
  - Analizar nuevos mecanismos dentro del ciclo *CBR*. Entre otros: i) favorecer unos sensores respecto de otros durante la fase de recuperación; ii) emplear otro tipo de adaptación durante la fase reusar, como por ejemplo reutilizar la solución propuesta por el caso recuperado; y iii) evaluar la calidad de las soluciones adaptadas imponiendo algún criterio para la retención de los casos.

  El estudio debería comparar todas las alternativas para determinar cuál es la mejor opción en el diseño de los comportamientos con el *CBR*. El diseño e implementación efectuados en el sistema presentado en esta Tesis cumple con los objetivos planteados, por lo que es posible que otras soluciones no mejoraran la operación del agente.

- Profundizar en el tema del control asistido. El éxito de los experimentos hace que el esquema propuesto sea prometedor para asistir a personas con discapacidades a la hora de navegar en una silla de ruedas. Se plantea también la posibilidad de evaluar el estado del paciente a la hora de combinar las órdenes del sistema autónomo con las órdenes del humano.

- Extender el mecanismo de cooperación a nuevos comportamientos reactivos. En primer lugar, habría que ampliar el número de comportamientos reactivos del sistema. En segundo lugar, habría que modificar el mecanismo de cálculo de los factores de ponderación para adaptarse a los nuevos comportamientos. Requeriría de nuevos patrones y de un nuevo cálculo de los pesos asignados a cada comportamiento. La ventaja del método propuesto en esta Tesis es que, debido a la independencia en la obtención del vector de características, en el cálculo de la base y en el cálculo de los patrones, solamente habría que modificar una pequeña parte del algoritmo.

- Implementar un método de localización global con posición inicial desconocida para el robot. La ventaja del vector de características propuesto es su dimensión reducida, sirviendo para representar una posición del entorno. Ello hace suponer que es apropiado para localizar globalmente al agente en un entorno conocido. De hecho, ya se ha empezado ha trabajar en esta línea. Con el vector de características propuesto se capturan marcas en distintas posiciones del espacio. Se está investigando el uso de una estructura piramidal que segmenta el entorno por marca y por probabilidad de ocupación. Con la segmentación, se está desarrollando una técnica que emplea los modelos ocultos de Markov para localizar al agente.

- Distribuir el algoritmo de cálculo de la ruta de exploración completa en varias máquinas. De este modo se conseguiría mejorar la calidad de la solución sin un aumento del tiempo necesario. En este sentido se podrían usar algoritmos más complejos como los algoritmos meméticos, o algoritmos que combinaran un técnica heurística con la búsqueda exhaustiva.

- Aumentar la funcionalidad del sistema incluyendo alguna capa jerárquicamente superior al planificador de rutas de exploración completa que permita escoger los nodos a visitar de acuerdo a criterios variados.
- Ampliar la interfaz visual del sistema para dar cabida a representaciones 3D. También se podría mejorar esta interfaz incluyendo mecanismos vocales de interacción entre el usuario y el sistema, útiles para ayudar a personas con dificultades motoras. En esta línea ya se ha trabajado, como se presenta en [MW04], Proyecto Fin de Carrera dirigido por el autor de esta Tesis. En ese trabajo se controla la plataforma robótica real *Pioneer P2AT* mediante la voz.
- Aumentar el campo de visión del mapa probabilístico sobre el que se planifican las rutas de exploración completa para abarcar entornos mucho más grandes. A tal efecto habría que: i) ampliar la funcionalidad de la estructura jerárquica propuesta; y ii) mejorar la localización local con técnicas más complejas y exactas.
- Buscar la aplicación real del sistema desarrollado dentro de la robótica de servicio. Entre otras, se proponen aplicaciones domésticas, de vigilancia, exploración de entornos hostiles y asistencia. También se propone la integración de una o varias partes del sistema en otros sistemas, cuyo objetivo sea diferente. Así, se plantea el uso en aplicaciones de guiado, de seguimiento o de planificación. Para ello, es posible que se necesite realizar modificaciones como la inclusión y consideración de nuevos comportamientos, tales como el guiado y/o seguimiento de una persona u objeto, el seguimiento de un camino prefijado o mantenerse alejado de un objeto.

3 Publicaciones

La producción científica relacionada con la presente Tesis consta de Revistas, Capítulos de libro y actas de Congresos. Se enumeran a continuación.

3.1 Revistas

1. **RAS02.** [PPB+02]

Título
Efficient Integration of Metric and Topological Maps for Directed Exploration of Unknown Environments
Revista
Robotics and Autonomous Systems
Resumen
Este artículo describe un algoritmo de exploración completa para entornos parcial o totalmente inexplorados. El algoritmo se basa en una representación que integra los paradigmas métrico y topológico de representación de entornos. La planificación se realiza a dos niveles. A nivel topológico mediante un algoritmo genético. A nivel métrico mediante una aproximación basada en los campos de potencial.

2. **CP02.** [PPU+02c]

Título
Survey Navigation for a Mobile Robot by Using a Hierarchical Cognitive Map
Revista
Cognitive Processing
Resumen
En este artículo se presenta un esquema cognitivo de navegación avanzado para un sistema autónomo móvil. La aproximación está basada en un mapa jerárquico del entorno donde los niveles altos están relacionados directamente con el nivel más bajo. El sistema se emplea para realizar una planificación de rutas a dos niveles.

3. **IASC05.** [PPUS05]

Título
A Hybrid Path Planning Technique for Partially Unknown Indoor Environments
Revista
Intelligent Automation and Soft Computing
Resumen
En este artículo se describe un algoritmo de integración de los paradigmas métrico y topológico de representación del entorno. Está basado en una estructura jerárquica donde existen distintos niveles de resolución. Esta representación se emplea para la planificación de rutas óptimas que un agente autónomo móvil debe seguir.

4. **SC07.** [PUdTS07]

Título
A New Sonar-Based Landmark for Localization in Indoor Environments
Revista
Soft Computing
Resumen
Se presenta en este artículo una nueva marca basada en la lectura de sensores sonar. Su utilidad radica en la posibilidad de emplearlas con el propósito de localizar un agente autónomo en entornos de interior, y caracterizar lugares.

5. **SMCA08.** [PUPS08] [En Revisión]

Título
A New Efficiency-Weighted Strategy for Continuous Human/Robot Cooperation in Navigation
Revista
IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans
Resumen
En este artículo se propone un método que permite la cooperación humano/robot en todos los puntos de una trayectoria, de manera que ambos participen del comportamiento emergente. Para probar el sistema se realiza un estudio con usuarios con diferentes características.

3.2 Capítulos de libro

1. **CISAR02.** [PPU+02b]

Título
Intelligent Navigation in Partially Unknown Indoor Environments

Libro
Computational Intelligent Systems for Applied Research

Resumen
Este capítulo de libro presenta un algoritmo de exploración para entornos total o parcialmente inexplorados. Se basa en una representación métrico-topológica del entorno que planifica rutas a alto nivel mediante un algoritmo genético, el cual proporciona como solución un conjunto de regiones para explorar de forma completa un entorno de forma óptima.

2. **ACSMHC03.** [UPA+03b]

Título
A Topological Map for Scheduled Navigation in a Hospital Environment

Libro
e-Health: Application of Computing Science in Medicine and Health Care

Resumen
En este capítulo de libro se presenta un método para planificar caminos en un entorno hospitalario, con objeto de optimizar diversas tareas relacionadas con un entorno sanitario.

3. **ARS03.** [UBP+03]

Título
Hierarchical Planning in a Mobile Robot for Map Learning and Navigation

Libro
Autonomous Robotic Systems. Soft Computing and Hard Computing Methodologies and Applications

Resumen
Este capítulo se centra en el desarrollo de un sistema híbrido para navegación en entornos desconocidos. El sistema trabaja a alto nivel de un modo deliberativo en función de una representación topológica del entorno, mientras que la navegación reactiva es soportada por un nivel jerárquico inferior que se basa en un mapa métrico.

4. **ACI04** [PUTS04a]

Título
A New Sonar Landmark for Place Recognition

Libro
Applied Computational Intelligence

Resumen
Este capítulo de libro aborda el tema del reconocimiento de lugares en un entorno a través de marcas de localización construidas a partir de sensores sonar. Para llegar a la obtención de la marca se aplica un procesado basado en el análisis de componentes principales.

5. **CAI07** [FEPUS07b]

Título
Collaborative Emergent Navigation Based on Biometric Weighted Shared Control

Libro
Computational and Ambient Intelligence, Lecture Notes in Computer Science 4507

Resumen
En este capítulo se presenta una aproximación novedosa al control asistido. Consiste en combinar las órdenes del humano con las de un agente autónomo móvil. La combinación se basa en dos aspectos: i) la lectura del estado del humano por medio de sensores biométricos; y ii) la medida de la eficiencia de los comandos del humano y del robot.

3.3 Congresos

1. **FLINS02.** [PPU+02a]

Título
Intelligent Navigation in Partially Unknown Indoor Environments

Congreso
5th International FLINS Conference on Computational Intelligent Systems for Applied Research (FLINS2002)

Resumen
Este artículo está dedicado a la planificación de rutas de modo deliberativo, cuyo fundamento es una estructura jerárquica con distintos niveles de resolución, que permite al robot la división del entorno en regiones. La ruta calculada por medio de un algoritmo genético será el orden en el que el robot debe visitar las regiones inexploradas del entorno.

2. **EULAT03.** [UPA+03a]

Título
A Topological Map for Scheduled Navigation in a Hospital Environment

Congreso
EULAT e-Health 2003

Resumen
Este artículo presenta una solución basada en una estructura jerárquica para planificar caminos que resuelvan de manera eficiente ciertas tareas en un entorno hospitalario.

3. **FLINS04.** [PUTS04b]

Título
A New Sonar Landmark for Place Recognition

Congreso
6th International FLINS Conference on Computational Intelligent Systems for Applied Research (FLINS2004)

Resumen
Este artículo aborda el reconocimiento de lugares distintos en un entorno desconocido por medio de marcas sonar. La lectura de los sensores sonar es procesada mediante un análisis de componentes principales que permite comparar y distinguir las marcas entre sí.

4. **ICRA07.** [PUS07]

Título
A CBR Approach to Behaviour-Based Navigation for an Autonomous Mobile Robot
Congreso
2007 IEEE International Conference on Robotics and Automation (ICRA07)
Resumen
En este artículo se presenta un esquema CBR nuevo para navegación basado en tres comportamientos: Seguir Pared, Seguir Pasillo y Cruzar Puerta. El sistema conmuta entre los comportamientos en función de un mapa local de evidencia construido en el entorno cercano al robot.

5. **IWANN07.** [FEPUS07a]

Título
Collaborative Emergent Navigation Based on Biometric Weighted Shared Control
Congreso
9th International Work-Conference on Artificial Neural Networks (IWANN'2007)
Resumen
En este artículo se presenta una aproximación al problema del control asistido. Se aborda desde el punto de vista de la combinación adaptativa, en función del estado de un humano, de las órdenes que dicho humano da a través de un joystick y de las órdenes del agente autónomo.

6. **URSI07.** [UPS$^+$07]

Título
Control Compartido a Nivel Reactivo basado en Eficiencia
Congreso
Actas del XXII Simposium Nacional de la Unión Científica Internacional de Radio (URSI'07)
Resumen
En este artículo se propone un método que permite la cooperación entre un humano y un robot a lo largo de una trayectoria, de manera que ambos participen de un mejor comportamiento emergente, cuyo campo de aplicación futuro es el de las sillas de ruedas.

7. **IROS07.** [UPG$^+$07]

Título
Efficiency based Reactive Shared Control for Collaborative Human/Robot Navigation
Congreso
Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)
Resumen
Este artículo presenta los resultados de navegación obtenidos tras combinar a nivel reactivo la entrada de un humano a través de un joystick con la salida de un campo de potencial puro. La combinación se efectúa en base a una métrica definida que evalúa factores locales.

8. **MELECON08.** [PFEUS08]

Título
Place Characterization for Navigation via Behaviour Merging for an Autonomous Mobile Robot
Congreso
Proceedings of the 14th IEEE Mediterranean Electrotechnical Conference (MELECON 2008)
Resumen
Este artículo presenta una aproximación novedosa para caracterizar y reconocer lugares en entornos desconocidos con el propósito de poder alcanzar una navegación basada en comportamientos. La técnica permite clasificar un lugar en una de tres categorías: pared, pasillo y puerta. No solamente permite seleccionar el comportamiento que mejor casa con la situación detectada, sino que sirve para alcanzar una fusión eficiente de los comportamientos.

Bibliografía

[Aam91] Agnar Aamodt. *A Knowledge Intensive Approach to Problem Solving and Sustained Learning*. Tesis Doctoral, University of Trondheim, Norwegian Institute of Technology, 1991.

[AC87] Philip Agre y David Chapman. Pengi: An Implementation of a Theory of Activity. En *Proceedings of the 6th National Conference on Artificial Intelligence*, pp. 268–272, Seattle, Estados Unidos, Julio 1987.

[AC01] Ercan U. Acar y Howie Choset. Robuts Sensor-Based Coverage of Unstructured Environments. En *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, pp. 61–68, Hawaii, Estados Unidos, Octubre 2001.

[ACF+98] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, y Felix Ingrand. An Architecture for Autonomy. *International Journal of Robotics Research*, 17(4):315–337, 1998.

[AGMR03] Eugenio Aguirre, Moisés Gómez, Rafael Muñoz, y Carmen Ruiz. Un Sistema Multi-Agente que Emplea Visión Activa y Ultrasonidos Aplicado a Navegación con Comportamientos Difusos. En *Actas del IV Workshop on Physical Agens (WAF'2003)*, pp. 63–74, Alicante, España, 2003.

[Alb91] James S. Albus. Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473–509, 1991.

[Alt89] Klaus D. Althoff. Knowledge Acquisition in the Domain of CNC Machine Centers: The MOLTKE Approach. En *Proceedings of the 3rd European Workshop on Knowledge-Based Systems*, pp. 180–195, París, Francia, Julio 1989.

[ALZT04] Chengwan An, Guizhi Li, Yongqian Zhang, y Min Tan. Doorplate Adaptive Detection and Recognition for Indoor Mobile Robot Self-Localization. En *Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics*, pp. 339–343, Shenyang, China, Agosto 2004.

[AM00] Peter Aigner y Brenan J. McCarragher. Modeling and Constraining Human Interactions in Shared Control Utilizing a Discrete Event Framework. *IEEE Transac-*

tions on Systems, Man and Cybernetics - Part A: Systems and Humans, 30(3):369–379, Mayo 2000.

[AMAM98] Muhammed Al-Mulhem y Tareq Al-Maghrabi. Efficient Convex-Elastic Net Algoritm to Solve the Euclidean Travelling Salesman Problem. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 28(4):618–620, Agosto 1998.

[AMF99] Angelo Arleo, José R. Millán, y Dario Floreano. Efficient Learning of Variable-Resolution Cognitive Maps for Autonomous Indoor Navigation. *IEEE Transactions on Robotics and Automation*, 15(6):990–1000, Diciembre 1999.

[Ang89] Colin Angle. Tooth Docs: the Paper. 1989.

[AOA99] N. Aras, B. J. Oommen, y I. K. Altinel. The Kohonen Network Incorporating Explicit Statistics and its Application to the Travelling Salesman Problem. *Neural Networks*, 12(9):1273–1284, Noviembre 1999.

[AP94] Agnar Aamodt y Enric Plaza. Case Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *Artificial Intelligence Communications*, 7(1):39–52, 1994.

[APV00] Benjamín Arenas, Alejandro Pavez, y Rodrigo Vidal. Algoritmo ACO Aplicado al TSP: Resumen de una Experiencia Práctica. En *Workshop Advances & Trends in Artificial Intelligence for Problems Solving, Jornadas Chilenas de Computación*, Santiago de Chile, Chile, 2000.

[Ark89] Ronald C. Arkin. Motor Schema-Based Mobile Robot Navigation. *The International Journal of Robotics Research*, 8(4):92–112, 1989.

[Ark90] Ronald C. Arkin. Integrating Behavioral, Perceptual and World Knowledge in Reactive Navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[Ark98] Ronald C. Arkin. *Behaviour-Based Robotics*. MIT Press, Cambridge, Estados Unidos, 1998.

[Ash91] Kevin D. Ashley. *Modeling Legal Arguments: Reasoning with Cases and Hypotheticals*. MIT Press, Cambridge, Estados Unidos, 1991.

[Asi42] Isaac Asimov. Runaround. *Astounding Science Fiction*, Marzo 1942.

[AY95] Yoshinobu Ando y Shinichi Yuta. Following a Wall by an Autonomous Mobile Robot with a Sonar-Ring. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1995)*, pp. 2599–2606, Nagoya, Japón, Mayo 1995.

[AZC+01] Ercan U. Acar, Yangang Zhang, Howie Choset, Mark Schervish, Albert Costa, Renata Melamud, David Lean, y Amy Graveline. Path Planning for Robotic

Demining and Development of a Test Platform. En *Proceedings of the 2001 International Conference on Field and Service Robotics*, pp. 161 – 168, Helsinki, Finlandia, 2001.

[Bar89] E. Ray Bareiss. *Exemplar Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning*. Academic Press, San Diego, Estados Unidos, 1989.

[BCME+98] Jean-Jacques Borrelly, Eve Coste-Manière, Bernard Espiau, Konstantinos Kapellos, Roger Pissard-Gibollet, Daniel Simon, y Nicolas Turro. The Orccad Architecture. *International Journal of Robotics Research*, 17(4):338–359, 1998.

[BEF96] Johann Borenstein, B. Everett, y Liquiang Feng. *Navigating Mobile Robots: Sensors and Techniques*. A.K. Peters Ltd., Wellesley, Estados Unidos, 1996.

[BFB+05] David J. Bruemmer, Douglas A. Few, Ronald L. Boring, Julie L. Marble, Miles C. Walton, y Curtis W. Nielsen. Shared Understanding for Collaborative Control. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, 25(4):494–504, Julio 2005.

[BK91a] Johann Borenstein y Yoram Korem. The Vector Field Histogram – Fast Obstcle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, Junio 1991.

[BK91b] Johann Borenstein y Yoram Koren. Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance. *IEEE Journal of Robotics and Automation*, 7(4):535–539, 1991.

[BLL92] Jérôme Barraquand, Bruno Langlois, y Jean C. Latombe. Numerical Potencial Field Techniques for Robot path Planning. *IEEE Transactions on Sytems, Man and Cybernectis*, 22(2):224–241, 1992.

[BMT00] Alberto Bemporad, Mauro Di Marco, y Alberto Tesi. Sonar-Based Wall-Following Control of Mobile Robots. *Transactions of the ASME*, 122:226–230, Marzo 2000.

[Bon91] R. Peter Bonasso. Integrating Reaction Plans and Layered Competences Through Synchronous Control. En *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1225–1233, Sydney, Australia, Agosto 1991.

[Bra91] L. Karl Branting. Exploiting the Complementarity of Rules and Precedents with Reciprocity and Fairness. En *Proceedings of the DARPA Case-Based Reasoning Workshop*, pp. 39–50, Washington, Estados Unidos, Mayo 1991.

[Bre03] Cynthia Breazeal. Toward Sociable Robots. *Robotics and Autonomous Systems*, 42(3-4):167–175, Marzo 2003.

[Bro86] Rodney A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, Marzo 1986.

[BSE95] Reinhard Braunstigl, Pedro Sanz, y José M. Ezkerra. Fuzzy Logic Wall Following of a Mobile Robot Based on the Concept of General Perception. En *Proceedings of the 7th International Conference on Advanced Robotics (ICAR'95)*, pp. 367–376, Sant Feliu de Guixols, España, Septiembre 1995.

[Cañ04] Francisco J. Cañadas. Diseño e Implementación de Algoritmos para la Optimización de Rutas de un Agente Autónomo Móvil. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, Málaga, España, 2004.

[Cap20] Karel Capek. *Rossum's Universal Robots*. Oxford University Press, Oxford, Gran Bretaña, 1920.

[CB94] J. Colegrave y A. Branch. A Case Study of Autonomous HouseHold Vacuum Cleaner. En *American Institute of Aeronautics and Astronautics (AIAA)/NASA Conference on Intelligent Robotics, Field, Factory, Service, and Space (CIRFFSS)*, Houston, Estados Unidos, 1994.

[CC98] Jill D. Crisman y Michael E. Cleary. Progress on the Deictic Controlled Wheelchair. En Mittal et al, editor, *Assistive Technology and AI. LNAI 1458*, pp. 137–149. Springer-Verlag, 1998.

[Cha05] Carolina Chang. Using Sensor Habituation in Mobile Robots to Reduce Oscillatory Movements in Narrow Corridors. *IEEE Transactions on Neural Networks*, 16(6):1582–1589, Noviembre 2005.

[CL85] Raja Chatila y Jean P. Laumond. Position Referencing and Consistent World Modelling for Mobile Robots. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 138–170, St. Louis, Estados Unidos, 1985.

[Cla68] Arthur C. Clarke. *2001: A Space Odyssey*. Hutchinson, Gran Bretaña, 1968.

[CMS00] Eve Coste-Manière y Reid Simmons. Architecture: The Backbone of Robotic Systems. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 67–72, San Francisco, Estados Unidos, Abril 2000.

[CO03] Ricardo Carelli y Eduardo Oliveira. Corridor Navigation and Wall-Following Stable Control for Sonar-Based Mobile Robots. *Robotics and Autonomous Systems*, 45:235–247, 2003.

[Con89] Jonathan H. Connell. A Colony Architecture for an Artificial Creature. Technical Report AITR-1151, MIT Artificial Intelligence Laboratory, 1989.

[Con92] Jonathan H. Connell. SSS: A Hybrid Architecture Applied to Robot Navigation. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2719–2724, Niza, Francia, Mayo 1992.

[CP97] Howie Choset y Philippe Pignon. Coverage Path Planning: the Boustrophedon Cellular Decomposition. En *Proceedings of the International Conference on Field and Service Robotics*, Camberra, Australia, Diciembre 1997.

[CP01] James L. Crowley y Fabrice Pourraz. Continuity Properties of the Appearance Manifold for Mobile Robot Position Estimation. *Image and Vision Computing*, 19:741–752, 2001.

[Cro85] James L. Crowley. Navigation for an Intelligent Mobile Robot. *IEEE Journal on Robotics and Automation*, 1:31–41, Marzo 1985.

[CSU97] Dany Z. Chen, Robert J. Szczerba, y John J. Uhran. A Framed-Quadtree Approach for Determining Euclidean Shortest Paths in 2D Environment. *IEEE Transactions on Robotics and Automation*, 12(5):668–680, Octubre 1997.

[CV90] Jonathan H. Connell y Paul Viola. Cooperative Control of a Semi-Autonomous Mobile Robot. En *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1118–1121, Cincinnati, Estados Unidos, 1990.

[CWS98] James L. Crowley, Frank Wallner, y Bernt Schiele. Position Estimation Using Principal Components of Range Data. *Robotics and Autonomous Systems*, 23(4):267–276, Julio 1998.

[Día96] Adenso Díaz. *Optimización Heurística y Redes Neuronales*. Paraninfo, Madrid, España, 1996.

[Dar59] Charles Darwin. *The Origin of Species*. John Murray, Londres, Gran Bretaña, 1859.

[Dav85] Lawrence Davis. Applying Adaptive Algorithms to Epistatic Domains. En *Proceedings of the International Conference on Artificial Intelligence*, pp. 162–164, Princeton, Estados Unidos, Abril 1985.

[DFJ54] G. B. Dantzig, D. R. Fulkerson, y S. M. Johnson. Solution of a Large-Scale Travelling Salesman Problem. *Operations Research*, 2:393–410, 1954.

[DG97] Marco Dorigo y Luca M. Garbandella. Ant Colonies for the Travelling Salesman Problem. *Biosystems*, 43:73–81, 1997.

[DH04] Gu Dongbing y Hu Huosheng. Teaching Robots to Coordinate its Behaviours. En *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA 2004)*, pp. 3721–3726, Nueva Orleans, Estados Unidos, Abril 2004.

[Dij59] Edsger W. Dijkstra. A Note on Two Problems in Conexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DJ96] Hansye S. Dulimarta y Anil K. Jain. A Client/Server Control Architecture for Robot Navigation. *Pattern Recognition*, 29(8):1259–1284, Agosto 1996.

[DJ06] Quoc V. Do y Lakhmi C. Jain. A Visual Landmark Recognition System for Autonomous Robot Navigation. En *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, pp. 237–242, Sydney, Australia, 2006.

[DK07] Christian Dornhege y Alexander Kleiner. Fully Autonomous Planning and Obstacle Negotiation on Rough Terrain Using Behavior Maps. En *Proceedings of the 2007 IEEE/RSJ International Conferencer on Intelligent Robots and Systems (IROS 2007)*, pp. 2561–2562, San Diego, Estados Unidos, Noviembre 2007.

[DL42] G.C. Danielson y C. Lanczos. Some Improvements in Practical Fourier Analysis and their Application to X-Ray Scattering from Liquids. *Journal of the Franklin Institute*, 233:365–380, 435–452, 1942.

[DMC97] Marco Dorigo, Vittorio Maniezzo, y Alberto Colorni. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics*, 26(1):29–41, Febrero 1997.

[DMS99] Göksel Dedeoglu, Maja J. Mataric, y Gaurav S. Sukhatme. Incremental, On-Line Topological Map Building with a Mobile Robot. En *Proceedings of Mobile Robots IV (SPIE'99)*, pp. 129–139, Boston, Estados Unidos, Septiembre 1999.

[DN99] Tom Duckett y Ulrich Nehmzow. Exploration of Unknown Environments Using a Compass, Topological Map and Neural Network. En *Proceedings of the 1999 International Symposium on Computational Intelligence in Robotics and Automation (CIRA'99)*, pp. 312–317, Monterey, Estados Unidos, Noviembre 1999.

[Dor92] Marco Dorigo. *Optimization, Learning and Natural Algorithms.* Tesis Doctoral, Politecnico di Milano, Milán, Italia, 1992.

[Elf87] Alberto Elfes. Sonar-Based Real-World Mapping and Navigation. *IEEE Journal of Robotics and Automation*, 3:249–265, Junio 1987.

[FBSC04] Eduardo Freire, Teodiano Bastos, Mario Sarcinelli, y Ricardo Carelli. A New Mobile Robot Control Approach Via Fusion of Control Signals. *IEEE Transactions on Systems, Man and Cybernetics-Parb B: Cybernetics*, 34(1):419–429, Febrero 2004.

[FBT97] Dieter Fox, Wolfram Burgard, y Sebastian Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation*, 4(1):23–33, Marzo 1997.

[FEPUS07a] Blanca Fernández-Espejo, Alberto Poncela, Cristina Urdiales, y Francisco Sandoval. Collaborative Emergent Navigation Based on Biometric Weighted Shared Control. En *Proceedings of the 9th International Work-Conference on Artificial Neural Networks 2007 (IWANN 2007)*, pp. 814–821, San Sebastián, España, Junio 2007.

[FEPUS07b] Blanca Fernández-Espejo, Alberto Poncela, Cristina Urdiales, y Francisco Sandoval. Collaborative Emergent Navigation Based on Biometric Weighted Shared Control. En Francisco Sandoval, Alberto Prieto, Joan Cabestany, y Manuel Graña, editors, *Computational and Ambient Intelligence, Lecture Notes in Computer Science 4507*, pp. 814–821. Springer, 2007.

[Fir87] R. James Firby. An Investigation into Reactive Planning in Complex Domains. En *Proceedings of the 6th National Conference on Artificial Intelligence*, pp. 202–207, Seattle, Estados Unidos, Julio 1987.

[Fir89] R. James Firby. *Adaptive Execution in Complex Dynamic Worlds.* Tesis Doctoral, Department of Computer Science, Yale University, New Haven, Estados Unidos, 1989.

[FL95] Susan Fox y David B. Leake. Combining Case-Based Planning and Introspective Reasoning. En *Proceedings of the 6th Midwest Artificial Intelligence and Cognitive Science Society Conference*, Carbondale, Estados Unidos, 1995.

[FLD05] Udo Frese, Per Larsson, y Tom Duckett. A multigrid algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[FM00] Matthias O. Franz y Hanspeter A. Mallot. Biomimetic Robot Navigation. *Robotics and Autonomous Systems*, 30(1-2):133–153, Enero 2000.

[FND03] Terrence Fong, Illah Nourbakhsh, y Kerstin Dautenhahn. A Survey of Socially Interactive Robots. *Robotics and Autnomous Systems*, 42:143–166, 2003.

[For88] J. C. Fort. Solving a Combinatorial Problem Via Self-Organizing Process: an Application of the Kohonen Algorithm to the Travelling Salesman Problem. *Biological Cybernetics*, 59:33–40, 1988.

[FS93] James A. Freeman y David M. Skapura. *Redes Neuronales: Algoritmos, Aplicaciones y Técnicas de Programación.* Addison Wesley, Nueva York, Estados Unidos, 1993.

[FS00] Elisabetta Fabrizi y Alessandro Saffiotti. Extracting Topology-Based Maps from Gridmaps. En *Proceedings of the IEEE Conference on Robotics and Automation (ICRA 2000)*, pp. 2972–2978, San Francisco, Estados Unidos, Abril 2000.

[FTB03] Terrence Fong, Charles Thorpe, y Charles Baur. Robot, Asker of Questions. *Robotics and Autonomous Systems*, 42:235–243, 2003.

[Gal90] C.R. Gallistel. *The Organisation of Learning*. MIT Press, Cambridge, Estados Unidos, 1990.

[Gar00] Jaime García. *Manual para el Cubo Rubik*. Ediciones García, Madrid, España, 2000.

[Gat91] Erann Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots*. Tesis Doctoral, Virginia Polytechnic Institute and State University, Blacksburg, Estados Unidos, 1991.

[Gat98] Erann Gat. On Three-Layer Architectures. En David Kortenkamp, R. Peter Bonasso, y Robin Murphy, editors, *Artificial Intelligence and Mobile Robots*, pp. 195–210. AAAI Press, 1998.

[GB05] Grzegorz Granosik y Johann Borenstein. The OmniTread Serpentine Robot with Pneumatic Joint Actuation. En *Proceedings of the 5th International Workshop on Robot Motion and Control*, pp. 105–110, Dymaczewo, Polonia, Junio 2005.

[GB06] Yi Guo y Mohanakrishnan Balakishnan. Complete Coverage Control for Non-holonomic Mobile Robots in Dynamic Environments. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2006)*, pp. 1704–1709, Orlando, Estados Unidos, Mayo 2006.

[GGRG85] John J. Grefenstette, Rajeev Gopal, Brian J. Rosmaita, y Dirk Van Gucht. Genetic Algorithms for the Traveling Salesman Problem. En *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, pp. 160–165, 1985.

[GL93] Fred Glover y Manuel Laguna. Tabu Search. En C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pp. 71–140. Blackwell Scientific Publishing, 1993.

[Glo77] Fred Glover. Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences*, 8:156–166, 1977.

[Glo89] Fred Glover. Tabu Searh: Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[GM03] Fred Glover y Belén Melián. Tabu Search. *Revista Iberoamericana de Inteligencia Artificial*, 19:29–48, 2003.

[GP95] Andrew H. Gee y Richard W. Prager. Limitations of Neural Networks for Solving Traveling Salesman Problems. *IEEE Transactions on Neural Networks*, 6(1):280–282, Enero 1995.

[GT94] Chris Gourley y Mohan Trivedi. Sensor Based Obstacle Avoidance and Mapping for Fast Mobile Robots. En *Proceedings of the 1994 IEEE International Conference*

on Robotics and Automation, pp. 1306–1311, San Diego, Estados Unidos, Mayo 1994.

[GTR07] GTRob. *Libro Blanco de la Robótica: De la Investigación al Desarrollo Tecnológico y Aplicaciones Futuras.* Comité Español de Automática, 2007.

[GZ95] Philippe Gaussier y Stéphane Zrehen. Perac: a Neural Architecture to Control Artificial Animals. *Robotics and Autonomous Systems*, 16(2-4):291–320, Diciembre 1995.

[Ham89] Kristian J. Hammond. *Case Based Planning: Viewing Planning as a Memory Task.* Academic Press, San Diego, Estados Unidos, 1989.

[HB96] Huosheng Hu y Michael Brady. A Parallel Processing Architecture for Sensor Based Control of Intelligent Mobile Robots. *Robotics and Autonomous Systems*, 17:235–257, 1996.

[HHW02] John D. Holliday, Chang Y. Hu, y Peter Willet. Grouping of Coefficients for the Calculation of Inter-Molecular Similarity and Dissimilarity Using 2D Fragment Bit-Strings. *Combinatorial Chemistry & High Throughput Screening*, 5(2):155–166, 2002.

[Hin92] Thomas R. Hinrichs. *Problem Solving in Open Worlds: A Case Study in Design.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1992.

[HLD03] J.B. Hayet, F. Lerasle, y M. Devy. Visual Landmark Detection and Recognition for Mobile Robot Navigation. En *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, Estados Unidos, Junio 2003.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1975.

[Hop82] John Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. En *Proceedings of the National Academy of Sciences of the USA*, pp. 2554–2558, Abril 1982.

[Hot33] Harold Hotelling. Analysis of Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24:417–441, 498–520, 1933.

[Hro03] Juraj Hromkovic. *Algorithms for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, Heuristics.* Springer Verlag, Nueva York, Estados Unidos, 2003.

[HS95] Christian Hofner y Günther Schmidt. Path Planning and Guidance Techniques for an Autonomous Mobile Cleaning Robot. *Robotics and Autonomous Systems*, 14:199–212, 1995.

[HS05] Yukio Horiguchi y Tetsuo Sawaragi. Effects of Probing to Adapt Machine Autonomy in Shared Control Systems. En *Proceedings of the International Conference on Systems, Man and Cybernetics*, pp. 317–323, Hawaii, Estados Unidos, Octubre 2005.

[HSWW03] John D. Holliday, Naomie Salim, Martin Whittle, y Peter Willett. Analysis and Display of the Size Dependence of Chemical Similarity Coefficients. *Journal of Chemical Information and Computer Sciences*, 43:819–828, 2003.

[HTW95] Alain Hertz, Eric Taillard, y Dominiquede Werra. A Tutorial on Tabu Seach. En *Proceedings of the Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)*, pp. 13–24, Italia, 1995.

[HV95] Karen Z. Haigh y Manuela M. Veloso. Route Planning by Analogy. En *Proceedings of the International Conference on Case-Based Reasoning*, pp. 160–180, Berlín, Alemania, 1995.

[Jon93] Judson P. Jones. Real-Time Construction of Three-Dimensional Occupancy Grids. En *Proceedings of the 1993 International Conference on Robotics and Automation*, pp. 52–57, Atlanta, Estados Unidos, Mayo 1993.

[JP85] David S. Johnson y Christos H. Papadimitriou. Computational Complexity. En E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, y D.B Shmoys, editors, *The Travelling Salesman Problem*, pp. 37–85. John Wiley and Sons, 1985.

[Kae86] Leslie P. Kaelbling. An Architecture for Intelligent Reactive Systems. Technical Report 400, SRI International, 1986.

[Kae88] Leslie P. Kaelbling. Goals as Parallel Program Specifications. En *Proceedings of the National Conference on Artificial Intelligence*, pp. 60–65, Saint Paul, Estados Unidos, Agosto 1988.

[Kal60] Rudolph E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the American Society of Mechanical Engineers, Journal of Basic Engineering*, 82(D):35–45, 1960.

[KB91] Benjamin Kuipers y Yung T. Byun. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Robotics and Autonomous Systems*, 8:47–63, Noviembre 1991.

[KB99] Benjamin J. Krose y R. Bunschoten. Probabilistic Localization by Appearance Models and Active Vision. En *Proceedings of the 1999 International Conference on Robotics and Automation (ICRA 1999)*, pp. 2255–2260, Detroit, Estados Unidos, Mayo 1999.

[KBRS97] David Kortenkamp, R. Peter Bonasso, Dan Ryan, y Debbie Schreckenghost. Traded Control with autonomous Robots as Mixed Initiative Interaction. En *Proceedings of the AAAI Spring Symposium on Mixed Initiative Interaction*, Stanford, Estados Unidos, 1997.

[Kha86] Oussama Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[Kir84] S. Kirkpatrick. Optimization by Simulated Annealing: Quantitative Studies. *Journal of Statistical Physics*, 34:975–986, 1984.

[KK05] Shahida Khatoon y Sofia Khatoon. Behavior Coordination of Autonomous Mobile Robot Navigation by Neuro-Fuzzy System. En *Proceedings of the IEEE 31st Annual Northeast Bioengineering Conference*, pp. 56–60, Estados Unidos, Abril 2005.

[KM98] Kurt Konolige y Karen Myers. The Saphira Architecture for Autonomous Mobile Robots. En David Kortenkamp, R. Peter Bonasso, y Robin Murphy, editors, *AI-Based Mobile Robots: Case Studies of Successful Robot Systems*, pp. 211–242. MIT Press, 1998.

[Koc85] E. Koch. Simulation of Path Planning for a System with Vision and Map Updating. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 146–160, St. Louis, Estados Unidos, 1985.

[Koh82] Teuvo Kohonen. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43:59–69, 1982.

[Kol83] Janet Kolodner. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7:243–280, 1983.

[Kol93] Janet L. Kolodner. *Case Based Reasoning*. Morgan Kaufmann, San Mateo, Estados Unidos, 1993.

[Kot89] Phyllis Koton. *Using Experience in Learning and Problem Solving*. Tesis Doctoral, Laboratory of Computer Science, Massachusetts Institute of Technology, Massachusetts, Estados Unidos, 1989.

[Kru03] Maarja Kruusmaa. Global Navigation in Dynamic Environments Using Case-Based Reasoning. *Autonomous Robots*, 14:71–91, 2003.

[KS90] M. Kirby y L. Sirovich. Application of the Karhunen Loeve Procedure for the Characterization of Human Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, Enero 1990.

[KSEP00] Gerhard Kraetzschmar, Stefan Sablatnög, Stefan Enderle, y Günther Palm. Application of Neurosymbolic Integration for Environment Modelling in Mobile Robots.

En Wermter & Sun editors, editor, *Hybrid Neural Systems*, pp. 387–401. Springer Verlag, 2000.

[KST+97] N. L. Katevas, N. M. Sgouros, S. G. Tzafestas, G. Papakonstantinou, P. Beattie, J. M. Bishop, P. Tsanakas, y D. Koutsouris. The Autonomous Mobile Robot SENARIO: A Sensor-Aided Intelligent Navigation System for Powered Wheelchairs. *IEEE Robotics Automation Magazine*, 4(4):60–70, 1997.

[KWLV05] Jonathan Kofman, Xianghai Wu, Timothy J. Luu, y Siddharth Verma. Teleoperation of a robot manipulator using a vision-based human-robot interface. *IEEE Transactions on Industrial Electronics*, 52(5):1206–1219, Octubre 2005.

[LA01] Maxim Likhachev y Ronald C. Arkin. Spation-Temporal Case-Based Reasoning for Behavioral Selection. En *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA 2001)*, pp. 1627–1634, Seúl, Corea del Sur, Mayo 2001.

[LA05] Mesbahi Larbi y Benyettou Aek. Mobile Robot Recognition Using Bayesian Penalization with Neural Approach. En *Proceedings of the 2005 ISSC Congress on Computational Intelligence Methods and Applications*, Estambul, Turquía, Diciembre 2005.

[LAP+04] Guizhi Li, Chengwan An, Jie Pang, Min Tan, y Xuyan Tu. Place Recognition Based on Gabor Filters and SVMs for Mobile Robot Self-Localization. En *Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 1112–1116, Singapur, Singapur, Diciembre 2004.

[LC94] Jang G. Lee y H. Chung. Global Path Planning for Mobile Robot with Grid-Type World Model. *Robotics and Computer-Integrated Manufacturing*, 11(1):13–21, Marzo 1994.

[LKKA05] Maxim Likhachev, Michael Kaess, Zsolt Kira, y Ronald C. Arkin. Spatio-Temporal Case-Based Reasoning for Efficient Reactive Robot Navigation. Technical Report ADA442283, Georgia Institute of Technology, Carnegie Mellon University, 2005.

[LKM+99] P. Larrañaga, C. M. Kuijpers, R. H. Murga, I. Inza, y S. Dizdarevic. Genetic Algorithms for the Travelling Salesman Problem: A Review on Representations and Operators. *Artificial Intelligence Review*, 13(2):129–170, Abril 1999.

[LL90] Tod S. Levitt y Daryl T. Lawton. Qualitative Navigation for Mobile Robots. *Artificial Intelligence*, 44:305–360, 1990.

[LM97] Feng Lu y Evangelos Milios. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. *Journal of Intelligent Robotic Systems*, 18:249–275, 1997.

[LOC00] Mattias Lindström, Anders Orebäck, y Henrik I. Christensen. BERRA: a Research Architecture for Service Robots. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000)*, pp. 3278 –3283, San Francisco, Estados Unidos, Abril 2000.

[LP81] Tomás Lozano-Pérez. Automatic Planning of Manipulator Transfer Movements. *IEEE Transactions on Systems, Man and Cybernetics*, 11(10):781–798, Octubre 1981.

[LR02] Axel Lankenau y Thomas Röfer. Mobile Robot Self-Localization in Large-Scale Environments. En *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002)*, pp. 1359–1365, Washington, Estados Unidos, Mayo 2002.

[LV05] Randel A. Lindemann y Chris J. Voorhees. Mars Exploration Rover Mobility Assembly Design, Test and Performance. En *Proceedings of the 2005 IEEE Conference on Systems, Man and Cybernetics*, pp. 450–455, Hawaii, Estados Unidos, Octubre 2005.

[MAL+05] Skye McLachlan, Jaimal Arblaster, D. K. Liu, Jaime Valls, y L. Chenoweth. A Multi-Stage Shared Control Method for an Intelligent Mobility Assistant. En *Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics*, pp. 426–429, Chicago, Estados Unidos, Julio 2005.

[Mar93] Dario Maravall. *Reconocimiento de Formas y Visión Artificial*. RA-MA, Madrid, España, 1993.

[Mar02] Manuel J. Martínez. Integración de Representaciones Métrico-Topológicas para un Agente Autonómo Móvil. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, Málaga, España, 2002.

[Mar03] Rafael Martí. Procedimientos metaheurísticos en optimización combinatoria. *Matemátiques*, 1(1):3–62, 2003.

[Mat91] Maja J. Mataric. *A Neurobiologically-Inspired Model for Robot Spatial Representation*. MIT Press, Cambridge, Estados Unidos, 1991.

[Mat92] Maja J. Mataric. Integration of Representation into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, Junio 1992.

[Mat97] Maja J. Mataric. Behavior-Based Control: Examples from Navigation, Learning and Group Behavior. *Journal on Experimental Theoretical Artificial Intelligence, Special Issue: Software Architecture Physics Agents*, 9:46–54, 1997.

[MDG+92] David P. Miller, Rajiv S. Desai, Erann Gat, Robert Ivlev, y John Loch. Reactive Navigation Through Rough Terrain: Experimental Results. En *Proceedings of the National Conference on Artificial Intelligence*, pp. 823–828, San Jose, Estados Unidos, Julio 1992.

[MDK+03] Aaron Morris, Raghavendra Donamukkala, Aanuj Kapuria, Aaaron Steinfeld, Judith T. Matthews, Jacqueline Dunbar-Jacob, y Sebastian Thrun. A Robotic Walker that Provides Guidance. En *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003)*, pp. 25–30, Taipei, Taiwan, Septiembre 2003.

[MDS93] David J. Musliner, Edmund H. Durfee, y Kang G. Shin. CIRCA: a Cooperative Intelligent Real-Time Control Architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561–1574, 1993.

[Med98] Josep R. Medina. Algoritmos Genéticos para la Optimización de Redes de Distribución. En *Actas del X Congreso Panamericano de Ingeniería y Tránsito*, pp. 339–347, Santander, España, 1998.

[Mil85] David P. Miller. A Spatial Representation System for Mobile Robots. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 122–128, St. Louis, Estados Unidos, 1985.

[Mil98] David P. Miller. Assistive Robotics: an Overview. En Mittal et al., editor, *Assistive Technology and AI. LNAI 1458*, pp. 126–136. Springer-Verlag, 1998.

[Mis04] Alfonsas Misevicius. Using Iterated Tabu Search for the Travelling Salesman Problem. *Information Technology and Control*, 3(32):29–40, 2004.

[MJ98] Fernando Matía y Agustín Jiménez. Multisensor Fusion: An Autonomous Mobile Robot. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 22(2):129–141, 1998.

[MM00] Javier Mínguez y Luis Montano. Nearness Diagram Navigation (ND): a New Real Time Collision Avoidance Approach. En *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, pp. 21–26, Takamatsu, Japón, Noviembre 2000.

[MMBB06] M. Mucientes, D.L. Moreno, A. Bugarín, y S. Barro. Evolutionary Learning of a Fuzzy Controller for Wall-Following Behavior in Mobile Robotics. *Soft Computing*, 10:881–889, 2006.

[Mor88] Hans P. Moravec. Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, 9:61–74, 1988.

[MPSU01] Alessandro Micarelli, Stefano Panzieri, Lorenzo Sciavicco, y Giovanni Ulivi. Landmark Recognition in Indoor Navigation by Fuzzy Maps and CBR. *Lecture Notes in Control and Information Sciences*, 270:227–250, 2001.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall Rosenbluth, Augusta H. Teller, y Edward Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21(6):1087–1092, Junio 1953.

[MSB05] Oscar Martínez, Cyrill Stachniss, y Wolfram Burgard. Supervised Learning of Places from Range Data Using AdaBoost. En *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pp. 1730–1735, Barcelona, España, Abril 2005.

[MV01] Aleix M. Martínez y Jordi Vitriá. Clustering in Image Space for Place Recognition and Visual Annotations for Human-Robot Interaction. *IEEE Transactions on Systems, Man, And Cybernetics–Part B: Cybernetics*, 31(5):669–682, Octubre 2001.

[MW04] Ivson P. Lemos Monteiro-Wanderley. Interfaz para el Control Remoto de un Agente Autónomo Móvil. Proyecto Fin de Carrera, Departamento de Tecnología Electrónica, Universidad de Málaga, Málaga, España, 2004.

[MYM05] Seo Min, Kim Young, y Lim Myo. Door Traversing for a Vision-Based Mobile Robot Using PCA. *Lecture Notes in Artificial Intelligence*, 3684:525–531, Agosto 2005.

[NCOA95] Paul Nisbet, John Craig, Phil Odor, y Stuart Aitken. 'Smart' Wheelchairs for Mobility Training. *Technology and Disability*, 5:49–62, 1995.

[Nil82] Nils J. Nilsson. *Principles of Artificial Intelligence*. Springer Verlag, Berlin, 1982.

[NVVR97] R.Ñeumann, H. A. Vidal, P. Vieira, y M. I. Ribeiro. Complete Path Planning and Guidance for Cleaning Robots. En *IEEE International Symposium on Industrial Electronics*, pp. 677–682, Guimaraes, Portugal, Julio 1997.

[OC03] Anders Orebäck y Henrik I. Christensen. Evaluation of Architectures for Mobile Robotics. *Autonomous Robots*, 14:33–49, 2003.

[Oeh92] R. Oehlmann. Learning Causal Models by Self-Questioning and Experimentation. En *Proceedings of the AAAI-92 Workshop on Communicating Scientific and Technical Knowledge*, pp. 73–80, Menlo Park, Estados Unidos, Julio 1992.

[OPC01] Joon S. Oh, Jin B. Park, y Yoon H. Chai. Complete Coverage Navigation of Clean Robot Based on Triangular Cell Map. En *Proceedings of the 2001 IEEE International Symposium on Industrial Electronics (ISIE 2001)*, pp. 2089–2093, Pusa, Corea del Sur, 2001.

[OVU95] Giuseppe Oriolo, Marilena Vendittelli, y Giovanni Ulivi. On-Line Map Building and Navigation for Autonomous Mobile Robots. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2900–2906, Nagoya, Japón, Mayo 1995.

[OY82] C. O'Dúnlaing y C. K. Yap. A Retraction Method for Planning the Motion of a Disk. *Journal of Algorithms*, 6:104–111, 1982.

[Pay86] David W. Payton. An Architecture for Reflexive Autonomous Vehicle Control. En *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1838–1845, San Francisco, Estados Unidos, Mayo 1986.

[PCJC06] Andrzej Pronobis, Barbara Caputo, Patric Jensfelt, y Henrik I. Christensen. A Discriminative Approach to Robust Visual Place Recognition. En *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, pp. 3829–3836, Pekín, China, Octubre 2006.

[Pea01] Karl Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2:559–572, 1901.

[Pea84] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley, Nueva York, 1984.

[PFEUS08] Alberto Poncela, Blanca Fernández-Espejo, Cristina Urdiales, y Francisco Sandoval. Place Characterization for Navigation via Behaviour Merging for an Autonomous Mobile Robot. En *Proceedings of the 14th IEEE Mediterranean Electrotechnical Conference (MELECON 2008)*, Ajaccio, Francia, Mayo 2008.

[PGKO04] Sarangi P. Parikh, Valdir Grassi, Vijay Kumar, y Jun Okamoto. Incorporating User Inputs in Motion Planning for a Smart Wheelchair. En *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA 2004)*, pp. 2043–2048, New Orleans, Estados Unidos, Abril 2004.

[PGKO05] Sarangi P. Parikh, Valdir Grassi, Vijay Kumar, y Jun Okamoto. Usability Study of a Control Framework for an Intelligent Wheelchair. En *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pp. 4745–4750, Barcelona, España, Abril 2005.

[PGS04] S. Parasuraman, V. Ganapathy, y B. Shirinzadeh. Behaviour Based Mobile Robot Navigation Technique for Real World Environments using Fuzzy Logic System. En *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, pp. 3359–3364, Holanda, 2004.

[PH92] Sandeep Pandya y Seth Hutchinson. A Case-Based Approach to Robot Motion Planning. En *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, pp. 492–497, Chicago, Estados Unidos, Octubre 1992.

[PKWN03] Liam Pedersen, David Kortenkamp, David Wettergreen, y Illah Nourbakhsh. A Survey of Space Robotics. En *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-03)*, Nara, Japón, Mayo 2003.

[PL90] Enric Plaza y Ramón López. A Case-Based Apprentice that Learns from Fuzzy Examples. En *Proceedings ISMIS*, pp. 420–427, Knoxville, Estados Unidos 1990.

[PNDW98] Daniel Pagac, Eduardo M. Nebot, y Hugh F. Durrant-Whyte. An Evidential Approach to Map-Building for Autonomous Vehicles. *IEEE Transactions on Robotics and Automation*, 14(4):623–629, 1998.

[PPB+02] Alberto Poncela, Eduardo J. Pérez, Antonio Bandera, Cristina Urdiales, y Francisco Sandoval. Efficient Integration of Metric and Topological Maps for Directed Exploration of Unknown Environments. *Robotics and Autonomous Systems*, 41(1):21–39, Octubre 2002.

[PPU+02a] Alberto Poncela, Eduardo J. Pérez, Cristina Urdiales, Antonio Bandera, y Francisco Sandoval. Intelligent Navigation in Partially Unknown Indoor Environments. En *Proceedings of the 5th International FLINS Conference (FLINS 2002)*, pp. 495–502, Gante, Bélgica, Septiembre 2002.

[PPU+02b] Alberto Poncela, Eduardo J. Pérez, Cristina Urdiales, Antonio Bandera, y Francisco Sandoval. Intelligent Navigation in Partially Unknown Environments. En *Computational Intelligent Systems for Applied Research*, pp. 495–502. World Scientific, 2002.

[PPU+02c] Eduardo J. Pérez, Alberto Poncela, Cristina Urdiales, Antonio Bandera, y Francisco Sandoval. Survey Navigation for a Mobile Robot Using a Hierchical Cognitive Map. *Cognitive Processing*, 3(3-4):99–106, 2002.

[PPUS05] Alberto Poncela, Eduardo J. Pérez, Cristina Urdiales, y Francisco Sandoval. A Hybrid Path Planning Technique for Partially Unknown Indoor Environments. *Intelligent Automation and Soft Computing*, 11(3):155–166, 2005.

[Pér04] María L. Pérez. Sistemas Multiagente Aplicados a la Resolución del Problema del Viajante de Comercio. *IEEE América Latina*, 2(1):1–5, Marzo 2004.

[Pér06] Eduardo J. Pérez. *Arquitectura de Navegación Distribuida para Agentes Robóticos (ISBN: 84-690-3326-3, 2006).* Tesis Doctoral, Servicio de Publicaciones de la Universidad de Málaga, Málaga, España, 2006.

[PUdTS07] Alberto Poncela, Cristina Urdiales, Carmen de Trazegnies, y Francisco Sandoval. A New Sonar-Based Landmark for Localization in Indoor Environments. *Soft Computing*, 11(3):281–285, Febrero 2007.

[PUPS08] Alberto Poncela, Cristina Urdiales, Eduardo J. Pérez, y Francisco Sandoval. A New Efficiency-Weighted Strategy for Continuous Human/Robot Cooperation in Navigation. *Enviado a IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, En Revisión, 2008.

[PUS07] Alberto Poncela, Cristina Urdiales, y Francisco Sandoval. A CBR Approach to Behaviour-Based Navigation for an Autonomous Mobile Robot. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2007)*, pp. 3681–3686, Roma, Italia, Abril 2007.

[PUTS04a] Alberto Poncela, Cristina Urdiales, Carmen De Trazegnies, y Francisco Sandoval. A New Sonar Landmark for Place Recognition. En *Applied Computational Intelligence*, pp. 456–462. World Scientific, 2004.

[PUTS04b] Alberto Poncela, Cristina Urdiales, Carmen De Trazegnies, y Francisco Sandoval. A New Sonar Landmark for Place Recognition. En *Proceedings of the 6th International FLINS Conference (FLINS 2004)*, pp. 456–462, Blankenberge, Bélgica, Septiembre 2004.

[QK93] Sean Quinlan y Oussama Khatib. Elastic Bands: Connecting Path Planning and Control. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 802–807, Atlanta, Estados Unidos, 1993.

[QLY04] Xuena Quiu, Shirong Liu, y Simon X. Yang. A Rolling Method for Complete Coverage Path Planning in Uncertain Environments. En *Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics*, pp. 146–151, Shangai, China, Agosto 2004.

[RAMC97] Ashwin Ram, Ronald C. Arkin, Kenneth Moorman, y Russell J. Clark. Case-Based Reactive Navigation: A Method for On-Line Selection and Adaptation of Reactive Robotic Control Parameters. *IEEE Transactions on Systemss, Man and Cybernetics - Part B: Cybernetics*, 27(3):376–394, Junio 1997.

[RCJ+02] R. S. Rao, K. Conn, S. H. Jung, J. Katupitiya, T. Kientz, V. Kumar, J. Ostrowski, S. Patel, y C. J. Taylor. Human Robot Interaction: Applications to Smart Wheelchairs. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002)*, Washington, Estados Unidos, 2002.

[Rei91] Gerhard Reinelt. A Travelling Salesman Problem Library. *ORSA Journal on Computing*, 3-4:367–385, 1991.

[RJ05] Nasir Rahman y Ali Raza Jafri. Two Layered Behaviour Based Navigation of a Mobile Robot in an Unstructured Environment using Fuzzy Logic. En *Proceedings of the IEEE 2005 International Conference on Emerging Technologies*, pp. 230–235, Islamabad, Pakistán, Septiembre 2005.

[RKH04] Bandi B. Reddy, Bahram Kimiaghalam, y Abdollah Homaifar. Reactive Real Time Behavior for Mobile Robots in Unknown Environments. En *Proceedings of the 2004 IEEE International Symposium on Industrial Electronics*, pp. 693–697, Ajaccio, Francia, 2004.

[Ros06] Mark Rosheim. *LEONARDO's Lost Robots*. Springer, Berlín, Alemania, 2006.

[RP89] Julio K. Rosenblatt y David W. Payton. A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control. En *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 317–324, Washington, Estados Unidos, Junio 1989.

[RPW+03] Petru Rusu, Emil M. Petriu, Thom E. Whalen, Aurel Cornell, y Hans J.W. Spoelder. Behavior-Based Neuro-Fuzzy Controller for Mobile Robot Navigation. *IEEE Transactions on Instrumentation and Measurement*, 52(4):1335–1340, Agosto 2003.

[RW91] M. M. Richter y S. Wess. Similarity, Uncertainty and Case-Based Reasoning in PATDEX. En R. S. Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pp. 249–266. Kluwer, 1991.

[SA77] Roger C. Schank y Robert P. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, Estados Unidos, 1977.

[SB90] Peter Scott y César Bandera. Hierarchical Multiresolution Data Structures and Algorithms for Foveal Vision Systems. En *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 832–834, Los Ángeles, Estados Unidos, Noviembre 1990.

[SC96] Mary Shaw y Paul Clements. A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems. En *Proceedings of the 2nd International Software Architecture Workshop*, pp. 50–54, San Francisco, Estados Unidos, Octubre 1996.

[Sch82] Roger C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, Nueva York, Estados Unidos, 1982.

[SD98] Saul Simhon y Gregory Dudek. A Global Topological Map formed by Local Metric Maps. En *Proceedings of ghe 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1708–1714, Victoria, Canadá, Octubre 1998.

[SDC05] Majura F. Selekwa, Damion D. Dunlap, y Emmanuel G. Collins. Implementation of Multi-Valued Fuzzy Behavior Control for Robot Navigation in Cluttered Environments. En *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pp. 3688–3695, Barcelona, España, Abril 2005.

[Sim85] R.L. Simpson. A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation. Technical Report GIT-ICS-85/18, Georgia Institute of Technology, School of Information and Computer Science, Atlanta, Estados Unidos, 1985.

[Sim94] Reid G. Simmons. Structured Control for Autonomous Robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, 1994.

[SK97] Hagit Shatkay y Leslie P. Kaelbling. Learning Topological Maps with Weak Local Odometric Information. En *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI97)*, pp. 920–927, San Francisco, Estados Unidos, 1997.

[SL97] Richard C. Simpson y Simon P. Levine. Development and Evaluation of Voice Control for a Smart Wheelchair. En *Proceedings of the Annual RESNA Conference*, pp. 417–419, Washington, Estados Unidos, 1997.

[SL98] Richard C. Simpson y Simon P. Levine. NavChair: An Assistive Wheelchair Navigation System with Automatic Adaptation. En Mittal et al., editor, *Assistive Technology and AI. LNAI 1458*, pp. 235–255. Springer-Verlag, 1998.

[SL02] Richard C. Simpson y Simon P. Levine. Voice Control of a Powered Wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(2):122–125, Junio 2002.

[SMCB+99] Miquel Sánchez-Marré, Ulises Cortés, Javier Béjar, Ignasi R. Roda, y Manel Poch. Reflective Reasoning in a Case-Based Reasoning Agent. *Lecture Notes in Artificial Intelligence*, 1624:143–158, 1999.

[SMCR+97] Miquel Sànchez-Marrè, Ulises Cortés, Ignasi R. Roda, Manel Poch, y Javier Lafuente. Learning and Adaptation in WWTP through Case-Based Reasoning. *Special issue on Machine Learning of Microcomputers in Civil Engineering*, 12(4):251–266, 1997.

[Smi96] Kate Smith. An Argument for Abandoning the Traveling Salesman Problem as a Neural-Network Benchmark. *IEEE Transactions on Neural Networks*, 7(6):1542–1544, Noviembre 1996.

[SMM+91] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, y C. Whitley. A Comparison of Genetic Sequencing Operators. En *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 69–76, Los Altos, Estados Unidos, 1991.

[SMS+07] F. M. Sánchez, F. Millán, J. Salvador, J. Palou, F. Rodríguez, S. Esquena, y H. Villavicencio. Historia de la Robótica: de Arquitas de Tarento al Robot Da Vinci (Parte I). *Actas Urológicas Españolas*, 31(2):69–76, Febrero 2007.

[SR92] David B. Skalak y Edwina L. Rissland. Arguments and Cases: An Inevitable Twining. *Artificial Intelligence and Law, An International Journal*, 1(1):3–48, 1992.

[SS88] S. Sharma y D. Sleeman. REFINER: A Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement. En *Proceedings of the European Working Session on Learning*, pp. 201–210, Londres, Inglaterra, Octubre 1988.

[Sto00] Bryan Stout. The Basics of A* for Path Planning. En *Game Programming Gems*, pp. 254–263. Charles River Media, Hingham, Estados Unidos, 2000.

[SY04] Li Shou y Li Yuan. Neuro/Fuzzy Behavior-Based Control of a Mobile Robot in Unknown Enrironments. En *Proceedings of the 3rd International Conference on Machine Learning and Cybernetics*, pp. 806–811, Shanghai, China, Agosto 2004.

[Syc88] Katia P. Sycara. Using Case Based Reasoning for Plan Adaptation and Repair. En *Proceedings of the Workshop on Case-Based Reasoning*, pp. 425–434, Clearwater Beach, Estados Unidos, Mayo 1988.

[TBB+98] Sebastian Thrun, Arno Bücken, Wolfram Burgard, Dieter Fox, Daniel Henning, Thorsten Fröhlinghaus, Thomas Hofmann, Michael Krell, y Timo Schmidt. Map Learning and High-Speed Navigation in RHINO. En David Kortenkamp, R. Peter Bonasso, y Robin Murphy, editors, *AI-Based Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, 1998.

[TBB+99] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B. Cremers, Frank Dellaert, Dieter Fox, Dirk Hähnel, Charles Rosenberg, Nicholas Roy, Jamieson Schultz, y Dirk Schulz. MINERVA: a Second-Generation Museum Tour-Guide Robot. En *Proceedings of the 1999 International Conference on Robotics and Automation (ICRA 1999)*, pp. 1999–2005, Detroit, Estados Unidos, Mayo 1999.

[Thr98] Sebastian Thrun. Learning Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99:21–71, Febrero 1998.

[TNS03] Nicola Tomatis, Illah Nourbakhsh, y Roland Siegwart. Hybrid Simultaneous Localization and Map Building: a Natural Integration of Topological and Metric. *Robotics and Autonomous Systems*, 44:3–14, 2003.

[TWBM97] Olivier Trullier, Sidney I. Wiener, Alain Berthoz, y Jean-Arcady Meyer. Biologically Based Artificial Navigation Systems: Review and Prospects. *Progress in Neurobiology*, 51:483–544, 1997.

[TYTK04] H.K. Tsai, J.M. Yang, Y.F. Tsai, y C.Y. Kao. Some Issues of Designing Genetic Algorithms for Traveling Salesman Problems. *Soft Computing*, 8:689–697, 2004.

[TZ04] Hashem Tamini y Andreas Zell. Vision Based Localization of Mobile Robots using Kernel Approaches. En *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, pp. 1896–1901, Sendai, Japón, Septiembre-Octubre 2004.

[UBP+03] Cristina Urdiales, Antonio Bandera, Eduardo J. Pérez, Alberto Poncela, y Francisco Sandoval. Hierarchical Planning in a Mobile Robot for Map Learning and Navigation. En D. Maravall, D. Ruan, y C. Zhou, editors, *Autonomous Robotic Systems: Soft Computing Methodologies*, pp. 188–203. Springer Verlag, 2003.

[UPA+03a] Cristina Urdiales, Alberto Poncela, Roberta Annicchiarico, F. Rizzi, Francisco Sandoval, y Carlo Caltagirone. A Topological Map for Scheduled Navigation in a Hospital Environment. En *Proceedings of the EULAT e-Health 2003*, Cuernava, México, Diciembre 2003.

[UPA+03b] Cristina Urdiales, Alberto Poncela, Roberta Annicchiarico, F. Rizzi, Francisco Sandoval, y Carlo Caltagirone. A Topological Map for Scheduled Navigation in a Hospital Environment. En J. Vázquez-Salceda I. Rudomín y J.L. Díaz de León Santiago (eds), editors, *e-Health: Application of Computing Science in Medicine and Health Care*, pp. 119–132. 2003.

[UPG+07] Cristina Urdiales, Alberto Poncela, Francesco Galluppi, Marta Olivetti, y Francisco Sandoval. Control Compartido a Nivel Reactivo Basado en Eficiencia. En *Actas del XXII Simposium Nacional de la Unión Científica Internacional de Radio, URSI'07*, Tenerife, España, Septiembre 2007.

[UPS+07] Cristina Urdiales, Alberto Poncela, Isabel Sánchez, Francesco Galluppi, Marta Olivetti, y Francisco Sandoval. Efficiency Based Reactive Shared Control for Collaborative Human/Robot Navigagion. En *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, San Diego, Estados Unidos, Octubre 2007.

[Urd99] Cristina Urdiales. *Arquitectura de Control de Movimiento y Exploración para un Agente Autónomo*. Tesis Doctoral, Departamento de Tecnología Electrónica, Universidad de Málaga, Málaga, España, 1999.

[Vap98] Vladimir Vapnik. *Statistical Learning Theory*. Wiley and Son, Nueva York, Estados Unidos, 1998.

[vdBG96] Jan van den Berg y Jock H. Geselschap. An Analysis of Various Elastic Net Algorithms. Technical Report EUR-CS-95-06, Erasmus University Rotterdam, 1996.

[VK99] Nikos Vlassis y Ben Kröse. Robot Environment Modeling via Principal Component Regression. En *Proceedings of the 1999 IEEE/RSJ International Conference on*

Intelligent Robots and Systems (IROS 1999), pp. 677–682, Kyöngju, Corea del Sur, Octubre 1999.

[VMK00] Nikos Vlassis, Yoichi Motomura, y Ben Kröse. Supervised Linear Feature Extraction for Mobile Robot Localization. En *Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*, pp. 2979–2984, San Francisco, Estados Unidos, Abril 2000.

[VMK02] Nikos Vlassis, Yoichi Motomura, y Ber Kröse. Supervised Dimension Reduction of Intrinsically Low-Dimensional Data. *Neural Computation*, 14(1):191–215, Enero 2002.

[VR95] Julie R. Vanderheide y Nageswara S. Rao. Terrain Coverage of an Unknown Room by an Autonomous Mobile Robot. Technical Report ORNL/RM-13117, Oak Ridge National Laboratory, 1995.

[Wat95] Ian Watson. An Introduction to Case Based Reasoning. En I. Watson, editor, *Progress in Case-Based Reasoning*, pp. 3–16. Springer Verlag, 1995.

[WBD98] Peter Willet, John M. Barnard, y Geoffrey M. Downs. Chemical Similarity Searching. *Journal of Chemical Information and Computer Science*, 38:983–996, 1998.

[WC95] John J. Weng y Shaoyun Chen. Autonomous Navigation Through Case-Based Learning. En *Proceedinngs of the IEEE International Symposium on Computer Vision*, pp. 359–364, Coral Glabes, Estados Unidos, Noviembre 1995.

[WL04] Meng Wang y James Liu. Autonomous Robot Navigation using Fuzzy Logic Controller. En *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, pp. 691–696, Shanghai, China, Agosto 2004.

[WSF89] Darrell Whitley, Timothy Starkweather, y D'Ann Fuquay. Scheduling Problems and Traveling Salesman: the Genetic Edge Recombination Operator. En *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 133–140, 1989.

[YB96] Brian Yamauchi y Randall Beer. Spatial Learning for Navigation in Dynamic Environments. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 26(3):496–505, 1996.

[YHH01] Takehisa Yairi, Kosuke Hirama, y Koichi Hori. Fast and Simple Topological Map Construction Based on Cooccurrence Frequency of Landmark Observation. En *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, pp. 1263–1268, Hawaii, Estados Unidos, Noviembre 2001.

[YKY98] Teruko Yata, Lindsay Kleeman, y Shin'ichi Yuta. Wall Following Using Angle Information Measured by a Single Ultrasonic Transducer. En *Proceedings of the*

1998 IEEE International Conference on Robotics and Automation, pp. 1590–1596, Leuve, Bélgica, Mayo 1998.

[YL04] Simon X. Yang y Chaomin Luo. A Neural Network Approach to Complete Coverage Path Planning. *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics*, 34(1):718–725, Febrero 2004.

[YTH02] Takehisa Yairi, Masahito Togami, y Koichi Hori. Learning Topological Maps fron Sequential Observation and Action Data under Partially Observable Environment. En *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2002)*, pp. 305–314, Tokyo, Japón, Agosto 2002.

[Zel92] Alexander Zelinsky. A Mobile Robot Navigation Exploration Algorithm. *IEEE Transactions on Robotics and Automation*, 8:707–717, 1992.

[ZGPP02] Eduardo Zalama, Jaime Gómez, Mariano Paul, y José R. Perán. Adaptive Behavior Navigation of a Mobile Robot. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 32(1):160–169, Enero 2002.

[Zim95] Uwe R. Zimmer. Self-Localization in Dynamic Environments. En *Proceedings of the IEEE/SOFT International Workshop BIES'95*, Tokyo, Japón, Mayo 1995.

[ZJ93] Alexander Zelinsky y Ray A. Jarvis. Planning Paths of Complete coverage of an Unstructured Environment by a Mobile Robot. En *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pp. 533–538, Tokyo, Japón, Noviembre 1993.

[ZLY06] Huidi Zhang, Shirong Liu, y Simon X. Yang. A Hybrid Robot Navigation Approach Based on Partial Planning and Emotion-Based Behavior Coordination. En *Proceedings of the 2006 IEEE/RSJ Interantional Conference on Intelligent Robots and Systems (IROS 2006)*, pp. 1183–1188, Pekín, China, Octubre 2006.

Apéndice A

Descripción del Software del Sistema

Este primer apéndice se dedica a la descripción del *software* desarrollado para implementar el sistema de exploración completa para entornos total o parcialmente desconocidos por parte de un agente autónomo móvil.

Para que el sistema funcione correctamente hay que lanzar los nueve módulos del esquema general de la Figura 2.1. Al arrancar, todo el sistema está a la espera de que el usuario dé la indicación necesaria para iniciar la exploración completa del entorno. Esta orden se manda mediante el módulo *Interacción*, pulsando el botón *INIC*. Al ejecutar esta acción, el módulo *Interacción* envía un comando a todos los módulos para que el sistema comience la exploración.

Una vez iniciado el proceso de exploración, el módulo *ControlRobot* se encarga, en todo momento, de acceder a la información sensorial y sobre la posición y orientación del robot. Esta información la comparte con el resto de módulos para que puedan realizar las tareas para las que están diseñados. Este módulo también está enviando continuamente comandos de movimiento a la plataforma robótica.

Partiendo de la lectura de los sensores que el módulo *ControlRobot* pone a disposición del sistema, los tres comportamientos de la capa reactiva (*Comportamientos*), calculan por separado las velocidades de rotación y traslación que habría que aplicar al robot para implementar la tarea para la que están diseñados. Los tres módulos de la capa envían sus comandos de movimiento al módulo *ControlRobot* para que éste actúe en consecuencia.

Al mismo tiempo, gracias a la información sensorial, el módulo *MapaMétrico* va generando y actualizando la representación del entorno. Con esta representación, el módulo *PlanificadorRuta* es capaz de calcular la ruta de exploración completa que cubre todas las zonas inexploradas del mapa métrico. Esta planificación está sincronizada por el módulo *MapaMétrico*. Hasta que no se detecte que el mapa métrico ha cambiado sustancialmente, no se vuelve a planificar una nueva ruta, trabajando el sistema, por tanto, con la última ruta obtenida. Esa ruta consiste en un conjunto ordenado de puntos del entorno que el robot debería visitar para explorarlo totalmente.

La información sensorial también le sirve al sistema para generar y actualizar la representación local del entorno más cercano al robot (módulo *MapaLocal*). Este modelo local es

usado en todo momento por el módulo *Fusión* para calcular los factores de ponderación de los comportamientos reactivos, utilizados por el módulo *ControlRobot*.

El módulo *ControlRobot* pondera los comandos de movimiento que los comportamientos reactivos han enviado con los factores calculados por el módulo *Fusión*. Así mismo, se tiene en cuenta la ruta ordenada de exploración completa. Con todo ello, se genera un único comando de movimiento, es decir, una velocidad de rotación y otra de traslación, que es enviado a la plataforma. Este módulo trata de seguir la ruta en el orden especificado. Si se alcanza el punto al que el robot debe dirigirse, el sistema automáticamente escoge el siguiente de la ruta ordenada como su destino. Normalmente, como todos los módulos trabajan en paralelo, al robot no le da tiempo a seguir de forma íntegra toda la ruta, porque antes de que lo pueda hacer ya habrá cambiado significativamente el mapa métrico y se habrá planificado una nueva ruta. Este hecho abriga todavía más sentido si se tiene en cuenta que al dirigirnos hacia zonas desconocidas el mapa métrico tendrá que estar continuamente modificándose y, por ende, obligando al planificador de rutas a recalcular su salida.

El sistema funciona de un modo completamente autónomo, siendo el usuario un mero espectador que únicamente arranca el proceso de exploración. El fin de la exploración también es detectado de forma automática. Cuando el módulo *PlanificadorRuta* entregue una ruta sin ningún punto al que deba dirigirse el robot, significará que la exploración ha finalizado, ya que no habrá ninguna zona inexplorada en el mapa métrico de partida de la planificación. En esta situación, el robot se para, dando por finalizado el proceso de exploración.

Todo el sistema de exploración se ha implementado en el lenguaje de programación *C* para *Linux*. La Figura A.1 es similar al esquema general de la Figura 2.1 del Capítulo 2. La única diferencia estriba en el hecho de que la Figura A.1 muestra el principal *software* desarrollado en *C* para implementar los distintos módulos del sistema de exploración.

Además del *software* que implementa cada uno de los módulos del sistema también se han desarrollado una serie de librerías en el lenguaje de programación *C*, necesarias para el correcto funcionamiento de algunos de los módulos. Así mismo, también se han desarrollado e implementado un conjunto de utilidades de ayuda al diseño, análisis y depuración del sistema. Estas aplicaciones se han desarrollado en distintos lenguajes de programación, como son *C* y *Matlab*.

En los restantes apartados de este apéndice se describe todo el *software* desarrollado. La descripción se realiza en función de las distintas capas implementadas. Para cada una de las capas se ha agrupado el *software* en tres grupos:

- Módulos. Distintos módulos en *C* que implementan la capa.
- Librerías. Librerías en *C* necesarias para el correcto funcionamiento de los módulos de la capa. Existen capas que requieren de alguna librería y otras que no.
- Utilidades. Son aplicaciones implementadas como complemento para ayudar en el diseño,

análisis y depuración del sistema. El lenguaje de programación empleado para estas aplicaciones es variado. Como ocurre con las librerías, hay capas para las que se ha empleado alguna utilidad, mientras que hay otras que no han requerido ninguna.

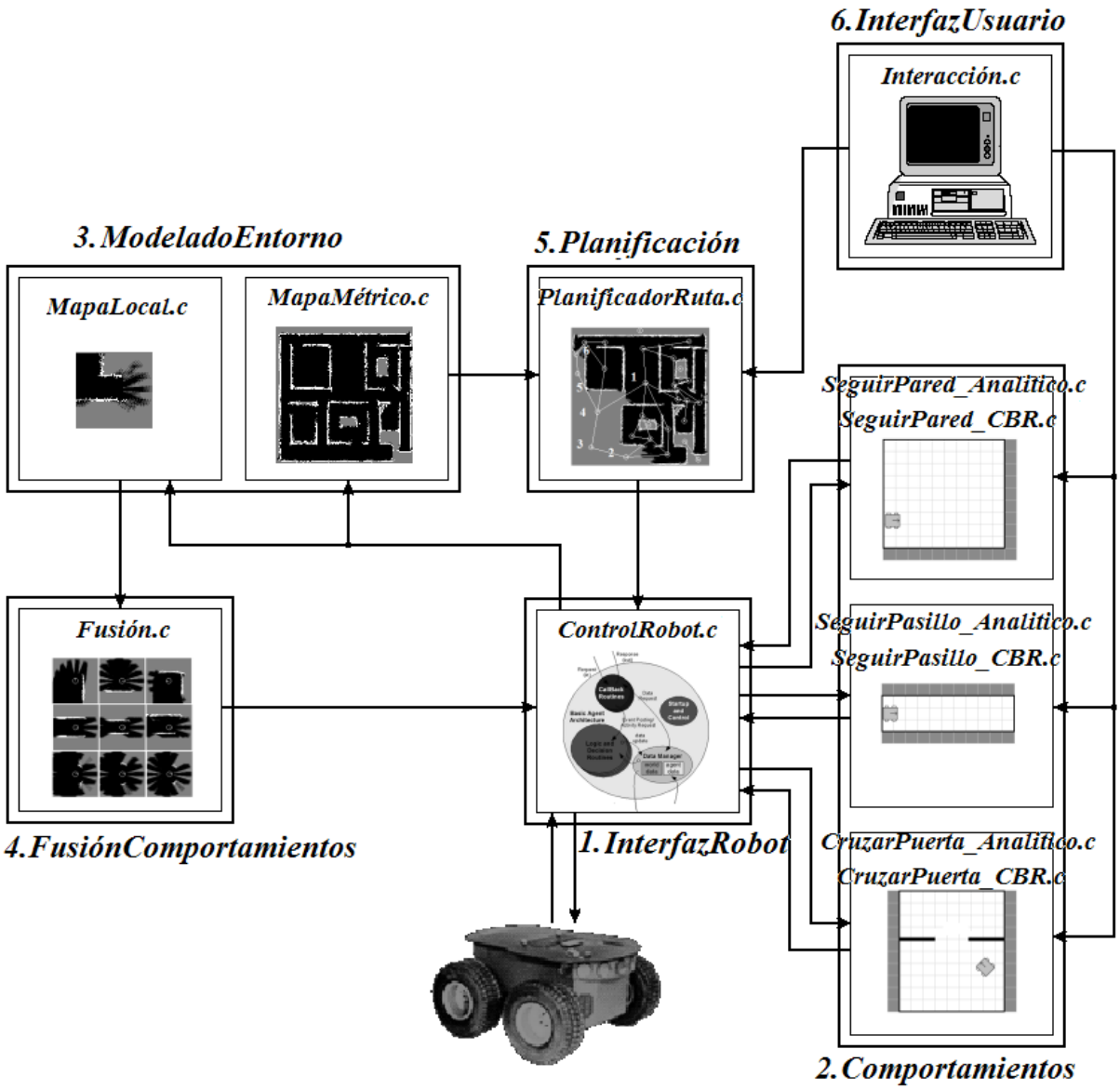

Figura A.1: Esquema general del sistema de exploración junto con el principal *software* desarrollado en *C*.

Antes de pasar a la descripción de todo el *software*, presentamos un resumen por capas de todo el que se va a analizar a continuación:

1. Capa *InterfazRobot*
   - Módulos
     - *ControlRobot.c*

- Librerías. Ninguna
- Utilidades
  - *AprendizajeAsistido.m*, en *Matlab*

2. Capa *Comportamientos*

   - Módulos
     - *SeguirPared_Analitico.c*
     - *SeguirPasillo_Analitico.c*
     - *CruzarPuerta_Analitico.c*
     - *SeguirPared_CBR.c*
     - *SeguirPasillo_CBR.c*
     - *CruzarPuerta_CBR.c*
   - Librerías. Ninguna
   - Utilidades
     - *EntrenarCBR.c*, en *C*
     - *ClasificarCBR.c*, en *C*
     - *ComportamientosRAW.c*, en *C*

3. Capa *ModeladoEntorno*

   - Módulos
     - *MapaMétrico.c*
     - *MapaLocal.c*
   - Librerías. Ninguna
   - Utilidades. Ninguna

4. Capa *FusiónComportamientos*

   - Módulos
     - *Fusión.c*
   - Librerías
     - *FusiónLib.h* y *FusiónLib.c*
   - Utilidades
     - *CapturaBase.c*, en *C*
     - *FusiónRAW.c*, en *C*
     - *PCA.m*, en *Matlab*
     - *CompararPCA.m*, en *Matlab*

5. Capa *Planificación*
   - Módulos
     - *PlanificadorRuta.c*
   - Librerías
     - *PAIC.h* y *PAIC.c*
     - *TSP.h* y *TSP.c*
   - Utilidades. Ninguna
6. Capa *InterfazUsuario*
   - Módulos
     - *Interacción.c*
   - Librerías. Ninguna
   - Utilidades. Ninguna

1 La capa *InterfazRobot*

El módulo *ControlRobot* se implementa mediante el programa *ControlRobot.c*. Se encarga de dos tareas principales. Por un lado, recibir de la plataforma robótica la información dada por los sensores sonar, así como su posición y orientación determinada por el sistema odométrico del agente. Por otro, enviar los comandos de movimiento que el robot debe aplicar a sus motores, en la forma de velocidad de traslación y velocidad de rotación. En el módulo se considera una combinación de los comandos de movimiento de los tres comportamientos como orden del robot, Seguir Pared, Seguir Pasillo y Cruzar Puerta . También se permite la combinación de las órdenes del robot y del humano cuando se trabaja con el control asistido.

1.1 Módulos

ControlRobot.c

Objetivo

Leer el estado del robot y enviarle los comandos de movimiento que debe aplicar. El estado del robot consiste en la lectura de sus sensores sonar, su posición y su orientación. En cuanto a los sensores sonar, su número varía en función de la plataforma robótica. Si se emplea el simulador de la plataforma robótica *Nomad200* de *Nomadics*, el robot consta de un anillo de 16 sensores equiespaciados. Si se usa la plataforma robótica real *Pioneer P2AT*, ésta estará equipada con 8 sensores sonar *Polaroid* frontales. Para ambas plataformas, los comandos de movimiento consisten en la velocidad de traslación y la velocidad de rotación que el robot debe aplicar a

sus motores. También se guarda la traza con la trayectoria recorrida por el agente. Contiene información sobre la posición, la orientación, la lectura de todos los sensores y el comando de movimiento enviado al robot. Cuando se trabaja con el control asistido, para los comandos de movimiento del robot, del humano y del control asistido se guardan la velocidades de traslación y rotación, la eficiencia total y los distintos factores de la eficiencia involucrados (ver Sección 4.2 del Capítulo 3).

Algoritmo

A partir de los comandos de movimiento generados por cada uno de los tres comportamientos del sistema de exploración, los factores de ponderación para estos tres comportamientos y el siguiente punto de la ruta de exploración completa, se obtiene un único comando de movimiento (velocidad de traslación y velocidad de rotación), que es enviado al robot para que actúe en consecuencia. Cuando se trabaja con el control asistido, a partir de los comandos de movimiento de los comportamientos y las órdenes aplicadas por un humano mediante un joystick, se obtiene un único comando de movimiento (velocidad de traslación y velocidad de rotación), que es enviado al robot para que actúe en consecuencia. El control asistido se alcanza mediante una combinación lineal ponderada de los comandos del robot y los comandos introducidos por el humano. El algoritmo de combinación se analiza en la Sección 4.2 del Capítulo 3.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes de la plataforma robótica.
- Comando de exploración: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.
- Comando de movimiento Seguir Pared: comando de movimiento para implementar el comportamiento *SeguirPared*, generado por el módulo *SeguirPared*.
- Comando de movimiento Seguir Pasillo: comando de movimiento para implementar el comportamiento *SeguirPasillo*, generado por el módulo *SeguirPasillo*.
- Comando de movimiento Cruzar Puerta: comando de movimiento para implementar el comportamiento *CruzarPuerta*, generado por el módulo *CruzarPuerta*.
- Pesos: factores de ponderación para los tres comportamientos, generados por el módulo *Fusión*.
- Ruta: ruta de exploración completa de todas las zonas inexploradas del entorno, generada por el módulo *PlanificadorRuta*. Incluye el punto de destino siguiente al que debe dirigirse el robot.
- Joystick: cuando se trabaja con el control asistido, comando de movimiento que el humano intenta aplicar al robot mediante el joystick.

Salidas

- Estado del robot: puesto a disposición de todos los módulos.
- Comando de movimiento para el robot: velocidad de traslación y velocidad de rotación que el robot debe aplicar a sus motores.

Salidas para

El comando de movimiento lo usa la plataforma robótica. El estado del robot es usado por los módulos *SeguirPared*, *SeguirPasillo*, *CruzarPuerta*, *MapaMétrico*, *MapaLocal*, *Fusión*, *PlanificadorRuta* e *Interacción*.

1.2 Librerías

Ninguna.

1.3 Utilidades

AprendizajeAsistido.m

Objetivo

Obtener de forma gráfica, estadística o algorítmica, distinta información sobre una traza con la trayectoria recorrida por el agente mientras navega con control asistido. La información gráfica consiste en una representación 3D donde el plano XY simula el espacio geométrico del entorno de trabajo, mientras que al eje Z se le asigna la eficiencia en cada punto de la trayectoria. A cada uno de los factores involucrados en la eficiencia se le asigna un canal de color *RGB* para hacer más atractiva la representación. Mediante un simple vistazo a la misma se puede tener una noción del comportamiento del robot a lo largo de la trayectoria. La información estadística consiste en medias y varianzas de distintos parámetros, como por ejemplo la curvatura. También se genera un mapa métrico con la trayectoria recorrida por el agente.

Algoritmo

Los datos estadísticos se obtienen mediante las funciones *mean* y *std* de *Matlab*. La curvatura emplea el algoritmo para el cálculo de la función de curvatura de [PUdTS07]. La representación gráfica 3D es triple. Se dibuja la eficiencia considerando únicamente los comandos reactivos del robot, considerando únicamente los comandos introducidos por el humano mediante el joystick y teniendo en cuenta la combinación de ambos comandos para conseguir el control asistido.

Entradas

- Fichero con la trayectoria recorrida por el agente. Contiene información sobre la posición, orientación y lectura de los sensores del robot. Así mismo también está presente la

información sobre los comandos de movimiento del robot, humano y el control asistido. Consiste en las velocidades de traslación y rotación, la eficiencia y los distintos factores involucrados en la eficiencia.

Salidas

Ninguna.

Salidas para

Ningún otro programa.

2 La capa *Comportamientos*

Esta capa consta de tres módulos, uno por cada comportamiento del sistema, Seguir Pared, Seguir Pasillo y Cruzar Puerta. Cada uno de ellos ha sido desarrollado siguiendo un doble enfoque. Por un lado, según un modelo analítico (*SeguirPared_Analitico.c*, *SeguirPasillo_Analitico.c* y *CruzarPuerta_Analitico.c*). Por otro, mediante la filosofía del *CBR* (*SeguirPared_CBR.c*, *SeguirPasillo_CBR.c* y *CruzarPuerta_CBR.c*). Los tres comportamientos son puramente reactivos. Ante una determinada lectura sensorial proporcionan la velocidad de traslación y la velocidad de rotación necesaria para implementar la funcionalidad deseada. En todos los programas desarrollados se guarda la traza con la trayectoria recorrida por el agente. En el caso de la implementación basada en el *CBR* se anota además en la traza si un determinado caso durante la trayectoria del robot ha sido adaptado o si, por el contrario, se soluciona a partir de la base de casos.

2.1 Módulos

SeguirPared_Analitico.c

Objetivo

Generar los comandos de movimiento necesarios, velocidades de traslación y de rotación, para que el robot se desplace de forma paralela a la pared que está a su derecha a una distancia determinada a partir de la información sensorial.

Algoritmo

Se basa en un modelo analítico que suma tres fuerzas. La primera mantiene el robot paralelo a la pared. La segunda intenta mantener la distancia prefijada a la pared. Y la tercera evita los obstáculos que aparecen.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.

- Comando de inicio: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.

Salidas

- Comando de movimiento Seguir Pared: velocidad de traslación y velocidad de rotación para implementar el comportamiento Seguir Pared.
- Fichero de traza: con la trayectoria recorrida por el agente.

Salidas para

Módulo *ControlRobot*.

SeguirPasillo_Analitico.c

Objetivo

Generar los comandos de movimiento necesarios, velocidades de traslación y de rotación, para que el robot navegue por el centro de un pasillo a partir de la información proporcionada por los sensores sonar.

Algoritmo

Se basa en un modelo analítico que suma tres fuerzas. La primera intenta mantener el robot paralelo a las paredes del pasillo. La segunda centra al robot en el pasillo. Y la tercera, evita los obstáculos.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.
- Comando de inicio: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.

Salidas

- Comando de movimiento Seguir Pasillo: velocidad de traslación y velocidad de rotación para implementar el comportamiento Seguir Pasillo.
- Fichero de traza: con la trayectoria recorrida por el agente.

Salidas para

Módulo *ControlRobot*.

CruzarPuerta_Analitico.c

Objetivo

Generar los comandos de movimiento necesarios, velocidades de traslación y de rotación, para que el robot cruce una puerta por su centro para entrar o salir de una habitación a partir de la información sensorial.

Algoritmo

Se basa en un modelo analítico que suma dos fuerzas. La primera mantiene al robot de forma perpendicular al marco de la puerta. Y la segunda evita la colisión con el marco de la puerta.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.
- Comando de inicio: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.

Salidas

- Comando de movimiento Cruzar Puerta: velocidad de traslación y velocidad de rotación para implementar el comportamiento Cruzar Puerta.
- Fichero de traza: con la trayectoria recorrida por el agente.

Salidas para

Módulo *ControlRobot*.

SeguirPared_CBR.c

Objetivo

Generar los comandos de movimiento necesarios, velocidades de traslación y de rotación, para que el robot se desplace de forma paralela a la pared que está a su derecha a una distancia determinada a partir de la información sensorial.

Algoritmo

Se basa en el *CBR* para implementar la funcionalidad de Seguir Pared. Para ello, se sigue el proceso de diseño de comportamientos de la Sección 3.2 del Capítulo 3.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.

- Comando de inicio: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.
- Base de casos: experiencia adquirida mediante aprendizaje por observación y/o aprendizaje por experiencia. La base de casos puede estar clasificada o no.

Salidas

- Comando de movimiento Seguir Pared: velocidad de traslación y velocidad de rotación para implementar el comportamiento Seguir Pared.
- Fichero de traza: con la trayectoria recorrida por el agente. Los casos adaptados se anotan.

Salidas para

Módulo *ControlRobot*.

SeguirPasillo_CBR.c

Objetivo

Generar los comandos de movimiento necesarios, velocidades de traslación y de rotación, para que el robot navegue por el centro de un pasillo a partir de la información sensorial.

Algoritmo

Se basa en el *CBR* para implementar la funcionalidad de Seguir Pasillo. Para ello, se sigue el proceso de diseño de comportamientos de la Sección 3.2 del Capítulo 3.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.
- Comando de inicio: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.
- Base de casos: experiencia adquirida mediante aprendizaje por observación y/o aprendizaje por experiencia. La base de casos puede estar clasificada o no.

Salidas

- Comando de movimiento Seguir Pasillo: velocidad de traslación y velocidad de rotación para implementar el comportamiento Seguir Pasillo.
- Fichero de traza: con la trayectoria recorrida por el agente. Los casos adaptados se anotan.

Salidas para

Módulo *ControlRobot*.

CruzarPuerta_CBR.c

Objetivo

Generar los comandos de movimiento necesarios, velocidades de traslación y de rotación, para que el robot cruce una puerta por su centro para entrar o salir de una habitación a partir de la información proporcionada por los sensores sonar. La entrada o salida de la habitación debe hacerse sin colisionar con el marco de la puerta.

Algoritmo

Se basa en el *CBR* para implementar la funcionalidad de Cruzar Puerta. Para ello, se sigue el proceso de diseño de comportamientos de la Sección 3.2 del Capítulo 3.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.
- Comando de inicio: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.
- Base de casos: experiencia adquirida mediante aprendizaje por observación y/o aprendizaje por experiencia. La base de casos puede estar clasificada o no.

Salidas

- Comando de movimiento Cruzar Puerta: velocidad de traslación y velocidad de rotación para implementar el comportamiento Cruzar Puerta.
- Fichero de traza: con la trayectoria recorrida por el agente. Los casos adaptados se anotan.

Salidas para

Módulo *ControlRobot*.

2.2 Librerías

Ninguna.

2.3 Utilidades

EntrenarCBR.c

Objetivo

Implementar el aprendizaje por observación de la fase Retener del ciclo *CBR*, almacenando el aprendizaje adquirido en una base de casos. Dependiendo del comportamiento que se desee implementar, el entrenamiento variará.

Algoritmo

Un operador humano guía el agente autónomo móvil por medio de un teclado. El robot es movido de manera que se ejecute uno de los tres comportamientos del sistema, Seguir Pared, Seguir Pasillo o Cruzar Puerta. Durante la navegación se salva en un fichero la traza con la trayectoria recorrida por el agente. Este fichero constituye una base de casos para el comportamiento seleccionado, pudiendo servir de experiencia del correspondiente módulo implementado según la filosofía del *CBR*.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes de la plataforma robótica.
- Información del guiado: velocidades de traslación y de rotación que el humano intenta aplicar al agente.

Salidas

- Comando de movimiento para el robot: velocidad de traslación y velocidad de rotación que el robot debe aplicar a sus motores.
- Base de casos: fichero con la experiencia adquirida mediante aprendizaje por observación. La base de casos resultante no está clasificada.

Salidas para

La base de casos es para *SeguirPared_CBR.c*, *SeguirPasillo_CBR.c*, *CruzarPuerta_CBR.c* o *ClasificarCBR.c*.

ClasificarCBR.c

Objetivo

Clasificar una base de casos con experiencia de navegación reactiva acumulada sobre alguno de los tres comportamientos del sistema, Seguir Pared, Seguir Pasillo o Cruzar Puerta. Esta base de casos contiene aprendizaje por observación y/o aprendizaje por experiencia.

Algoritmo

MaxMin. Clasifica la base de casos en un número indeterminado de clases de forma no supervisada. La clasificación únicamente depende de un parámetro, la distancia *Dist_MaxMin* mínima entre las clases. Los prototipos de las clases obtenidas conforman la base de casos clasificada.

Entradas

- *Dist_MaxMin*: distancia mínima entre las clases.
- Base de casos: fichero con la experiencia adquirida mediante aprendizaje por observación y/o aprendizaje por experiencia.

Salidas

- Base de casos clasificada: fichero con la experiencia adquirida mediante aprendizaje por observación y/o aprendizaje por experiencia clasificada.

Salidas para

SeguirPared_CBR.c, *SeguirPasillo_CBR.c* o *CruzarPuerta_CBR.c*.

ComportamientosRAW.c

Objetivo

Generar un mapa métrico con la trayectoria recorrida por el agente superpuesta en gris. Los puntos de la trayectoria en los que los casos tuvieron que ser adaptados se representan con círculos blancos.

Algoritmo

A partir de los datos de la traza con la trayectoria recorrida por el agente se ejecuta el algoritmo de generación de un mapa métrico presentado en la Sección 3.3 del Capítulo 2, representando en gris los puntos por los que pasa el agente. El algoritmo es válido para trazas almacenadas durante la navegación con comportamientos analíticos o con el *CBR*. En este último caso, cuando se adapten casos se anota la situación en el mapa métrico resultante con un círculo blanco.

Entradas

- Fichero de traza: con la trayectoria recorrida por el agente.

Salidas

- Fichero RAW: con el mapa métrico que tiene la trayectoria del robot superpuesta.

Salidas para

Ningún otro programa.

3 La capa *ModeladoEntorno*

La misión de esta capa es proporcionar una representación del entorno que sirva como punto de partida al proceso de planificación de la ruta de exploración completa que lleva a cabo el módulo *PlanificadorRuta*, labor realizada por el módulo *MapaMétrico*. También es responsable de generar y mantener, mediante el módulo *MapaLocal*, un modelo de entorno adecuado para el sistema de ayuda a la cooperación de comportamientos (módulo *Fusión*), que calcula los factores de ponderación para cada uno de los tres comportamientos del sistema, Seguir Pared, Seguir Pasillo y Cruzar Puerta. Ambos modelos del entorno consisten en un mapa métrico probabilístico, diferenciándose en su rango de alcance.

3.1 Módulos

MapaMétrico.c

Objetivo

Obtener un mapa métrico probabilístico del entorno por el que navega el robot que sirva como punto de partida adecuado para el proceso de planificación de rutas de exploración completa que se realiza en el módulo *PlanificadorRuta*.

Algoritmo

El algoritmo de construcción del mapa probabilístico se describe y analiza de forma detallada en la Sección 3.3 del Capítulo 2. Se basa en un modelo sencillo que realiza medidas frecuentas en vez de emplear una compleja regla de actualización. Así mismo, se efectúa la integración temporal de las sucesivas lecturas de los sensores sonar para actualizar la probabilidad de ocupación de cada celda del espacio.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, generados por el módulo *ControlRobot*.
- Comando de exploración: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.

Salidas

- Mapa métrico: modelo generado del entorno.

Salidas para

Módulos *PlanificadorRuta* e *Interacción*.

MapaLocal.c

Objetivo

Obtener un mapa métrico probabilístico del entorno más cercano al robot que sirva como punto de partida para el proceso de cálculo de los factores de ponderación de los tres comportamientos.

Algoritmo

El algoritmo de construcción del mapa probabilístico es similar al de *MapaMétrico.c*. Se describe en la Sección 3.3 del Capítulo 2.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, generados por el módulo *ControlRobot*.
- Comando de exploración: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.

Salidas

- Mapa local: modelo generado del entorno más cercano al robot.

Salidas para

Módulos *Fusión* e *Interacción*.

3.2 Librerías

Ninguna.

3.3 Utilidades

Ninguna.

4 La capa *FusiónComportamientos*

Esta capa consta de un único módulo, *Fusión*, implementado mediante el programa *Fusión.c*. Su misión es la de obtener los factores de ponderación que se deben aplicar a los comandos de movimiento de los tres comportamientos del sistema de exploración. Partiendo del mapa local y siguiendo el procedimiento analizado en el Capítulo 4 se llega a los tres factores necesarios. Como el proceso de obtención de estos tres factores es bastante laborioso, se ha desarrollado una librería que encapsula las principales funciones que soportan el algoritmo.

4.1 Módulos

Fusión.c

Objetivo

Obtener los tres factores de ponderación que el sistema de exploración debe usar para pesar los comandos de movimiento de los comportamientos Seguir Pared, Seguir Pasillo y Cruzar Puerta.

Algoritmo

El presentado en el Capítulo 4. El algoritmo parte del mapa local generado en el módulo *MapaLocal*. Esta representación es procesada para obtener el contorno de la $RLOI$ en su entorno más próximo. El contorno se representa a su vez mediante su mapa de profundidad, el cual es transformado en el módulo de su FFT. Finalmente, el $|FFT|$ es proyectado sobre la base del sistema para obtener el vector de características del mapa local en la posición del robot. Una vez representado un mapa local mediante su vector de características, es comparado con los patrones del sistema para hallar los factores de ponderación.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, generados por el módulo *ControlRobot*.
- Comando de exploración: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.
- Mapa local: modelo generado del entorno más cercano al robot en el módulo *MapaLocal*.
- Base: base de componentes principales, generada con la aplicación $PCA.m$.

Salidas

- Pesos: factores de ponderación para los tres comportamientos.
- Fichero de traza: con la trayectoria recorrida por el agente, anotando los factores de ponderación en cada punto de la trayectoria.

Salidas para

Módulo *ControlRobot*.

4.2 Librerías

FusiónLib.h y *FusiónLib.c*

Esta librería, desarrollada en *C*, es necesaria para el correcto funcionamiento del módulo *Fusión*. Incluye las funciones de cálculo que implementan todos los pasos del algoritmo de obtención de los factores de ponderación para los tres comportamientos del sistema a partir de un mapa local. A continuación se presentan las funciones que forman parte de la librería.

*void Umbralización(unsigned char *imagen);*

Descripción
Lleva a cabo la fase de *Umbralización* del algoritmo para el cálculo del contorno del mapa local dentro del proceso de extracción del vector de características. La imagen de entrada con el mapa local es sobrescrita, siendo a la finalización de la función, el mapa local umbralizado.
Parámetros
imagen: mapa local de entrada.
Valor devuelto
Ninguno.
Llamada por
Módulo *Fusión*.

*void Filtrado(unsigned char *imagen);*

Descripción
Procesa el mapa local umbralizado mediante el filtro de la mediana para suprimir el ruido, eliminando de la imagen muchas de las regiones de tamaño pequeño que aparecen debidas al propio proceso de la umbralización. La máscara empleada es 3x3. La imagen de entrada se sobrescribe, estando filtrada al final del procesado. Esta imagen presenta un conjunto de regiones no conectadas que, potencialmente, pueden constituir la región de interés.
Parámetros
imagen: mapa local de entrada umbralizado.
Valor devuelto
Ninguno.
Llamada por
Módulo *Fusión*.

*void ObtenerRLOI(unsigned char *imagen);*

Descripción
Analiza el mapa local umbralizado y filtrado para extraer la Región Libre de Obstáculos de Interés (*RLOI*), la región de interés del mapa local. Es la región en la que se encuentra el robot, delimitando el espacio libre en su entorno más próximo. La imagen de entrada es analizada para clasificar todos sus pixeles. Debido al proceso involucrado, es necesaria la fusión de las clases que están conectadas, por formar parte de la misma región. Una vez obtenidas todas las regiones, se escoge la *RLOI*. La imagen de entrada es modificada para obtener al concluir el procesado el mapa local umbralizado, filtrado y con la región de interés.
Parámetros
imagen: mapa local de entrada umbralizado y filtrado.
Valor devuelto
Ninguno.
Llamada por
Módulo *Fusión*.

*int *ObtenerContorno(unsigned char *imagen, int *K);*

Descripción
Obtiene el contorno $\overrightarrow{C}$ de la *RLOI*. $\overrightarrow{C}$ es un vector de tamaño variable que depende de la configuración del entorno en el mapa local.

Parámetros
imagen: imagen con la *RLOI* del mapa local umbralizado y filtrado.
K: número de puntos del contorno. Es un valor de salida modificado por la función.

Valor devuelto
Vector de enteros de tamaño variable con las coordenadas de los puntos del contorno.

Llamada por
Módulo *Fusión*.

*int *CalcularMapaProfundidad(int *contorno, int K);*

Descripción
Implementa la fase de *Cálculo del mapa de profundidad* del algoritmo de extracción del vector de características de un mapa local. A partir de un vector de tamaño variable se obtiene un vector de tamaño fijo que modela la distancia medida por N sensores equiespaciados en un anillo de sensores en la posición del robot.

Parámetros
contorno: vector con las coordenadas de los puntos del contorno.
K: número de puntos del contorno.

Valor devuelto
Vector $\overrightarrow{MP}$ con el mapa de profundidad de tamaño fijo.

Llamada por
Módulo *Fusión*.

*float *CalcularFFT(int *MP, int N);*

Descripción
Calcula la Transformada Discreta de Fourier del mapa de profundidad de entrada mediante el algoritmo de la Transformada Rápida de Fourier (FFT).

Parámetros
MP: vector con el mapa de profundidad del contorno de la *RLOI* del mapa local.
N: tamaño del mapa de profundidad, y de la FFT de salida.

Valor devuelto
Vector con la FFT del mapa de profundidad de entrada.

Llamada por
Módulo *Fusión*.

*float *CalcularMóduloFFT(float *FFT, int N);*

Descripción
Calcula el módulo de la FFT de un mapa de profundidad.

Parámetros
FFT: vector con la FFT de un mapa de profundidad.
N: tamaño de la FFT.

Valor devuelto
Vector de tamaño N con el módulo de la FFT.

Llamada por
Módulo *Fusión*.

*float *ProyectarSobreBase(float *modFFT, int N, float **base, int P);*

Descripción
Implementa la fase de extracción del vector de características de un mapa local a partir del módulo de la *FFT* del mapa de profundidad del contorno de su *RLOI*. Para ello, proyecta el $|FFT|$ sobre la base, obteniéndose un vector de P componentes, que es el vector de características.

Parámetros
modFFT: módulo de la *FFT* del mapa de profundidad del contorno de la *RLOI* de un mapa local.
N: tamaño del vector de entrada y de los vectores de la base.
base: base de componentes principales sobre la que proyectar.
P: número de vectores de la base, tamaño del vector de características de salida.

Valor devuelto
El vector de características de P componentes.

Llamada por
Módulo *Fusión*.

*float *ObtenerFactores(float *vector, int P);*

Descripción
Calcula los factores de ponderación que se deben aplicar a los comandos de movimiento de los tres comportamientos del sistema, tomando como punto de partida un vector de características. Se calcula en primer lugar la distancia de Tanimoto del vector de características a los 9 patrones del sistema. En segundo lugar se analizan estas distancias con el procedimiento descrito en el Capítulo 4 para calcular los factores de ponderación finales.

Parámetros
vector: vector de características de entrada.
P: tamaño del vector de características.

Valor devuelto
Vector con los tres factores de ponderación para los tres comportamientos del sistema.

Llamada por
Módulo *Fusión*.

4.3 Utilidades

CapturaBase.c

Objetivo

Capturar y procesar mapas locales adquiridos en entornos arbitrarios, tanto simulados como reales, como punto de partida del cálculo de la base de componentes principales del sistema.

Algoritmo

El robot es guiado manualmente por un operador humano mediante un teclado. Al navegar por el entorno, se va generando y actualizando el mapa local del sistema. Cuando se crea conveniente, se captura el mapa local en la posición actual del robot. Esta captura se efectúa

tantas veces como se desee. Los mapas locales son procesados hasta obtener el $|FFT|$ del mapa de profundidad del contorno de su *RLOI*. Los $|FFT|$ son almacenados en un fichero para su posterior procesamiento mediante la aplicación *PCA.m*, y así calcular la base del sistema.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes de la plataforma robótica.
- Mapa local: modelo generado del entorno más cercano al robot en el módulo *MapaLocal*.
- Información del guiado: velocidades de traslación y de rotación que el humano intenta aplicar al agente.

Salidas

- Fichero $|FFT|$: fichero con el módulo de la FFT de los mapas locales capturados.

Salidas para

Aplicación *PCA.m*.

FusiónRAW.c

Objetivo

Generar las representaciones *RS* y *RF* presentadas en el Capítulo 4. Ambas representaciones consisten en el mapa métrico con la trayectoria recorrida por el agente en color. Dependiendo de la representación (*RS* o *RF*), la interpretación de los colores de la trayectoria es diferente.

Algoritmo

A partir de los datos de la traza con la trayectoria recorrida por el agente se ejecuta el algoritmo de generación de un mapa métrico presentado en la Sección 3.3 del Capítulo 2, representando en color los puntos por los que pasa el agente.

Entradas

- Fichero de traza: con la trayectoria recorrida por el agente, junto con los factores de ponderación en cada punto de dicha trayectoria.

Salidas

- Fichero RAW *RS*: imagen RAW con la *Representación de Selección*.
- Fichero RAW *RF*: imagen RAW con la *Representación de Fusión*.

Salidas para

Ningún otro programa.

PCA.m

Objetivo

Calcular la base de componentes principales del sistema de exploración. Sobre esta base se proyectará cualquier $|FFT|$ del mapa de profundidad del contorno de la región libre de obstáculos de un mapa local. El resultado de esta proyección es el vector de características para el mapa local de entrada.

Algoritmo

Expansión de Karhunen-Loëve o Análisis de Componentes Principales. La descripción matemática del método se presenta en el Capítulo 4.

Entradas

- Fichero $|FFT|$: fichero con un conjunto arbitrario de módulos de la FFT, generado con *CapturaBase.c*.

Salidas

- Base: base de componentes principales.

Salidas para

Módulo *Fusión*.

CompararPCA.m

Objetivo

Servir de ayuda para el análisis y depuración del módulo *Fusión*. Esta aplicación permite la comparación tanto gráfica como numérica de los $|FFT|$ de dos ficheros distintos, así como de los vectores de características obtenidos al proyectar esos $|FFT|$ sobre la base que se toma a la entrada. Gráficamente, se pueden representar conjuntamente los dos $|FFT|$ que se están comparando o sus dos vectores de características asociados. Numéricamente, se indica la distancia de Tanimoto entre los dos vectores de características comparados.

Algoritmo

A partir de los dos ficheros con varios $|FFT|$, se obtienen los vectores de características de todos ellos proyectando sobre la base que se emplea. También se obtiene la distancia de Tanimoto de cualquier vector de características obtenido a partir de los datos del fichero 1 con respecto

a cualquier vector de características calculado a partir de la información del fichero 2. Una vez se dispone de todos estos valores, es el usuario el que selecciona la información que se desea conocer.

Entradas

- Fichero 1 $|FFT|$: fichero con el $|FFT|$ de varios mapas de profundidad correspondientes a varios contornos de su *RLOI*.
- Fichero 2 $|FFT|$: fichero con el $|FFT|$ de varios mapas de profundidad correspondientes a varios contornos de su *RLOI*.
- Base: base de componentes principales, generada con la aplicación *PCA.m*.

Salidas

Ninguna.

Salidas para

Ningún otro programa.

5 La capa *Planificación*

Esta capa se implementa a través del módulo *PlanificadorRuta*, gracias al programa *PlanificadorRuta.c*. Su tarea consiste en calcular la ruta de exploración completa que cubre todas las zonas no exploradas del entorno por el que navega el robot. El método está soportado por una estructura jerárquica que explícitamente representa las regiones no exploradas. Esta estructura integra los paradigmas métrico y topológico de modelado de entornos.

5.1 Módulos

PlanificadorRuta.c

Objetivo

Obtener la ruta de exploración completa que el robot debe seguir a través de las zonas no exploradas del entorno, partiendo del modelo de entorno generado en el módulo *MapaMétrico*.

Algoritmo

La ruta de exploración completa se calcula con el algoritmo genético diseñado que se presenta en el Capítulo 5. Para poder aplicar este algoritmo se necesita disponer de las regiones no exploradas del entorno y de la matriz de distancias entre regiones. Para poder calcular estos datos el método propuesto se sustenta en una estructura jerárquica que integra los paradigmas métrico y topológico. Sobre el mapa métrico se construye de forma no supervisada una estructura

piramidal que permite obtener un mapa topológico del entorno con las regiones del mismo y las conexiones entre ellas.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, generados por el módulo *ControlRobot*.
- Comando de exploración: orden para iniciar/parar el proceso de exploración, generada por el módulo *Interacción*.
- Mapa métrico: modelo generado del entorno en el módulo *MapaMétrico*.

Salidas

- Ruta: ruta de exploración completa de todas las zonas inexploradas del entorno. Incluye el punto de destino siguiente para el robot.

Salidas para

Módulos *ControlRobot* e *Interacción*

5.2 Librerías

PAIC.h y *PAIC.c*

Esta librería, desarrollada en *C*, es necesaria para el correcto funcionamiento del módulo *PlanificadorRuta*. Incluye las funciones de cálculo que implementan todos los pasos del algoritmo de construcción del mapa topológico a partir de un mapa métrico. También existen funciones para obtener las regiones no exploradas del entorno y la matriz de distancias entre estas regiones. Cualquier celda de cualquier nivel de la estructura pertenece al tipo de datos *Celda* definido en la librería. Este tipo de datos incluye todos campos asociados a una celda, definidos en la Sección 2.2.2 del Capítulo 5. A continuación se presentan las funciones que forman parte de la librería.

*Celda ***Inicializar(unsigned char *imagen);*

Descripción
Implementa la fase de *Creación e Inicialización* de la estructura piramidal jerárquica. Se reserva la memoria necesaria. Posteriormente, se inicializan todos los niveles de la estructura, siendo su base la imagen con el mapa métrico que se le pasa como entrada.
Parámetros
imagen: mapa métrico que sirve como base de la estructura jerárquica.
Valor devuelto
La estructura jerárquica creada e inicializada.
Llamada por
Módulo *PlanificadorRuta*.

*void Enlazar(Celda ***Piramide);*

Descripción
Implementa la fase de *Enlazado* de la estructura jerárquica. Se fusionan zonas adyacentes que son parecidas, disminuyendo el número final de regiones.

Parámetros
Piramide: pirámide creada e inicializada.

Valor devuelto
La estructura jerárquica enlazada.

Llamada por
Módulo *PlanificadorRuta*.

*void Clasificar(Celda ***Piramide, int *NumClase);*

Descripción
Implementa la fase de *Clasificación* de la estructura jerárquica. Se obtiene la clase a la que pertenece cada celda de cualquier nivel. Al finalizar esta etapa se completa la estructura de enlaces, estando las celdas de la base de la pirámide clasificadas en regiones. Únicamente se representan las regiones de espacio libre o inexploradas.

Parámetros
Piramide: estructura jerárquica enlazada.
NumClase: número de clases o regiones. Valor de salida modificado por la función.

Valor devuelto
La estructura jerárquica clasificada.

Llamada por
Módulo *PlanificadorRuta*.

*int *BoundingBox(Celda ***Piramide, int NumClase);*

Descripción
Calcula la *bounding box* de las regiones en la base.

Parámetros
Piramide: la estructura jerárquica creada e inicializada, enlazada y clasificada.
NumClase: número de clases o regiones.

Valor devuelto
Vector con la *bounding box* de todas las regiones.

Llamada por
Módulo *PlanificadorRuta*.

*int **EstudioConectividad(Celda ***Piramide, int NumClase, int *BB);*

Descripción
Analiza de forma no supervisada la relación de conexión entre todas las parejas posibles de regiones en función de la relación espacial definida por sus *bounding boxes* y las celdas pertenecientes a ellas en la base de la pirámide.

Parámetros
Piramide: la estructura jerárquica creada e inicializada, enlazada y clasificada.
NumClase: número de clases o regiones.
BB: vector con la *bounding box* de todas las regiones.

Valor devuelto
Matriz de conexión entre regiones.

Llamada por
Módulo *PlanificadorRuta*.

*int *RegionesNoExploradas(Celda ***Piramide, int NumClase, int **MC, int *NumRegiones);*

Descripción
Obtiene una lista con todos los nodos no explorados accesibles desde la posición del robot. En primer lugar se analiza la lista de nodos del mapa topológico para extraer todos los nodos no explorados del mapa topológico. Después se estudian todos estos nodos no explorados para considerar únicamente aquéllos a los que el robot puede llegar desde su posición actual.

Parámetros
Piramide: la estructura jerárquica creada e inicializada, enlazada y clasificada.
NumClase: número de clases o regiones.
MC: matriz de conexión entre regiones.
NumRegiones: número de regiones no exploradas accesibles. Valor modificado por la función.

Valor devuelto
Vector con las regiones no exploradas accesibles desde la posición del robot.

Llamada por
Módulo *PlanificadorRuta*.

*float **MatrizDistancias(Celda ***Piramide, int *Regiones, int **MC, int NumRegiones);*

Descripción
Obtiene la matriz de distancias entre todas las parejas posibles de nodos no explorados accesibles desde la posición del robot . Si dos nodos no explorados están conectados directamente entre sí, la distancia entre ellos será la correspondiente distancia euclídea de sus centroides asociados. Si no lo están, esta función implementa el algoritmo de búsqueda de caminos de Dijkstra para calcular el camino para ir del nodo origen al nodo destino. Con este camino calculado se puede obtener la distancia entre ellos como la suma de las distancias parciales entre los nodos ordenados del camino.

Parámetros
Piramide: la estructura jerárquica creada e inicializada, enlazada y clasificada.
Regiones: lista de nodos no explorados accesibles.
MC: matriz de conexión entre regiones.
NumRegiones: número de regiones no exploradas accesibles.

Valor devuelto
Matriz de distancias entre cualquier pareja de nodos no explorados accesibles, en la forma de una matriz de *float*.

Llamada por
Módulo *PlanificadorRuta*.

TSP.h y *TSP.c*

Esta librería, también desarrollada en *C*, es necesaria para la correcta planificación de rutas de exploración completa en el módulo *PlanificadorRuta*. Incluye las funciones que implementan el algoritmo para el cálculo de la mejor ruta de exploración de todas las regiones no exploradas accesibles desde la posición del robot, minimizando la distancia recorrida por el agente y visitando cada región una única vez. A continuación se presentan las funciones que forman parte de la librería.

*int *BúsquedaExhaustiva(int *Regiones, float **MD, int NumRegiones);*

Descripción
Implementa el algoritmo recursivo de búsqueda exhaustiva de la ruta óptima de exploración completa.

Parámetros
Regiones: lista de nodos no explorados accesibles.
MD: matriz de distancias entre las regiones.
NumRegiones: número de regiones no exploradas accesibles.

Valor devuelto
Ruta de exploración completa. Es el vector Regiones de entrada ordenado según el orden en el que cada región debe ser visitada.

Llamada por
Módulo *PlanificadorRuta*.

*int *AlgoritmoGenético(int *Regiones, float **MD, int NumRegiones);*

Descripción
Implementa el algoritmo genético para encontrar la mejor ruta de exploración completa que se emplea en esta Tesis. Este algoritmo se presenta en el Capítulo 5. Las regiones se codifican con la *representación por camino*. La población inicial de individuos tiene una tamaño igual al número de nodos del problema, generándose de forma aleatoria. La función objetivo es la longitud total de la ruta. El criterio de selección es el de la selección por torneo. El operador de cruce es el denominado *order crossover*. No se utiliza operador de mutación y el criterio de reducción consiste en seleccionar la mejor población de individuos cuyo tamaño sea igual al de la población inicial.

Parámetros
Regiones: lista de nodos no explorados accesibles.
MD: matriz de distancias entre las regiones.
NumRegiones: número de regiones no exploradas accesibles.

Valor devuelto
Ruta de exploración completa. Es el vector Regiones de entrada ordenado según el orden en el que cada región debe ser visitada.

Llamada por
Módulo *PlanificadorRuta*.

5.3 Utilidades

Ninguna.

6 La capa *InterfazUsuario*

El módulo *Interacción* que implementa esta capa se ha desarrollado en el programa *Interacción.c*. Su misión es la de servir de interfaz entre el usuario del sistema de exploración y la plataforma robótica, tanto simulada como real. Esta capa es analizada con detalle en la Sección 3.4 del Capítulo 2.

6.1 Módulos

Interacción.c

Objetivo

Dos son los objetivos que este módulo persigue. Por un lado, dotar al usuario de un mecanismo para iniciar y parar el proceso de exploración completa en cualquier momento. Por otro, servir de interfaz visual para mostrar en todo instante el estado del robot y de la exploración del entorno de trabajo.

Algoritmo

La interfaz visual se ha desarrollado utilizando la biblioteca *GTK* para *Linux*. Toma como entradas los modelos de entorno generados (en los módulos *MapaMétrico* y *MapaLocal*), la ruta planificada y el estado del robot, para mostrar visualmente esta información. La aplicación permite también guardar las imágenes mostradas para poder extraer resultados.

Entradas

- Estado del robot: lectura de los sensores sonar, posición y orientación, procedentes del módulo *ControlRobot*.
- Mapa métrico: modelo generado del entorno en el módulo *MapaMétrico*.
- Mapa local: modelo generado del entorno más cercano al robot en el módulo *MapaLocal*.
- Ruta: ruta de exploración completa de todas las zonas inexploradas del entorno, generada por el módulo *PlanificadorRuta*. Incluye el punto de destino siguiente para el robot.

Salidas

- Comando de exploración: orden para iniciar/parar el proceso de exploración.

Salidas para

Módulos *ControlRobot*, *SeguirPared*, *SeguirPasillo*, *CruzarPuerta*, *MapaMétrico*, *MapaLocal*, *Fusión* y *PlanificadorRuta*.

6.2 Librerías

Ninguna.

6.3 Utilidades

Ninguna.

Apéndice B

Cuestionarios de Usuario del Control Asistido

En este apéndice se presenta el modelo de cuestionario empleado para las pruebas del aprendizaje asistido del Capítulo 3, así como los cuestionarios reales rellenados por los 13 usuarios participantes.

Tal y como se comentó en la Sección 5.2 del Capítulo 3, al finalizar la sesión de test cada uno de los usuarios rellenó el cuestionario que se muestra en la Tabla B.1 para conocer su opinión respecto del funcionamiento del sistema y poder obtener una realimentación sobre las pruebas. Durante la sesión de test de un usuario se realizaron 5 iteraciones en cada uno de los entornos reales de prueba, dando lugar a 20 trayectorias. Por otro lado, los usuarios realizaron las pruebas en distinto orden, con el objeto de analizar la influencia de la complejidad de las pruebas en los resultados obtenidos. Se puede observar en las Tablas B.2-B.14 que los cuestionarios completados por los usuarios indican el orden en el que se llevaron a cabo dichas pruebas.

Los 4 entornos reales de test se muestran en la Figura 3.43 del Capítulo 3 en su Sección 5.2. Recordamos a continuación los códigos nemónicos empleados para referirnos a cada uno de ellos:

- FW. Entorno donde hay que seguir el contorno de la pared derecha.
- NC. Pasillo estrecho a recorrer por su centro.
- DNC. Pasillo de doble anchura, más ancho que el NC y que también se debe recorrer por el centro.
- PD. Puerta que se debe cruzar sin colisionar.

Una vez puestos en situación, se presenta el modelo de cuestionario (Tabla B.1) y cada uno de los rellenados por los 13 usuarios (Tablas B.2-B.14).

			Sujeto Nº
1. Edad:			
2. Sexo:			
☐ Hombre	☐ Mujer		
3. Conduzco:			
☐ Bicicletas	☐ Motos	☐ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	☐ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	☐ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 ()	Prueba 2 ()	Prueba 3 ()	Prueba 4 ()
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 ()	Prueba 2 ()	Prueba 3 ()	Prueba 4 ()
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 ()	Prueba 2 ()	Prueba 3 ()	Prueba 4 ()
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.1: Modelo de cuestionario de usuario.

			Sujeto Nº 1
1. Edad: 35			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	■ Regular	☐ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	■ No		
6. Uso videojuegos:			
☐ Nunca	■ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	☐ Poco	☐ Bastante	■ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	■ 1	☐ 1	☐ 1
■ 2	☐ 2	■ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	■ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
■ 4	☐ 4	■ 4	☐ 4
☐ 5	■ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	■ 1	■ 1	☐ 1
■ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.2: Cuestionario del Sujeto Nº1.

			Sujeto Nº 2
1. Edad: 27			
2. Sexo:			
■ Hombre	☐ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	☐ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	☐ Ocasionalmente	■ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	☐ Poco	☐ Bastante	■ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	■ 1	☐ 1	■ 1
■ 2	☐ 2	■ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
■ 4	☐ 4	■ 4	☐ 4
☐ 5	■ 5	☐ 5	■ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	■ 1	■ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.3: Cuestionario del Sujeto Nº2.

			Sujeto Nº 3
1. Edad: 34			
2. Sexo:			
■ Hombre	☐ Mujer		
3. Conduzco:			
☐ Bicicletas	☐ Motos	☐ Coches	
4. Mi pulso es:			
☐ Malo	■ Regular	☐ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	☐ Ocasionalmente	☐ Semanalmente	■ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (PD)	Prueba 2 (FW)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	☐ 1	☐ 1
■ 2	☐ 2	■ 2	■ 2
☐ 3	■ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (PD)	Prueba 2 (FW)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
■ 3	■ 3	☐ 3	■ 3
☐ 4	☐ 4	■ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (PD)	Prueba 2 (FW)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	■ 2	☐ 2
■ 3	■ 3	☐ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.4: Cuestionario del Sujeto Nº3.

			Sujeto Nº 4
1. Edad: 35			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
☐ Bicicletas	☐ Motos	☐ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	■ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (NC)	Prueba 2 (DNC)	Prueba 3 (PD)	Prueba 4 (FW)
■ 1	■ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	■ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (NC)	Prueba 2 (DNC)	Prueba 3 (PD)	Prueba 4 (FW)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	■ 3	■ 3
■ 4	■ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (NC)	Prueba 2 (DNC)	Prueba 3 (PD)	Prueba 4 (FW)
☐ 1	☐ 1	☐ 1	☐ 1
■ 2	■ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	■ 4	■ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.5: Cuestionario del Sujeto Nº4.

			Sujeto Nº 5
1. Edad: 26			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	■ Regular	☐ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	■ No		
6. Uso videojuegos:			
■ Nunca	☐ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (PD)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
■ 3	☐ 3	☐ 3	☐ 3
☐ 4	■ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (PD)	Prueba 3 (NC)	Prueba 4 (DNC)
■ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	■ 3	☐ 3	☐ 3
☐ 4	☐ 4	■ 4	☐ 4
☐ 5	☐ 5	☐ 5	■ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (PD)	Prueba 3 (NC)	Prueba 4 (DNC)
■ 1	☐ 1	■ 1	■ 1
☐ 2	■ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.6: Cuestionario del Sujeto Nº5.

			Sujeto Nº 6
1. Edad: 23			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	☐ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	■ No		
6. Uso videojuegos:			
☐ Nunca	■ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (PD)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	■ 1	☐ 1
■ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	■ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (PD)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	■ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	■ 4
■ 5	☐ 5	■ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (PD)	Prueba 3 (NC)	Prueba 4 (DNC)
■ 1	☐ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
☐ 4	■ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.7: Cuestionario del Sujeto Nº6.

			Sujeto Nº 7
1. Edad: 35			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
☐ Bicicletas	☐ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	■ Regular	☐ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	■ No		
6. Uso videojuegos:			
■ Nunca	☐ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (PD)	Prueba 2 (FW)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
■ 5	■ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (PD)	Prueba 2 (FW)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	■ 2	☐ 2	☐ 2
■ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	■ 4
☐ 5	☐ 5	■ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (PD)	Prueba 2 (FW)	Prueba 3 (NC)	Prueba 4 (DNC)
☐ 1	☐ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
■ 3	☐ 3	☐ 3	☐ 3
☐ 4	■ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.8: Cuestionario del Sujeto Nº7.

			Sujeto Nº 8
1. Edad: 30			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
■ Bicicletas	■ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	■ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	■ 1	☐ 1	☐ 1
■ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	■ 4
☐ 5	☐ 5	■ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
☐ 3	☐ 3	☐ 3	☐ 3
■ 4	☐ 4	☐ 4	☐ 4
☐ 5	■ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	■ 4
☐ 5	☐ 5	■ 5	☐ 5
Observaciones:			

Tabla B.9: Cuestionario del Sujeto Nº8.

			Sujeto Nº 9
1. Edad: 32			
2. Sexo:			
■ Hombre	☐ Mujer		
3. Conduzco:			
■ Bicicletas	■ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	☐ Ocasionalmente	■ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	☐ Poco	☐ Bastante	■ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	■ 1	☐ 1	☐ 1
■ 2	☐ 2	■ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
■ 4	☐ 4	■ 4	■ 4
☐ 5	■ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.10: Cuestionario del Sujeto Nº9.

			Sujeto Nº 10
1. Edad: 31			
2. Sexo:			
■ Hombre	☐ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	■ Regular	☐ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	■ No		
6. Uso videojuegos:			
☐ Nunca	☐ Ocasionalmente	■ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	☐ Poco	☐ Bastante	■ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	☐ 1	☐ 1
☐ 2	☐ 2	■ 2	■ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
■ 4	☐ 4	■ 4	■ 4
☐ 5	■ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	■ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.11: Cuestionario del Sujeto Nº10.

			Sujeto Nº 11
1. Edad: 34			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
☐ Bicicletas	☐ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
■ Nunca	☐ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
■ Ninguno	☐ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	■ 2	☐ 2	☐ 2
☐ 3	☐ 3	■ 3	☐ 3
■ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	■ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
☐ 4	☐ 4	■ 4	☐ 4
■ 5	■ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
■ 3	■ 3	■ 3	☐ 3
☐ 4	☐ 4	☐ 4	■ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.12: Cuestionario del Sujeto Nº11.

			Sujeto Nº 12
1. Edad: 35			
2. Sexo:			
■ Hombre	☐ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	☐ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
■ Sí	☐ No		
6. Uso videojuegos:			
☐ Nunca	■ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	■ Poco	☐ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	■ 1	☐ 1	☐ 1
■ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	■ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
■ 4	☐ 4	■ 4	■ 4
☐ 5	■ 5	☐ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
☐ 3	☐ 3	■ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.13: Cuestionario del Sujeto Nº12.

			Sujeto Nº 13
1. Edad: 34			
2. Sexo:			
☐ Hombre	■ Mujer		
3. Conduzco:			
■ Bicicletas	☐ Motos	■ Coches	
4. Mi pulso es:			
☐ Malo	☐ Regular	■ Bueno	☐ Muy bueno
5. Uso gafas habitualmente:			
☐ Sí	■ No		
6. Uso videojuegos:			
☐ Nunca	■ Ocasionalmente	☐ Semanalmente	☐ Casi diariamente
☐ Todos los días			
7. Mi nivel de relación con la robótica es:			
☐ Ninguno	☐ Poco	■ Bastante	☐ Mucho
8. Puntúe la dificultad de las pruebas de 1 a 5 (1 mínima dificultad, 5 máxima dificultad):			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	☐ 1	☐ 1
☐ 2	☐ 2	■ 2	☐ 2
☐ 3	☐ 3	☐ 3	■ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
9. Puntúe el grado de control que ha experimentado en cada una de las pruebas de 1 a 5 (1 mínimo control, 5 máximo control)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
☐ 1	☐ 1	☐ 1	☐ 1
☐ 2	☐ 2	☐ 2	■ 2
☐ 3	☐ 3	☐ 3	☐ 3
■ 4	☐ 4	☐ 4	☐ 4
☐ 5	■ 5	■ 5	☐ 5
10. Puntúe el grado de estrés en cada prueba de 1 a 5 (1 mínimo estrés, 5 máximo estrés)			
Prueba 1 (FW)	Prueba 2 (NC)	Prueba 3 (DNC)	Prueba 4 (PD)
■ 1	■ 1	■ 1	■ 1
☐ 2	☐ 2	☐ 2	☐ 2
☐ 3	☐ 3	☐ 3	☐ 3
☐ 4	☐ 4	☐ 4	☐ 4
☐ 5	☐ 5	☐ 5	☐ 5
Observaciones:			

Tabla B.14: Cuestionario del Sujeto Nº13.