Departamento de Sistemas Telemáticos y Computación (GSyC)
Universidad Rey Juan Carlos

Tesis Doctoral

Técnicas de percepción compartida aplicadas a la asignación dinámica de roles en equipos de robots móviles

Carlos Enrique Agüero Durán
carlos.aguero@urjc.es

Directores de Tesis:
Jose María Cañas Plaza
josemaria.plaza@urjc.es
Vicente Matellán Olivera
vicente.matellán@unileon.es

Carlos Enrique Agüero Durán

Departamento de Sistemas Telemáticos y Computación (GSyC)

Escuela Técnica Superior de Ingeniería de Telecomunicación

Universidad Rey Juan Carlos

Av. Tulipán s/n

Móstoles 28933 Madrid

Correo electrónico: carlos.aguero@urjc.es

Web: http://gsyc.es/∼caguero

Autorizamos la defensa de la tesis doctoral *Técnicas de percepción compartida aplicadas a la asignación dinámica de roles en equipos de robots móviles* cuyo autor es D. Carlos Enrique Agüero Durán.

José María Cañas Plaza Vicente Matellán Olivera

Móstoles, 10 de Noviembre de 2009

TESIS DOCTORAL: Técnicas de percepción compartida aplicadas a la asignación dinámica de roles en equipos de robots móviles

AUTOR: D. Carlos Enrique Agüero Durán
DIRECTORES: Dr. José María Cañas Plaza
Dr. Vicente Matellán Olivera

El tribunal nombrado para juzgar la Tesis arriba indicada, compuesto por los siguientes doctores:

PRESIDENTE:

VOCALES:

SECRETARIO:

acuerda otorgarle la calificación de:

Móstoles, de de

El Secretario del Tribunal

A todos aquellos que mantienen la ilusión por vivir y a los que ya no están pero vivieron con la misma ilusión

Agradecimientos

Dr. Son sólo dos letras que marcan el fin de una etapa y abren las puertas de otra. No voy a negar que reconforta llegar al final del camino, pues echando la mirada atrás se pueden ver aún las pisadas humeantes de la senda caminada. Sin embargo, este viaje no ha hecho más que empezar y tras la euforia debe reanudarse la marcha con más fuerza y decisión si cabe.

Como en el cine, la película que aquí se narra no podría haber llegado a buen puerto sin los directores Vicente y Jose Mª. Desde que escribí mis primeros guiones ellos fueron mi fuente de inspiración y modelo a seguir. Ahora han conseguido que tenga un conejo, un dinosaurio y un perro robot en mi mesa. Vicente, gracias por enseñarme cómo divertirse con la robótica. Jose Mª, gracias por enseñarme cómo se logran las cosas. Gracias a los dos por confiar en mí.

Pero qué sería de una película sin sus actores protagonistas... A Paco, a todos los miembros de la historia del grupo de Robótica, a todos los integrantes del equipo TeamChaos y a toda la comunidad robótica que ha influido en mí, gracias. Gracias con mayúscula pues he pasado muchas horas con vosotros, horas de valor incalculable en esfuerzo y aprendizaje pero también en diversión. Vosotros sois parte de este trabajo pues me habéis alentado cuando lo necesitaba y criticado cuando era preciso, pero sobre todo me habéis acompañado a lo largo de esta andadura. Gracias por tan mágicos momentos.

Por supuesto que los actores secundarios tienen que tener su reconocimiento, ¡faltaría más! Sois vosotros, todos los miembros del GSyC. Pueden cambiar el nombre del departamento una y otra vez pero habéis demostrado que la filosofía del trabajo bien hecho no depende de nombres, solo de personas y tajo. Gracias por enseñarme a enseñar y por sentir con dedicación y transparencia vuestra profesión, alejada de los géneros que poco tienen que ver con la labor de un docente.

Mi familia y amigos siempre han estado aquí. Es mi equipo técnico desde que aparecí por primera vez en sus retinas. Les he cogido cariño y parece que ellos a mí también, así que no me queda más que agradecerles su desinteresada y siempre dispuesta gratitud, cariño y amistad. Nunca habéis fallado, eso es profesionalidad. Gracias por quererme, divertirme, acompañarme, escucharme, protegerme y educarme como lo habéis hecho.

Hay una persona que también forma parte del elenco de esta película y merece ser felicitada con todos los honores. Muky (si, Muky), gracias, gracias, gracias, por entender lo que significa este trabajo y sacrificar parte de tu tiempo. Sé que en nuestra *biblioteca* tú también estudias, pero además de hacerlo muy bien has permitido que yo me relajara y disfrutara de esta tesis. Sabes muy bien que la vida es dura, pero también tiene momentos buenos si consigues levantarte. Tú lo has hecho y éste es uno de ellos: Recoge tu premio.

Gracias.

Resumen

El camino que ha seguido la robótica a lo largo de sus escasos sesenta años de vida ha estado marcado por los avances y progresos en áreas bien conocidas como la manipulación, navegación, construcción de mapas, auto-localización, percepción del entorno o desarrollo de arquitecturas, entre otras. Debido al auge de la robótica y a la aparición de aplicaciones donde es necesario el empleo de múltiples robots, ha surgido en los últimos años el concepto de sistema multi-robot. Lejos de consistir en una mera agrupación pasiva de miembros, los individuos colaboran entre sí para obtener un beneficio mayor.

La cooperación multi-robot es un terreno activo de investigación, que busca desarrollar algoritmos que saquen provecho del empleo de múltiples agentes o individuos trabajando en una tarea común. Las tareas que mejor se adaptan a equipos de robots cooperantes son aquellas fácilmente divisibles en subtareas, que serán asignadas a cada miembro de la sociedad robótica.

Esta tesis aborda el problema de la percepción y la asignación de tareas en equipos de robots. Por un lado, hoy en día las cámaras son uno de los sensores principales de cualquier robot. Son sensores baratos y su flujo de información es elevado. Sin embargo, el procesamiento visual asociado no está exento de errores e imprecisiones que dificultan las tareas de estimación de objetos. La percepción distribuida es una aplicación potencial de un conjunto de robots cooperativos. Mediante ella es posible mejorar la percepción que cada robot tiene de manera individual en muchos aspectos como la precisión, robustez, comportamiento ante falsos negativos o positivos, etc. En esta tesis se plantea el problema de la percepción distribuida y se proponen diferentes soluciones, que utilizan la probabilidad como motor para acumular evidencias y para modelar la incertidumbre. A la hora de estimar la posición de un objeto del entorno, suelen existir dos fuentes principales de incertidumbre. La primera de ellas es la incertidumbre asociada a la propia observación. La segunda es la incertidumbre unida a la posición del observador. Ambas deterioran la estimación del objeto y deben ser tenidas en cuenta de manera apropiada para lograr una estimación óptima.

Por otro lado se aborda la descomposición en roles dentro de un equipo colaborativo de robots. Cada uno de estos roles tiene asociado un comportamiento específico, que complementa el trabajo del resto de miembros. En ocasiones no es buena idea que estos roles sean asignados de manera fija, pues ante situaciones inesperadas, cambios en el entorno, etc. es posible que exista otra configuración de roles más adecuada. El segundo pilar de esta tesis estudia este problema y propone una solución al problema del intercambio dinámico de roles en equipos de robots. Esta solución establece cuándo y cómo se han de realizar los intercambios, especifica cuál es la formación más adecuada del equipo de robots según las condiciones actuales del entorno y resuelve el intercambio de información entre los diferentes individuos del sistema multi-robot.

Los cuestiones aquí abordadas han sido implementadas en *software* y los resultados han sido obtenidos gracias a experimentos en dos plataformas robóticas diferentes: el robot aiBo de la empresa Sony y el robot Nao de la empresa Aldebaran Robotics. La elección de estos dos robots no es casual, puesto que han sido las plataformas de referencia de la liga de plataforma estándar de la competición robótica internacional RoboCup. Este prestigioso evento reúne a centros de todo el mundo para competir en diferentes ligas y servir de estímulo y campo de experimentación para promover el desarrollo de la Robótica. La liga que nos ocupa enfrenta a dos equipos de robots en un partido de fútbol.

Este entorno de experimentación ha guiado el desarrollo de esta tesis, pues las técnicas de percepción distribuida aquí propuestas pretenden mejorar el comportamiento del equipo ante la estimación del objeto más importante de la competición: la pelota de juego. La división de tareas presentada en este trabajo de investigación ha sido aplicada a este problema para asignar los diferentes roles de juego de manera dinámica dentro del transcurso de cada partido. La técnica de percepción distribuida desarrollada se integra de manera efectiva en este algoritmo de selección de tareas. La posición de la pelota es un parámetro clave a la hora de decidir la mejor configuración de roles. La incorporación de la estimación distribuida otorga mayor robustez a la decisión y mejora la selección de acción del equipo.

Abstract

The evolution of robotics along its sixty years old of history has been guided by the improvements and advances in the classical and well known topics as manipulation, navigation, map's creation, self-location, perception of the environment or the development of software architectures, among others. Due to the growth of robotics and the increase of applications where the use of several robots are necessary, the term multi-robot system has emerged. Far from being simple and passive members, the robots collaborate themselves to obtain some benefit or advantage.

Multi-robot cooperation is an active research field that tries to develop algorithms that obtain some kind of profit, due to the use of a team of robots working in a common task. The best suited tasks for cooperative robot teams are those that can be decomposed in multiple subtasks. Each subtask will be associated to each member of the robotics society.

This doctoral thesis tackles the problem of perception and task assignment in robot teams. Nowadays the cameras are one of the most used sensor in robotics. They are cheap and its information stream is high. However, the visual processing phase associated is not perfect and adds errors and imprecisions that make difficult the object estimation. Shared perception is a potential application of a cooperative multi-robot group. Using shared perception it can improve the local perception in several aspects like accuracy, robustness, behavior in the presence of false positives or false negatives, etc. Inside this thesis we raise the problem of distributed perception and we propose some solutions that make use of the probability as the tool for accumulate evidences and deal with uncertainty. When a robot tries to estimate the position of an object in the environment, it normally has two sources of uncertainty. One is the uncertainty associated to the observation and the other is the uncertainty associated to the position of the observer. Both cause a deterioration of the object estimation and we should take into account both sources of uncertainty to fulfill an optimal estimation

This thesis also raises the roles decomposition inside a shared robot team. Each role is combined with an specific behavior that complements the work of the other

members. Sometimes it is not a good idea to assign those roles in a fix way, since changes in the environment or unexpected situations create the necessity to find a better role configuration. The second topic of this thesis studies this matter and proposes a solution to the problem of dynamic roles assignment in multi robot teams. This solution defines when the robot should exchange their roles and the way of doing it, it also specifies the most appropriate formation according to the actual conditions of the environment, and it solves the information communication process among the members of the team.

The questions proposed have been implemented in software and the results have been extracted from experiments using two different robotics platforms: the aiBo robot from Sony and the Nao robot from Aldebaran-Robotics. The choice of those robots is not casual. They have been the reference robots of the standard platform league of the RoboCup international competition. This prestigious contest meets research centers for competing in different leagues. It serves as an incentive for promoting the robotics research and offers an experimentation frame for researchers all over the world. The standard platform league faces up two teams of robots in a soccer match.

The experimentation field related to the RoboCup has guided the development of this thesis, since the distributed perception techniques proposed expect to improve the behavior of the team, estimating the most important object of the competition: the ball. The task decomposition presented in this research work has also been applied to this problem to assign different roles in a dynamic manner during the matches. The distributed perception technique developed is fully integrated in an effective way inside the task selection algorithm. The position of the ball is a key element when the team is trying to decide the best role configuration. The addition of the distributed ball estimator grants better robustness to the decision and improves the selection of actions in the team.

ÍNDICE GENERAL

1. Introducción **1**

1.1. Robótica 2
1.2. Robótica Móvil 12
1.3. Robótica cooperativa 17
1.4. Entorno de trabajo: RoboCup - Liga de plataforma estándar 21
1.4.1. Plataforma de experimentación: Robots aiBo y Nao 24
1.5. Objetivos 27
1.6. Propuesta de jugador robótico cooperativo 31
1.7. Estructura de la tesis 32

2. Estado del arte **35**

2.1. Robótica cooperativa 36
2.1.1. Control Centralizado Vs Distribuido 38
2.1.2. Homogeneidad Vs heterogeneidad 39
2.1.3. Mecanismos de coordinación 41
2.1.4. Tipos de comunicación 44
2.1.5. Interferencias y conflictos 45
2.2. Aplicaciones de los sistemas multi-robot 46
2.2.1. Recolección o *foraging* 46

2.2.2. Formaciones 47
2.2.3. Gestión de alcenamiento 48
2.2.4. Manipulación coordinada 49
2.2.5. Fútbol robótico 50
2.2.6. Construcción distribuida de mapas 52
2.2.7. Percepción distribuida 55
2.2.8. Coordinación de actuadores 56
2.3. Percepción coordinada de objetos 58
2.3.1. Fusión sensorial basada en operadores estadísticos 62
2.3.2. Fusión sensorial basada en filtro de Kalman 63
2.3.3. Fusión sensorial basada en métodos de Monte Carlo 64
2.3.4. Fusión sensorial basada en métodos híbridos 64
2.4. División de tareas en equipos de robots 68
2.4.1. GermanTeam 69
2.4.2. rUNSWift 70
2.4.3. NUbots 73
2.4.4. CAMBADA 75
2.4.5. CS Freiburg 77
2.5. Discusión 77

3. Estimación coordinada de objetos dinámicos **81**
3.1. Fundamentos teóricos 82
3.1.1. Filtros Bayesianos 82
3.1.2. Rejillas probabilísticas 85
3.1.3. Filtro de Kalman 87
3.1.4. Filtro de partículas 94
3.2. Percepción distribuida de la pelota en la RoboCup 97
3.2.1. Modelo de observación visual 98
3.2.2. Fusión analítica 109
3.2.3. Fusión de rejillas probabilísticas 115
3.2.4. Fusión basada en filtro de Kalman 120
3.2.5. Fusión basada en filtros de partículas 123

3.3. Experimentos 130
3.3.1. Precisión 132
3.3.2. Mediciones de eficiencia computacional 140
3.3.3. Comportamiento ante oclusiones en uno o varios robots . . . 142
3.3.4. Comportamiento ante falsos positivos 149
3.3.5. Comportamiento ante alta incertidumbre en la posición del observador 161
3.3.6. Comportamiento ante objetos dinámicos 169
3.4. Discusión 179

4. Asignación dinámica de roles **183**
4.1. Fundamentos teóricos 185
4.1.1. Asignación óptima 185
4.1.2. Utilidad 186
4.1.3. Roles y mecanismos de comunicación 187
4.2. *Switch!:* Asignación dinámica de roles en la RoboCup 188
4.2.1. Especificación de roles para el juego 188
4.2.2. Funciones heurísticas para el cálculo de utilidad 190
4.2.3. Posicionamiento de cada rol de juego 193
4.3. Experimentos 197
4.3.1. Coordinación con *Switch!* en el simulador 197
4.3.2. Coordinación con *Switch!* en robots aiBo reales 200
4.3.3. Reto del pase en la RoboCup 203
4.3.4. Competición *German Open* 2007 209
4.4. Conclusiones 212

5. Conclusiones **215**
5.1. Percepción distribuida aplicada a la asignación dinámica de roles en equipos de robots móviles 216
5.2. Aportes realizados 220
5.3. Líneas futuras 222

A. Software del equipo Team Chaos para el robot aiBo **225**
A.1. Arquitectura general 226
A.2. Módulo de comunicaciones 230
A.3. La aplicación *ChaosManager* 232

B. Software BICA para el robot NAO **237**
B.1. Arquitectura basada en comportamientos 242
B.2. Diseño del jugador de fútbol robótico 244
B.2.1. Percepción del jugador de fútbol 245
B.2.2. Movimientos básicos 247
B.2.3. Comportamiento para seguir la pelota con la mirada 250
B.2.4. Comportamiento para seguir la pelota con el cuerpo 251
B.2.5. Comportamiento de búsqueda de la pelota 252
B.2.6. Comportamiento de búsqueda de porterías 253
B.2.7. Comportamiento de jugador de campo 254
B.3. *Manager* 255
B.3.1. Herramienta de depuración de componentes 256
B.3.2. Herramienta de calibración del sistema de percepción visual . 256
B.3.3. Herramienta de generación de movimientos predefinidos . . . 257

ÍNDICE DE FIGURAS

1.1. Robots de la fábrica de automóviles SEAT en Martorell (España) ensamblando piezas (izquierda) y realizando labores de pintado (derecha) 4
1.2. Robot Kawasaki para transporte pesado de mercancía (izquierda) y robots móviles de la empresa Kiva Systems para transporte ligero (derecha) 5
1.3. Robots destinados al servicio particular: Aspiradora Roomba®(izquierda) y limpiador de piscinas Verro (derecha) 6
1.4. Tercera generación del robot ARMAR destinado a convivir en viviendas 7
1.5. Robot guía Rhino (izquierda) del Deutsches Museum Bonn y robot guía Minerva (derecha) del Smithsonian's National Museum of American History 7
1.6. Robot Kismet (izquierda) y robot Nexi (derecha) propiedad del MIT destinados a fomentar la investigación en interacción social 8
1.7. Robot PackBot (izquierda) y robot MARCbot (derecha) utilizados en conflictos armados reales 9
1.8. Robot Spirit (izquierda) e imagen real tomada en una situación complicada de atasco en la arena de Marte (derecha) 9
1.9. Robot humanoide Robonova (izquierda) y construcción de un plotter usando el kit Lego NXT por el estudiante Krystian Majewski (derecha) 10

1.10. Robot Pioneer equipado con un sensor láser y una cámara motorizada (izquierda) y robot Nao promocionando la competición RoboCup 2009 (derecha) 11
1.11. Imagen original del robot Elsie sin su caparazón protector (izquierda) y robots Shakey (centro) y Flakey (derecha) diseñados en el SRI . . . 13
1.12. Robot Asimo subiendo escaleras (izquierda), captura de movimiento del robot de Toyota con los dos pies en el aire mientras corría (centro) y robot mula Big Dog (derecha) 15
1.13. Sony AIBO ERS7 (Foto de Sony) 25
1.14. Robot NAO de la compañía Aldebaran Robotics 27
1.15. En este instante de la competición *Roboludens 2006*, la casualidad hizo que esta niña vistiera con los mismos colores que la baliza mostrada en el campo, dificultando la auto-localización de algunos robots 29

2.1. Esquema de intercambio de información en la liga de robots de pequeño tamaño (izquierda) e instante de una competición real (derecha) 39
2.2. Instantes de las competiciones de robots de tamaño medio (izquierda) y humanoides (derecha) de la edición 2008 de la RoboCup celebrada en Suzhou (China) 39
2.3. Robots Ganymede, Io y Callisto 40
2.4. Robots del proyecto *The Nerd Herd* (izquierda) de la Universidad South California (EEUU), junto con su autora Maja Mataric (derecha) 42
2.5. Robots *Sally* y *Shannon* (izquierda) pertenecientes a Georgia Institute of Technology participando en la competición de saqueo *Find Life on Mars* (derecha) dentro del *AAAI-97 Robot Competition* 47
2.6. Equipo de cuatro camiones militares propiedad de DARPA durante el proyecto *Demo II Project* (izquierda) y formaciones en línea, columna, diamante y cuña (derecha) 48
2.7. Robots de *Kiva Systems* (izquierda) e instalaciones en Denver (derecha) donde los robots realizan trabajos de almacén 49
2.8. Seis barcos autónomos remolcando de manera coordinada una barcaza simulada 50

2.9. Simulador *Soccer server* empleado en la competición RoboCup (izquierda) e instante de un partido de una competición organizada por la FIRA (derecha) 51
2.10. Mapa creado dentro del proyecto Centibots por un enjambre de robots 52
2.11. Imagen obtenida por el robot que monitoriza a su compañero 54
2.12. Robot Silo6, propiedad del Instituto de Automática Industrial (España) 57
2.13. Reconstrucciones 3D realizadas por el robot *Junior* en distintos momentos de la competición *DARPA Urban Challenge* 59
2.14. Diagrama de posicionamiento de roles. Estrategia *Striker / Defender* (izquierda) y *Supporter / Defender* (derecha) (Foto original de rUNSWift) 71

3.1. Ejemplo de función de densidad de probabilidad bidimensional para expresar la posición de un objeto en un plano 84
3.2. Rejilla probabilística de dos dimensiones para estimar la posición de un objeto 86
3.3. Función de densidad de probabilidad para expresar la posición condicionada a la observación z_1 88
3.4. Función de densidad de probabilidad para expresar la posición condicionada a la observación z_2 90
3.5. Función de densidad de probabilidad para expresar la posición condicionada a las observaciones z_1 y z_2 91
3.6. Propagación de la función de densidad de probabilidad en el tiempo . 93
3.7. Función de densidad de probabilidad (a) que puede ser representada mediante una función Gaussiana parametrizable mediante su media y desviación típica y función de densidad de probabilidad (b) que no puede representarse así. En ambos casos puede usarse una aproximación para representar las funciones formada por un conjunto de partículas cuyos pesos representan la densidad de probabilidad 95
3.8. Proceso de extracción de información de una imagen 99

3.9. Escenario de pruebas para la obtención del modelo de observación visual. Se puede observar la posición del robot aiBo y los diferentes puntos de control a diferentes distancias y ángulos 101
3.10. Sistema de *log* incluido en ChaosManager 102
3.11. Error en la estimación de distancia a la pelota 103
3.12. Error en la estimación de ángulo a la pelota 104
3.13. Diferentes etapas del procesamiento visual realizado sobre una pelota y una portería en el simulador Webots 106
3.14. Error en la estimación de distancia a la pelota 107
3.15. Error en la estimación de ángulo a la pelota 108
3.16. FDP Gaussiana correspondiente a la observación relativa al robot . . . 110
3.17. FDP Gaussiana correspondiente a la posición absoluta del robot . . . 111
3.18. FDP Gaussiana correspondiente a la estimación individual absoluta de la pelota 112
3.19. Esquema de funcionamiento de la variante analítica de percepción distribuida 114
3.20. Fusión (elipse azul) procedente de tres estimaciones locales (elipse naranja) 115
3.21. Rejilla probabilística asociada a una posición de pelota 116
3.22. Rejilla probabilística asociada a la posición del observador (x, y, θ) . . 117
3.23. Proceso de fusión entre la FDP con la observación relativa y la FDP de auto-localización 118
3.24. Combinación de dos rejillas de probabilidad con la estimación de la pelota 119
3.25. Diferentes aproximaciones con 30º, 60º, 90º y 120º de incertidumbre de orientación en el robot observador 121
3.26. Esquema de funcionamiento del estimador distribuido basado en filtro de Kalman 122
3.27. Ejemplo simplificado de varias partículas 3D representando posibles posiciones del observador 124
3.28. Ejemplo simplificado de FDP muestreada 2D representando la observación relativa de la pelota 125

3.29. Ejemplo simplificado de FDP muestreada 2D representando la posición de la pelota en coordenadas globales 127
3.30. Esquema del algoritmo de fusión sensorial basado en filtro de partículas 128
3.31. Esquema con las hipótesis del filtro de partículas (blanco) e hipótesis recibidas del resto de compañeros (azul, naranja y rosa) 129
3.32. Detalle del paso de corrección para evaluar el nuevo factor de importancia de una hipótesis del filtro de partículas 129
3.33. Diversos instantes de los experimentos de percepción distribuida tanto en simulación como en entorno real con las dos plataformas utilizadas: Robot aiBo y robot Nao 131
3.34. Montaje experimental para pruebas de precisión 133
3.35. Comparación del error cometido en la estimación local y la estimación compartida mediante fusión analítica 134
3.36. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en rejilla probabilística 135
3.37. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman . . 136
3.38. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de partículas . 138
3.39. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas 139
3.40. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas 140
3.41. Distribución del consumo de CPU dentro de cada iteración 141
3.42. Montaje para pruebas de oclusiones en las observaciones de los robot 142
3.43. Resultados obtenidos ante oclusiones en un robot 143
3.44. Resultados obtenidos ante oclusiones en dos robots 143
3.45. Resultados obtenidos ante oclusiones en un robot usando rejillas . . . 144
3.46. Resultados obtenidos ante oclusiones en dos robots usando rejillas . . 144

3.47. Resultados obtenidos ante oclusiones en un robot utilizando filtro de Kalman . . . 145
3.48. Resultados obtenidos ante oclusiones en dos robots utilizando filtro de Kalman . . . 145
3.49. Resultados obtenidos ante oclusiones en un robot utilizando filtro de partículas . . . 147
3.50. Resultados obtenidos ante oclusiones en dos robots utilizando filtro de partículas . . . 148
3.51. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas . . . 149
3.52. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas . . . 150
3.53. Montaje experimental para pruebas de falsos positivos . . . 151
3.54. Resultados obtenidos ante un falso positivo con baja incertidumbre . . 152
3.55. Resultados obtenidos ante un falso positivo con alta incertidumbre . . 152
3.56. Resultados obtenidos ante dos falsos positivos con incertidumbre media 153
3.57. Resultados obtenidos ante cuatro falsos positivos puntuales . . . 153
3.58. Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con baja incertidumbre utilizando rejillas probabilísticas . . . 154
3.59. Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con alta incertidumbre utilizando rejillas probabilísticas . . . 154
3.60. Resultados obtenidos ante dos falsos positivos con incertidumbre media 155
3.61. Resultados obtenidos ante un falso positivo puntual . . . 155
3.62. Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con baja incertidumbre utilizando un filtro de Kalman . . . 156

3.63. Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con alta incertidumbre utilizando un filtro de Kalman 156
3.64. Resultados obtenidos ante dos falsos positivos con incertidumbre media 157
3.65. Resultados obtenidos ante un falso positivo puntual 157
3.66. Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con baja incertidumbre utilizando filtros de partículas 158
3.67. Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con alta incertidumbre utilizando filtros de partículas 158
3.68. Resultados obtenidos ante dos falsos positivos con incertidumbre media 159
3.69. Resultados obtenidos ante tres falsos positivos puntuales 159
3.70. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas 160
3.71. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas 161
3.72. Escenario para experimentos con elevada incertidumbre en la posición del observador 162
3.73. Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots 162
3.74. Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots 163
3.75. Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots 163
3.76. Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots 164
3.77. Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots 165

3.78. Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots 166
3.79. Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots 166
3.80. Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots 167
3.81. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas 168
3.82. Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas 169
3.83. Escenario para experimentos con la pelota en movimiento 170
3.84. Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método analítico de fusión 171
3.85. Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método de fusión de rejillas probabilísticas 173
3.86. Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de Kalman 174
3.87. Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de partículas 175
3.88. Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de Kalman 176
3.89. Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de partículas 177

3.90. Diagrama comparativo entre las técnicas propuestas de percepción distribuida 180

4.1. Puntos de interés para el cálculo de la posición óptima de defensa . . 194
4.2. Puntos de interés para el cálculo de la posición óptima de atacante . . 196
4.3. Posición del defensa ante diferentes situaciones de juego 198
4.4. Posición del atacante ante diferentes situaciones de juego 199
4.5. Experimento de asignación del rol de chutador 201
4.6. Secuencia de fotogramas donde se aprecia una situación clásica de *patio de colegio*, donde todos los robots tratan de ir a por la pelota sin ninguna coordinación entre sí 202
4.7. Secuencia de fotogramas de un instante del experimento de *Switch!* utilizando roles fijos 202
4.8. Secuencia de fotogramas de un instante del experimento de *Switch!* utilizando roles dinámicos 204
4.9. Instantánea del entorno de pruebas recreando el desafío del pase de la RoboCup 2006 205
4.10. Captura de la herramienta HFSM que muestra los estados y transiciones que conforman el diagrama de comportamientos para el desafío del pase 207
4.11. Secuencia de fotogramas de un instante de juego real en la competición German Open 2007 donde se aprecia el funcionamiento de *Switch!* 211

5.1. Diagrama comparativo entre las técnicas propuestas de percepción distribuida 219

A.1. Arquitectura del TeamChaos para el robot aiBo 228
A.2. Entradas y salidas del módulo GM 228
A.3. Objeto de tipo ExternalMessage con sus campos relevantes 231
A.4. Esquema de comunicación entre los módulos y el TCM 232
A.5. Editor de lenguaje LUA para escribir y depurar comportamientos básicos 234

A.6. Pestaña integrada en la aplicación ChaosManager para visualizar información sobre la estimación compartida 235

B.1. Ejemplos de árbol de *brokers* 238
B.2. Módulos incluidos en el *MainBroker* 239
B.3. Dimensiones de la cabeza del robot Nao y disposición de sus cámaras 240
B.4. Simulador Webots y robot real 241
B.5. Entradas y salidas de la entidad componente 242
B.6. Árbol de activación con varios niveles de componentes (izquierda) y árbol con un componente utilizado por dos componentes *padre* (derecha) 243
B.7. Árbol de activación con el componente *root* en el nivel más alto. Típicamente a mayor nivel de abstracción de un módulo, menor es su frecuencia 245
B.8. Posición 3D de la pelota y su sistema de referencia 247
B.9. Extracción de características visuales de la portería 248
B.10. Componente *Body* y sus componentes de nivel inferior que utilizan NaoQi para desplazar el robot 249
B.11. Árbol de activación del componente *FaceBall* 251
B.12. Árbol de activación del componente *FollowBall* 252
B.13. Árbol de activación en cada posible estado del componente *SearchBall* 253
B.14. Máquina de estados del componente *Player* con su correspondiente árbol de activación 255
B.15. Herramienta de depuración de componentes trabajando sobre el componente *Turn* 256
B.16. Ajuste de los parámetros de una de las cámaras del robot Nao 257

CAPÍTULO 1

Introducción

I can't define a robot, but I know one when I see one.

– Joseph Engelberger

La resolución de problemas de manera cooperativa entre varios robots es el objetivo principal que persigue esta tesis. En este capítulo se presenta el marco de referencia que engloba a este problema. En la sección 1.1 presentamos el contexto de la robótica, disciplina que estudia todo lo relacionado con los robots. Aunque todo el mundo tiene una primera intuición del significado de la palabra robot, profundizaremos en este término y aclararemos algunas ideas sobre qué son y qué no son los robots. Recorreremos su historia reciente desde los primeros artefactos mecánicos a las aplicaciones que existen en la actualidad. La sección 1.2 circunscribe el entorno general de la robótica a la parcela más específica de la robótica móvil. La aparición de *equipos* de robots no es casual y en la sección 1.3 pondremos de manifiesto la motivación y ventajas que supone que un grupo de robots colabore en una tarea común. El entorno de experimentación y las plataformas de pruebas serán presentados en la sección 1.4. Los objetivos concretos serán ordenados y descritos en la sección 1.5 y la propuesta concreta que da respuesta a los desafíos planteados será esbozada en la sección 1.7.

1.1. Robótica

La robótica persigue el diseño y construcción de sistemas que de manera automática sean capaces de realizar alguna tarea útil. Estos sistemas o automatismos que desarrolla la robótica reciben el nombre de robots. Veamos una primera definición formal de los términos *robótica* y *robot* según la Real Academia de la Lengua Española [RAE02]. La robótica es la *técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales*. Asimismo, define el término robot como *máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas sólo a las personas*.

Según la RAE, la robótica aplica la informática al diseño de los robots. Sin embargo, esta definición es algo incompleta pues la robótica es una rama ingenieril que bebe de diversas especialidades: Mecánica, electrónica, informática, inteligencia artificial e ingeniería de control, entre otras.

Vamos a ampliar el concepto de robot haciendo un símil con el ser humano. Los humanos disponemos de un esqueleto que nos dota de una estructura sólida. Además tenemos un complejo sistema muscular que nos permite movernos e interactuar con el mundo que nos rodea: gatear, andar, correr, saltar, manipular objetos, comunicarnos con otras personas, etc. Nuestros sentidos nos avisan de los peligros que nos rodean y nos permiten percibir el mundo. Disponemos de un complicado sistema digestivo y circulatorio que transforman los alimentos que ingerimos en energía y la reparten por todo el cuerpo. Finalmente todos nosotros tenemos un cerebro y un sistema nervioso, que coordinan todos nuestros sentidos y envían a nuestros miembros motores las órdenes para movernos, y nos dotan de la capacidad de pensar sobre las decisiones que tomamos en nuestra vida.

Los robots son una copia artificial de los seres humanos y, aunque no es fácil encontrar una única definición en la comunidad científica del término robot, sí que hay consenso sobre algunos puntos que todo robot debe tener. El primer elemento es el cuerpo, chasis o estructura. Un robot interactúa con el entorno. Su morfología puede tener forma de brazo articulado, forma de humano, forma de animal, etc. pero tiene que ser una máquina que integre todos sus elementos mecánicos. Los elementos

que permiten la interacción con el entorno se denominan actuadores. Algunos ejemplos de actuadores son los motores de las ruedas o patas para desplazar el robot por superficies lisas o terrenos abiertos, los motores equipados en manos mecánicas para recoger objetos, altavoces para comunicarse con personas, etc.

El siguiente componente que todo robot debe tener es un sistema sensorial. Los sensores son dispositivos que miden alguna magnitud física como presión, temperatura, etc. y la transmiten adecuadamente. Los robots necesitan estos mecanismos para conocer su entorno, percibir lo que les rodea y decidir en consecuencia. Los sensores más habituales son las cámaras, sensores de ultrasonido, sensores de luz, sensores de infrarrojos, sensores de presión, brújulas electrónicas, GPS, odómetros, etc.

Los sensores por sí mismos, raramente pueden activar directamente actuadores. Es necesario que haya una etapa de procesado y análisis de la información sensorial para extraer del flujo de datos la parte de interés. El procesador es el cerebro del robot y se encarga de esta labor. Los robots son sistemas programables y en el programa contenido en la memoria del robot están incluidas todas las decisiones que se deberán tomar en función de los datos sensoriales de entrada. La salida generada por el procesador será una serie de órdenes para sus propios actuadores. En ocasiones es habitual escuchar o leer la expresión *inteligencia* asociada a los robots. Es importante resaltar que, como cualquier computador, un robot por sí mismo no es inteligente. Es más apropiado utilizar la expresión *comportamiento inteligente* cuando nos referimos a la conducta que está desplegando un robot. El objetivo final de todo robot es que su trabajo sea útil, que aspire el polvo del suelo de nuestra casa, que pinte las carrocerías en las fábricas de automóviles o que desactive explosivos. Si su labor es exitosa a los ojos de un observador externo su comportamiento será inteligente. El éxito o fracaso del robot vendrá dado en gran medida por la calidad de su *software*. El programa debe contemplar todos los casos posibles, para que el robot se comporte bien ante cualquier eventualidad.

A día de hoy no existen robots de propósito general, como el caso de los populares y ficticios C3PO y R2D2 de la saga cinematográfica *Star Wars*. Desde sus orígenes los robots fueron concebidos para extender las capacidades del hombre, llegar hasta donde nuestros sentidos no llegan, trabajar hasta límites donde nuestra fatiga no alcanza a superar o mejorar la productividad. Desde la Revolución Industrial se intentó

la construcción de automatismos que ayudasen o reemplazaran al hombre. Entre ellos destacaron los Jaquemarts, muñecos de dos o más posiciones que golpean campanas accionados por mecanismos de relojería china y japonesa.

Esta idea de autómata creado para ayudar al hombre fue la semilla del término robot. Karel Kapek dio lugar a esta palabra en su obra *Rossum's Universal Robots* escrita en 1917. En ella, el científico Rossum desarrolla una sustancia que le permite a él y su hijo fabricar robots para servir a la clase humana. El término final robot apareció con el tiempo al derivar al inglés la palabra checa *robota*, que significa servidumbre. El escritor Isaac Asimov, también contribuyó con sus novelas de ciencia ficción a acuñar el término robótica desde 1939.

A pesar de que la mágica atmósfera de la ciencia ficción siempre ha rodeado al mundo de la robótica, ésta ha servido también para alejar de la realidad la verdadera imagen de los robots. El desarrollo de la automatización en fábricas ha llevado a emplear brazos manipuladores robotizados en cadenas de montaje. Es habitual ver este tipo de robots en varias fases del ensamblaje y pintado de automóviles como puede observarse en la figura 1.1. El periodo actual de amortización de estas máquinas ha hecho que paulatinamente haya descendido el número de personas que se encarga de estas tareas en las factorías.

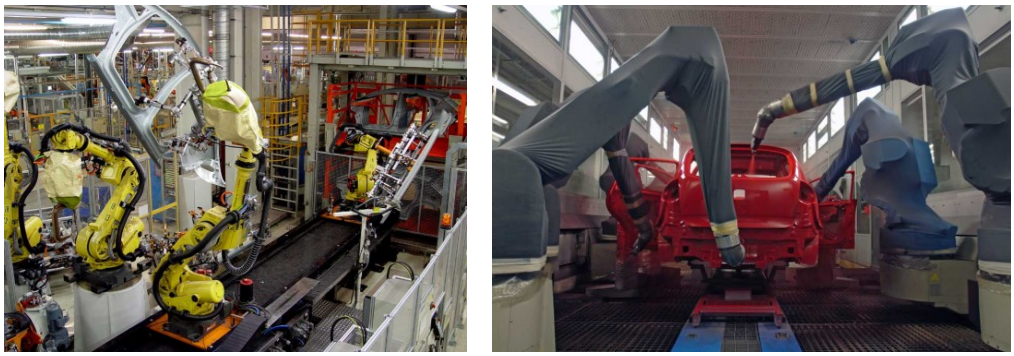


Figura 1.1: Robots de la fábrica de automóviles SEAT en Martorell (España) ensamblando piezas (izquierda) y realizando labores de pintado (derecha)

El transporte de mercancías es otro de los nichos de mercado que están encontrando los fabricantes de robots. Primero fueron los brazos articulados estáticos que

desplazaban la carga de un lugar a otro (figura 1.2 izquierda). Sus mayores limitaciones eran su elevado precio y su reducida capacidad móvil que impedían desplazar grandes distancias la carga. A menudo se utilizaban cintas transportadoras para paliar las limitaciones de los brazos articulados. Actualmente los centros logísticos más punteros están empezando a emplear pequeños robots móviles que transportan de manera automática la mercancía hasta diferentes puntos del almacén (imagen 1.2 derecha). Son incluso capaces de recorrer diferentes plantas y coordinarse entre sí.

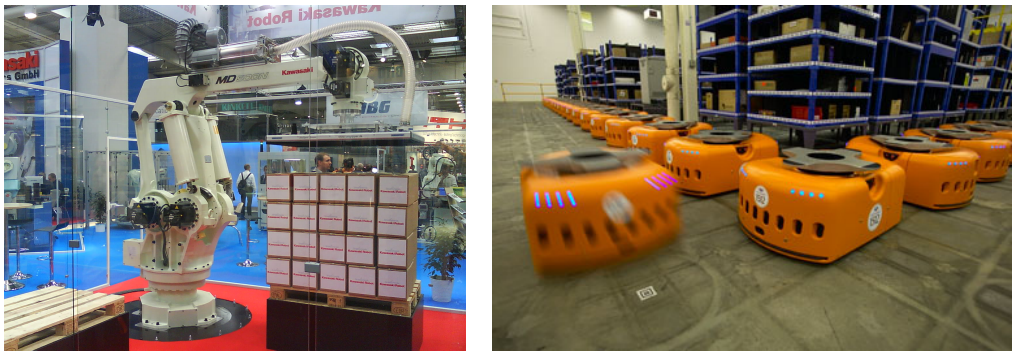


Figura 1.2: Robot Kawasaki para transporte pesado de mercancía (izquierda) y robots móviles de la empresa Kiva Systems para transporte ligero (derecha)

Los servicios sanitarios también ha encontrado en la robótica un camino por donde avanzar en su lucha contra las enfermedades. No tanto como mecanismo nuevo de curación, sino como medio para reducir el impacto de las operaciones se han desarrollado robots que asisten a los cirujanos y permiten operar con una precisión mayor que la lograda en una intervención clásica. El robot Da Vinci es el máximo exponente de este tipo de robots, reemplazando la cirugía abierta por una cirugía avanzada laparoscópica mínimamente invasiva. Una de sus aplicaciones estrella es la prostatectomía donde el paciente obtiene los mismos beneficios clínicos y oncológicos que con la técnica clásica, pero optimizados en confort para los cirujanos y precisión en las cirugías para pacientes con incisiones mínimas, menos anestesia, menor pérdida de sangre y hospitalizaciones más cortas, lo que se traduce en mejor calidad de vida.

La robótica no es territorio exclusivo del sector industrial. El desarrollo tecnológico y el abaratamiento de costes ha hecho posible que podamos adquirir robots para uso personal. El ejemplo más representativo es el de la empresa *iRobot* creada

en 1990 por Colin Angle, Helen Greiner y Rodney Brooks. Desde septiembre desde 2002 hasta la actualidad (2009) han vendido según su propia información más de tres millones de aspiradoras Roomba®, siendo el robot de consumo más vendido del mundo. Su aspiradora es capaz de recorrer por completo la superficie de una o varias habitaciones, evitando escaleras y obstáculos y garantizando que todas las zonas han sido aspiradas. Además de este producto, la empresa también comercializa robots limpiadores de piscinas, limpiadores de canaletas y variantes de Roomba®para fregar y encerar suelos.

Figura 1.3: Robots destinados al servicio particular: Aspiradora Roomba®(izquierda) y limpiador de piscinas Verro (derecha)

El consorcio formado por Universität Karlsruhe, Research Center Computer Science (FZI), Research Center Karlsruhe y Fraunhofer Institute Karlsruhe (IITB) ha desarrollado el robot humanoide ARMAR, como una primera aproximación a integrar robots dentro de nuestras casas. Por tanto, deben ser capaces de desenvolverse y desempeñar su labor en un entorno creado para que habiten personas y no robots y para que sean capaces de comunicarse con seres humanos.

Esta interacción persona-robot es un camino que la robótica tiene aún que atravesar. No sólo basta con que los robots dispongan de potentes programa de reconocimiento del habla e inteligencia artificial para comunicarse con los seres humanos. Si verdaderamente queremos tener robots de servicios, éstos deben ser capaces de simular las sensaciones humanas, tener capacidad gestual y oral para comunicarse e incluso tener un aspecto amigable para que cualquier persona no familiarizada con la robótica se sienta cómoda. Rhino y Minerva son dos de los primeros robots guía

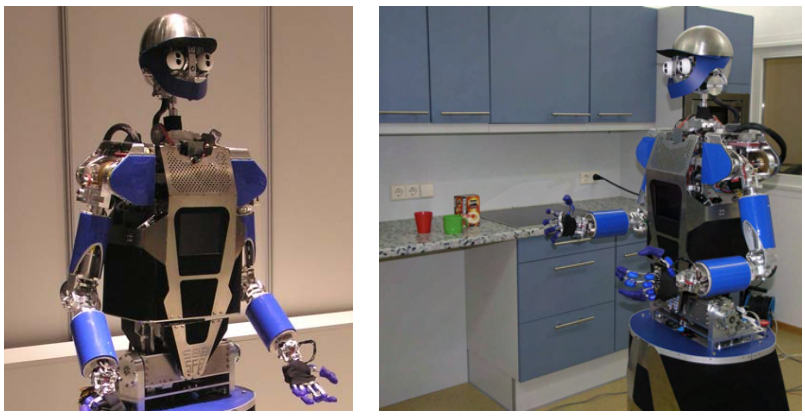


Figura 1.4: Tercera generación del robot ARMAR destinado a convivir en viviendas

que permitían interactuar con los visitantes de sus respectivos museos para ir a las diferentes estancias. Minerva era capaz de emitir sonidos, hablar, controlar su cabeza e incluso disponía de un interfaz web para que a través de Internet se pudiera conocer el estado del robot. Sin embargo, su apariencia distaba mucho de lo que se podría considerar como robot expresivo o amigable.

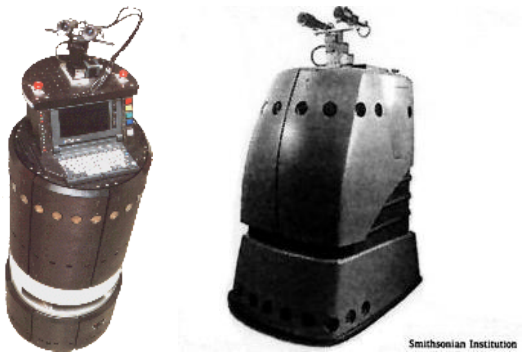


Figura 1.5: Robot guía Rhino (izquierda) del Deutsches Museum Bonn y robot guía Minerva (derecha) del Smithsonian's National Museum of American History

La profesora Cynthia Breazeal lidera el grupo de robótica personal del Instituto Tecnológico de Massachusetts (MIT). El robot Kismet fue desarrollado durante su tesis doctoral a finales de los años noventa. Es considerado uno de los primeros esfuerzos para lograr una interacción más natural y gestual para comunicación entre

robots y personas. El proyecto Nexi es la apuesta más actual del MIT para avanzar en esta campo. El nivel de integración alcanzado y su capacidad gestual es ya muy avanzado y el robot simula las expresiones y tamaño de un niño de tres años.

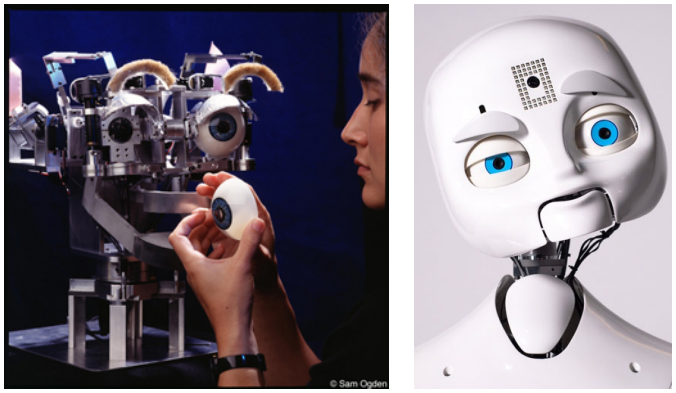


Figura 1.6: Robot Kismet (izquierda) y robot Nexi (derecha) propiedad del MIT destinados a fomentar la investigación en interacción social

Otros entornos donde tiene cabida el empleo de robots son las aplicaciones que entrañan peligro para el ser humano. Ante zonas afectadas por catástrofes naturales o actuaciones terroristas o militares es posible encontrar robots, en muchas ocasiones teleoperados, encomendados a labores de búsqueda de víctimas, desactivación de explosivos o exploración del entorno. Un ejemplo del empleo de estos robots lo podemos encontrar en el ejército de Estados Unidos, que ha desplegado una gran cantidad de robots PackBot (figura 1.7 izquierda) y MARCbot (figura 1.7 derecha) en los conflictos de Irak y Afganistán. Su misión principal es la detección de explosivos y la inspección de objetos sospechosos.

La acción de explorar y reconocer nuestro entorno es algo que siempre ha atraído a la raza humana. No en vano en el año 2009 celebramos el cuarenta aniversario de la primera exploración lunar humana. ¿Por qué deberían los robots sustituir a los astronautas? Los robots son capaces de aguantar impasibles ante situaciones inhóspitas. Venus cuenta con temperaturas de más de 482 ^{o}C. , alta presión y lluvias ácidas. Desde luego no es un sitio donde las personas nos sintamos cómodas. Los robots no tienen las necesidades de oxígeno, agua y alimentos que los humanos requerimos, ni la necesidad vital de regresar a la tierra. La radiación espacial, así como la gravedad

Figura 1.7: Robot PackBot (izquierda) y robot MARCbot (derecha) utilizados en conflictos armados reales

reducida puede llegar a causar problemas médicos a las personas. Los robots Sojourner, Spirit y Opportunity representan las apuestas de la NASA por la exploración de Marte. Sus fotografías y análisis están siendo claves para conocer la historia del planeta rojo. La exploración espacial será sin duda, en un futuro no muy lejano, territorio de vehículos robotizados que alejen a las personas de los riesgos extremos de estos viajes extraterrestres.

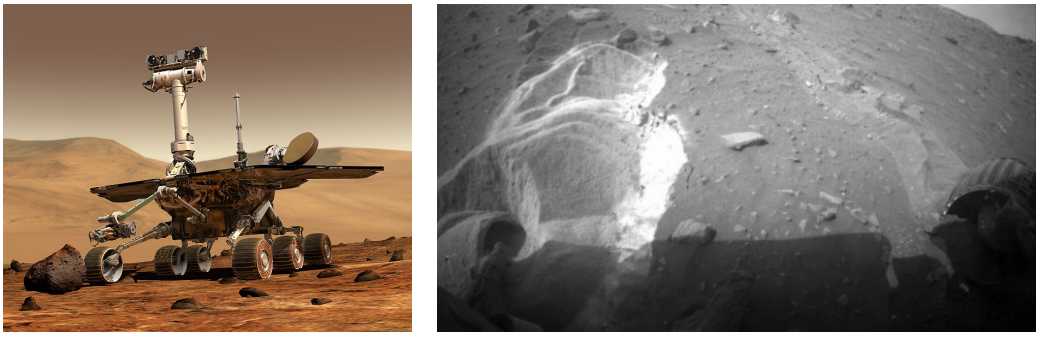


Figura 1.8: Robot Spirit (izquierda) e imagen real tomada en una situación complicada de atasco en la arena de Marte (derecha)

Un avance imparable en la tecnología ha hecho que el ocio de las personas viva un momento de revolución. Los clásicos salones de juego de los años noventa con las primeras sagas de máquinas recreativas han cedido el paso a los simuladores de jue-

go 3D montados sobre plataformas móviles. La industria del videojuego ha superado en facturación al mercado cinematográfico. Las empresas Space Adventures y Virgin Galactic ofertan incluso viajes espaciales al público de a pie. El progreso avanza de manera imparable y sin duda, la robótica también está comenzando a inundar el mundo del ocio. En el año 1999 Sony presentó su robot aiBo. Esta mascota virtual con forma de perro vendió mas de 150.000 unidades hasta que en 2006 se anunciara su retirada del mercado. La empresa WowWee's creó en el año 2000 el exitoso Robosapiens. Su diseñador, Mark Tilden ideó un humanoide manejado por control remoto capaz de ejecutar secuencias de movimientos. El robot es capaz de caminar, tiene cierto grado de programación básico y capacidad para manipular objetos con sus manos con relativa fuerza. La empresa Hitec vende como juguete educativo su robot humanoide Robonova. Es de pequeño tamaño pero cuenta con 16 grados de libertad, posibilidad de acoplar sensores, software de programación, etc. La marca Lego, conocida por su larga tradición de juguetes de construcción modular también dispone de una división dedicada a acercar la robótica al público en general. Su sistema *Robotic Invention System* goza ya de varias generaciones y en la actualidad su última versión es la del kit Lego Mindstorms NXT 2.0 lanzada a principios de 2009. Se compone de una unidad de procesamiento completamente programable que dispone de varios puertos para conectar sensores y motores. Entre sus usuarios podemos encontrar desde público infantil o adulto, hasta grupos de estudiantes que aprovechan sus cualidades para hacer prácticas académicas sobre temas de robótica.

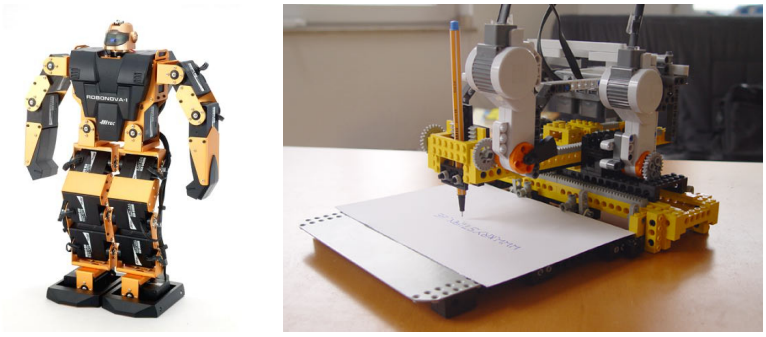


Figura 1.9: Robot humanoide Robonova (izquierda) y construcción de un plotter usando el kit Lego NXT por el estudiante Krystian Majewski (derecha)

La comunidad universitaria y científica alrededor de la robótica es otra gran *consumidora* de robots. ¿Cómo llegar de un sitio a otro de mi entorno?¿Cómo construir un mapa alrededor de mi posición?¿En qué sitio me encuentro de mi entorno?¿Cómo percibir de manera apropiada la realidad que me rodea?¿Cómo sacar partido a un equipo de robots para realizar una tarea común? Estas son algunas de las preguntas clásicas de la robótica y que la comunidad científica ha tratado de dar respuesta a lo largo de los años de vida de esta disciplina. Para validar todas las propuestas y soluciones es necesario contar con robots reales con los que realizar los experimentos. El robot Pioneer de la empresa ActivMedia ha sido un referente durante muchos años en lo que se refiere a robots con ruedas. Básicamente consiste en una plataforma que contiene sensores de ultrasonido y odométricos, sobre la que se puede conectar un ordenador portátil. En este ordenador se tiene total libertad para programar los comportamientos que procesan la información sensorial y comandan movimientos a los motores de la plataforma para dirigir el robot. Es habitual colocar otros sensores externos para ampliar las capacidades perceptivas del robot (sensores láser, cámaras, etc.). Más recientemente la empresa francesa Aldebaran Robotics comercializa su robot humanoide Nao desde 2008. Su principal atractivo es su precio (5000€-10.000€) y su buena terminación y características técnicas que escapan de los desorbitados precios que existían hasta el momento para adquirir este tipo de robots.

Figura 1.10: Robot Pioneer equipado con un sensor láser y una cámara motorizada (izquierda) y robot Nao promocionando la competición RoboCup 2009 (derecha)

Como hemos descrito en esta sección la robótica ha evolucionado a lo largo de la historia pero siempre con un denominador común: ayudar al hombre en alguna tarea. A medida que transcurren los años podemos vislumbrar que los robots más flexibles y productivos son aquellos que gozan de una mayor autonomía. Si el desarrollo avanza lo suficiente y no necesitamos un operador para controlar un robot, su tarea podrá hacerse 24h. al día y será realizada siempre de la misma manera evitando posibles fallos debidos al cansancio o errores humanos. No olvidemos que nosotros sí somos inteligentes y los robots no, ahí radica el reto, en lograr dotar a los robots de suficiente grado de inteligencia a sus comportamientos para que logran mayor autonomía en tareas donde todavía no la tienen.

1.2. Robótica Móvil

En esta sección vamos a acercarnos al entorno específico en el que se sitúa este trabajo: robótica móvil. El discurso se concentrará en la historia, características concretas, clasificaciones posibles e influencias de esta rama de la robótica que está destinada a dar un salto de calidad en los próximos años.

En la robótica industrial, dominada por brazos manipuladores, los robots están anclados en sus bases y el entorno de trabajo está completamente controlado y libre de obstáculos. Las posibles interacciones con objetos o personas del exterior están muy acotadas. Los robots móviles son aquellos que tienen capacidad para desplazarse por el entorno, ya sea en interiores (pasillos, casas, naves industriales, etc) o en exteriores (carreteras, caminos, etc.), agua, tierra o aire. Las capacidades motoras de los robots móviles se obtienen gracias a una base con ruedas, patas articuladas o hélices. Si además no necesitan de la supervisión de un operador para comandar movimientos estaremos ante un robot móvil autónomo.

La especialidad de la robótica móvil se diferencia de la robótica clásica de manipuladores, la visión por computador o la inteligencia artificial, en que el énfasis se centra en el estudio de los problemas y sus soluciones en relación con el entorno que rodea al robot. Hay que dar respuesta a retos como desplazarse a través del espacio, percibir el entorno y razonar acerca de éste para interactuar con él. Ahora las posibilidades de actuación se multiplican y los robots tienen que ser capaces de desenvolverse

con éxito ante una casuística de situaciones mucho mayor, sorpresas, etc. La interacción con el entorno es mucho más compleja, pues la heterogeneidad de los objetos y sucesos que pueden aparecer y su total desconocimiento resultan un reto mucho más difícil de resolver.

ELSIE (Electro-Light-Sensitive Internal-External) se considera el primer robot móvil de la historia. Fue construido por Grey Walter en Inglaterra alrededor de 1949 y su comportamiento era muy limitado para lo que estamos acostumbrados hoy en día. ELSIE, un robot con forma de tortuga, se limitaba a seguir un rastro de luz utilizando un sistema mecánico realimentado sin incorporar ningún tipo de inteligencia adicional. A pesar de sus restricciones, era capaz de percibir la luz recibida y actuar en consecuencia moviéndose hacia ella: un robot móvil sencillo pero innovador porque su movimiento no estaba preprogramado, sino que atendía a las variaciones de luz de manera similar a como reacciona el mecanismo de reflejo humano, es decir, una acción (luz) implicaba una reacción (movimiento).

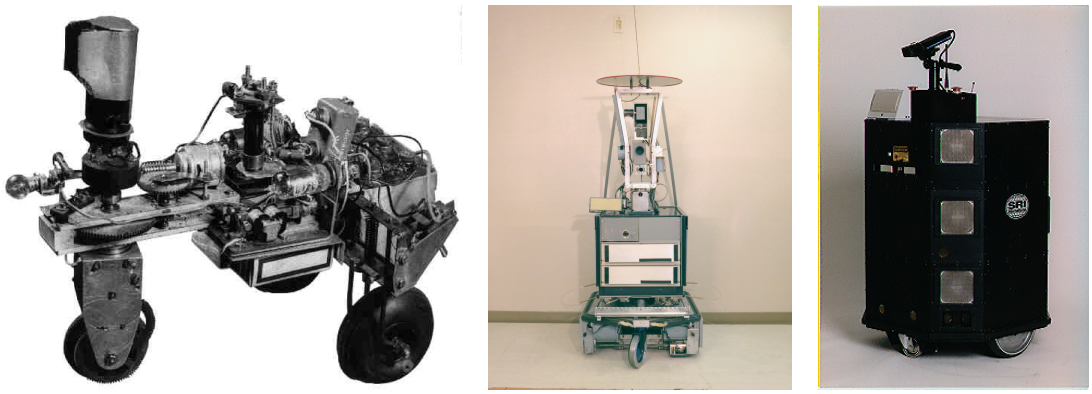


Figura 1.11: Imagen original del robot Elsie sin su caparazón protector (izquierda) y robots Shakey (centro) y Flakey (derecha) diseñados en el SRI

Desde 1966 a 1972, el equipo liderado por el Dr. Charles Rosen del Standford Research Institute (SRI) desarrolló el robot Shakey. Su sistema sensorial constaba de una cámara, sensores de ultrasonido, de contacto y odométricos. Además podía desplazarse por el suelo gracias a unas ruedas con las que contaba su base motora. El procesamiento se llevaba a cabo empleando dos ordenadores conectados por radio, uno embarcado para controlar los motores y el otro fuera del robot para el proce-

samiento de imágenes. Shakey era capaz de razonar sobre su entorno para localizar bloques, dirigirse hacia ellos y empujarlos suavemente. Aunque la apariencia de Shakey es la de una caja inestable con ruedas, representó un gran avance en la época en lo que a inteligencia artificial y capacidad de interactuar con el entorno se refiere. En 1985, el mismo centro diseñó y construyó al robot Flakey, el sucesor de Shakey. La mayor mayor innovación recaía en que ahora su autonomía era completa y todo el procesamiento se realizaba a bordo del robot.

En 1993, el Robotics Institute de Carnegie Mellon University creó el robot Dante con la misión de explorar el interior del volcán Erevus situado en la Antartida. Debido a la especial orografía del terreno no era posible emplear ruedas, por lo que el sistema motriz de Dante constaba de ocho patas articuladas. Durante su misión de recogida de muestras el robot se rompió una pata y no se pudo recuperar. Los investigadores John Bares y William Whittaker no cejaron en su misión y construyeron un año más tarde la segunda generación Dante II mejorando su robustez mecánica. El volcán Spurr situado en Alaska fue el destino de Dante II y esta vez la misión fue todo un éxito: se consiguió transmitir vídeo, el robot consiguió navegar durante cinco días por terrenos escarpados, recogió muestras de gases y se pudo recuperar. Hay que resaltar que el robot era teleoperado pero sus capacidades de exploración supusieron un gran avance para la época.

Los avances en la robótica móvil durante el siglo XXI han sido espectaculares. Prueba de ello es el trabajo de la Universidad de Stanford y el vehículo/robot Stanley [TMD+06] ganadores del Grand Challenge 2005. Este automóvil robótico fue capaz de conducir a alta velocidad por el desierto sin ninguna intervención humana durante aproximadamente 100 Km., atravesando pasajes estrechos y negociando más de un centenar de curvas. Como es de esperar, su completo *software* de control incluye numerosas tecnologías punteras en cuanto a IA y robótica se refiere (aprendizaje, razonamiento probabilístico, fusión sensorial, navegación, etc.). Desde el año 2007, este reto se denomina Urban Challenge y se desarrolla en una ciudad simulada. Los vehículos deben recorrer los casi 100 Km. que conforman la ruta respetando las normas de circulación y ante situaciones de tráfico real con otros vehículos.

Los robots humanoides también están siendo objeto de investigaciones constantes en el ámbito de la robótica móvil. En concreto, la locomoción es un problema

abierto pero los avances se suceden año tras año. El robot Asimo de Honda ha demostrado que es capaz de caminar, subir escaleras e incluso correr. Toyota, por su parte, ha presentado en julio de 2009 su robot humanoide corriendo a 7 Km/h. En la imagen 1.2 (centro) podemos ver un instante de la carrera donde podemos apreciar como ninguno de sus pies apoya en el suelo. La empresa Boston Dynamics llamó la atención de la comunidad científica cuando presentó en sociedad su robot *Big Dog*. Este robot de transporte de carga con dos pares de patas enfrentadas y 110 Kg. de peso es capaz de remontar pendientes de 35 grados de inclinación, transportar 150 Kg. de carga y avanzar a 6.4 Km/h en terreno abierto. Su sistema de control avanzado le permite incluso caminar por hielo, recuperarse ante resbalones imprevistos, golpes externos, etc.

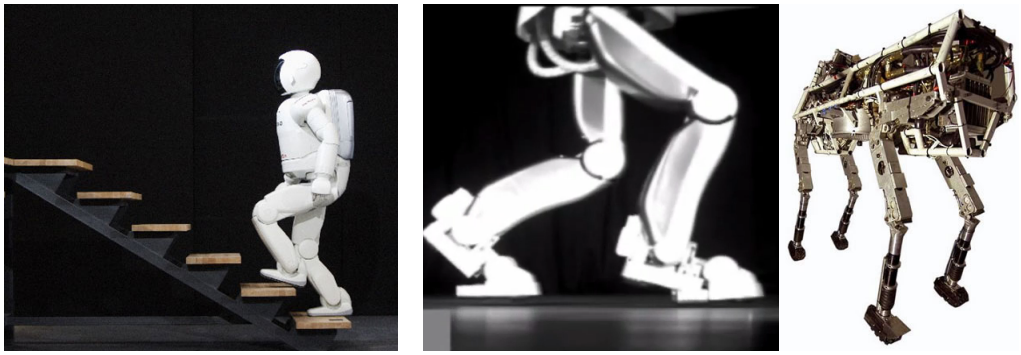


Figura 1.12: Robot Asimo subiendo escaleras (izquierda), captura de movimiento del robot de Toyota con los dos pies en el aire mientras corría (centro) y robot mula Big Dog (derecha)

Atentiendo a diversos criterios podríamos clasificar los robots móviles en varios subgrupos:

Según el medio donde se mueven: Con este criterio encontramos tres tipos principales de robots móviles. El primero de ellos lo forman los robots terrestres. Estos robots emplean ruedas, patas o cadenas similares a los tanques para desplazarse. La mayor velocidad y su sencillez de control son las principales ventajas de las ruedas o las cadenas, contra la mayor flexibilidad para recorrer superficies heterogéneas de las patas. Las tareas habituales de estos robots son la navegación y exploración tanto por interiores como por exteriores, trabajos

agrícolas, labores de entretenimiento o incluso recogida de muestras o limpieza.

Los robots aéreos o UAVs (Unmanned Aerial Vehicles) realizan habitualmente trabajos de vigilancia o escolta y reconocimiento. Su morfología habitual es la de aviones o helicópteros de radio-control adaptados para vuelo autónomo.

Los robots subacuáticos reciben el nombre de AUVs (Autonomous Underwater Vehicles). Emplean diversas hélices que, con el correcto sistema de navegación y control, permiten desplazarse al robot en cualquier plano debajo del agua. Sus tareas típicas son las de exploración del fondo marino, labores de rescate o incluso búsqueda de tesoros como los empleados por la polémica empresa Odyssey.

Según si el entorno es de interiores o exteriores: Los entornos de interiores o de oficinas son un ambiente muy popular para determinado tipo de robots. Su superficie lisa y controlada facilita la tarea de navegar y deambular por él. Estos entornos son muy adecuados para labores de investigación, ya que se pueden dedicar esfuerzos a otros problemas sin tener que resolver el reto de la navegación en exteriores, un verdadero desafío tecnológico. Los robots de exteriores son territorio de vehículos agrícolas, coches autónomos, robots con patas, UAVs y AUVs de tamaño medio/grande que tienen que hacer frente a corrientes de aire y de agua. También son el entorno ideal para robots domésticos, diseñados para desplazamientos de cortas distancias y tareas puntuales de vigilancia, limpieza o entretenimiento.

La magnitud del problema que aborda la robótica móvil es grande y comprende múltiples subproblemas relacionados con la cantidad de cuestiones a resolver. Es una área interdisciplinar, que comprende campos de la informática (algoritmos, sistemas de control, comunicaciones, programación, etc.), mecánica (diseño e integración hardware, locomoción, etc.) e incluso de la biología o psicología (inspiración animal para diseño hardware o incluso software, comportamientos de grupo, etc.).

Para hacer una recapitulación de esta sección me gustaría citar unas palabras de Hans Moravec al hilo de la robótica móvil escritas en 1999, que resumen lo complicado del problema que afronta la robótica: *Los primeros resultados fueron como una*

ducha de agua fría. Mientras los programas de razonamiento puro hicieron su trabajo tan bien como un ágil y joven colegial, los mejores programas de control robótico, aún siendo mucho más complicados de escribir, tardaron horas en encontrar unos cuantos bloques y colocarlos sobre un tablón. Con frecuencia fallaban estrepitosamente, haciéndolo mucho peor que un niño de seis meses.

Ha pasado una década desde que Moravec hizo esta frustrante declaración sobre el estado embrionario de la robótica. A día de hoy la robótica móvil ha progresado mucho como hemos visto en esta sección, se han desarrollado máquinas capaces de andar sobre hielo, recuperarse de caídas, correr, saltar, subir escaleras, volar, explorar el entorno submarino, conducir de manera autónoma, etc.

1.3. Robótica cooperativa

Como sucede en la sociedad humana, hay determinadas tareas en las que es mejor estar asociado en comunidad para llevar a cabo un trabajo. En ocasiones es incluso un requisito indispensable. Imaginemos que queremos realizar un mapa de la superficie lunar. Por muy buenas que fueran las características exploratorias del citado robot, por muy precisos que fueran sus sensores o por muy veloz que fueran sus motores, tardaría años en lograr tener un mapa completo de la superficie del satélite. Si en lugar de un robot pudiéramos enviar miles o millones, y éstos coordinaran su trabajo se podría lograr el trabajo en un tiempo razonable. Esta sección introduce los sistemas multi-robot, que son el marco central de esta tesis.

A medida que el progreso avanza de manera imparable en comportamientos mono-robot, es lógico que comiencen a coexistir vehículos autónomos por las carreteras, equipos de robots de transporte de mercancías en centros de distribución, cadenas de brazos manipuladores en fábricas o incluso varios robots en pequeños comercios como las farmacias. Esto requiere que los investigadores desarrollen algoritmos para coordinar todos estos comportamientos individuales.

El uso de varios robots para lograr un objetivo común viene motivado por varios factores. La *rapidez* es uno de ellos. A mayor cantidad de individuos organizados para realizar la misma tarea, el ahorro en tiempo será mayor y el objetivo se alcanzará con mayor velocidad. Cuando la tarea a realizar es fácilmente divisible en subtareas

es más fácil la asignación de subobjetivos a los miembros del equipo. Otra ventaja adicional es la mayor *tolerancia a fallos*. Si alguno de los robots del equipo sufre algún problema inesperado, el resto de compañeros pueden cubrir su puesto reorganizándose entre sí y consiguiendo mayor robustez general. La *mejora de las prestaciones individuales* es otro factor positivo. Las capacidades de percepción de cada robot pueden verse ampliadas al compartir información. De esta manera un robot podría percibir más allá de lo que captan sus sensores. Otra ventaja del uso de sistemas multi-robot coordinados es el *paralelismo*, es decir, se podrían llevar a cabo diversas acciones de manera simultánea. El *coste económico* también puede beneficiarse del uso de equipos de robots. La compra de un *super-robot* encarecerá el coste total más que la suma de los precios individuales de varios robots convencionales.

La otra cara de la moneda son los efectos laterales o adversos que también pueden producirse. El problema de las interferencias es uno de los más comunes. Habitualmente los miembros del equipo comparten el medio y se pueden producir colisiones o demoras para evitar las mismas, empeorando el rendimiento. Si los robots se coordinan utilizando comunicaciones existe un coste añadido en tiempo de proceso y en gasto energético. Otro de los requisitos para coordinarse adecuadamente es conocer en cierta medida las intenciones del resto de individuos. Los instantes en que haya demasiada incertidumbre en este aspecto podrían provocar colisiones ante la falta de entendimiento.

Aunque el concepto de coordinar comportamientos en agentes autónomos es relativamente nuevo, muchas de sus técnicas están influenciadas de los trabajos existentes en el campo de la DAI (*Distributed Artificial Intelligente*), disciplina que cuenta con una relativa madurez de dos décadas. En [BG88] se repasan los problemas clásicos abordados por la DAI.

A su vez, la naturaleza siempre es fuente de inspiración y la robótica cooperativa no es una excepción. En la naturaleza podemos encontrar multitud de ejemplos de grupos de individuos coordinados en mayor o menor medida. El deseo de conocer y analizar el funcionamiento de la naturaleza, junto con la necesidad de diseñar y construir robots o sistemas artificiales, tienen puntos en común. Los trabajos de Ronald Arkin [Ark98] siguen esta línea y tratan de diseñar sistemas complejos inspirados en el mundo natural. El concepto de *Swarm robotics* ha surgido al hilo de las grandes

colonias de robots al igual que ocurren en la naturaleza con las bandadas de aves o bancos de peces.

En el mundo que nos rodea también podemos encontrar influencias sobre grupos de agentes. Lynne E. Parker utiliza un ejemplo muy visual en [Par08]. Las personas nos organizamos en puestos específicos para desempeñar trabajos en una empresa. Formando una sociedad cada trabajador se especializa en una parcela concreta. En el ámbito médico sucede lo mismo, podemos encontrar especialistas en neurología, estomatología, podología, etc. Existen multitud de ejemplos en nuestra vida donde aparecen equipos de individuos. Las personas hemos aprendido que un equipo coordinado es más eficiente a la hora de resolver problemas que un único individuo.

Marvin Minsky propuso en su libro *The society of Mind*[Min85] la idea de que la inteligencia se consigue como una sociedad de agentes que interactúan entre sí y el resultado es un comportamiento global. Sus ideas estaban fundamentadas bajo su opinión sobre el funcionamiento del cerebro y la inteligencia. La base de su argumentación es que la mente no es una única entidad monolítica, sino que está formada por la sociedad de agentes especializados en tareas específicas. Estos agentes se relacionan produciendo como resultado un comportamiento o un pensamiento inteligente.

Típicamente los mecanismos de cooperación se emplean en aplicaciones formadas por equipos de robots, denominados sistemas multi-robot. Un ejemplo muy conocido son las cadenas de brazos robots realizando labores de soldadura o pintura en las fábricas de automóviles. En nuestro caso, los robots además son móviles y completamente autónomos entre sí. Sin embargo, también es posible encontrar sistemas colaborativos dentro de un mismo robot (por ejemplo gestionando el control de los actuadores de un robot con patas) o en sistemas de percepción distribuida (por ejemplo coordinando varias cámaras diseminadas por el entorno para tareas de vigilancia).

Existen numerosas aplicaciones donde se extrae todo el potencial a mantener un equipo de robots. Las tareas de recolección donde se pretende recoger una serie de objetos, conservación de formaciones específicas en tareas de navegación, construcción de mapas, percepción distribuida, etc. son algunos ejemplos de tareas paradigmáticas en este campo. Estas aplicaciones se repasarán, junto con las técnicas distribuidas usadas en ellas, en el capítulo 2.

La comunidad de robótica distribuida goza de una buena actividad investigadora respaldada por eventos periódicos. En la tabla 1.1 mostramos las principales conferencias en este ámbito. También se editan números especiales en revistas que dedican su contenido a publicar artículos del campo como [AS08a, CM08, CHI+09], e incluso revistas específicas como [Dor08]. La red Europea de investigación en robótica EURON [Eur09] incluso organizó un curso de verano celebrado en Darmstadt (Alemania) en agosto del 2008 con el título *Monitoring and Coordination Across Networked Autonomous Entities*.

Nombre de la conferencia	**Fecha y lugar**
IFAC Workshop on Networked Robotics	Octubre 2009, Colorado, USA
Practical Applications of Agents and Multi-Agent Systems	Junio 2009, Salamanca, España
International Workshop on Robotic Wireless Sensor Networks	Junio 2009, California, USA
Network Robot Systems at ICRA	Mayo 2009, Kobe, Japón
Robot Communication and Coordination - *RoboComm*	Marzo 2009, Odense, Dinamarca
New Challenges for Cooperative Robotics	Octubre 2008, Lisboa, Portugal
Network Robot Systems	Septiembre 2008, Niza, Francia
Multisensor Fusion and Integration for Intelligent Systems - *MFI*	Agosto 2008, Seoul, Korea
Formal Models and Methods for Multi-Robot Systems - *AAMAS*	Mayo 2008, Estoril, Portugal
Cooperative Robotics in Europe	Abril 2008, Hannover, Alemania

Cuadro 1.1: Principales eventos relacionados con la robótica cooperativa en 2008/2009

Toda esta actividad pone de manifiesto que existe una corriente de trabajo continua alrededor de esta disciplina. Los problemas abiertos todavía son numerosos y giran alrededor de algunas preguntas: ¿Cuál es la mejor estrategia de posicionamiento de los miembros del equipo para explorar de manera óptima un entorno, para construir un mapa o para vigilar uno o varios objetos móviles? ¿Cuál es el mejor algoritmo de fusión sensorial ante entornos dinámicos, con observaciones imprecisas y a menudo erróneas? ¿Cómo es posible mejorar la calidad de los algoritmos de auto-localización

utilizando información procedente de otros robots? ¿De qué manera podemos sincronizar diferentes robots para manipular simultáneamente un objeto? En este trabajo trataremos de abordar alguna de estas preguntas aportando nuestra propia propuesta de solución.

1.4. Entorno de trabajo: RoboCup - Liga de plataforma estándar

Con el fin de fomentar el desarrollo de los robots cooperativos es habitual encontrar competiciones entre robots organizadas en todo el mundo. Estas competiciones sirven de excusa y motivación para poner en práctica las técnicas desarrolladas por los investigadores en todos los campos de la robótica. En muchos casos los avances en robótica cooperativa son el resultado de los trabajos en este área de universidades o grupos de investigación participantes. Entre las competiciones más prestigiosas podríamos citar las organizadas dentro de la RoboCup [KAK+95] o por la FIRA [oIRsAF09]. En estas competiciones dos equipos de robots autónomos compiten entre sí en un partido de fútbol. Es habitual que todos los equipos tengan cierto grado de cooperación entre sus miembros. Dentro de las aplicaciones más usuales de cooperación se encuentran la asignación de roles de juego entre todos los robots del equipo. La competición RoboCup tiene un papel importante en esta tesis debido a que es el marco de referencia principal donde se han centrado los experimentos de este trabajo.

La RoboCup es una competición internacional que pretende fomentar la investigación en campos como la inteligencia artificial, robótica y disciplinas relacionadas con la Ingeniería. Es una apuesta para promover los avances en los mencionados campos ofreciendo un problema conocido y popular, donde puedan examinarse y evaluarse multitud de tecnologías. RoboCup utiliza el fútbol como tema principal de investigación, esperando que las innovaciones y técnicas presentadas año tras año puedan revertir en la sociedad y en la industria. Su meta final es lograr que en el año 2050 un equipo de robots venza al equipo humano campeón del mundo en un partido.

Los requisitos para que un equipo pueda jugar un partido de fútbol comprenden varias tecnologías: Diseño de arquitecturas software, colaboración multi-agente,

estrategia, razonamiento en tiempo real, percepción, auto-localización, etc. En definitiva, es una tarea que involucra a dos equipos móviles de robots bajo un entorno altamente dinámico.

La competición se organiza en varias ligas según el tipo de robots, su tamaño y objetivo de la competición. En un primer nivel podríamos encontrar tres tipos de competiciones: Rescate, hogar (*at home*) y fútbol robótico. La liga de rescate simula un entorno asolado por una catástrofe donde incluyen víctimas simuladas. El objetivo es teleoperar un robot, para que únicamente empleando sus sensores, el operador sea capaz de explorar la mayor superficie en busca de víctimas. Es una liga donde prima el diseño mecánico y las capacidades de navegación y exploración. La competición *at home* persigue conseguir que los robots sean capaces de desenvolverse con autonomía en entornos cotidianos. Para ello, se simula el entorno de una casa y los participantes tienen que lograr puntuar en una serie de pruebas. Seguir a una persona alrededor de la casa, ser capaz de percibir y recoger un objeto concreto especificado por un juez o reconocer a determinadas personas en habitaciones diferentes son algunas pruebas a las que se enfrentan los robots participantes. Por último la liga de fútbol robótico es la que más éxito de participación ha tenido a lo largo de los años congregando a miles de investigadores. Esta liga se subdivide en varias categorías que se describen a continuación:

Simulación En esta liga la organización proporciona a los participantes un simulador común. Este simulador funciona como un servidor al que los clientes se pueden conectar para conocer el estado del juego y, enviar a su vez los comandos de cada equipo. Los partidos son de once contra once jugadores y la estrategia juega un papel primordial.

Liga de pequeño tamaño En esta liga los robots están limitados a un diámetro máximo de 18 cm. y 15 cm. de altura. Se desplazan por el campo de juego mediante ruedas omnidireccionales y los participantes son libres de construir sus robots según la normativa de la liga. La autonomía no es completa, sino que reciben los comandos de movimiento desde un ordenador al que están conectados mediante un enlace inalámbrico. Cada equipo es responsable de situar una o dos cámaras sobre el campo para extraer la posición de los jugadores y de

la pelota. Estos son los parámetros con los que el programa de control de cada equipo toma las decisiones y comanda los movimientos a cada uno de sus cinco jugadores.

Liga de tamaño medio Los robots que participan en esta liga tienen un tamaño de 50 cm. de diámetro y 80 cm. de altura, como máximo. Son completamente autónomos. Se desplazan mediante ruedas y usan como sensor principal una cámara omnidireccional. El campo de juego destaca por su gran tamaño (18 x 12 m.) y los equipos están formados por cinco jugadores, uno de ellos actuando de portero.

Liga de humanoides Esta liga busca utilizar robots similares en forma a los humanos, es decir, dos piernas, dos brazos y cabeza. La liga se divide en dos categorías dependiendo del tamaño (tamaño niño y tamaño adolescente) y el *hardware* es completamente abierto, es decir, cada equipo es responsable del diseño y construcción completo de sus robots. Locomoción dinámica, carrera y golpeo de la pelota mientras se mantiene la estabilidad son algunos de los retos de esta categoría.

Liga de plataforma estándar La principal característica de esta liga es que el hardware es común a todos los equipos. El robot NAO es el seleccionado para participar desde el año 2008. Debido a la unificación de la plataforma, el esfuerzo se centra en el software que gobierna el robot. Los equipos están formados por tres robots cada uno y uno de ellos puede jugar de portero si así lo cree oportuno. La pelota es de color naranja para facilitar su detección y las porterías son azul o amarilla según el lado del campo. Todos los robots juegan sobre una moqueta verde y las líneas de las áreas, bandas y centro del campo se delimitan con líneas de color blanco. El sensor principal de los robots de esta categoría siempre ha sido una cámara (o dos en el caso del robot Nao). La gran riqueza de información que proporciona una cámara se contrapone a la limitada capacidad de recursos que suele contener el hardware de esta categoría. Por tanto, la influencia de un buen diseño algorítmico y una cuidada programación son requisitos necesarios para abordar con éxito la participación en este exigente escenario.

> Debido a que el resultado de los partidos únicamente depende de las capacidades del software desarrolladas por cada equipo, es muy importante sacar el máximo rendimiento a todos los elementos disponibles. Uno de los ingredientes más interesantes y que ha demostrado ser muy beneficioso en el impacto global del juego es la capacidad de coordinación. Un equipo cooperativo puede mejorar sus cualidades perceptivas y de actuación, mejorando sustancialmente el rendimiento frente al enfoque no cooperativo.

Las ligas de fútbol robótico, independientemente de la categoría, tienen algunas características comunes. La primera de ellas es el gran dinamismo que obtienen los partidos. La pelota viaja con relativa facilidad de un lado a otro del campo y los robots deben afinar sus sistemas perceptivos y locomotores para no perderla de vista y alcanzarla en el menor tiempo posible. Además el entorno es competitivo, otro equipo exactamente igual que el nuestro pero con objetivos contrarios trata de ganarnos en un equipo de fútbol.

El trabajo fundamental de esta tesis persigue la programación de un jugador robótico cooperativo para participar en la liga de plataforma estándar. La RoboCup nos ofrece un marco de experimentación muy exigente y competitivo, donde poder medir las capacidades de nuestra apuesta de colaboración. Debido a la evolución de esta categoría a lo largo de los años, es posible que en esta tesis hagamos referencia a instantes o normas que no corresponden exactamente con la normativa actual. Un ejemplo lo encontramos en el cambio de plataforma que se experimentó durante el año 2008. El robot Nao sucedió al exitoso aiBo en la liga de plataforma estándar. Esa es la razón principal por la que en esta tesis se emplearán ambas plataformas.

1.4.1. Plataforma de experimentación: Robots aiBo y Nao

A continuación se describen las dos plataformas de ensayo empleadas en esta tesis: El robot aiBo de la empresa Sony y el robot Nao de la compañía Aldebaran Robotics. La elección de estos dos robots no es casual y está motivada por la conexión de este trabajo con la liga de plataforma estándar de la RoboCup, cuyos robots de referencia son los que se detallan a continuación.

El primero de los robots utilizados en los experimentos de esta tesis es el robot aiBo. Lo comercializó la empresa Sony desde 1999 hasta 2006 y durante sus ocho años de vida hubo tres generaciones diferentes. En nuestro caso el modelo empleado es el ERS-7M3, último modelo del fabricante japonés.

Este robot con morfología de perro y diseñado como robot de compañía utiliza un avanzado mecanismo de locomoción con cuatro patas y numerosos sensores. Su sensor principal es una cámara a color de 350.000 pixeles, varios sensores de infrarrojos situados en su pecho para detección de distancias, un acelerómetro, un micrófono, sensores de contacto en su lomo, cabeza y en la parte inferior de la pata. Dispone de 20 grados de libertad para mover sus diferentes actuadores (patas, orejas, boca, rabo y cabeza), altavoces, *leds* y una tarjeta de red inalámbrica 802.11b.

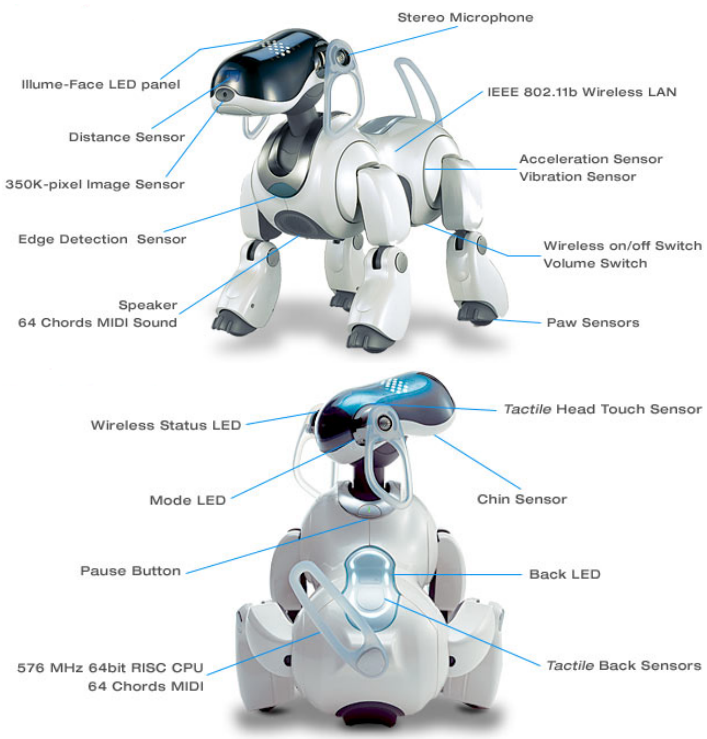


Figura 1.13: Sony AIBO ERS7 (Foto de Sony)

El cerebro del robot está gobernado por un procesador MIPS de 576 MHz. y 64 Mb. de memoria principal. El robot dispone de una tarjeta de memoria externa que hace las veces de disco o memoria secundaria y donde residen los programas que

ejecuta. El fabricante proporciona un entorno de programación denominado OPEN-R[1] detallado con profundidad en [MGCRJ04].

Sin duda era un plataforma comercial muy interesante pues disponía de un equilibrado conjunto de sensores y actuadores a un precio asequible (2000€). Si a ello le añadimos una excelente robustez mecánica, un buen servicio postventa y una gran comunidad de usuarios alrededor entenderemos por qué fue la seleccionada por la RoboCup para la liga de plataforma estándar.

El segundo robot empleado en los experimentos es el robot NAO. Su fabricante es la compañía francesa Aldebaran Robotics, que debido al cese en la producción del robot aiBo, supo coger el testigo y reemplazar con su robot humanoide a su homólogo japonés en la liga de plataforma estándar de la RoboCup desde el año 2008. Este año fue el año de lanzamiento del robot NAO y únicamente los participantes en la RoboCup tuvieron acceso al mismo.

NAO es un robot humanoide, es decir, su morfología se asemeja a la de una persona, aunque más bien podríamos decir a un niño. Su altura es de 58 cm. y su peso de 4,3 Kg. Cuenta con 23 grados de libertad en la versión RoboCup y 25 en la versión académica que permiten controlar sus piernas, brazos, cabeza, cadera y manos en caso de la versión académica. Su unidad de procesamiento cuenta con un microprocesador AMD Geode a 500 MHz. Además cuenta con dos altavoces para emitir sonidos o música y varios *leds* repartidos por su cuerpo. Su equipamiento sensorial también es amplio: Dos cámaras situadas en la cabeza, sensores de ultrasonido en el pecho, sensores inerciales en el torso, varios sensores de presión y contacto en la planta y puntera de cada pie y dos micrófonos. En cuanto al equipamiento de comunicaciones cuenta con una conexión *ethernet* y una tarjeta de red inalámbrica 802.11g.

Una de los aspectos más ventajosos de NAO es que utiliza GNU/Linux como sistema operativo, solución muy flexible y que permite utilizar una gran cantidad de software ya existente. Para el acceso a los sensores y actuadores la compañía francesa proporciona una capa de software denominada *NaoQi*. Además, es posible utilizar varios simuladores comerciales como Webots o Microsoft Robotics Studio.

El precio de este robot está alrededor de 10.000 €, que comparado con los desorbitados precios de cualquier otro fabricante de humanoides es sin duda una interesan-

[1] http://www.openr.org/

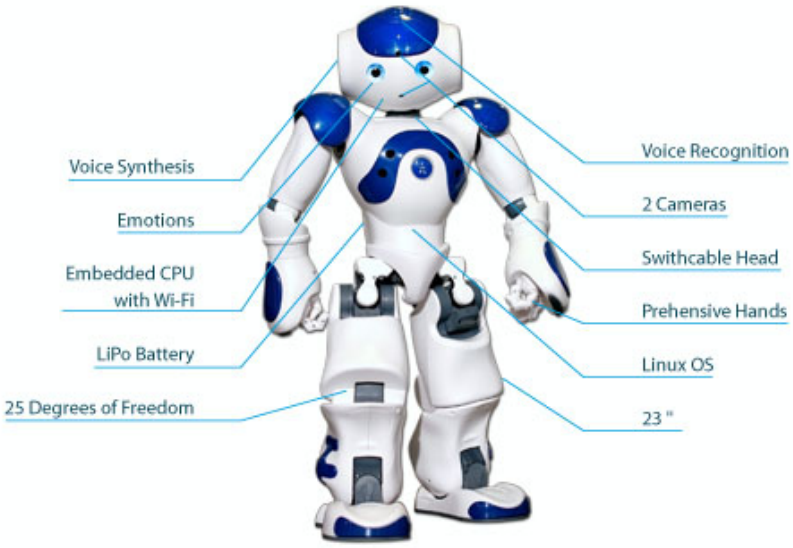


Figura 1.14: Robot NAO de la compañía Aldebaran Robotics

te opción. La robustez mecánica alcanzada después de varias versiones es aceptable aunque no llega a las cotas logradas por el robot aiBo. Sin embargo, la expansión de NAO está siendo alta, así como la actividad y soporte del fabricante, por lo que es de esperar que en pocos años dispondrán de una plataforma excelente para la robótica y la participación en la liga de plataforma estándar de la RoboCup.

1.5. Objetivos

En las secciones anteriores ya hemos fijado el contexto en el que se enmarca esta tesis. Dentro del campo de la robótica móvil nos vamos a concentrar en dos problemas concretos que atañen a los sistemas multi-robot. El primero de ellos es la estimación de un objeto con gran dinamismo (la pelota de juego), estimación que será realizada de manera coordinada y mediante técnicas probabilísticas por un equipo de robots. La segunda propuesta a abordar es la de establecer un mecanismo de cooperación basado en un intercambio dinámico de roles entre los individuos del equipo para aumentar el rendimiento conjunto. Para validar experimentalmente la solución propuesta nos vamos a ceñir al entorno propuesto por la RoboCup. Por lo tanto, el objetivo principal

de esta tesis es el análisis, diseño, implementación, experimentación y validación de estas dos técnicas de cooperación aplicadas al fútbol robótico dentro del marco de la RoboCup.

Al hilo de la estimación compartida de la pelota pueden surgir varias preguntas. ¿Por qué colaborar y no mantener una estimación sencillamente local? La primera respuesta a esta pregunta es que esta competición garantiza un dinamismo relativamente alto. Existen varios robots moviéndose, las condiciones de luz no están controladas, por lo que las percepciones se complican dependiendo del público que haya alrededor del campo, la pelota se mueve con alta velocidad, etc. A su vez, la locomoción de estos robots se basa en el movimiento coordinado de sus patas o piernas. Las oscilaciones de la cabeza son mayores comparadas con las que pueden tener los robots que se mueven usando ruedas. Además, el campo de juego es relativamente grande y es difícil percibir la pelota si está en la otra punta del campo. Estas características hacen que sea difícil mantener una percepción estable de la pelota en todo momento. Si utilizamos un enfoque cooperativo existe una ventaja fundamental: Si el robot no está viendo la pelota pero existe al menos otro compañero que si la está percibiendo, siempre tendrás una idea aproximada de la posición de la misma. La precisión y la estabilidad en la estimación son otras ventajas añadidas. Como veremos en la sección 3.3 las pruebas demuestran que el error medio cometido utilizando una estimación común es menor que el error medio obtenido únicamente con la estimación local. Intuitivamente podríamos adelantar que las percepciones están ponderadas según el grado de incertidumbre de cada estimación. Si uno de los robots considera que su estimación es muy buena tendrá más peso en la estimación compartida que otro robot que no crea tanto en su percepción.

La percepción distribuida de la pelota tiene que ayudar a facilitar el seguimiento de la misma por parte de los robots. La robustez tiene que ser uno de los aportes principales de esta estimación, asegurando que los robots dispondrán permanentemente de una estimación incluso ante oclusiones locales que impidan ver la pelota a algún robot. La precisión también será un parámetro a mejorar, pues buscamos una estimación más precisa que la que se puede obtener de manera individual. La incertidumbre asociada a las observaciones y los procesos de auto-localización debe ser tenida en

Figura 1.15: En este instante de la competición *Roboludens 2006*, la casualidad hizo que esta niña vistiera con los mismos colores que la baliza mostrada en el campo, dificultando la auto-localización de algunos robots

cuenta e integrada en el proceso de percepción distribuida, para garantizar que ante situaciones reales de juego el diseño realizado es válido.

El segundo pilar desarrollado para dotar de un alto nivel de cooperación a los jugadores robot de nuestro equipo de RoboCup ha sido la técnica de asignación dinámica de roles bautizada como Switch!. Esta capa de software mantiene un flujo constante de datos entre todos los jugadores indicando diferentes parámetros del juego (posición, distancia la pelota, incertidumbres, etc.). Estos parámetros son utilizados para calcular periódicamente el grado de adaptación a cada uno de los roles existentes en el equipo. Cada robot debe desarrollar el comportamiento más conveniente para el equipo según una serie de parámetros. Es deseable que ante un cambio en el juego, los jugadores sean capaces de reconfigurar sus roles para adaptarse a la nueva situación creada. Switch! hace todo esto y materializa diferentes roles de juego para los jugadores de campo de liga de plataforma estándar: *Defender, supporter o kicker*. Además se han utilizado diferentes heurísticas para calcular la posición óptima de ca-

da jugador según su rol y garantizar una formación adecuada a la situación del juego. Switch! no sólo debe garantizar el dinamismo y el cambio de roles cuando sea necesario, también debe protegerse de excesivos cambios que desestabilicen la formación del equipo. Todos los detalles de esta técnica de intercambio de roles serán ampliados en el capítulo 4.

Junto con el objetivo principal de esta tesis existen varios objetivos secundarios que se detallan a continuación:

1. Analizar las referencias bibliográficas más representativas de la comunidad robótica relacionadas con el campo de la fusión sensorial y los mecanismos de intercambio de información.

2. Desarrollar la arquitectura necesaria para poner en marcha las técnicas de cooperación propuestas en la tesis. Los comportamientos de coordinación podrían clasificarse como de alto nivel, pues es imprescindible disponer de otras conductas básicas (locomoción, percepción, comunicaciones, etc.) para poder abordar este problema y realizar experimentos en el robot real.

3. Validar los aportes de la tesis en la competición RoboCup, donde se establece un riguroso control sobre el entorno de experimentación. En particular habrá que determinar si son viables las técnicas de fusión multi-robot sensorial con un objeto tan rápido y cambiante como es la pelota.

La fase inicial de esta tesis persigue crear toda la infraestructura software necesaria para poder gobernar las dos plataformas. Para el caso del robot aiBo se ha empleado principalmente la infraestructura del equipo *TeamChaos*. Para el caso del robot Nao se ha creado completamente desde cero todo lo necesario para poder integrar los algoritmos de cooperación en el robot. Los anexos A y B describen las citadas arquitecturas.

1.6. Propuesta de jugador robótico cooperativo

En las secciones anteriores hemos presentado el marco donde se sitúa este trabajo y cuáles son los objetivos que se persiguen. En esta sección se va a esbozar la solución propuesta a los problemas planteados en esta tesis.

El desarrollo de un comportamiento cooperativo en un equipo de robots puede lograrse de varias maneras según los objetivos que se persigan. En el caso que nos ocupa hemos apostado por dos técnicas combinadas, una para estimar la posición de la pelota con mayor precisión, y la otra para conseguir una formación óptima según las condiciones dinámicas del entorno. Es interesante resaltar que el entorno de la RoboCup es el marco seleccionado para las pruebas, pero las técnicas aquí presentadas son aplicables a un abanico mucho más amplio de entornos y robots, pues la base matemática que soporta estas técnicas es robusta, contrastada y otorga solidez a las soluciones propuestas.

Los métodos aquí presentados son puramente probabilísticos, es decir, las técnicas empleadas hacen uso de artefactos matemáticos para modelar la incertidumbre de cada estimación y estimar no sólo la posición de la pelota en el campo, sino también el grado de incertidumbre que presenta la estimación. En el capítulo 3 se detallarán todos los aspectos de las diferentes técnicas planteadas.

La propuesta de algoritmo de percepción distribuida de la pelota no es única. Se han realizado varios diseños, cada uno de ellos con una teoría matemática subyacente diferente. La primera propuesta utiliza un método analítico basado en modelos de observación bidimensionales Gaussianos y ganancia de Kalman. La siguiente proposición aborda la discretización del entorno. Hemos dividido el campo de juego en celdas formando una rejilla de probabilidad. Aplicando reglas de combinación basadas en la regla de Bayes hemos mantenido una estimación distribuida y múltiples hipótesis de la pelota. Una tercera alternativa emplea filtros de Kalman para integrar las diferentes observaciones procedentes de cada robot. Finalmente hemos diseñado una cuarta solución para este problema utilizando varios filtros de partículas. Cada una de estas soluciones tiene sus problemas e inconvenientes. El análisis y comparativa extraído tras los experimentos de cada una de estas técnicas, supone un aporte de esta tesis para el estudio de las diferentes técnicas de fusión multi-robot.

El algoritmo de intercambio dinámico de roles *Switch!*, que presentaremos en el capítulo 4, emplea roles para organizar los diferentes comportamientos en cada robot. Cada rol tendrá asignado una estrategia de posicionamiento, cuyo objetivo es cubrir la mayor parte de juego posible. Además, es necesario tener un balance equilibrado entre formaciones defensivas y ofensivas. El posicionamiento propuesto combina ambos perfiles y trata de maximizar las recuperaciones de pelota en caso de disparos o pases. Es muy importante que el equipo sea capaz de auto-configurarse e intercambiar los roles si la situación de juego así lo requiere. Hemos adoptado los conceptos de utilidad, que presentaremos en el capítulo 2, y hemos creado nuestras propias funciones de cálculo de utilidad para cada rol, donde influye principalmente la posición de la pelota y la posición de los miembros del equipo. Algunas ideas planteadas por equipos participantes en la RoboCup han sido adoptadas y modificadas para incluirlas en nuestra propuesta, como la heurística para calcular cuál es el robot más adecuado para ir hacia la pelota o la idea de añadir bonificaciones o penalizaciones para favorecer o evitar el intercambio de roles en determinadas circunstancias.

1.7. Estructura de la tesis

En este capítulo se ha presentado el contexto general y particular en el que se desarrolla esta tesis. Hemos comenzado describiendo el marco general de la robótica, su historia, los componentes que conforman un robot, sus aplicaciones más habituales, etc. Dentro del amplio campo de la robótica, se ha dirigido el discurso de este capítulo hasta la disciplina de la robótica móvil. Se han detallado las características comunes en este tipo de máquinas, hemos realizado un repaso histórico a los robots más paradigmáticos y se han presentado diversas clasificaciones posibles atendiendo a varios criterios. La coordinación entre robots es el tema principal que aborda esta tesis y para ello es necesario disponer de un sistema multi-robot. Hemos descrito las razones que motivan el uso de este tipo de sistemas, sus cualidades y justificado el interés que han suscitado en la comunidad científica en los últimos años. Las posibilidades de cooperación dentro de un equipo de robots son muy amplias y en este trabajo nos hemos circunscrito a dos temas concretos: La estimación coordinada de objetos y el intercambio dinámico de roles. Una vez presentados los detalles de estos

temas se ha pasado a definir los objetivos concretos de esta tesis, así como la plataforma y el entorno de experimentación. Por último, se ha presentado la propuesta de jugador robótico cooperativo para la competición RoboCup, que hace uso de las técnicas mencionadas anteriormente.

El capítulo 2 muestra el resultado del estudio pormenorizado que se ha realizado como complemento al diseño e implementación de la solución propuesta. Ha sido necesaria una labor de estudio de la literatura relacionada con la robótica cooperativa y los sistemas multi-robot, para crear un sustrato teórico sobre el que fijar todo este trabajo. Para afrontar cada una de las propuestas que compone nuestro jugador colaborativo, también se ha realizado un repaso minucioso del estado del arte sobre fusión sensorial y reparto de tareas en sistemas multi-robot. Las conclusiones más importantes de cada estudio se plasman en sendas secciones finales de este segundo capítulo.

El capítulo 3 se dedica por completo a la estimación de objetos dinámicos. En una primera sección se dan a conocer los fundamentos teóricos de la fusión sensorial (filtros Bayesianos, filtro de Kalman, filtro de partículas). A continuación se desgranan todos los aspectos de la propuesta de esta tesis dentro del marco de la estimación coordinada de objetos. El completo análisis de los resultados obtenidos en la fase de experimentación es descrito en una sección propia dentro de este capítulo. Para finalizar se presentan las conclusiones de la solución aportada, tras el análisis de las pruebas.

El capítulo 4 mantiene la misma estructura que el anterior. Comienza analizando los fundamentos teóricos asociadas al reparto de tareas dentro de un equipo de robots. La siguiente sección detalla y analiza la proposición concreta (*Switch!*) que se ha realizado al hilo de este tema, y que fundamenta el segundo pilar de esta tesis para disponer de un sistema cooperativo dentro del entorno de la RoboCup. Además, se presentan las pruebas realizadas para validar el diseño, así como las conclusiones obtenidas de las aportaciones de esta fase del trabajo.

El capítulo 5 realiza un resumen de las contribuciones más significativas que ha abordado este trabajo. Adicionalmente elabora una lista de tareas que se consideran importantes y se tendrán en cuenta en líneas futuras.

Para concluir, se han añadido dos anexos, que no forman parte del núcleo teórico abordado pero son un paso imprescindible para la experimentación con las técnicas propuestas. Por un lado, se presenta la estructura del equipo *TeamChaos* utilizada para las pruebas en el robot cuadrúpedo aiBo; y por otro la arquitectura software *BICA (Behavior based Iterative Control Architecture*, para los experimentos en el robot humanoide Nao.

CAPÍTULO 2

Estado del arte

If knowledge can create problems, it is not through ignorance that we can solve them.

– Isaac Asimov

En este capítulo realizaremos un repaso bibliográfico a través de aplicaciones y de manera más específica en las dos líneas de trabajo de esta tesis en robótica cooperativa: Percepción coordinada y división de tareas. Comenzaremos con los trabajos sobre robótica cooperativa más representativos, para conocer el panorama actual que la comunidad robótica ha dispuesto en los últimos años. Se estudiarán los diferentes tipos de enfoque a la hora de utilizar robots cooperantes, ya sea mediante redes de sensores, conjuntos de actuadores o equipos de robots. Algunos de los conceptos de mayor importancia en esta disciplina serán también repasados, haciendo referencia a los autores o creadores originales de los términos o definiciones. A su vez estableceremos una breve taxonomía de los sistemas cooperativos según diferentes criterios expuestos por la literatura clásica y actual en este ámbito.

Otro de los puntos donde profundizaremos en este capítulo es realizar una revisión de las aplicaciones más representativas que sacan partido al uso de sistemas

multi-robot. Exploraremos algunos trabajos sobre recolección, control de formaciones, gestión de almacenamiento, manipulación coordinada, fútbol robótico, construcción distribuida de mapas, percepción distribuida y coordinación de actuadores.

Una vez expuesto el estado del arte general alrededor de la robótica cooperativa y los sitemas multi-robot nos acercaremos a detallar los trabajos más representativos en los ámbitos de percepción coordinada de objetos mediante fusión sensorial y reparto de tareas en equipos de robots. Estos son los dos temas centrales de esta tesis y, por tanto, reciben un tratamiento especial en este capítulo donde los trabajos más importantes son analizados y presentados, como paso previo a la descripción de los aportes fundamentales de esta tesis.

2.1. Robótica cooperativa

Una cuestión clave a la hora de diseñar un sistema cooperativo es conocer todas las posibles configuraciones del sistema, ventajas, inconvenientes, etc. De esta manera, el diseño será más apropiado a las necesidades requeridas. En esta sección exponemos los conceptos habituales en entornos cooperativos, estableciendo una breve taxonomía de estos sistemas.

Cuando hablamos de sistemas cooperativos podemos diferencias tres familias o niveles: sistemas multi-sensoriales que cooperan entre sí, sistemas con múltiples actuadores dentro de un mismo robot y sistemas formados por varios robots independientes.

El caso de varios sensores cooperantes incluye todas aquellas aplicaciones que involucran a más de un sensor y comparten información. Esta información es útil para modelar de manera más rica el entorno que quieren percibir.

Un ejemplo de estos sistemas son las *redes de sensores*. Es un concepto relativamente nuevo en adquisición y tratamiento de datos de manera coordinada. Estas redes están compuestas por un amplio número de sensores con capacidades comunicativas, que se interconectan formando redes ah-hoc. Disponen de una reducida capacidad de cálculo e incapacidad motora pero sus capacidades cooperativas permiten a estos sistemas ser una buena opción en determinados entornos: Por ejemplo zonas de difícil acceso (zonas asoladas por catástrofes, otros planetas, etc.) o con alto riesgo de

bloqueos (volcanes, desiertos, etc.). Berkeley Motes[HC01], Pico-Radio[RASP00] o Smart-Dust[KFP] son el resultado de varios trabajos en el ámbito de las redes de sensores.

Los sistemas cooperantes con múltiples actuadores incluyen el control de actuadores dentro de un mismo robot o la manipulación coordinada de objetos. La locomoción con patas requiere que haya una sincronización y planificación de todos los movimientos para que se produzca de manera suave y efectiva. En el caso de la manipulación coordinada sucede algo parecido, es vital que haya cooperación para evitar choques entre los manipuladores, interferencias, etc. Un ejemplo de estos últimos sistemas los podemos encontrar en las cadenas de montaje de automóviles responsabilizándose de tareas de soldadura o pintura.

La tercera familia de sistemas cooperantes está formada por varios robots independientes. Estos robots forman un equipo en caso de no ser excesivamente numeroso o un enjambre (*swarm robotics*) en caso de pertenecer a un conjunto muy numeroso. Las aplicaciones habituales de estos equipos son la recolección de objetos, exploración, navegación en formación, etc.

El término comportamiento colectivo describe cualquier comportamiento en el que intervienen múltiples robots. La definición de comportamiento cooperante es mucho más interesante y denota un comportamiento colectivo que incluye algún mecanismo de coordinación implícito o explícito entre los robots para organizar y ordenar la tarea común aumentando el rendimiento global.

Existe una nueva corriente dentro del mundo de la coordinación multi-robot llamada *swarm robotics*. Este término hace referencia a grandes grupos de robots, formados por cientos de miembros. En general, la coordinación entre los miembros de estas hordas o enjambres es muy baja. El resultado es lo que se conoce como comportamiento emergente, es decir, su comportamiento aparece como resultado de la ejecución sin apenas comunicación de todos los comportamientos individuales (sin que haya una programación explícita de esa tarea colectiva). El proyecto Centibots [KFO$^+$04], es un buen ejemplo donde 100 robots coordinados tratan de construir mapas y realizar tareas de vigilancia.

Dado un grupo de robots, un entorno y una tarea, ¿Cómo emerge un comportamiento cooperante que la realiza?¿Cómo deberían coordinarse los diferentes indi-

viduos para culminar su objetivo común?¿Qué tipos de robots pueden intervenir en los equipos?¿Qué tipo de información se puede intercambiar? Las respuestas no son triviales y hay que analizar los elementos que conforman este tipo de sistemas para poder contestar a estas preguntas. A continuación se presentan los ejes que soportan cualquier sistema multi-robot coordinado.

2.1.1. Control Centralizado Vs Distribuido

Gregory Dudek presenta en [DJW96] una taxonomía para clasificar los diferentes aspectos que forman parte de un sistema robótico multi-agente. En los sistemas centralizados uno de los robots actúa de líder y toma las decisiones (sistemas maestro-esclavo) o un ordenador externo se encarga de comandar las mismas. Habitualmente el líder o maestro también es el encargado de almacenar el modelo perceptivo del entorno, pues en función de él decidirá sobre el resto de robots a su servicio. En los sistemas descentralizados no existe este líder y, o bien todos los agentes tienen el mismo peso en las decisiones de control (distribuido), o bien existen niveles jerárquicos. En los sistemas distribuidos cada robot es encargado de modelar su entorno con sus capacidades perceptivas y decidir en consecuencia. Parece bastante aceptado que las arquitecturas descentralizadas tienen mayores ventajas [LL94], [TK93].

En la RoboCup podemos encontrar con frecuencia ejemplos de sistemas centralizados y distribuidos. En la categoría de robots de pequeño tamaño dos equipos de 5 miembros cada uno compiten entre sí en un entorno real simulando un partido de fútbol. La mayoría de los equipos emplean una o varias cámaras dispuestas en posición cenital a 4m. del suelo, desde donde capturan todos los movimientos de la pelota y los robots usando un ordenador central. La fase perceptiva y estratégica se realiza en este PC externo, donde además se deciden los comandos individuales a ejecutar por cada robot. Estos comandos son enviados periódicamente a traves de comunicación inalámbrica a cada uno de los miembros del equipo. Esta liga es un claro ejemplo de sistema centralizado.

En las categorías de robots de tamaño medio, humanoides o de plataforma estándar todos los robots son complemente autónomos (salvo determinadas situaciones que requieren detenciones del juego, penalizaciones, etc.). No hay un centro único

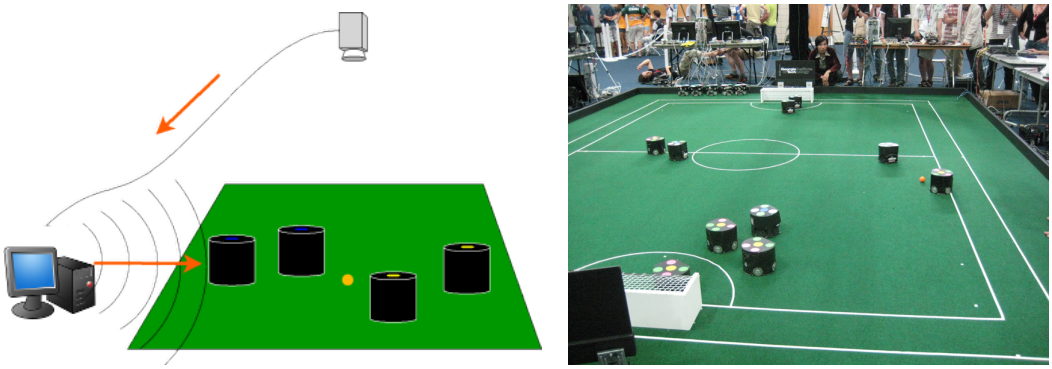


Figura 2.1: Esquema de intercambio de información en la liga de robots de pequeño tamaño (izquierda) e instante de una competición real (derecha)

de toma de decisiones. Los robots disponen de cámaras a bordo, toman sus propias decisiones de movimiento y realizan todo el procesamiento perceptivo, estratégico, etc. sin depender de un sistema externo o de un robot que actúe como líder. En este caso estos sistemas son totalmente distribuidos.

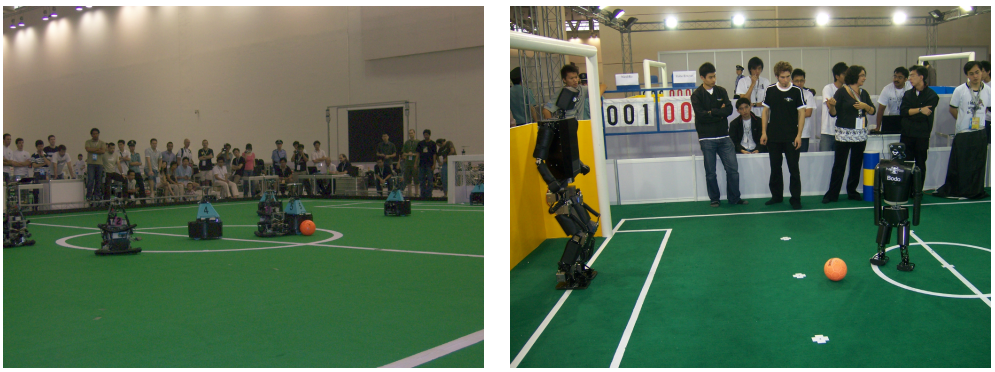


Figura 2.2: Instantes de las competiciones de robots de tamaño medio (izquierda) y humanoides (derecha) de la edición 2008 de la RoboCup celebrada en Suzhou (China)

2.1.2. Homogeneidad Vs heterogeneidad

Robin Murphy describe en [Mur00] esta característica como el grado de similitud entre los robots que forman el grupo. Un grupo heterogéneo está formado por al

menos dos miembros con *hardware* o *software* diferente, mientras que en un grupo homogéneo todos los miembros son idénticos. Los robots *Ganymede, Io* y *Callisto* diseñados en Georgia Tech son un buen ejemplo de equipo homogéneo. Estos robots ganaron el primer premio de la competición *Pick up the Trash* del AAAI Mobile Robot Competition en 1994. El resto de competidores diseñaron robots individuales más sofisticados con sistemas complejos de visión para reconocer los objetos a recoger. Por su parte *Ganymede, Io* y *Callisto* tenían un diseño mecánico sencillo y un programa de control basado en una secuencia de acciones reactivas siguiendo la arquitectura AuRa[AB97]. Los robots no se comunicaban entre sí, pero eran capaces de detectase mutuamente creando fuerzas repulsivas.

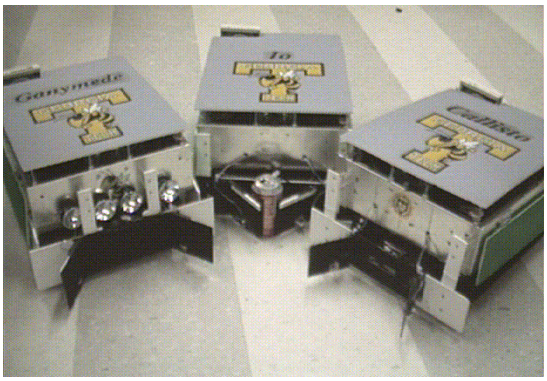


Figura 2.3: Robots Ganymede, Io y Callisto

En la categoría de *plataforma estándar* de la competición RoboCup encontramos a menudo ejemplos de equipos heterogéneos. La heterogeneidad la encontramos no tanto en la mecánica, sino en la parte del *software*, donde cada robot ejecuta roles distintos. En [PV08] se detalla uno de los últimos trabajos usando los robots aiBo, que permite establecer estrategias de decisión para posicionar los robots de un mismo equipo. Es habitual, establecer diversos roles como el de portero, chutador, defensa o delantero. Mientras que la labor del chutador es bastante clara y definida, consistente en alcanzar la pelota y golpearla hacia la portería adversaria; la labor del delantero (que apoya al chutador) es más difusa. En este rol se trata de maximizar las opciones ante un posible pase o una pérdida de la pelota. ACTRESS [AMI89] y ALLIANCE [Par98] son ejemplos reales de arquitecturas que manejan grupos heterogéneos.

El concepto de cobertura de tarea o *task coverage* [Par94] expresa la habilidad de un robot para completar una tarea dada, es decir, el grado de independencia para la tarea especificada. Este parámetro es un cuantificador de la necesidad de cooperación. Si este valor es elevado, habrá poca cooperación ya que los individuos se valdrán por ellos mismos para solucionar la tarea. Por el contrario, si la cobertura de tarea es baja, habrá que coordinarle con el resto de miembros para paliar las deficiencias de cada individuo. En grupos homogéneos, su valor es mínimo y éste crece a medida que aumenta la heterogeneidad.

El término de entropía social, acuñado por Tucker Balch en [Bal97], indica el grado de heterogeneidad del equipo. La entropía es una medida del grado de variación en un sistema. Este indicador asigna un número para expresar el grado de desorden en el grupo. Un valor de 0 significará que la sociedad es homogénea. La entropía social será máxima si todos los miembros son diferentes. Este valor aumentará a medida que el número de robots heterogéneos se incremente. La fórmula para expresar la entropía social de un grupo R es la siguiente:

$$Entropía(R) = -\sum_{i=1}^{c} p_i \, log_2(p_i) \tag{2.1}$$

siendo c el número de castas, es decir, el número de robots diferentes. Imaginemos que queremos calcular el valor de entropía social de un equipo de 5 robots participante en la liga de robots de cuatro patas de la RoboCup. Supongamos que se han diseñado 4 roles diferentes: Portero, defensa, chutador y jugador de campo. Si dos de los robots ejecutaran el rol de jugador de campo y los tres restantes los de portero, defensa y chutador respectivamente, tendríamos 4 castas. El valor de p_i corresponde al porcentaje de robots que pertenecen a la casta c_i, en nuestro ejemplo los valores serían $p_1 = p_2 = p_3 = 0{,}20$ y $p_4 = 0{,}40$. Si desarrolláramos la ecuación 2.1 obtendríamos un valor de *1.91* de entropía social.

2.1.3. Mecanismos de coordinación

Un equipo de robots puede cooperar de maneras muy diversas: Desde la más básica en la que cada individuo no tiene conciencia del resto, hasta las más elaboradas

basadas en negociaciones de alto nivel y con un elevado grado de comunicación. Ilustraremos varios mecanismos de cooperación con el siguiente ejemplo.

Maja Mataric ha dedicado parte de su investigación a estudiar el comportamiento emergente en grupos de agentes. Uno de sus experimentos más conocidos es el denominado *The Nerd Herd*, que consistió en preparar un entorno con 20 robots exactamente iguales. El objetivo del experimento era conseguir que todos los agentes alcanzaran el mismo objetivo: Llegar a otra habitación conectada por una puerta por donde sólo podía pasar un robot simultáneamente.

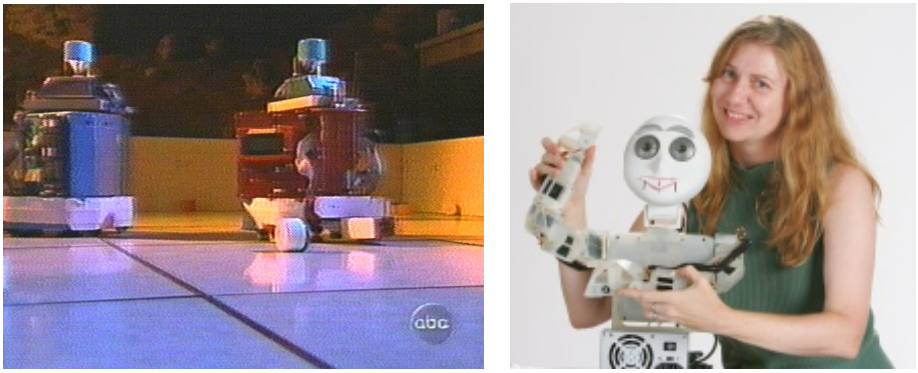


Figura 2.4: Robots del proyecto *The Nerd Herd* (izquierda) de la Universidad South California (EEUU), junto con su autora Maja Mataric (derecha)

En el primer conjunto de pruebas, cada robot no conocía la existencia de otros robots, es decir, operaban bajo una *coexistencia ignorante*. El conjunto de comportamientos reactivos que podían desplegar era muy simple y se limitaba a *mover-a-objetivo* y *esquivar-obstáculo*. Debido a que la cercanía con otro robot era tratado como si se estuviera cerca de un obstáculo, los robots estaban constantemente desviando su rumbo y alejándose del camino hacia el destino. El equipo era muy lento en culminar su tarea y a medida que se incorporaban nuevos robots empeoraba la tarea.

A continuación se modificó el experimento para que los robots fueran capaces de detectar la presencia de sus congéneres. A esta conciencia colectiva se le denomina *coexistencia informada*. Se añadió un tercer comportamiento *esquiva-robot* que simplemente detenía su camino durante un determinado tiempo. Los resultados fue-

ron mejores debido a que se reducían los atascos de robots, aunque el resultado final seguía siendo malo.

En el experimento definitivo se cambió la heurística del comportamiento *esquiva-robot*: Ahora los robots eran repelidos por otros robots pero después el robot trataba de dirigirse en la misma dirección que la mayoría de sus compañeros. El comportamiento emergente obtenido fue similar al de un rebaño o bandada de animales avanzando al unísono hasta el mismo lugar. Ahora no sólo se redujeron las colisiones, sino que la tarea se completó mucho más rápido que de cualquier otra manera anterior. La definición para esta nueva sociedad es la de *coexistencia inteligente*.

Modelar las intenciones, creencias, acciones, capacidades, ... del resto de miembros del grupo lleva a una menor necesidad de comunicación explícita. Este modelado permite en cierta medida presuponer en todo momento la acción que otro individuo va a acometer sin emitir ningún mensaje.

La coordinación de equipos de robots en entornos dinámicos es uno de los problemas fundamentales en grupos de agentes. Coordinarse normalmente significa sincronizarse. Este nivel de sincronización o intercambio de información están altamente ligados a la tarea a realizar. Por ejemplo y en general, tareas de transporte coordinado requieren mayor nivel de coordinación que trabajos de saqueo o recolección de objetos.

Los mecanismos de cooperación dinámica son adecuados para favorecer la flexibilidad y adaptabilidad a la hora de afrontar una tarea coordinada. En [CKC04] se propone un mecanismo de asignación dinámico de roles, que permite asociar tareas concretas a robots o grupos de robots sobre la marcha. Este mecanismo es descentralizado y cada individuo toma sus propias decisiones basadas en la información local y global. Podemos enumerar tres tipos de cambio de rol:

- Asignación o *Allocation*: Se asume un nuevo rol cuando se termina la ejecución una tarea.
- Reasignación o *Reallocation*: El rol activo se aborta y comienza la ejecución de otro nuevo.
- Intercambio o *exchange*: Dos o más robots se sincronizan e intercambian sus papeles.

¿Cuándo se produce el cambio de rol? Este es un aspecto importante y varios autores han propuesto el concepto de utilidad. Cuando un nuevo rol está disponible se calcula la utilidad de ejecutarlo y si la diferencia entre la utilidad del nuevo rol y el antiguo supera cierto umbral, el robot cambiará al nuevo rol.

2.1.4. Tipos de comunicación

La comunicación determina las posibilidades de interacción entre los miembros del grupo, siendo el vehículo para que la coordinación se manifieste. Podemos identificar tres tipos de interacción que tienen que ver con el *intercambio de información*: A través del medio, a través de los propios sensores del robot o explícita empleando comunicaciones.

La interacción a través del entorno es el método más simple y limitado. Se modifica el entorno como medio de comunicación sin existir una comunicación explícita. Esta interacción suele darse principalmente en robots reactivos produciendo comportamientos emergentes. Algunos ejemplos pueden verse en [Ark92] y [SSH94]. Este tipo de comunicación aparece en la naturaleza y se conoce como estigmergia, comportamiento innato o *eusocial behavior*. En las colonias de hormigas, los diferentes componentes colaboran a través de pautas o hitos dejados en el medio: feromonas, acumulación de objetos o cualquier otro tipo de cambio físico, como la temperatura.

La interacción sensorial se produce cuando continúa sin haber comunicación explícita pero a través de sus sensores (visión, infrarrojos, ...) cada miembro del equipo es capaz de percibir a otros individuos del mismo. De esta manera se pueden crear comportamientos colectivos como pueden apreciarse en [KKR$^+$94] y [KRI$^+$94].

Finalmente, la interacción usando comunicaciones utiliza comunicación explícita entre los robots como herramienta de coordinación, ya sea entre pares o mediante *broadcast*. Buscando el símil con la naturaleza, ahora la comunicación sería similar a la encontrada en animales superiores. La interacción entre los miembros es más elevada y existe mayor intencionalidad para maximizar el beneficio del grupo. Lo habitual aquí es utilizar protocolos existentes, aunque para robots concretos y limitados surgen protocolos específicos [Fer92], [AMdlHC03].

Otra definición interesante es el concepto de *negociación*, un tipo concreto de comunicación explícita. Las negociaciones permiten intercambiar información sobre el estado de los robots o sus intenciones, favoreciendo la cooperación. Los robots negocian sus pasos a seguir y puede entenderse como una comunicación de alto nivel. El lenguaje *KQML*[FF] es un esfuerzo por unificar protocolos de comunicación. *KQML* define tanto un formato de mensajes como un protocolo de manejo de los mismos, para proporcionar intercambio de conocimiento entre varios agentes. En [Syc90] se presenta una aproximación al modelado de negociaciones entre agentes, un concepto absorbido desde el campo de la Inteligencia Artificial.

Cuando un equipo de robots tiene necesidad de intercambiar información surge el concepto de *sincronización*. En general, existen dos tipos de comunicación explícita para coordinarse: Síncrona y asíncrona. Con comunicación síncrona los mensajes son enviados y recibidos constantemente, mientras que en el modo asíncrono se desencadena la comunicación sólo ante situaciones excepcionales o destacadas.

2.1.5. Interferencias y conflictos

La coordinación multi-robot es ventajosa pero también tiene sus riesgos. Uno de los problemas típicos surge cuando varios miembros quieren acceder a un recurso compartido, que puede ser por ejemplo una intersección de carreteras, un pasillo de una casa o el suelo de un almacén. Si dos o más individuos acceden a él en el mismo instante de tiempo se produce una colisión. Por tanto, es necesario que exista algún mecanismo de exclusión mutua para evitar estos problemas cuando sea necesario.

Otro ejemplo ilustrativo se manifiesta en la RoboCup. Si dos jugadores del mismo equipo deciden ir a por la pelota a la vez, ambos se estorbarán mutuamente. En [Tea05] podemos encontrar un protocolo de reserva de pelota para que sólo un jugador intente acceder a ese recurso en un instante de tiempo determinado.

Durfee [Dur95, VD95] introdujo el concepto de *Recursive Modeling Method* o RMM para modelar explícitamente las creencias sobre los estados de otros agentes. El objetivo de modelar las creencias del resto de individuos es predecir sus acciones para mantener un alto grado de cooperación y evitar en mayor medida las interferencias.

2.2. Aplicaciones de los sistemas multi-robot

Una vez que hemos barrido los fundamentos teóricos y los conceptos más interesantes de los sistemas robóticos cooperativos vamos a hacer un repaso a las aplicaciones típicas que se abordan con grupos de robots. Por cada una de ellas identificaremos las propuestas más representativas y la clasificaremos según los criterios presentados en la sección anterior, repasando también con ello parte de la bibliografía relevante de la robótica cooperativa.

2.2.1. Recolección o *foraging*

La recolección consiste en encontrar y recoger una serie de objetos diseminados por el entorno. En situaciones reales estas técnicas se pueden usar para recoger sustancias tóxicas, participar en situaciones de rescate, detectar de minas, etc. Los mecanismos de cooperación en estos entornos pueden ser variados, desde repartir los robots por zonas disjuntas, hasta construir cadenas con ellos para recolectar los objetos. Incluso pueden repartirse aleatoriamente y que emerja un comportamiento colectivo. Los equipos de robots para recolección suelen ser homogéneos y el control es totalmente distribuido. En cuanto al tipo de comunicación empleado lo habitual es utilizar interacción sensorial para evitar colisiones. Los sistemas más avanzados emplean comunicaciones para hacer un mejor reparto de las zonas a explorar y optimizar el tiempo empleado.

La figura 2.2.1 muestra instantes de la competición *Find Life on Mars*[Six98], inspirada en la exploración de Marte. Los robots participantes deben recolectar objetos y pelotas específicamente coloreados y depositarlas en una zona común. Los robots *Sally* y *Shannon* fueron programados por Georgia Institute of Technology para abordar el problema desde un punto de vista cooperativo. Los dos robots estaban gobernados por varias capas de control. La capa de bajo nivel, más reactiva, se encargaba de implementar los comportamientos reactivos como por ejemplo *esquivar_obstáculo* o *ir_a_objetivo*. El resultado final era un comportamiento emergente donde los dos robots no comunicaban explícitamente entre sí pero si colaboraban minimizando el tiempo de recolección.

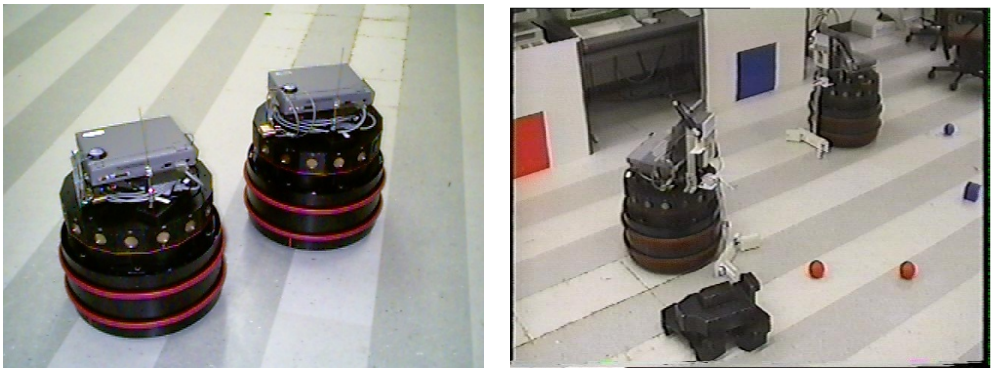


Figura 2.5: Robots *Sally* y *Shannon* (izquierda) pertenecientes a Georgia Institute of Technology participando en la competición de saqueo *Find Life on Mars* (derecha) dentro del *AAAI-97 Robot Competition*

2.2.2. Formaciones

Esta especialidad aparece cuando se combina la navegación y los grupos de robots. Consiste en el control de múltiples individuos moviéndose juntos en formación. Los trabajos en el ámbito de las formaciones en vehículos no tripulados despertaron un interés militar en el Ministerio de Defensa de Estados Unidos durante la década de los 90. El control de las formaciones de vehículos resulta interesante para realizar funciones de escolta o exploración.

Un primer ejemplo de formaciones lo podemos encontrar en [DOK98]. El método propuesto usa sensores locales y no externos y asigna a uno de los robots el rol de líder, por tanto este sistema es centralizado. Al haber un robot actuando como líder, el equipo es por definición heterogéneo, pues los roles diferenciados de cada tipo de robot marcan la heterogeneidad. En este caso concreto la cooperación se produce mediante interacción sensorial detectando la distancia y orientación entre los vehículos.

El robot que juega el papel de líder se mueve según un planificador de trayectorias disponible y el resto tratan de seguir sus pasos. El objetivo es mantener una distancia aproximada entre las parejas de robots y que todos tengan una orientación determinada. Los parámetros involucrados en las decisiones son la distancia entre los robots, la velocidad y la orientación de cada uno. Mediante cálculos matemáticos se

consiguen dar los comandos de movimiento adecuados a cada robot para que la formación se mantenga en equilibrio. La presencia de obstáculos podría desencadenar cambios de formación y pasar a formas mas lineales para evitar el contacto, etc.

Un segundo ejemplo interesante es el caso del proyecto desarrollado en el *Georgia Tech Mobile Robot Laboratory*, utilizando vehículos terrestres no tripulados. En la figura 2.2.2 se muestra un convoy real de vehículos durante una demostración, así como las cuatro formaciones clásicas más empleadas (línea, columna, diamante y cuña). En este caso, uno de los vehículos actuaba de líder y no tenía que preocuparse de mantener la formación. Esto nos indica que el sistema es heterogéneo. Los vehículos estaban equipados con GPS y esto permitía conocer con relativa exactitud la posición de cada uno. En cuanto al mecanismo de comunicación el proyecto desarrolló varias técnicas siendo las más empleadas las basadas en comunicación explícita.

Figura 2.6: Equipo de cuatro camiones militares propiedad de DARPA durante el proyecto *Demo II Project* (izquierda) y formaciones en línea, columna, diamante y cuña (derecha)

2.2.3. Gestión de alcenamiento

En grandes centros de distribución, almacenes o pequeños negocios con gran volumen de movimiento de productos es habitual dedicar mucho tiempo a la gestión del inventario. Esta gestión involucra tareas de llenado de determinados compartimentos o vaciado cuando los clientes lo solicitan. Un equipo de robots cooperantes puede ser una muy buena solución para este problema, liberando a las personas de esta tarea

repetitiva y mejorando la eficiencia de la gestión de los recursos. Cuando disponemos de varios agentes compartiendo el mismo medio físico también pueden surgir colisiones. Este problema puede interpretarse como un problema de acceso a un recurso y se puede resolver mediante reglas, prioridades o comunicación. Desde otro punto de vista también puede verse como un problema de planificación de movimientos entre varios robots.

Figura 2.7: Robots de *Kiva Systems* (izquierda) e instalaciones en Denver (derecha) donde los robots realizan trabajos de almacén

La empresa *Kiva Systems*[Gui08, Kiv09] ofrece al mercado de los centros de distribución un equipo de cientos robots formando un enjambre (*swarm*) para encargarse del inventario. Esta empresa afincada en Boston ha desarrollado un sistema que permite transportar estanterías de carga a diferentes sitios de un almacén de manera coordinada. Hasta donde sabemos *Kiva Systems* no ha publicado datos concretos sobre la tecnología de control de los robots, ni de su diseño mecánico o su algoritmo de asignación de recursos. Sin embargo, los resultados que han obtenido hablan por sí solos y se podría decir que representan la punta de lanza en cuanto a resultados reales en este área se refiere.

2.2.4. Manipulación coordinada

Hay muchos trabajos sobre este tema consistente por ejemplo en arrastrar cajas usando varios robots con brazos, o de manera más genérica, manipular objetos entre varios agentes. Un dato interesante de esta especialidad es que la cooperación puede

llevarse a cabo sin que cada robot sepa de la existencia del resto. Algunos trabajos interesantes en esta línea son [SSH94] y [TK93].

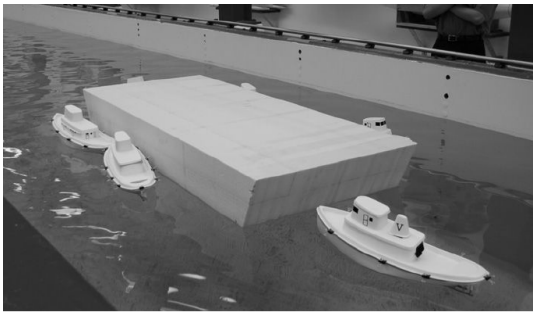


Figura 2.8: Seis barcos autónomos remolcando de manera coordinada una barcaza simulada

En la figura 2.8 podemos observar los experimentos realizados por Joel M. Espósito en [Esp08], donde plantea la cuestión del posicionamiento de nuevos miembros a la hora de transportar coordinadamente objetos. En su caso, los robots son grandes grupos de barcos autónomos a escala. Su trabajo está inspirado en la literatura actual sobre manipulación y agarre de objetos con manos robóticas. Su enfoque es genuino porque afronta el problema con un control distribuido y utilizando un equipo de robots, equipados con módulos de comunicación inalámbrica para intercambiar información entre ellos.

En [BMB90] se puede encontrar otro ejemplo donde dos brazos manipuladores industriales cooperan para trabajar sobre el mismo objeto. Esta necesidad surge para soportar objetos muy grandes o pesados que de manera individual no podrían sujetarse. El trabajo explora los mecanismos para trazar trayectorias compatibles con los dos manipuladores y que eviten cualquier obstáculo.

2.2.5. Fútbol robótico

El fútbol robótico es un campo de pruebas interesante que está cobrando mayor interés año a año. Además del reto tecnológico que requiere desarrollar un sistema completo perceptivo, locomotor, estratégico, que incluya auto-localización y cooperación entre sus miembros, tiene el ingrediente extra de ser competitivo. Un equipo de robots se enfrenta a otro equipo con objetivos opuestos a los tuyos. Desde el punto

de vista de coordinación es un escenario muy propicio a establecer roles entre los miembros de cada equipo (portero, defensor, delantero, etc.) con estrategias de posicionamiento y comportamientos específicos.

La percepción también se beneficia de la cooperación (como veremos en la sección 2.2.7) y la estimación de los objetos interesantes del partido, como la pelota, puede realizarse de manera más precisa intercambiando información entre varios robots.

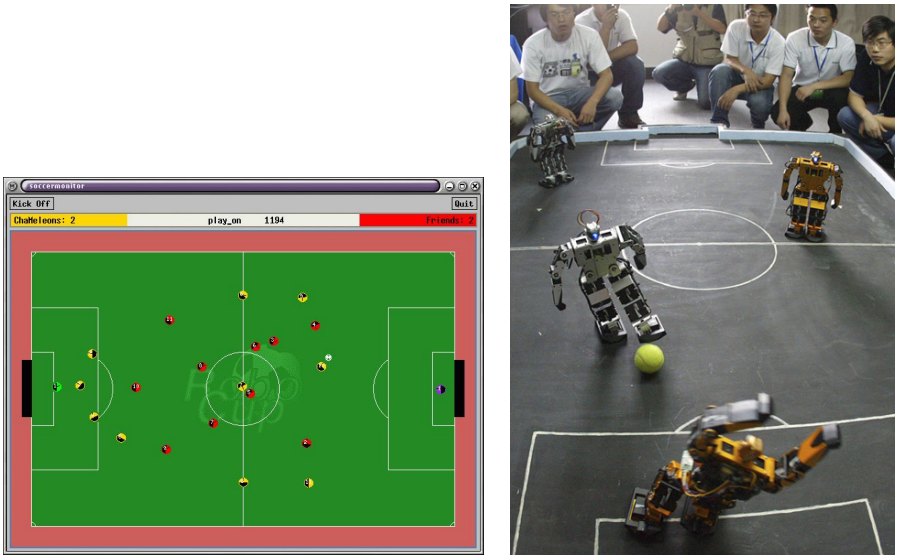


Figura 2.9: Simulador *Soccer server* empleado en la competición RoboCup (izquierda) e instante de un partido de una competición organizada por la FIRA (derecha)

Existe otra variante del fútbol robótico con las mismas reglas pero con diferente entorno: Un entorno virtual simulado. Los simuladores como *sserver* [NNM+98] proporcionan una excelente plataforma para trabajar con sistemas multiagente. Simulan los limitados sistemas perceptivos de los robots, permiten comunicar información entre los miembros del equipo utilizando primitivas de tipo *say*, e incluso se simulan los errores habituales tanto en los comandos de movimiento como en las capturas sensoriales.

2.2.6. Construcción distribuida de mapas

La construcción de mapas es una tarea ampliamente trabajada en el campo de la Robótica, que permite ampliar la autonomía del robot. Consiste en la creación conjunta de un modelo del entorno. Abarca subtareas como la exploración, que persigue deambular por el entorno descubriendo y percibiendo sus características más relevantes. La construcción de mapas también requiere incorporar algoritmos de autolocalización para enriquecer el mapa de manera adecuada. En el capítulo III del libro *Probabilistic Robotics*[TBF05] se hace un repaso muy detallado de las técnicas probabilísticas de construcción de mapas.

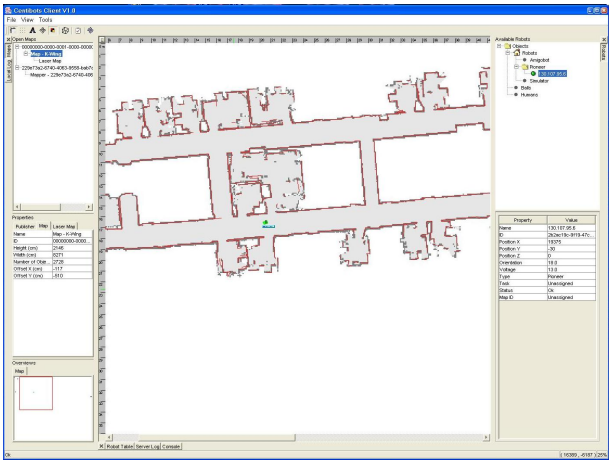


Figura 2.10: Mapa creado dentro del proyecto Centibots por un enjambre de robots

El problema de explorar el entorno es ampliamente conocido y existen muchas técnicas para conseguirlo usando un solo robot. Sin embargo, la ventaja asociada al uso de un equipo de robots es clara: Se mejora el tiempo de exploración debido al reparto de las zonas a explorar. Atajar el problema usando un grupo de robots resulta algo más complejo por la dificultad añadida de tener que coordinarlos.

En [BMF+00] presentaron un sistema cuya cuestión clave era hacia dónde dirigir cada robot. Esta técnica utiliza un mapa de celdas de probabilidad. El entorno se divide en celdas o casillas y se almacena en cada una la probabilidad de que en ella haya un obstáculo. También se propone la definición de celda frontera como aquella

que limita con otra de la que no se conoce ningún dato (aún está sin explorar). El objetivo del algoritmo es asignar celdas frontera a los robots para que viajen hasta ellas. Esta asignación no la realizan los propios robots, sino un agente externo que recibe la información sensorial de cada individuo.

Este sistema es claramente centralizado, pues la asignación de las celdas la calcula y ordena un agente externo y no los propios robots. El sistema es heterogéneo al existir esta unidad de procesamiento central. En cuanto a la comunicación se emplea comunicación explícita para asignar celdas frontera a cada robot. Además cada robot tiene que localizarse dentro del escenario y comunicar al resto su posición y el mapa actual que lleva registrado. En principio, todos los robots deberían manejar aproximadamente el mismo mapa en un instante determinado.

Relacionado con este ámbito se encuentra el problema de la localización cooperativa. Tiene como fin el utilizar al resto de miembros del grupo consiguiendo una mejora de la auto-localización. Es habitual que el rendimiento en navegación aumente si la localización es más precisa. En [RDM00] proponen que la exploración del entorno se haga en parejas de robots. Cada robot está equipado con algún tipo de sensor de seguimiento para mantener la atención en su compañero. El entorno es modelado mediante una aproximación polinomial, es decir, se simplifica el entorno modelándolo como segmentos rectos. La estrategia propuesta en ese artículo consiste en que uno de los robots permanece detenido mientras el otro comienza a explorar el entorno. El robot que navega siempre observa al robot detenido. Este proceso va alternándose periódicamente. En la figura 2.11 podemos observar la imagen real obtenida por uno de los robots del experimento.

Los dos robots van confeccionando un mapa y éste va enriqueciéndose por la línea que une ambos robots. Si durante la exploración se deja de ver al robot estático, significa que ha aparecido un obstáculo. Este agente estático actúa de baliza para el que va explorando el entorno con el fin de corregir los errores odométricos.

Los robots están equipados con líneas negras alrededor de su chasis. La separación entre ellas, junto con la distancia estimada recorrida, son los parámetros con los que se corrige la odometría. Por supuesto, el sensor encargado de visualizar estas marcas es una cámara.

Figura 2.11: Imagen obtenida por el robot que monitoriza a su compañero

Otro ejemplo que hace énfasis en la localización y construcción cooperativa de mapas en vehículos no tripulados debajo del agua se detalla en [WL04]. A pesar de disponer de un grupo de robots, sólo algunos de ellos son los encargados de confeccionar el mapa y estimar las posiciones del resto de miembros del equipo. El entorno donde se desenvuelven estos vehículos hace imposible el uso de sensores láser y GPS. La estimación de la posición se realiza mediante sensores acústicos. El artículo presenta un grupo de robots heterogéneos y centralizado que disponen de este tipo de sensores y distingue entre dos jerarquías de individuos: Maestros y esclavos. Los maestros disponen de mejores sensores y de un sistema acústico para interrogar al resto de robots y compartir información. Los esclavos tienen sensores más limitados y no disponen de la capacidad de forzar una interacción. El algoritmo utilizado emplea una extensión del filtro de Kalman, que junto con las observaciones realizadas por el resto de robots se usan para construir el mapa y corregir las posiciones de cada uno. Es necesario utilizar algoritmos de correspondencia para mezclar los datos procedentes de varias fuentes.

2.2.7. Percepción distribuida

Otra aplicación típica es el uso de varios robots para percibir objetos. Las ventajas de los equipos de robots en estas aplicaciones son la mayor superficie de cobertura, la mejor precisión obtenida, mayor robustez ante fluctuaciones en la estimación, etc. Una de las aplicaciones de la percepción distribuida son las tareas de vigilancia o seguridad. Dentro de las aplicaciones de seguridad un aspecto importante consiste en visualizar y realizar seguimiento de objetos móviles. La cuestión clave en este tipo de sistemas reside en dónde colocar los sensores de visión o atención.

Un ejemplo detallado en [Par02] estudia el uso de un conjunto de robots para llevar a cabo esta tarea y minimizar el tiempo en el que algún objetivo escape a la atención de los sensores. El artículo establece una métrica que corresponde con el número medio de objetivos que están siendo seguidos por al menos un robot. Esta métrica es la que trata de maximizar el algoritmo propuesto. El trabajo define tres franjas concéntricas alrededor de cada robot:

1. Área sensorial: Zona alrededor de cada miembro del equipo que son capaces de percibir los sensores.

2. Área de predicción de seguimiento: Zona intermedia donde se sitúan objetos que no son directamente visibles por el robot pero si lo son por otros miembros.

3. Área de comunicación: Límite exterior que marca la frontera hasta la que es posible emitir señales para interactuar con el resto de robots. Este concepto es muy importante en las redes de sensores, pues define el alcance comunicativo de cada robot. Se debería garantizar que siempre existieran varios robots dentro de este área, para no romper en ningún caso la cadena de comunicación.

Cada robot monitoriza los objetos contenidos en su área de atención sensorial y envía al resto de individuos las posiciones de cada objeto. Con los datos recibidos del resto de robots, se realiza un seguimiento virtual de los objetos que no son directamente visibles pero que están situados dentro del área de predicción. Estos objetos situados en el área de predicción influyen en las decisiones de movimiento de cada robot.

En [JMCB08] se describe otra aplicación compuesta de cámaras fijas repartidas por el techo de una habitación. Se basa en un sistema centralizado que recoge la información de cada cámara y mantiene una estimación de la posición en tres dimensiones de las personas que hay en la habitación. Su objetivo es monitorizar la posición de las mismas para detectar situaciones de desvanecimiento o caídas. Está especialmente ideada para personas mayores que quieren mantener su autonomía viviendo en su propia casa y disponer de un sistema de vigilancia pasivo que reduzca el tiempo de asistencia en caso de accidente.

2.2.8. Coordinación de actuadores

En esta sección nos vamos a focalizar en la coordinación de los actuadores principales de los robots caminantes: sus patas. Algunas aplicaciones potenciales de los robots caminantes citadas en la literatura son el transporte militar, inspección en entornos peligrosos, trabajos agrícolas, en la construcción, tareas de rescate, etc. En la figura 2.12 se muestra un ejemplo de robot caminante que posee múltiples actuadores, sistema de percepción y coordinación de movimientos. Este robot es usado, entre otras tareas, para la exploración y detección de territorios minados.

Un robot caminante debe ser físicamente capaz de atravesar los trazados irregulares y los obstáculos que existan en el terreno sobre el que se desplace. Para esto, se debe conseguir una fiabilidad en el control autónomo del robot, una alta precisión y una adecuada eficiencia. Esto es, debe haber un alto nivel de coordinación entre los actuadores que posea teniendo en consideración la fusión de todo el sistema sensorial que interactúa con el entorno. La coordinación entre los actuadores de un robot caminante, entonces, debe brindar una alta capacidad de adaptabilidad al terreno, y lo más importante en su propia seguridad y de los objetos o personas que le rodean.

Las tareas que realizan los robots caminantes cada vez son más complejas, debido a los avances que se suscitan en el campo de la robótica y, por otro lado, en las respuestas que se ofrecen a la sociedad en que vivimos. El sistema de percepción del entorno y su fusión sensorial deben brindar la suficiente información para que por medio de algoritmos de control se proporcione una coordinación de movimientos

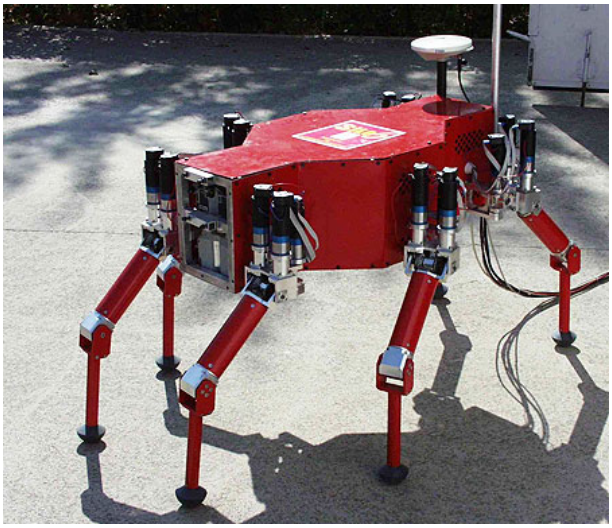


Figura 2.12: Robot Silo6, propiedad del Instituto de Automática Industrial (España)

adecuada entre las diferentes partes del cuerpo del robot, para que éste ejecute una acción como si se tratase de un *ser vivo*.

Por ejemplo, estrategias de control de fuerza aplicadas al robot humanoide SILO2 para que se acomode cuando una fuerza actúa sobre él. Para ello, el robot debe *sentir* la fuerza sobre uno de sus lados provocando que el sistema de control mande a realizar una tarea coordinada sobre los actuadores de manera independiente, con la finalidad que el robot se evada de la fuerza, pero manteniendo una postura de estabilidad[Mon05].

Otro ejemplo es la realización de trabajos conjuntos entre el robot humanoide HRP y un ser humano, en labores de transporte de piezas de un lugar a otro[HKK$^+$04]. El HRP posee algoritmos de control de impedancia que actúan sobre sus brazos. Dependiendo de los vectores de fuerzas que se producen durante la manipulación de la pieza, el algoritmo de control jerárquico realiza un proceso coordinado para que las patas del robot se acomoden de tal forma que el ZMP esté dentro del polígono de apoyo del humanoide manteniendo la estabilidad del mismo.

2.3. Percepción coordinada de objetos

El problema de la percepción en robótica ha sido habitualmente atacado desde un punto de vista individual. Como ya hemos mencionado en el capítulo anterior, los sistemas multi-robot son una apuesta para mejorar muchas de las aplicaciones y técnicas existentes en la realidad. En esta sección vamos a profundizar en uno de los problemas abordados en esta tesis, la estimación cooperativa de un objeto, y en particular a repasar los trabajos más importantes dentro de la RoboCup en este campo.

Este mecanismo de cooperación permite que cada robot sea capaz de modelar su entorno más allá de lo que percibe a través de sus propios sensores, apoyándose en las observaciones de otros robots del equipo.

Las técnicas de fusión sensorial se han empleado habitualmente en aplicaciones para un solo robot. Con ellas es posible modelar el entorno combinando varias fuentes de información multimodal procedentes del mismo robot y de diferentes instantes de tiempo. Por ejemplo, en las aplicaciones de construcción de mapas se integran observaciones procedentes de sensores láser, de ultrasonidos o incluso de cámaras con información de movimiento procedente de los sensores odométricos. En el capítulo III de [TBF05] se hace un repaso muy detallado de las técnicas probabilísticas de construcción de mapas.

En la figura 2.13 podemos observar otro ejemplo de fusión multisensorial dentro de un único robot. Las imágenes muestran la reconstrucción del entorno generada por el robot *Junior*[MBB+08] combinando múltiples sensores láser y radar. *Junior* fue el vehículo desarrollado por la Universidad de Stanford y liderado por Sebastian Thrun para participar en la competición DARPA Urban Challenge[urb] en noviembre de 2007.

La fusión sensorial multi-robot tiene como objetivo mantener una estimación más precisa, estable y robusta del entorno, que la que puede conseguir un robot por sí solo. Esto se consigue gracias a la combinación de información procedente de otros robots. Existen dos grandes familias predominantes de algoritmos de fusión sensorial: Los probabilísticos y los no probabilísticos. Los métodos probabilísticos han demostrado, en general, ser los más apropiados para problemas de estimación debido a sus propiedades para modelar la incertidumbre en el estado de el/los objetos

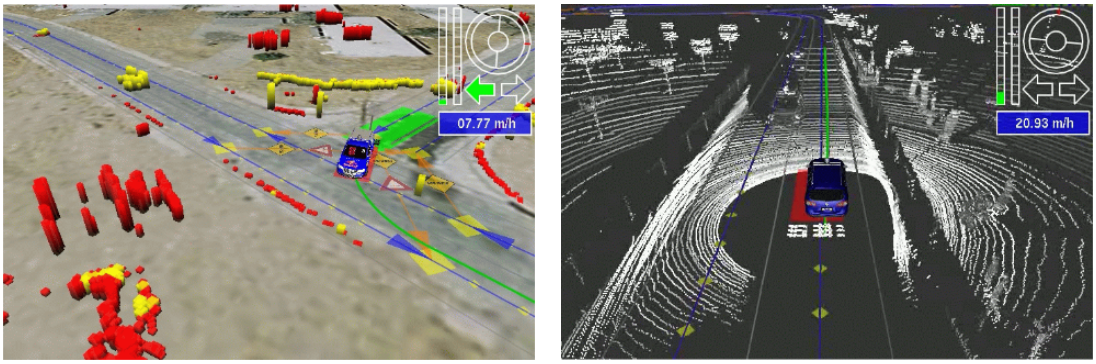


Figura 2.13: Reconstrucciones 3D realizadas por el robot *Junior* en distintos momentos de la competición *DARPA Urban Challenge*

a estimar. Aunque a continuación repasaremos las características fundamentales de unos y otros, en [SK08] se puede encontrar una revisión detallada de los algoritmos de ambas familias.

El núcleo de las técnicas probabilísticas es la regla de Bayes. Esta regla permite inferir el estado de un objeto x en base a una serie de observaciones z sobre él, ya sean del mismo robot o de otro. La inferencia Bayesiana necesita que exista una relación entre x y z modelada como una Función de Densidad de Probabilidad (en adelante FDP). La ecuación 2.2 muestra la regla de Bayes.

$$P(x|z) = \frac{P(z|x) * P(x)}{P(z)} \tag{2.2}$$

La potencia de la regla de Bayes estriba en que se pasa de calcular la probabilidad de estar en un estado x dada una observación z, a calcular la probabilidad de obtener la observación z suponiendo que se está en el estado x. Esta última probabilidad es más sencilla de calcular puesto que podemos construirnos a priori un *modelo de observación*. Este modelo se creará tomando observaciones en posiciones conocidas del entorno. De este modo, modelaremos la distribución de probabilidad del modelo de observación y tendremos una estimación de $P(z|x)$.

El conjunto de posibles posiciones donde el algoritmo de estimación puede concluir que se encuentra un objeto se denomina espacio de estados. Nuestro espacio

de estados va a estar modelado por una FDP. El manejo eficiente de FDP's puede realizarse de varias maneras.

Una primera aproximación es emplear métodos analíticos, donde se define una fórmula parametrizada para expresar la función. Por ejemplo utilizando la media y la varianza de la estimación para modelar una distribución gaussiana bidimensional.

Otra aproximación es utilizar métodos discretizados para manejar las FDP's. En este caso la función de densidad de probabilidad se modela con una rejilla, donde cada celda almacena la probabilidad de que la observación se encuentre en el área abarcada por ella. En general, los métodos basados en rejillas suelen ser muy costosos computacionalmente. La razón de su elevado tiempo de cómputo es la constante iteración por todas sus celdas de los algoritmos, ocasionando tiempos de proceso exponenciales.

Una última aproximación comprende los métodos muestreados, donde la FDP es representada por un conjunto de muestras con un peso asociado. Este último enfoque evalúa un subconjunto del espacio de estados posibles. De esta manera el tiempo de cómputo es radicalmente menor y los resultados pueden ser muy similares.

Los métodos probabilísticos no están exentos de problemas. Son sistemas complejos que necesitan manejar una gran cantidad de probabilidades para ser capaces de operar de manera adecuada y tienen problemas a la hora de asignar probabilidades en caso de desconocimiento o ambigüedad. Es por esto que han aparecido otras técnicas de fusión sensorial alternativas a las basadas en métodos probabilísticos. Las principales son el cálculo de intervalos, la lógica borrosa y la teoría de evidencias.

En el cálculo de intervalos se modela la incertidumbre sobre el conocimiento de cierto estado acotando los valores posibles entre un intervalo. Por ejemplo, $x \in [a, b]$ describe el parámetro *x* entre los valores *a* y *b*. Ésta es la única asunción posible y ni siquiera se puede afirmar que los valores de *x* se distribuyen de manera uniforme dentro del intervalo. Varios intervalos acotados pueden combinarse con reglas básicas de manipulación como la suma, resta, multiplicación, división, aritmética de matrices, etc. En general, el cálculo de intervalos es recomendable cuando no tenemos o es difícil obtener un modelo probabilístico de observación, pero los sensores están correctamente caracterizados y su error es acotado. En [Moo67] se puede encontrar una descripción detallada de esta técnica.

La lógica borrosa es otra alternativa a los métodos probabilísticos que ha alcanzado gran éxito en determinadas aplicaciones (especialmente el control borroso). En los sistemas lógicos clásicos podemos definir una función de membresía que asocia un determinado elemento a un grupo o conjunto. Únicamente podemos decir si dicho elemento pertenece o no al grupo. En el ámbito de la lógica borrosa se dice que todo elemento tiene un grado de membresía con el grupo expresado entre 0 y 1. Así, un elemento puede pertenecer a varios grupos en mayor o menor grado. Por supuesto las reglas borrosas o difusas se pueden componer usando los mismos operadores que en la teoría de conjuntos clásica: Unión, intersección, negación, etc. La lógica borrosa se recomienda en sistemas controlados por un supervisor o cuando se quieren fusionar tareas de alto nivel. En [DP80] se puede encontrar una amplia descripción de estos sistemas.

Los métodos basados en teoría de evidencias permiten modelar explícitamente la ignorancia para distinguir entre los posibles estados de una creencia, es decir, la ambigüedad. Ahora, el conjunto de estados posibles para una determinada variable estará formada por todas las combinaciones posibles de estados. Por ejemplo, para expresar la creencia de si una determinada celda de un mapa está ocupada o vacía tendríamos los siguientes conjuntos de estados:

$$2^x = \{\{\text{ocupado, vacío}\}, \{\text{ocupado}\}, \{\text{vacío}\}, \emptyset\}$$

Cada uno de estos conjuntos de estados tendría asignado un valor que expresa posibilidad. El objetivo del estado {ocupado, vacío} sería modelar la ignorancia sobre si la celda está ocupada o vacía.

El uso de conjuntos de estados permite una descripción más rica de la representación de creencias. Por supuesto, el coste añadido es un incremento exponencial en la complejidad del sistema. Las técnicas basadas en razonamiento de evidencias han alcanzado cierto éxito en el dominio del razonamiento automático. En [PNDW98] se puede encontrar una descripción ampliada de esta técnica.

A continuación se van a examinar las técnicas de fusión sensorial más utilizadas en el entorno de la RoboCup. Estos trabajos son muy importantes para el desarrollo de esta tesis, pues suponen la base sobre la que se ha diseñado el aporte aquí presentado.

Los trabajos se han dividido en cuatro familias de algoritmos: Técnicas de fusión basadas en operadores estadísticos, en filtros de Kalman, en métodos de Monte Carlo y en métodos híbridos. Para cada una de estas familias se ha realizado un trabajo de documentación para conocer los equipos y las técnicas concretas que existen en este ámbito.

2.3.1. Fusión sensorial basada en operadores estadísticos

CAMBADA

CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture) es el equipo participante en la liga de robots de tamaño medio de la Universidad de Aveiro (Portugal). Desde el año 2003 han participado en casi todas las competiciones internacionales RoboCup, así como en eventos nacionales e internacionales. Su mejor participación fue el tercer puesto logrado en la RoboCup 2009, donde además obtuvieron el primer puesto en la prueba de desafíos técnicos.

La arquitectura de CAMBADA está diseñada con la cooperación como uno de sus pilares fundamentales. Uno de los componentes más importantes del software de cada robot es una base de datos de tiempo real (*Real-Time DataBase o RTDB*). Esta base de datos replicada en cada miembro se actualiza periódicamente y es utilizada para compartir el modelo del mundo entre todo el equipo.

Su algoritmo de percepción cooperativo de la pelota [LLC08] está basado en uno de los principales estadísticos: la media aritmética. Cada robot i, de un total de n que sean capaces de detectar la pelota, contribuirá con su estimación a la pelota global. Este cálculo se realiza aplicando la media aritmética entre todas las estimaciones. El siguiente paso es descartar todas aquellas posiciones que sean díscolas, es decir, posibles falsos positivos debido a que sus estimaciones están relativamente lejos de la estimación media. Finalmente, la estimación del robot más cercana a la media es la que se toma como estimación compartida por el equipo.

$$\begin{aligned} SharedBall_x &= \tfrac{1}{n} \textstyle\sum_{i=0}^{n} Ball_x^i \\ SharedBall_y &= \tfrac{1}{n} \textstyle\sum_{i=0}^{n} Ball_y^i \end{aligned} \tag{2.3}$$

Como podemos observar a la vista de la ecuación 2.3, todas las estimaciones aportadas por los robots que perciben la pelota se ponderan de la misma manera. Por lo general, los modelos sensoriales son más precisos a corta distancia que a larga distancia. Otra alternativa para tener en cuenta este hecho es corregir cada estimación con un factor proporcional w_i a la distancia $dist_i$, e incluso incluir la calidad de la estimación $calidad^i_{pelota}$ y la calidad de la posición propia $calidad^i_{PosiciónRobot}$ según el algoritmo de auto-localización.

$$w_i = \frac{1}{dist_i} \cdot calidad^i_{pelota} \cdot calidad^i_{PosiciónRobot} \tag{2.4}$$

$$\begin{aligned} SharedBall_x &= \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} Ball^i_x \cdot w_i \\ SharedBall_y &= \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} Ball^i_y \cdot w_i \end{aligned} \tag{2.5}$$

La principal ventaja del algoritmo que emplea CAMBADA es su sencillez, que junto con la escasa capacidad de cálculo requerida lo convierten en una alternativa adecuada siempre y cuando el error en la posición del observador no sea demasiado grande. Esta es su principal limitación, la incertidumbre en la posición del robot no es tenida en cuenta en ningún momento. Además, este algoritmo ofrece una estimación instantánea, no realizando ninguna acumulación de evidencias a lo largo del tiempo, información muy valiosa para protegerse, por ejemplo, de falsos positivos esporádicos.

2.3.2. Fusión sensorial basada en filtro de Kalman

Pinheiro y Lima

Varios trabajos han utilizado un enfoque Gaussiano para modelar la estimación de la pelota. Pinheiro y Lima [PL04] modificaron el algoritmo de Durrant-Whyte's [DW87], considerando las anteriores posiciones de la pelota para la futura estimación. Además utilizaron una métrica para medir la calidad de las observaciones y de la fusión sensorial, con el fin de mejorar la toma de decisiones y mantener un modelo del mundo más estable y robusto.

Stroupe

Stroupe [SMB01a], por su parte, también utilizó una Gaussiana bidimensional en forma canónica para modelar cada estimación de la pelota. Este método permite fusionar las diferentes estimaciones del resto de robots multiplicando las Gaussianas. Para predecir la posición de la pelota se utiliza una aproximación similar al filtro de Kalman.

2.3.3. Fusión sensorial basada en métodos de Monte Carlo

Mostly Harmless

El equipo Mostly Harmless [SFF+03] de la Universidad Tecnológica de Graz en Austria han utilizado métodos de fusión sensorial para integrar las diferentes percepciones procedentes de los sensores con los que están equipados cada robot.

Han utilizado MIRO [KUS+02] como capa de abstracción para comunicar aplicaciones entre diferentes robots. Con el fin de obtener un modelo global compartido del mundo han utilizado una aproximación basada en técnicas de Monte Carlo.

2.3.4. Fusión sensorial basada en métodos híbridos

GermanTeam

GermanTeam es el nombre de un consorcio alemán formado por Humboldt-Universität de Berlin, Universität Bremen, Technische Universität Darmstadt y el Laboratorio DFKI de Bremen para participar en la liga de plataforma estándar de robots de cuatro patas en la RoboCup. Su participación comenzó en 2001 y finalizó en 2008, año en que disolvió la asociación junto con la desaparición de la liga de cuatro patas (robots aiBo) en favor de la de dos (robots Nao). Este equipo cosechó numerosos éxitos destacando sus victorias en la RoboCup en los años 2004, 2005 y 2008. Liderado por Thomas Röfer, este consorcio no sólo destacó por su nivel de juego, sino por su gran capacidad de organización integrando el código de cuatro centros de investigación y poniendo su código fuente a disposición de la comunidad.

En el año 2005, GermanTeam publicó un extenso documento técnico [RLB+05] describiendo por completo todos los aspectos de su código. Este equipo desplegó siempre un gran nivel de cooperación entre sus robots y los detalles de su sistema de gestión de roles fueron desgranados en este documento.

Utilizado por primera vez en la competición German Team 2008, este consorcio alemán emplea un enfoque híbrido [BBG+08] para mantener una estimación compartida de la pelota. Un filtro de partículas y varios filtros de Kalman son combinados como expondremos a continuación.

Cada robot utiliza un filtro de partículas para estimar la posición de la pelota de manera individual. Periódicamente el estado del filtro de partículas es comunicado entre los miembros del equipo. Cada una de las partículas es tratada como una medida potencial que alimenta un filtro de Kalman. Cada partícula recibida puede ser añadida como una observación en un filtro de Kalman ya existente (si la posición de esa partícula está a menos de 1 m. de distancia de la estimación del filtro de Kalman), o bien, puede ser utilizada para crear un nuevo filtro de Kalman inicializado en la posición de la partícula. A través de una comparación entre la varianza de cada filtro de Kalman se crea la hipótesis de la pelota.

Iterativamente, todos los filtros de Kalman son actualizados pero si alguno de ellos no recibe ninguna observación en 10 iteraciones será eliminado. El principal beneficio de esta técnica es que se mantiene la conocida eficiencia de los filtros de Kalman pudiendo mantener múltiples hipótesis.

ISocRob MSL team

El equipo ISocRob liderado por Pedro Lima y perteneciente al Laboratorio de Sistemas Inteligentes de la Universidad Técnica de Lisboa participa en la categoría Medium Size League de la RoboCup desde 1998. Su trabajo en el campo de los sistemas multi-robot es importante, como así lo demuestran las tesis leídas por muchos de sus miembros y la numerosa cantidad de publicaciones disponibles al respecto.

En [SL09] se describe su mecanismo de seguimiento cooperativo de la pelota basado en un enfoque Bayesiano. Su propuesta es utilizar un método distribuido, probabilístico y basado en un modelo no parametrizado. Con esta solución se pretende

evitar el punto de fallo único que existiría en caso de dedicar uno de los robots del equipo a la labor del fusionado sensorial. El adjetivo probabilístico hace referencia a que su técnica hace uso de la teoría probabilística para modelar la incertidumbre inherente a este tipo de sistemas. El empleo de modelos no parametrizados son el contrapunto a técnicas de estimación parametrizadas como por ejemplo filtros de Kalman. A pesar de no contar con la eficiencia computacional y simplicidad de la estimación de los modelos paramétricos, permiten modelar un abanico más amplio de situaciones y sistemas más complejos. Los filtros de partículas son un ejemplo de estimadores basados en modelos no paramétricos.

El equipo ISocRob se sirve de un filtro de partículas para mantener una estimación de la pelota tanto en coordenadas globales como en coordenadas locales al robot. La riqueza de la distribución de densidad de probabilidad que logra un filtro de partículas se confronta con el tamaño necesario para comunicar a otro robot esta distribución. Es necesario enviar todas y cada una de las partículas y, en general, suele haber en torno a varios cientos o miles de ellas por robot. Esta limitación ha motivado la idea de crear un resumen de estas partículas, que se aproxime a la distribución original. ISocRob utiliza un modelo de mezcla de Gaussianas (Gaussian Mixture Model o GMM) para aproximar la distribución de partículas por varias Gaussianas parametrizadas con su media, desviación típica y un peso para cada una de ellas. La obtención de estos parámetros se efectúa con un algoritmo EM [Har58, DLR77].

La arquitectura ideada permite mantener en paralelo dos estimadores: Por un lado el estimador local y por otro el estimador cooperativo o de equipo. En cada uno de los filtros se realizan los ciclos de predicción y corrección pero con algunas diferencias.

En el caso del estimador local, se utilizan tanto las observaciones locales como las observaciones procedentes del resto de miembros del equipo. Todas estas observaciones son la entrada del paso de corrección, que únicamente utilizará su modelo de observación local.

El estimador de equipo, por su parte, fusiona directamente las distribuciones de probabilidad obtenidas después del paso de corrección por cada uno de los robots.

En caso de que un robot perciba la pelota siempre utilizará su filtro local. Sin embargo, si el citado robot no es capaz de distinguir la pelota, empleará el estimador de

equipo. En cualquier caso ambas estimaciones se enriquecen empleando información del resto de miembros y, por tanto, se pueden considerar mecanismos de percepción coordinada.

AllemaniACs

La Universidad de Aachen (Alemania) dispone de un grupo de investigación denominado *RoboCup Working Group (AG RoboCup)*. Al hilo de sus investigaciones en robótica participan en tres competiciones diferentes: Liga de tamaño medio, liga *at home* y liga de plataforma estándar. Su equipo de MSL interviene en la competición mundial desde 2002 y también han colaborado a extender el estado del arte en cuanto a técnicas de fusión sensorial para la estimación coordinada de la pelota.

En [FFHL05] presentan su técnica híbrida combinando una rejilla ponderada (*weight grid*) y un filtro de Kalman. La rejilla ponderada es similar a una rejilla o *grid* de ocupación [ME85] pero la suma de todas sus celdas puede ser mayor que 1. Para cada estimación de la pelota se calcula un modelo Gaussiano según el modelo de observación del sensor instalado en cada robot.

Durante la fase de actualización de la rejilla, cada estimación es ponderada según una función similar a la mostrada en la ecuación 2.4. A continuación estos pesos w_i son multiplicados por sus respectivas distribuciones Gaussianas y el resultado de la combinación final es almacenado en las celdas. Esta rejilla conforma la denominada *pelota de referencia*.

Cada una de las estimaciones procedentes de los robots se utiliza de entrada para alimentar un filtro de Kalman. Si la estimación procedente del filtro recae fuera de una circunferencia virtual de $1m.$ de radio centrada en la *pelota de referencia*, el filtro de Kalman es inicializado.

En [WAD$^+$01, DGB01] también se describe otra técnica híbrida que combina localización Markoviana [GBFK98], que permite mantener estimaciones multi-hipótesis, para fusionar las diferentes observaciones; junto con un filtro de Kalman.

2.4. División de tareas en equipos de robots

El segundo problema que se aborda en esta tesis es el reparto de tareas en un equipo de robots. En [Par03] se enumeran gran cantidad de técnicas para abordar el problema de la coordinación en diferentes entornos. Podríamos hacer una distinción entre cooperación débil o cooperación fuerte. La primera únicamente establece que haya algún mecanismo de coordinación al principio de la tarea e incluso si hubiera que volver a planificar los objetivos de la misma. La generación conjunta de mapas y la exploración son algunos ejemplos de coordinación débil. En [GM02] y [DS03] se describen técnicas basadas en mercado y subastas respectivamente dentro del ámbito de la cooperación débil. La coordinación fuerte requiere que haya intercambio continuado de información, ya que la labor ha realizar necesita mayor grado de integración entre los miembros del grupo. El mantenimiento de formaciones en grupos de vehículos es una de las aplicaciones habituales de este tipo de cooperación. En [Par96] y [Ark92] se detallan técnicas basadas en comportamientos y esquemas respectivamente dentro del marco de la cooperación fuerte.

El reparto de tareas o roles que se presenta en esta tesis se enmarca en el ámbito de la cooperación fuerte. Cada miembro del grupo tendrá asignada siempre una tarea que determinará su comportamiento. El reparte de tareas responde al paradigma *divide y vencerás*, por el que un objetivo de mayor complejidad se divide en varios subobjetivos de menor dificultad de consecución. Cada uno de estos subobjetivos se asocia con la tarea de la que será responsable cada robot.

La asignación de roles o tareas puede clasificarse principalmente en dos grupos: Asignación estática o dinámica. La asignación dinámica de roles es un requisito indispensable en aquellos sistemas multi-robot que quieren abordar un problema común en un entorno cambiante. En entornos estáticos es posible asignar roles fijos a los miembros del grupo. Sin embargo, en ambientes donde el dinamismo es mayor necesitamos un mecanismo dinámico de intercambio de tareas para adaptar el equipo a los cambios del entorno. En [LJGM06] se analizan con más detalle las técnicas de asignación dinámica para entornos multi-robot.

A continuación se presenta un estudio de diferentes arquitecturas cooperativas, diseñadas para repartir tareas entre los miembros de un equipo dentro del ámbito de la RoboCup.

2.4.1. GermanTeam

Este potente equipo alemán, referencia durante muchos años en la categoría de plataforma estándar diseñó básicamente una arquitectura cooperativa con cuatro roles de juego: Portero (*goalie*), chutador (*striker*) y dos auxiliares (*supporters*). El portero es el único rol que no puede ser intercambiado con el resto de jugadores, debido a que éstos tienen prohibido el acceso al área de su propio campo. Su portero destacó por su carácter estático, evitando las escapadas fuera del área y otorgando gran importancia a la localización. El objetivo principal del chutador era recuperar la pelota en caso de encontrarse defendiendo y conducirla hasta la portería contraria en caso de situación ofensiva. De los dos jugadores auxiliares, uno de ellos ayuda en labores defensivas situándose por detrás de la pelota y sin sobrepasar su propio campo. El auxiliar restante acompaña al chutador en su labor ofensiva evitando posicionarse en la trayectoria hasta la portería contraria, maximizando la recepción de pases y minimizando el tiempo de recuperación de la pelota en caso de saque de banda.

Los roles de los jugadores de campo pueden ser intercambiados de manera dinámica según la situación del juego. Existe una negociación entre estos tres robots para escoger quién es el chutador, el auxiliar defensivo y el auxiliar ofensivo. Para erigir al chutador, todos los robots comunican al resto cuánto tiempo tardarían en alcanzar la pelota. Este tiempo en milisegundos es calculado a través de la siguiente fórmula:

$$\begin{aligned} TiempoAlcancePelota = & \\ & \frac{DistPelota}{0{,}2} + \\ & 400{,}0 * |Ángulo_{Pelota-PorteríaContraria}| + \\ & 2{,}0 * TiempoSinVerPelota \end{aligned} \tag{2.6}$$

Como podemos observar en la ecuación 2.6, por cada 10cm. de distancia se asume que el robot necesita 500 ms. para alcanzar la pelota. El ángulo que forma la pelota con la portería contraria es multiplicado por 400 ms. De esta manera se beneficia a los robots que están situados por detrás de la pelota sobre los robots que tienen que rotar para encarar la portería contraria. El último término añade una penalización de dos segundos por cada segundo que un robot permanece sin ver la pelota.

Durante el periodo de negociación de los roles, el robot con el menor tiempo estimado para alcanzar la pelota es elegido para el rol de chutador. Para garantizar la estabilidad de la decisión se otorga una bonificación de 500 ms. al robot que previamente ejercía de chutador. Para el rol de auxiliar ofensivo se utiliza la posición que ocupa cada robot en el campo, prefiriendo el más adelantado. De los dos jugadores que quedan sin rol asignado, aquel que tenga mayor valor en la citada coordenada ejercerá de auxiliar ofensivo (esta coordenada indica lo adelantado o retrasado que se encuentra el robot en el campo). Para evitar oscilaciones de roles demasiado rápidas, el robot que previamente actuaba de auxiliar ofensivo dispone de una bonificación extra de 30 cm.

2.4.2. rUNSWift

La Universidad de Nueva Gales del Sur (Australia) ha competido en la liga de plataforma estándar de la RoboCup desde 2002 hasta 2008. Su rendimiento ha sido muy elevado como demuestran sus tres victorias, tres segundos puestos y un tercer puesto en la competición mundial RoboCup.

En [Mor05] describen sus algoritmos de formación y comportamientos estratégicos. Su reparto de tareas sigue el modelo clásico de división de roles, asociando a cada rol una serie de comportamientos y una posición ideal.

Su diseño contempla dos estrategias: *Supporter / Defender* y *Striker / Defender*. La primera de ellas asigna tres roles a los jugadores de campo: *Attacker, supporter y defender*. El rol de *attacker* se encarga de atrapar la pelota, conducirla hasta la portería contraria o robarla en caso de posesión del equipo contrario. El papel de *supporter* se dedica a apoyar al atacante en labores de ataque. Como puede apreciarse en la figura 2.14 (derecha), este rol está muy ligado al de *attacker* y su posición es muy cercana

a éste pero situándose a su espalda. El *defender* mantiene su posición en el campo propio para mitigar un posible contra-ataque. En caso de emplear la estrategia *Striker / Defender* (figura 2.14 izquierda), el rol de *supporter* se reemplaza por el de *striker*, cuya posición en el campo se encuentra en el lado contrario del que ocupe la pelota.

La primera estrategia persigue una mayor agresividad cuando se dispone de la pelota pues se concentran dos jugadores cerca de ella. Es necesario que haya una buena interacción local entre los roles de *attacker* y *supporter* para no estorbarse entre sí. La segunda estrategia favorece el juego ante la tendencia actual de jugar en campos amplios, pues la recuperación de la pelota es más rápida debido a la mejor cobertura del campo.

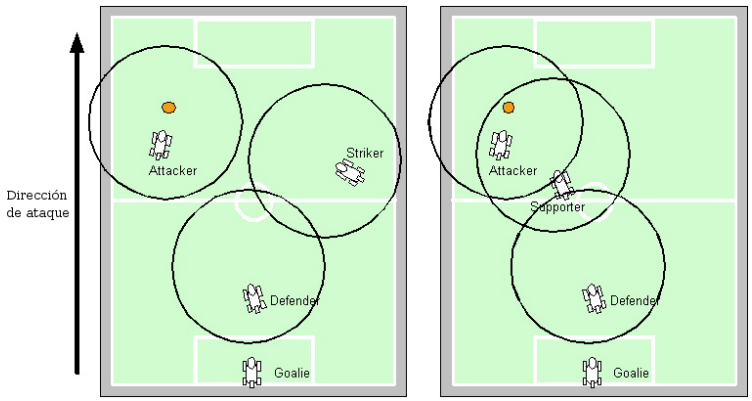


Figura 2.14: Diagrama de posicionamiento de roles. Estrategia *Striker / Defender* (izquierda) y *Supporter / Defender* (derecha) (Foto original de rUNSWift)

El algoritmo de asignación dinámico de roles está basado en dos parámetros fundamentales: Tiempo para llegar hasta la pelota con posición de tiro y tiempo para acceder a una determinada posición del campo. Estos conceptos ideados originalmente por el equipo German Team en [RLB+05] han sido extendidos y modificados por el equipo australiano.

Ahora la ecuación de cálculo de tiempo de alcance de la pelota queda así:

$$
\begin{aligned}
TiempoAlcancePelota = \\
& BonificaciónPorAttacker+ \\
& BonificaciónPorAgarrePelota+ \\
& TiempoGirarHastaPelota+ \\
& TiempoLlegarHastaPelota+ \\
& TiempoGirarConPelotaHastaPortería+ \\
& PenalizaciónObstáculos
\end{aligned}
\tag{2.7}
$$

`BonificaciónPorAttacker` - El jugador que ya dispone de este rol recibe una bonificación en forma de reducción de tiempo.

`BonificaciónPorAgarrePelota` - Si un robot ya está sujetando la pelota, éste no debería abandonar el rol de *attacker* hasta que no soltara la misma. En este caso se reciba una bonificación que garantiza su elección como *attacker*.

`TiempoGirarHastaPelota` - Este parámetro mide el tiempo estimado dedicado a girar sobre sí mismo para encarar la pelota.

`TiempoLlegarHastaPelota` - En función de la distancia hasta la bola, se calcula el tiempo necesario para alcanzar su posición.

`TiempoGirarConPelotaHastaPortería` - Supuesto que se alcanzara la pelota, este término estima el tiempo empleado en girar conduciendo la pelota hasta orientarse mirando a la portería contraria (posición de disparo).

`PenalizaciónObstáculos` - El tiempo aproximado de retardo debido a los obstáculos detectados entre el robot y la pelota.

El rol más prioritario es el de *attacker*, y su asignación se realiza evaluando la ecuación 2.7 en todos los robots. Aquel cuyo tiempo de alcance de la pelota sea menor se alzará con este rol. Para el resto de roles, se evalúa el tiempo de acceso a la posición de cada uno de los roles. Aquel robot que necesite menos tiempo para alcanzar la posición del siguiente rol más prioritario será su propietario.

Otro comportamiento cooperante que puede desplegar runSWIFT es el de búsqueda de la pelota. Periódicamente todos los robots comunican si perciben la pelota o no. Si tras un periodo aproximado de 2 segs. ninguno de los miembros detecta la bola, se establece una nueva formación que dispersa los robots por todo el campo en forma de triángulo equilátero centrado en el centro del campo. Cada uno de los robots se dirige a los vértices de este polígono imaginario maximizando las posibilidades de que alguno encuentre la pelota. La asignación de los puntos a cada robot se realiza de la misma manera que la asignación de roles de juego normal, es decir, se evalúa el tiempo de llegada hasta las posiciones de destino y los más apropiados para cada punto se asignan esa posición.

2.4.3. NUbots

El equipo australiano NUbots compuesto por miembros del laboratorio de robótica de Newcastle (Universidad de Newcastle) ha sido uno de los equipos de referencia de la liga de plataforma estándar de la RoboCup. Entre sus principales logros en la competición podríamos citar las victoria en las competiciones RoboCup 2006 y 2008 (participando como parte del equipo *NUManoid*), 2º clasificado en 2005 y 2007 y 3º clasificado en 2002, 2003 y 2004.

En [Lab06] se presentan los detalles de su arquitectura de comportamientos a nivel de equipo, influenciado por el trabajo de Michael J. Quinlan y Stephan K. CHalup descrito en [QC06].

En su propuesta remarcan una distinción entre roles y posiciones. Consideran estos términos diferentes y únicamente establecen dos roles: interceptor (*chasing*) y posicionador (*positioning*). Este enfoque contrasta con la corriente más utilizada de asociar a cada rol una posición dentro del campo. En [VRC$^+$04] se presenta una implementación que sigue este último modelo.

Si un robot selecciona el rol de interceptor no podrá ser posicionador y si es posicionador no podrá ser interceptor en ese mismo instante. Además, idealmente sólo uno de los robots deberá ser el interceptor, evitando sanciones de tipo *pushing*.

En el software utilizado por NUbots se define una serie de ocho posiciones posibles P que los robots pueden adoptar. Estas posiciones son *Goal Keeper (GK)* o

portero, *Sweeper (SW)* o chutador, *Left Back (LB)* o defensa izquierdo, *Centre Back (CB)* o defensa central, *Right Back (RB)* o defensa derecho, *Left forward (LF)* o delantero izquierdo, *Centre Forward (CF)* o delantero central y *Right Forward (RF)* o delantero derecho. Cada posición $P = \langle x, y, \theta \rangle$ tiene asignada una terna de coordenadas en el campo de juego.

También se maneja el concepto de formación $F = \langle P_a, P_b, P_c, P_d \rangle$ como un vector que contiene cuatro posiciones. Cuando un equipo se encuentre usando una formación específica, sus robots deberán ocupar las posiciones contenidas en el vector.

Existe una última definición para completar la jerarquía de tareas: el término estrategia $S = \langle F_{def}, F_{ofe} \rangle$. Este componente de alto nivel contiene dos formaciones, siendo una de ellas de carácter defensivo y la otra ofensivo.

NUbots dispone de varias estrategias disponibles según las situaciones del juego:

$$S_{normal} = \langle D_{normal}, O_{normal} \rangle$$
$$S_{interceptor} = \langle D_{interceptor}, O_{interceptor} \rangle$$
$$S_{agresivo} = \langle D_{agresivo}, O_{agresivo} \rangle$$
$$S_{ofensivo} = \langle D_{interceptor}, O_{agresivo} \rangle$$
$$S_{defensivo} = \langle D_{agresivo}, O_{normal} \rangle$$
$$S_{mantenimiento} = \langle D_{agresivo}, D_{agresivo} \rangle$$

donde

$$D_{normal} = \langle P_{GK}, P_{LB}, PRB, PCF \rangle, O_{normal} = \langle P_{GK}, P_{CB}, P_{LF}, P_{RF} \rangle$$
$$D_{chutador} = \langle P_{GK}, P_{LB}, PRB, PCF \rangle, O_{chutador} = \langle P_{GK}, P_{CB}, P_{LF}, P_{RF} \rangle$$
$$D_{agresivo} = \langle P_{GK}, P_{LB}, PRB, PCF \rangle, O_{agresivo} = \langle P_{GK}, P_{CB}, P_{LF}, P_{RF} \rangle$$

Una vez descrita la teoría creada por NUbots resta mencionar cómo la ponen en práctica. La asignación de roles se realiza sin utilizar negociaciones, es decir, cada robot toma su propia decisión basada en la información disponible y espera, en general, que la decisión sea coherente con la tomada por el resto de compañeros. Los datos intercambiados contienen la distancia hasta la pelota (ligeramente modificada para tener en cuenta varios parámetros) e información extra sobre las actuaciones que está

realizando cada robot (intento de sujetar la pelota, regatear a un contrincante, disparar a portería, etc.). Esta información extra se utiliza para decidir cuál es el mejor lugar para los robots posicionadores, según la tarea actual del robot interceptor.

El primer paso es seleccionar la mejor formación según la estrategia actual. En la competición RoboCup 2006, NUbots aplicó la siguiente regla:

$$F_{actual} = \begin{cases} D_{actual} & \text{si la pelota está en el campo propio} \\ O_{actual} & \text{si la pelota está en el campo contrario} \end{cases}$$

Según esta regla, si la estrategia actual de juego fuera S_{normal} y la pelota se encontrara en nuestro propio campo, la formación seleccionada sería D_{normal}. Si por el contrario la pelota se encontrara en el campo contrario la formación escogida sería O_{normal}.

Las estrategias también pueden variarse según el marcador actual y el tiempo restante. A continuación mostramos un fragmento de una regla escrita en *Python* para la RoboCup 2006:

```
if (secondHalf and (oppScore > ownScore)):
  scoreDiff = oppScore - ownScore
  if (timeLeft/60.0 < scoreDiff):
    newStrategy = AGGRESSIVE
```

2.4.4. CAMBADA

La arquitectura de CAMBADA, el equipo de la Universidad de Aveiro en Portugal, está diseñada teniendo en cuenta el reparto de tareas entre los diferentes miembros del equipo. Los dos componentes principales que intervienen en la asignación de tareas son los *comportamientos* y los *roles*.

El concepto de comportamiento es similar al de habilidad referido a los humanos. Las personas podemos tener habilidad para caminar, correr, saltar, buscar objetos, identificar caras, comer, beber, hablar, etc. En robótica, el conjunto de acciones (ya sean de actuación o donde intervengan los sensores) que culmina en el desarrollo de una conducta concreta recibe el nombre de comportamiento. Así, podemos encontrar

comportamientos robóticos para navegar hasta una posición (x, y, θ), identificar una pelota, golpearla, driblar a un contrario, etc.

La idea de rol define un componente de más alto nivel que el comportamiento y establece el conjunto de comportamientos a desplegar en un instante determinado. Por ejemplo, el rol de portero tratará de evitar movimientos demasiado bruscos que puedan alejar al robot de la portería. Sin embargo, un hipotético rol de chutador exigirá mayor dinamismo para alcanzar la pelota lo más rápido posible.

La asignación de roles en CAMBADA [LLC08] está basada en las posiciones absolutas de cada robot en el campo. Estas posiciones no pueden ser directamente observadas por otros robots, es decir, el comportamiento de detección de congénere o compañero no está implementado. Su cálculo es realizado por el algoritmo de auto-localización que ejecuta en cada robot y comunicado al resto utilizando la infraestructura de comunicaciones.

Como es habitual, el rol de portero se asigna de manera fija a uno de los robots. Esta decisión está aún más justificada en la liga MSL, pues el citado robot tiene una configuración *hardware* ligeramente diferente porque está permitido que amplíe su envergadura y altura de manera temporal (simulando una parada).

Los roles para los jugadores de campo se asignan según la estrategia de juego seleccionada. Cada estrategia establece una posición base para cada robot especificada en un fichero de texto. Con el fin de que la formación no sea completamente estática, sino que varíe según la situación de la pelota, estas posiciones son modificadas por fuerzas atractivas hacia la bola.

Para decidir qué rol de juego es asignado a cada robot se comienza calculando las distancias de cada robot a todos las posiciones base especificadas por la estrategia actual. De manera iterativa, se comienza asignando el rol más prioritario (*chutador*) al robot cuya distancia a la posición base de mayor prioridad (en este caso sería la pelota) sea menor. Una vez establecido el rol de mayor precedencia se continúa con el siguiente, y así sucesivamente hasta asignar todos.

Esta solución permite que los roles más importantes sean siempre asociados a algún robot, incluso en casos de inferioridad numérica provocada por alguna sanción o daño en algún robot.

2.4.5. CS Freiburg

La universidad de Freiburg (Alemania) participó en la categoría F2000 (precursora de la actual MSL) de la RoboCup desde el año 1998 hasta el 2002. En tres ocasiones fueron campeones mundiales, además de obtener buenos resultados en su campeonato nacional.

Según sus propias palabras, su título del año 2000 se pudo atribuir a su efectivo mecanismo de cooperación [WAD+01, DGB01] entre los robots, basado en un robusto y preciso algoritmo de localización y en un sofisticado conjunto de comportamientos.

Las unidades de coordinación utilizadas son los roles. En este caso encontramos tres papeles diferentes para cada jugador de campo: *active*, *support* y *strategic*. El primero de ellos se centrará en obtener la pelota, el rol de *support* tratará de asistir al *active* y el último rol ayudará en labores defensivas.

El mecanismo de asignación de roles se basa en combinar funciones de utilidad e intencionalidad. Periódicamente, cada robot comunica al resto cuál es el rol que está ejecutando y cuál es que le gustaría obtener tan pronto como fuera posible. Dos roles sólo pueden ser intercambiados si sus respectivos robots tienen los mejores valores de utilidad para los futuros roles y además, ambos tienen intención de cambiar al nuevo rol.

El posicionamiento de cada rol en el campo es dinámico y se tienen en cuenta factores como las posiciones desde las que se ve la pelota, la posición de ésta, el propio rol, los obstáculos cercanos, etc. Todos estos factores son combinados mediante un campo de potencial, similar al método SPAR creado por la Universidad de Carnegie Mellon para la liga de robots de pequeño tamaño [SVR99].

2.5. Discusión

A lo largo de este capítulo se ha escrito el estado del arte actual de la robótica cooperativa y de los dos problemas dentro de ella abordados en esta tesis. Se ha establecido una definición formal de los sistemas multi-robot y hemos presentado una taxonomía según las opiniones y trabajos de numerosos autores. Los dos proble-

mas que afrontamos en esta tesis son el desarrollo de técnicas de cooperación para el seguimiento y localización de un objeto dinámico, junto con el reparto de tareas alrededor de la competición RoboCup.

A la vista de la taxonomía descrita podríamos enmarcar nuestro problema como un problema de control distribuido. Cada robot es completamente autónomo y además no se necesita ningún componente externo ajeno a los propios miembros del equipo. Se podría decir que la inteligencia del comportamiento que queremos obtener debe salir de la propia comunidad y no de la decisión de un agente externo.

Nuestro entorno de trabajo, el definido por la liga de plataforma estándar de la RoboCup, asegura que los robots disponen del mismo *hardware*. Sin embargo, nuestro aporte para la división de tareas convierte el equipo en heterogéneo. Esto es así debido a que cada uno de los robots ejecutará una serie de comportamientos guiado por un rol diferente, a criterio de la estrategia de asignación de roles que se presentará en el capítulo 4.

Nuestro nivel de coordinación se basa en comunicación explícita, es decir, se utilizan las capacidades de red de los robots para enviar y recibir información. Esta información incluye las intenciones, creencias e incluso las acciones que cada miembro está realizando. Podríamos clasificar nuestra sociedad dentro del término *coexistencia inteligente*. Cada robot conoce la existencia de sus compañeros y las comunicaciones son el medio de diálogo entre nuestros algoritmos de coordinación, tanto en la parte de percepción distribuida como en el ámbito del reparto de roles.

Las interferencias y conflictos no están excluidas de nuestro entorno. Es necesario evitar que, por ejemplo, se produzcan gran cantidad de oscilaciones entre los diferentes roles que eviten el desarrollo normal de los comportamientos. Es deseable que los algoritmos de estimación coordinada de la pelota puedan soportar falsos positivos de algún robot o incluso medidas erróneas. En los capítulos 3 y 4 se presentarán las soluciones que evitan estos problemas asociados a las múltiples fuentes de información.

En la sección 2.2 hemos recorrido las aplicaciones más importantes de los sistemas multi-robot. Las aplicaciones de recolección tienen gran parecido con el objetivo principal de las competiciones robóticas: Capturar una pelota. La solución natural de dividir la búsqueda entre diferentes robots asignando a cada rol una zona de acción

es el enfoque habitual del reparto de tareas en la RoboCup. Nuestra aproximación en este campo tiene una innegable influencia procedente de este tipo de aplicaciones.

El control de formaciones es otra corriente de trabajo habitual dentro de la rama de robótica distribuida. En nuestro caso, los robots utilizan una formación dinámica en función de los parámetros de juego (posición de la pelota, posición de los robots dentro del campo, etc.) y de los roles de cada robot. El posicionamiento de cada robot y la necesidad de evitar choques entre los miembros son aspectos a resolver tratados por los trabajos sobre formaciones presentados en este capítulo.

En las aplicaciones de gestión de almacenamiento uno de los principales problemas a resolver es el acceso a un recurso compartido. Nuestro problema también puede verse en cierta manera como un ejercicio de acceso a un medio común: el campo de juego. La aproximación hacia la pelota también debe estar protegida, pues el comportamiento de *patio de colegio* (todos como locos detrás de la pelota) debe ser completamente evitado.

La coordinación de actuadores también aparece en este trabajo. Los efectores principales del robot aiBo son sus cuatro patas articuladas. Cada una de ellas dispone de tres articulaciones gobernadas por motores y todos ellos deben ser coordinados para que una locomoción estable y paramétrica emerja. El robot Nao también requiere un control muy preciso de sus articulaciones. Las articulaciones del cuello deben combinarse de manera concisa para que se realice convenientemente el seguimiento de la pelota. Las articulaciones del resto del cuerpo deben estar en armonía total para lograr una locomoción bípeda estable. Todos estos aspectos han sido abordados en las arquitecturas de TeamChaos y BICA y, aunque no son el objetivo principal de esta tesis, han tenido que ser resueltos para poder realizar los experimentos aquí presentados.

La sección 2.3 repasa el problema de la percepción distribuida dentro del marco de la RoboCup. Estos trabajos plantean diversas soluciones, que como nuestra propuesta, tratan de resolver el problema de la estimación de la pelota utilizando múltiples fuentes de información. A la vista de la heterogeneidad de las soluciones no parece existir un denominador común dentro de las técnicas escogidas. Existen trabajos que emplean filtros de Kalman, métodos de Monte Carlo, métodos analíticos, técnicas que discretizan el entorno, etc. Sin embargo, todos estos métodos se enmar-

can dentro de la familia de algoritmos probabilísticos, verdadero dominante de las propuestas. Un aspecto muy importante a destacar es que muchos de estos trabajos no tienen en cuenta la incertidumbre asociada a la posición del observador. Este hecho complica mucho la estimación de la pelota pero, en nuestra opinión, asumir un error despreciable es una relajación demasiado fuerte y alejada de la realidad.

El capítulo 3 aborda el problema de la estimación coordinada de la pelota bajo diferentes técnicas: Primero se propone una solución analítica, a continuación un método basado en rejillas de probabilidad y finalmente dos métodos basados en filtros de Kalman y métodos de Monte Carlo. La incertidumbre tanto de la observación como del observador serán integradas en nuestros algoritmos y la comparativa y resultados obtenidos serán presentados en la parte final del capítulo.

La sección 2.4 permite conocer cuáles son los mecanismos habituales para repartir tareas dentro de un equipo de RoboCup. Hemos expuesto los conceptos de comportamiento, rol, utilidad, asignación dinámica, etc. habituales dentro de este campo. Nuestro aporte presentado en el capítulo 4 está muy influenciado por todos estos trabajos y mezcla conceptos de casi todos los equipos mostrados. Uno de los requerimientos necesarios es que la asignación de roles se realice en tiempo real y de manera sencilla. Hemos descartado el empleo de negociaciones y hemos abordado el problema de manera similar al equipo NUbots. El concepto de utilidad también ha sido adoptado por nuestra arquitectura de cooperación, para medir el grado de adaptabilidad de cada robot a los roles. En prácticamente todos los trabajos estudiados, la función de utilidad evalúa diferentes parámetros que siempre son locales a cada robot. Por ejemplo, distancia observada de la pelota al robot, posición en el campo, etc. Nuestro aporte en este campo busca utilizar un parámetro como es la distancia hasta la pelota, obtenido como resultado de mantener una estimación distribuida. Las ventajas potenciales serán la mayor estabilidad de los parámetros para la asignación de roles y, por tanto, la mayor robustez de funcionamiento ante falsos positivos, medidas erróneas, etc.

CAPÍTULO 3

Estimación coordinada de objetos dinámicos

Do not expect to arrive at certainty in every subject which you pursue. There are a hundred things wherein we mortals... must be content with probability, where our best light and reasoning will reach no farther.

– Isaac Watts

En el capítulo anterior hemos repasado el estado del arte referente a los sistemas multi-robot, así como sus aplicaciones. Hemos descrito sus características principales, así como las aplicaciones más habituales que involucran a este tipo de robots. Dentro de las posibilidades existentes para coordinar un equipo de robots nuestra propuesta es combinar percepción y acción para enriquecer el comportamiento del equipo.

En este capítulo se describe el aporte cooperativo al problema perceptivo de la estimación de la pelota en el entorno de la RoboCup. Nuestro equipo de robots cuenta con cuatro miembros, tres de ellos son jugadores de campo y uno de ellos es el portero. Cada uno de los componentes del grupo mantendrá una estimación absoluta de la pelota, que dependerá de si ésta es directamente visible por el robot o no y

de la calidad de su propia estimación sobre la posición en el campo de juego (autolocalización). Con el fin de componer una estimación compartida común al equipo deberemos comunicar entre todos los miembros la estimación absoluta de la pelota. En este capítulo emplearemos varias técnicas de fusión de información para combinar las estimaciones de cada robot en una única estimación compartida.

En la sección 3.1 se exponen los fundamentos teóricos que son la base para la técnica aportada. Esta base teórica está formada, en esencia, por las técnicas actuales de fusión sensorial probabilística. Los aspectos más representativos de los filtros Bayesianos, rejillas probabilísticas, filtros de Kalman y filtros de partículas serán presentados en esta sección.

En la sección 3.2 se detallan las diferentes técnicas estudiadas y propuestas para afrontar el problema de la estimación de un objeto dinámico por un equipo de robots. La propuesta en este ámbito no ha sido única, se han estudiado, diseñado e implementado diferentes alternativas basadas en métodos analíticos, rejillas de probabilidad, filtros de Kalman o filtros de partículas. Cada una de estas técnicas está respaldada por una serie de experimentos para caracterizar su comportamiento, validando la calidad obtenida en la estimación, su estabilidad, etc. En la sección 3.3 se analizan las pruebas realizadas en los robots aiBo y Nao en el entorno de la RoboCup.

Las conclusiones a los métodos propuestos, así como las diferencias encontradas se exponen en la sección 3.4. Esta última sección sirve de discusión sobre los métodos presentados, problemas encontrados y principales aportes.

3.1. Fundamentos teóricos

3.1.1. Filtros Bayesianos

En este capítulo aparecerá en numerosas ocasiones la palabra filtro, comenzaremos por realizar una definición de este término. Un filtro tiene como función principal mantener una estimación del estado de un sistema, que evoluciona en el tiempo y además no es directamente observable. Las entradas del estimador serán muestras u observaciones periódicas obtenidas gracias a uno o varios sensores y que permitirán observar el estado del sistema de manera indirecta y aproximada. Las principales

aplicaciones de los filtros en el ámbito de la ingeniería, control, robótica, etc. son la estimación de la posición de objetos (*tracking*) y la estimación de la propia posición en el entorno (*self-localization*). En nuestro caso, dedicaremos el estimador para conocer con la mayor precisión y robustez posible la posición de la pelota dentro del campo de juego.

El problema general del filtrado puede formularse de manera Bayesiana. Este matiz es importante porque permite mantener una representación común para un amplio abanico de problemas de fusión sensorial tanto discretos como continuos. La teoría de Bayes posibilita representar y acumular conocimiento parcial a través de probabilidad. Además, el estimador Bayesiano es óptimo porque minimiza el error de estimación.

Denominaremos x_t al valor del estado que pretendemos estimar en el instante t. Este estado podría describir la posición y velocidad de un determinado objeto a seguir, la posición de un robot o el estado de cualquier proceso a monitorizar. En un instante de tiempo k, definiremos los siguientes términos:

x_k**:** El vector de estado a estimar en el instante k. Es un vector porque puede contener varios parámetros a estimar (posición, velocidad, orientación, etc.).

u_k**:** El vector de control es conocido y aplicado al sistema en el instante $k-1$ para conducir al mismo del estado x_{k-1} al estado x_k. Este vector de control se utiliza para modelar la dinámica del sistema, es decir, cuál será el estado ante una entrada determinada de control $p(x_k|x_k, u_k)$.

z_k**:** Representa la observación tomada del estado x_k en el instante k. Lo habitual es que el estado x_k no sea observable directamente, sino que haya que inferirlo en función de la información que se tiene hasta el momento y las medidas que nos proporcionan los sensores (con sus imprecisiones, incertidumbres, etc.) sobre el estado actual del sistema. Esta observación será empleada para modificar la estimación sobre el estado del sistema $p(x_k|x_k, zk)$.

También vamos a definir los siguientes conjuntos:

$X^k = x_0, x_1, ..., x_k = X^{k-1}, x_k$**:** La historia de todos los estados del sistema.

$U^k = u_1, u_2, ..., u_k = U^{k-1}, u_k$**:** El histórico de todas las entradas de control.

$Z^k = z_1, z_2, ..., z_k = Z^{k-1}, z_k$**:** El conjunto de todas las observaciones tomadas en el tiempo.

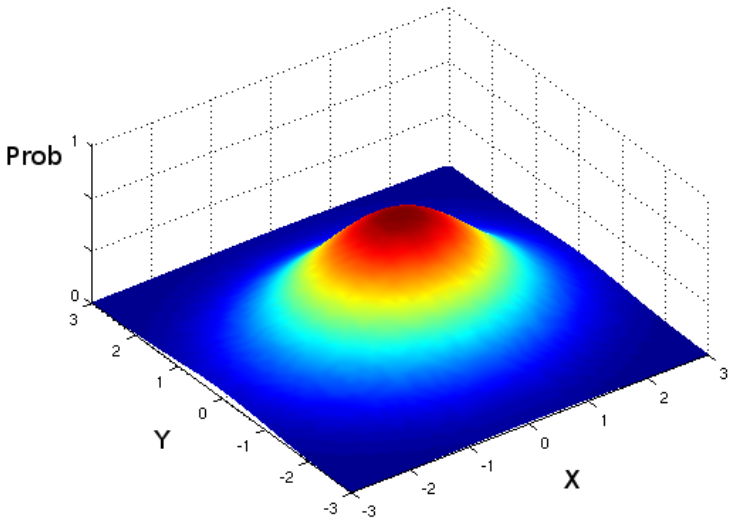


Figura 3.1: Ejemplo de función de densidad de probabilidad bidimensional para expresar la posición de un objeto en un plano

El problema de la estimación consiste en encontrar la función de densidad de probabilidad o *posterior* dadas un conjunto de observaciones y entradas de control a lo largo del tiempo y conocido el estado inicial x_0. La figura 3.1 ilustra un ejemplo de FDP bidimensional.

$$P(x_k|Z^k, U^k, x_0) \tag{3.1}$$

La ecuación 3.1 puede reescribirse utilizando la regla de Bayes, incluyendo el modelo de sensor $P(z_k|x_k)$ y el modelo de predicción de la función de densidad de probabilidad $P(x_k|Z^{k-1}, U^k, x_0)$ basado en las observaciones hasta el instante $k-1$.

$$P(x_k|Z^k, U^k, x_0) = \frac{P(z_k|x_k)P(x_k|Z^{k-1}, U^k, x_0)}{P(z_k|Z^{k-1}, U^k)} \tag{3.2}$$

Es importante resaltar que el proceso de evolución sigue un proceso de Markov de orden uno, por el que el estado del sistema resume toda la historia pasada. Esta asunción implica que $P(x_k)$ es suficiente para representar todos los estados anteriores del sistema $P(x_{0:k})$.

El denominador de la ecuación 3.2 es independiente del estado en el que se encuentre el sistema. El modelo de sensor asume que las observaciones son independientes, es decir:

$$P(z_1, \cdots, z_n|x) = P(z_1|x) \cdots P(z_n|x) = \prod_{i=1}^{n} P(z_i|x) \tag{3.3}$$

Usando el teorema de la probabilidad total podemos reescribir el segundo término del numerador de la ecuación 3.2 como:

$$\begin{aligned} & P(x_k|Z^{k-1}, U^k, x_0) \\ &= \int P(x_k, x_{k-1}|Z^{k-1}, U^k, x_0)dx_{k-1} \\ &= \int P(x_k|x_{k-1}, Z^{k-1}, U^k, x_0)P(x_{k-1}|Z^{k-1}, U^k, x_0)dx_{k-1} \\ &= \int P(x_k|x_{k-1}, u_k)P(x_{k-1}|Z^{k-1}, U^{k-1}, x_0)dx_{k-1} \end{aligned} \tag{3.4}$$

La última igualdad de la ecuación 3.4 implica que el futuro estado únicamente depende del estado actual y de la entrada de control efectuada en este momento. El modelo de transición entre estados también toma la forma de distribución de probabilidad $P(x_k|x_{k-1}, u_k)$.

3.1.2. Rejillas probabilísticas

Las rejillas probabilísticas se definen como una particularización de los filtros Bayesianos. El espacio de estados se discretiza en una rejilla con celdas del mismo tamaño. Para el caso de seguimiento de objetos $P(x_{ij})$ representa la probabilidad de que el objeto se encuentre en la casilla (i, j). Además, la suma de las probabilidades de todas las celdas debe ser igual a 1. La rejilla probabilística representa una FDP dis-

cretizada. La imagen 3.2 muestra una rejilla probabilística bidimensional estimando la posición de un objeto. Las zonas más oscuras representan posiciones más probables que las celdas claras.

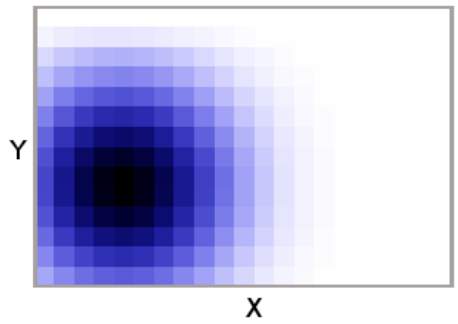


Figura 3.2: Rejilla probabilística de dos dimensiones para estimar la posición de un objeto

El procedimiento para incorporar una observación a la estimación requiere de un modelo de observación, que relacione el estado y las observaciones $P(z|x_{ij}) = \Lambda(x_{ij})$. La ecuación 3.5 describe el paso de corrección a aplicar en todas las casillas de la rejilla.

$$P^{+}(x_{ij}) = C \cdot \Lambda(x_{ij})P(x_{ij}), \forall i, j, \tag{3.5}$$

donde C representa una constante de normalización obtenida sumando la probabilidad de todas las celdas.

Esta operación puede resultar altamente ineficiente, especialmente para el caso de más de dos dimensiones. Sin embargo, una de las ventajas de esta técnica es que resulta relativamente fácil incorporar información a priori. Por ejemplo, en la RoboCup tras una fuera de banda la pelota siempre es situada por el árbitro en una línea imaginaria paralela a la línea de banda y a unos 20 cm. de ella. Si utilizásemos una rejilla probabilística para estimar la posición de la pelota tras un saque de banda, antes de obtener ninguna observación, ya podríamos asignar un valor de probabilidad mínimo a todas aquellas celdas que no estuvieran sobre las líneas imaginarias mencionadas de cada banda.

La técnica de fusión de información basada en rejilla de probabilidad está especialmente indicada para espacios de estados de pocas dimensiones y de reducido tamaño.

3.1.3. Filtro de Kalman

Rudolph Emil Kalman publicó en 1960 su trabajo[Kal60] describiendo el famoso filtro que heredó su apellido y que ha sido la técnica estrella en numerosas aplicaciones de control y estimación hasta la fecha. Un filtro de Kalman es un algoritmo de procesamiento de datos recursivo y óptimo, pues minimiza el error en la estimación. Es una particularización del estimador Bayesiano cuando las FDPs que intervienen son Gaussianas y utilizamos modelos lineales de observación y dinámica del sistema. En esta sección vamos a resumir sus principales características. Para profundizar más sobre esta técnica se puede consultar [WB04] y [TMD$^+$06].

El adjetivo *recursivo* de la definición de filtro de Kalman indica que no es necesario almacenar todos los datos recibidos hasta la fecha y utilizarlos para procesar una nueva observación. Toda esta información anterior queda representada por el estado del sistema en el instante anterior al que se encuentra en ese momento.

Como versa la definición del filtro de Kalman, este estimador es óptimo. Dados una serie de requisitos que veremos a continuación, el filtro de Kalman es capaz de estimar el estado de un determinado proceso modelando dicho sistema, conociendo su dinámica, sabiendo cuáles son los errores e incertidumbre en las observaciones y la dinámica del sistema, así como el punto de partida o estado inicial. El término óptimo hace referencia a que se minimiza el error medio cometido en la estimación comparado con cualquier otra técnica existente.

Todo filtro de Kalman requiere que el sistema pueda ser descrito mediante un modelo lineal y además, tanto el sistema como sus observaciones deben ser Gaussianas y contener *ruido blanco*. Los sistemas lineales son más adecuados para su manipulación con las herramientas ingenieriles que disponemos. Además, la teoría de sistemas lineales es más completa y práctica que la de sus hermanos no lineales. Por supuesto, no todos los fenómenos pueden ser modelados como sistemas lineales,

de ahí que sea una de las restricciones de este filtro. No obstante, existen herramientas matemáticas para aproximar un sistema no lineal a otro lineal en caso necesario.

El ruido blanco implica que los errores no están correlados en el tiempo. Es decir, dado un valor conocido de ruido en un instante determinado no implica ninguna información sobre el ruido que existirá en otro instante posterior.

A continuación se va a utilizar un ejemplo extraído de [May79], para ilustrar el funcionamiento del filtro de Kalman:

Supongamos que estamos perdidos en el mar, de noche y no tenemos idea de nuestra posición. Usaremos la posición de una estrella para predecir nuestra posición. Para simplificar el ejemplo supondremos que nuestra posición buscada es unidimensional. En el instante t_1 determinas tu posible posición a través de la primera observación z_1. Sin embargo, debido a la imprecisión de los instrumentos utilizados, del error humano y de que el método empleado no está exento de errores, el resultado de tu medida no es preciso. Podríamos expresar la incertidumbre de la medida tomada usando el operador estadístico desviación típica σ_{z_1} (o el equivalente varianza $\sigma^2_{z_1}$).

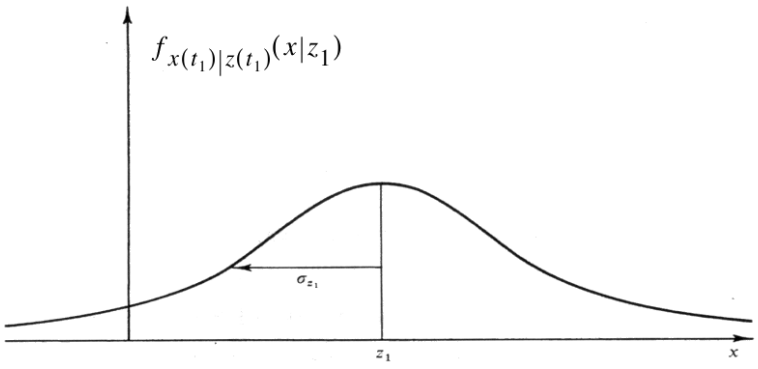


Figura 3.3: Función de densidad de probabilidad para expresar la posición condicionada a la observación z_1

Gracias a esta observación podríamos definir la probabilidad de que tu posición fuera $x(t_1)$ en el instante t_1 condicionada al valor de la observación z_1 como muestra la figura 3.3. Esta gráfica representa la probabilidad de estar en cualquier posición, dada la observación que tomaste. σ_{z_1} es una medida directa de la incertidumbre: cuando mayor sea su valor más ancha será la función de densidad de probabilidad y, por tanto, se repartirá entre más posiciones la probabilidad de estar en una determina-

da posición. En el caso de las funciones de densidad de probabilidad Gaussianas, el 68.3 % de la densidad de probabilidad se concentra dentro de la banda que delimita σ_{z_1}) unidades a ambos lados de la media.

A la vista de la función de densidad de probabilidad de la figura 3.3, la mejor estimación de la posición es:

$$\hat{x}(t_1) = z_1 \tag{3.6}$$

A su vez, la varianza del error en la estimación es:

$$\sigma_x^2(t_1) = \sigma_{z_1}^2 \tag{3.7}$$

Ahora supongamos que un marinero con más experiencia toma una medida independiente a la nuestra inmediatamente después a la tomada por nosotros, es decir, $t2 \cong t1$ por lo que la posición a estimar no habrá cambiado apenas. Este marinero obtiene una segunda observación z_2 con varianza σ_{z_2}. Debido a su mayor experiencia asumiremos que su varianza es menor que la nuestra. La figura 3.4 muestra la función de densidad de probabilidad de la posición estimada basada únicamente en la observación z_2. Obsérvese el mayor pico en la curva, representando la mayor precisión de la estimación.

En este momento disponemos de dos observaciones para estimar nuestra posición. ¿De qué manera podemos combinarlas? La función de densidad de probabilidad para expresar la posición condicionada a las dos observaciones anteriores es una función Gaussiana con media μ y varianza σ^2 como podemos apreciar en la figura 3.5 con:

$$\mu = [\sigma_{z_2}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_1 + [\sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_2 \tag{3.8}$$

$$1/\sigma^2 = (1/\sigma_{z_1}^2) + (1/\sigma_{z_2}^2) \tag{3.9}$$

Un dato interesante es que σ es menor tanto de σ_{z_1} como de σ_{z_2}, por lo que la incertidumbre de la estimación se ha reducido combinando las dos fuentes de in-

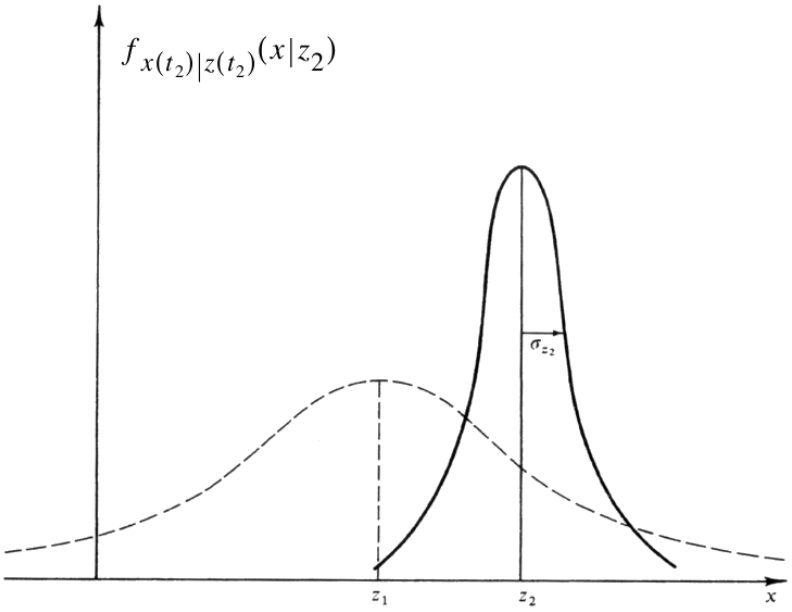


Figura 3.4: Función de densidad de probabilidad para expresar la posición condicionada a la observación z_2

formación. Dada la función de densidad de probabilidad de la figura 3.5, la mejor estimación es:

$$\hat{x}(t_2) = \mu \tag{3.10}$$

con una varianza de σ^2. Esta estimación es óptima, es decir, es lo mejor que se puede obtener bajo cualquier criterio razonable. La ecuación 3.8 parece tener sentido e intuitivamente nos muestra que si σ_{z_1} es igual a σ_{z_2}, la estimación obtenida será el resultado de calcular la media aritmética sobre ambas observaciones. Si alguna de las observaciones tiene mayor precisión, la ecuación otorgará mayor peso a la misma en la estimación final.

La ecuación 3.8 puede reescribirse como:

$$\hat{x}(t_2) = [\sigma_{z_2}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_1 + [\sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)]z_2 = z_1 + [\sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2)][z_2 - z_1] \tag{3.11}$$

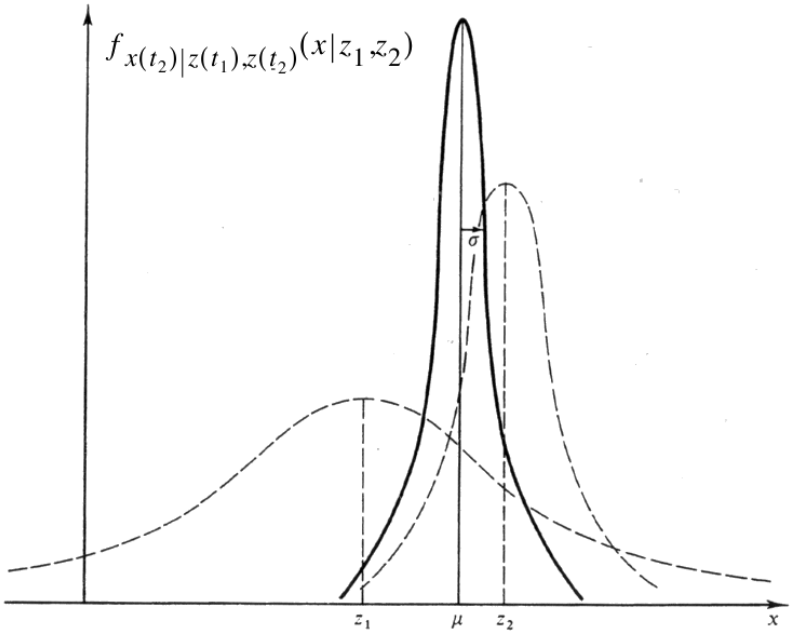


Figura 3.5: Función de densidad de probabilidad para expresar la posición condicionada a las observaciones z_1 y z_2

o en la forma en que se emplea en la mayoría de implementaciones del filtro de Kalman (teniendo en cuenta que $\hat{x}(t_1) = z_1$:

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)] \tag{3.12}$$

donde

$$K(t_2) = \sigma_{z_1}^2/(\sigma_{z_1}^2 + \sigma_{z_2}^2) \tag{3.13}$$

Estas ecuaciones nos indican que la mejor estimación $\hat{x}(t_2)$ de la posición en el instante t_2 es igual a la mejor *predicción* $\hat{x}(t_1)$ de su valor antes de disponer de la observación z_2, más un término de *corrección* que expresa un factor de ponderación multiplicado por la diferencia entre la nueva observación z_2 y la mejor predicción tenida hasta el momento $\hat{x}(t_1)$. Aquí se puede apreciar la estructura de predicción-corrección de este filtro.

Usando $K(t_2)$ de la ecuación 3.13, la fórmula de la varianza mostrada en 3.9 puede escribirse así:

$$\sigma_x^2(t_2) = \sigma_x^2(t_1) - K(t_2)\sigma_x^2(t_1) \tag{3.14}$$

Los valores de $K(t_2)$ y $\sigma_x^2(t_2)$ de las ecuaciones 3.13 y 3.13 engloban toda la información posible sobre la estimación de $\hat{x}(t_2)$ dadas las observaciones z_1 y z_2, es decir $f_{x(t_2)|z(t_1),z(t_2)}(x|z_1, z_2)$. Se podría decir que la estimación queda resuelta para sistemas estáticos, veamos como añadir dinámica al problema.

Volviendo a nuestro ejemplo supongamos que nos movemos antes de tomar una nueva observación. Asumiremos que el mejor modelo que tenemos de nuestro movimiento es

$$dx/dt = u + w \tag{3.15}$$

donde *u* representa la velocidad ordenada al motor de la barca y *w* es el valor del ruido de actuación que modela el desconocimiento exacto sobre nuestra velocidad. El ruido *w* será modelado como ruido blanco Gaussiano de media cero y varianza σ_w^2.

La figura 3.6 muestra cómo se desplaza la función de densidad de probabilidad que expresa la posición, dadas las observaciones z_1 y z_2. En el instante z_2 la función se encuentra como la derivamos anteriormente. A medida que el tiempo avanza, la función se desplaza a lo largo del eje *x* a velocidad *u* y se aplana sobre su media. Por tanto, la función de densidad de probabilidad comienza con la mejor estimación conocida, se mueve según el modelo de dinámica del sistema y se aplana con el tiempo debido a que cada vez estamos menos seguros de nuestra posición si no obtenemos una nueva observación. En el instante t_3^-, justo antes de que una nueva observación se tome en el instante t_3, la función de densidad de probabilidad $f_{x(t_3)|z(t_1),z(t_2)}(x|z_1, z_2)$ es la mostrada en la figura 3.6 y puede ser expresada matemáticamente como una función de densidad de probabilidad Gaussiana con media y varianza dadas por:

$$\hat{x}(t_3^-) = \hat{x}(t_2) + u[t_3 - t_2] \tag{3.16}$$

$$\sigma_x^2(t_3^-) = \sigma_x^2(t_2) + \sigma_w^2[t_3 - t_2] \tag{3.17}$$

Por tanto, $\hat{x}(t_3^-)$ es la mejor predicción sobre el valor de la posición en el instante t_3^-. La varianza $\sigma_x^2(t_3^-)$ es la varianza esperada de la predicción.

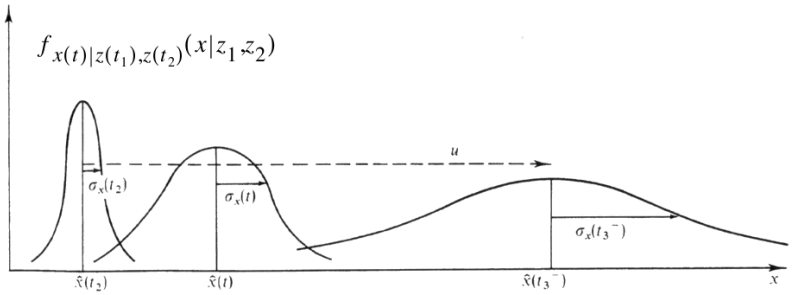


Figura 3.6: Propagación de la función de densidad de probabilidad en el tiempo

A continuación se realiza una nueva medición z_3 con varianza $\sigma_{z_3}^2$. Como sucedía anteriormente nos encontramos con dos funciones de densidad de probabilidad Gaussianas disponibles que proporcionan información. Una de ellas representa toda la información hasta el momento de la nueva observación y la otra representa la información ofrecida por la propia observación. Usando el mismo procedimiento que con anterioridad, la función con media $\hat{x}(t_3^-)$ y varianza $\sigma_x^2(t_3^-)$ será combinada con la función con media z_3 y varianza $\sigma_{z_3}^2$ para obtener una nueva función Gaussiana con media:

$$\hat{x}(t_3) = \hat{x}(t_3^-) + K(t_3)[z_3 - \hat{x}(t_3^-)] \tag{3.18}$$

y varianza:

$$\sigma_x^2(t_3) = \sigma_x^2(t_3^-) - K(t_3)\sigma_x^2(t_3^-) \tag{3.19}$$

donde la ganancia $K(t_3)$ viene dada por:

$$K(t_3) = \sigma_x^2(t_3^-)/[\sigma_x^2(t_3^-) + \sigma_{z_3}^2] \tag{3.20}$$

La estimación $\hat{x}(t_3)$ satisface la mismas ecuación vista en 3.12. La mejor predicción que se tenía antes de la observación z_3 es corregida por un factor de ponderación tantas veces como indica la diferencia entre z_3 y la predicción de su valor. De la misma manera, la varianza y la ganancia son exactamente iguales que las de las ecuaciones 3.14 y 3.13.

3.1.4. Filtro de partículas

Un filtro de partículas es una implementación alternativa no paramétrica de un filtro Bayesiano. Los filtros de partículas aproximan una función de densidad de probabilidad por un conjunto de muestras. Estas muestras o partículas son extraídas de la propia distribución. La aproximación empleada, al no ser parametrizada, permite modelar una mayor cantidad de funciones de densidad de probabilidad que por ejemplo una aproximación basada en Gaussianas.

Una de las principales limitaciones del filtro de Kalman es su carácter unimodal, es decir, sólo puede mantener una única hipótesis de la estimación. La causa de esta limitación es el modelo empleado para representar la densidad de probabilidad, un modelo Gaussiano. Los filtros de partículas son una alternativa que solventan en parte esta limitación. La figura 3.7 ilustra esta idea. La familia de filtros de partículas es amplia, en esta sección nos centraremos en introducir una de las variantes más utilizadas: el filtro de condensación [IB98].

Puesto que el problema a abordar es el de la fusión de información entre varios robots, empleando filtros de partículas podremos obtener una FDP muestreada para representar la posición de la pelota. La diferencia con los métodos paramétricos radica fundamentalmente en que tanto los estimadores individuales en cada robot, como el estimador compartido, podrán representar FDPs más ricas, con múltiples hipótesis y más heterogéneas.

Las muestras o partículas de la distribución a modelar se pueden expresar así:

$$X_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{3.21}$$

Cada partícula $x_t^{[m]}$, con $(1 \leq m \leq M)$, representa una posible estimación de todo el espacio de estados en el instante *t*. *M* expresa el número de partículas que

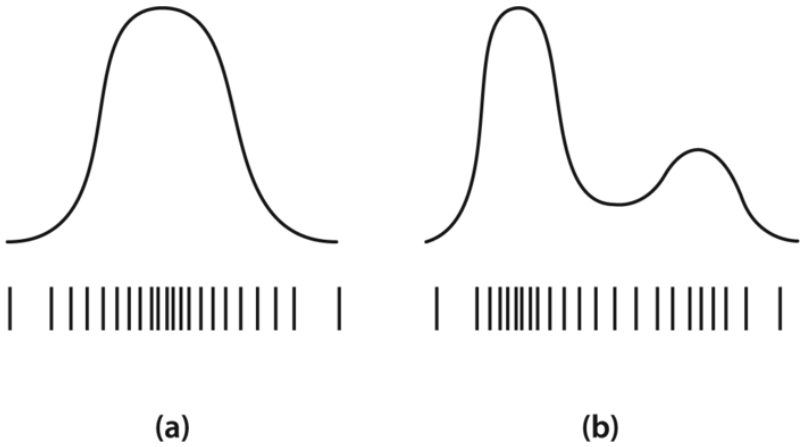


Figura 3.7: Función de densidad de probabilidad (a) que puede ser representada mediante una función Gaussiana parametrizable mediante su media y desviación típica y función de densidad de probabilidad (b) que no puede representarse así. En ambos casos puede usarse una aproximación para representar las funciones formada por un conjunto de partículas cuyos pesos representan la densidad de probabilidad

hay en el conjunto. Lo habitual es que el número de partículas *M* sea suficiente para mantener la representatividad.

La intuición del funcionamiento de los filtros de partículas es que aproximan una función de densidad de probabilidad o creencia sobre un determinado estado $prob(x_t)$ como un conjunto de partículas X_t. Se denomina *conjunto muestral* al conjunto de todas las partículas y éste representa la totalidad de las posibles hipótesis a la luz de la información disponible hasta el momento. La verosimilitud de cada partícula x_t del conjunto muestral debería ser proporcional a su expresión Bayesiana:

$$x_t^{[m]} \sim p(x_t|z_{1:t}, u_{1:t}) \tag{3.22}$$

La consecuencia de esta ecuación es que si una determinada zona del espacio de estados está dominada por una gran cantidad de partículas, el objeto que estamos tratando de estimar se encontrará en esa zona con alta probabilidad.

De forma similar a como ocurre con el filtro de Kalman, los filtros de partículas elaboran sus hipótesis de manera recursiva, es decir, no necesitan almacenar un histórico completo de la información en cada instante. Únicamente con la información de la iteración anterior $prob(x_{t-1})$ se puede calcular la estimación actual $prob(x_t)$.

La versión básica del funcionamiento del filtro de partículas está descrita en la tabla 1. La entrada del algoritmo de estimación es el conjunto muestral del instante anterior X_{t-1}, junto con la entrada de control más reciente u_t y la última observación z_t. Veamos con más detenimiento el pseudocódigo del algoritmo:

1 **Algoritmo filtro de partículas**(X_{t-1}, u_t, z_t)**:**
2 $\bar{X}_t = X_t = 0$
3 **for** $m = 1$ **to** M **do**
4 | Muestrear $x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$
5 | $w_t^{[m]} = p(z_t|x_t^{[m]})$
6 | $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7 **end**
8 **for** $m = 1$ **to** M **do**
9 | Extraer i con probabilidad $\propto w_t^{[i]}$
10 | Añadir $w_t^{[i]}$ a X_t
11 **end**
12 Devolver X_t

Algoritmo 1: Algoritmo de filtro de partículas

La línea 4 genera una nueva hipótesis para el instante t basada en la partícula x_{t-1} y la entrada de control u_t. Esta nueva hipótesis corresponde a la muestra número m, pues su fuente de datos ha sido la partícula m del conjunto muestral X_{t-1}. Este paso del algoritmo acarrea muestrear desde la distribución de transición de estado $p(x_t|u_t, x_{t-1}^{[m]})$. El conjunto de partículas obtenidas tras M iteraciones representan las hipótesis del filtro $\bar{bel}(x_t)$.

En la línea 5 observamos cómo se calcula el factor de importancia o peso $w_t^{[m]}$ para cada partícula $x_t^{[m]}$. Los factores de importancia se utilizan para incorporar la observación z_t al conjunto muestral. Este peso se corresponde con la probabilidad de obtener la observación z_t desde la partícula $x_t^{[m]}$, dado por la expresión $w_t^{[m]} = p(z_t|x_t^{[m]})$. El conjunto muestral ponderado por su factor de importancia es, por tanto, la aproximación que realiza el filtro para la estimación $bel(x_t)$.

Entre las líneas 8 y 11 se produce la fase de remuestreo. El algoritmo debe extraer con reemplazamiento M partículas del conjunto muestral temporal $\bar{X}t$. La probabilidad de que en la extracción se obtenga cada partícula vendrá dada por su

factor de importancia. En consecuencia el remuestreo transforma un conjunto muestral de M partículas en otro del mismo número. El hecho de incluir los factores de importancia en esta fase hace que la distribución de partículas cambie. Mientras que antes del remuestreo las partículas se distribuían según $\bar{bel}(x_t)$, después del mismo se distribuirán aproximadamente según la distribución $bel(x_t) = \eta p(z_t|x_t^{[m]})\bar{bel}(x_t)$. El conjunto muestral resultante puede contener partículas duplicadas (debido a que la extracción se hace con reemplazamiento). Sin embargo el hecho más importante es que las partículas no contenidas en X_t tenderán a ser las de menor factor de importancia, representando las hipótesis menos probables.

3.2. Percepción distribuida de la pelota en la RoboCup

La estimación distribuida de la pelota requiere de varias etapas previas en cada uno de los miembros del equipo. Es necesario que cada robot estime de manera individual la pelota y, después, se combine la información para obtener una estimación común. En esta sección se describen las sucesivas etapas desarrolladas para mantener la estimación compartida de la pelota en coordenadas absolutas, esto es, situada en el campo de juego. El estado a estimar será la posición (x, y) de la pelota desde un punto de vista métrico. A continuación se muestra el resumen con los pasos a completar:

```
1  Algoritmo estimación distribuida de la pelota:
2  Obtención de la FDP instantánea e individual en el robot 1
3      Proceso de extracción de información en cada imagen
4      Obtención de la FDP instantánea de la observación en coord. relativas
5      Obtención de la FDP de posición del observador
6      Composición de la FDP instantánea e individual en coord. absolutas
7  Obtención de la FDP instantánea e individual en el robot 2
8  ...
9  Obtención de la FDP instantánea e individual en el robot n
10 z = Combinación de FDPs instantáneas e individuales en FDP distribuida
11 Actualizar el estimador distribuido con la observación z
```

Algoritmo 2: Esqueleto del algoritmo de estimación distribuida de la pelota

Básicamente el algoritmo combina con diferentes técnicas de fusión sensorial las estimaciones individuales de cada robot. Para que cada robot mantenga estimaciones instantáneas e individuales de la pelota debe resolver varias tareas. La primera de ellas es extraer información de las imágenes. Las imágenes no son tratadas directamente como observaciones, sino que se utilizan para estimar la distancia, ángulo e incertidumbres desde el robot que las observa hasta la pelota. Esta información inferida de las imágenes supone la observación en cada robot. Combinando la FDP de la pelota en coordenadas relativas y la FDP de la posición del robot en el campo de juego se obtiene la FDP de la pelota instantánea en coordenadas absolutas. Este proceso es repetido por todos los miembros del equipo y, puesto que todos ahora mantienen una estimación en el mismo sistema de referencia, éstas pueden ser combinadas.

3.2.1. Modelo de observación visual

Una de las etapas fundamentales en el diseño de nuestro estimador es la construcción de un modelo de observación visual para la estimación en posición de la pelota. Atendiendo al esquema presentado en el algoritmo 2, esta etapa se encarga de la extracción de información de las imágenes obtenidas en los robots aiBo y Nao. En concreto la información que extraerá será una tupla (ρ, θ), que expresará la distancia y ángulo desde la posición actual del robot hasta la pelota, es decir, en coordenadas relativas. Además es necesario caracterizar la incertidumbre asociada a la estimación.

El objetivo del modelo de percepción es obtener una medida de distancia, ángulo e incertidumbre asociada de la pelota respecto del robot. La incertidumbre que asociamos a cada medida debe aproximarse a la real, medida experimentalmente. Por un lado, se pretenden detectar los *errores sistemáticos* o sesgos y corregirlos y por otro, caracterizar la medida de la *incertidumbre* asociada. Imaginemos que situamos la pelota a diferentes distancias y, siempre, la estimación se encuentra por debajo de la distancia real. A esto le llamaremos *error sistemático*, pues es fácilmente resoluble corrigiendo la estimación. En general, diremos que un error es sistemático si el sumatorio del error en distancia o ángulo al objeto en la posición i es diferente de cero.

$$\begin{aligned} ErrorSistemDist &= \sum_{i=1}^{n} ErrorEstimacDist_i \\ ErrorSistemAng &= \sum_{i=1}^{n} ErrorEstimacAng_i \end{aligned} \tag{3.23}$$

Una vez corregida esta situación, podemos encontrarnos con otro fenómeno. Las estimaciones, en media tienen error cero, pero unas veces las estimaciones se quedan cortas y otras corresponden a distancias más lejanas que la distancia real al objeto (lo mismo sería aplicable al ángulo). Las principales causas de este fenómeno son los cambios de iluminación, sombras, baja resolución de las imágenes, etc. Aquí no podemos fácilmente corregir el error sumando o restando una determinada distancia o ángulo, pero no es éste nuestro objetivo. Nuestra tarea es conocer el grado de incertidumbre de la estimación. Así, elaboraremos una tabla que asociará incertidumbres mayores o menores dependiendo de la distancia y ángulo al objeto.

Modelo de percepción en el robot aiBo

El sensor principal que usamos es la cámara del robot aiBo, que proporciona imágenes a 30Hz en el espacio YCrCb y a una resolución de 416x320 pixels. El primer objetivo es obtener una medida de posición de la pelota basada en distancia y radio en coordenadas relativas. Con estos dos parámetros se calculan las incertidumbres asociadas en base a estadísticas experimentales obtenidas a priori.

En TeamChaos, toda imagen atraviesa las siguientes etapas: Transformación a HSV, filtrado de color, extracción de *blobs*, reconocimiento de objetos y estimación de distancia/ángulo al objeto detectado. La imagen 3.8 resume todas las fases que intervienen en el proceso de extracción de información.

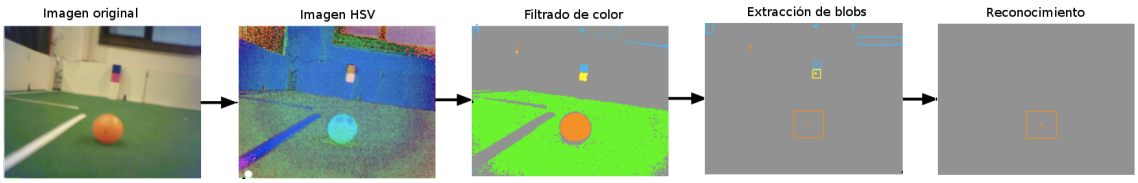


Figura 3.8: Proceso de extracción de información de una imagen

Transformación a HSV: La imagen original es transformada al espacio HSV. Este espacio define un modelo de color en términos de sus componentes constitu-

yentes en coordenadas cilíndricas. Sus siglas se corresponden con *Hue* o tonalidad, *Saturation* o saturación y *Brightness* o brillo. La principal ventaja de trabajar en este espacio de color es su mayor robustez a los cambios de iluminación.

Filtrado de color: A continuación se pasa un filtro sobre todos los píxeles de la imagen para que sean clasificados dentro del rango de posibles colores útiles. El entorno de la RoboCup define en su reglamento el color de cada objeto relevante (naranja para la pelota, verde para el suelo, amarillo y azul para las porterías, blanco para las líneas del campo). Estos rangos de colores en espacio HSV se calculan en la etapa de calibración.

Extracción de *blobs*: Todos los conjuntos de píxeles del mismo color y relativamente cercanos unos de otros son agrupados formando un posible objeto o *blob*. Se utiliza un algoritmo de crecimiento de regiones, por el que se parte de unos puntos *semilla* y se agregan puntos vecinos con características similares. El color de los pixels es utilizado como criterio de similitud.

Reconocimiento de objetos: Por cada uno de los posibles *blobs* se ejecuta un algoritmo de detección de objetos. Este algoritmo comprueba diferentes condiciones que deben cumplirse para que el *blob* sea considerado un objeto. Por ejemplo, para que sea considerado pelota, éste debe estar situado sobre otro *blob* verde (moqueta), y además debe tener un tamaño mínimo y máximo (correspondiente al tamaño de la pelota visto desde la posición más lejana o cercana posible). Un aspecto importante es que únicamente se escoge un blob candidato para la pelota. El código utilizado no permite obtener múltiples hipótesis de la pelota, por lo que únicamente nos quedaremos con la opción más probable.

Estimación de distancia y ángulo al objeto: Una vez reconocido el *blob* como objeto, se calcula la distancia y ángulo desde el robot hasta él. Los objetos tienen tamaño conocido, por lo que la distancia se obtiene realizando cálculos sobre el tamaño del *blob*. El ángulo se determina calculando el centro de masas del *blob* y midiendo el ángulo hasta ese punto. Este cálculo es realizado por el software ya existente dentro del módulo de percepción de *TeamChaos*.

Todos los factores anteriormente mencionados permiten modelar una versión simplificada del mundo que rodea al robot. Sin embargo, ninguna de las fases está exenta de errores, por lo que obtendremos un modelo aproximado del entorno que rodea al robot. Para caracterizar el grado de precisión de este modelo necesitamos realizar diversas pruebas que conforman el modelo de percepción visual del aiBo para la pelota.

Para realizar las pruebas se ha comenzado realizando una calibración de algunos parámetros de la cámara (resolución, balance de blancos, velocidad de obturación y ganancia). Después se ha realizado una calibración de todos los colores que el robot deberá detectar. En nuestro caso estos colores han sido: Naranja y verde.

Una vez terminado el proceso de calibración, hemos situado un robot en el centro del campo, por lo que su posición y orientación es conocida. Hemos distribuido en una de las mitades del campo 20 puntos a diferentes distancias y ángulos del robot. La figura 3.9 muestra la disposición del robot y los 20 puntos de control.

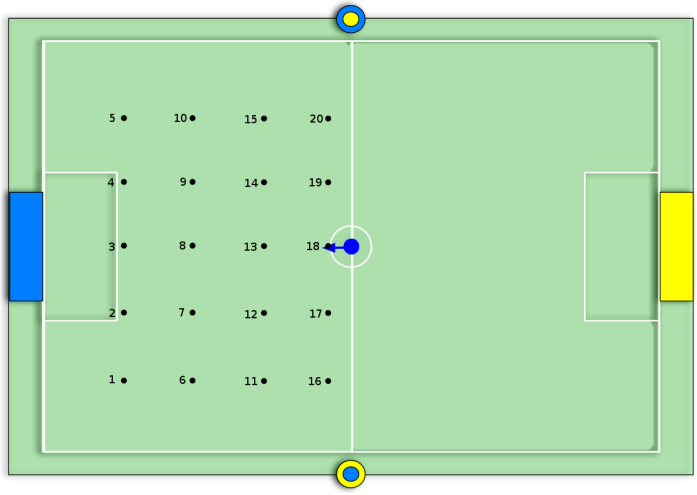


Figura 3.9: Escenario de pruebas para la obtención del modelo de observación visual. Se puede observar la posición del robot aiBo y los diferentes puntos de control a diferentes distancias y ángulos

Se ha diseñado y programado un sistema de *log* en el robot, que permite almacenar en un fichero todas las estimaciones en distancia y ángulo que se van calculando por cada objeto. En nuestro caso, hemos generado un fichero por cada punto de con-

trol. En cada fichero, se almacenan una serie de líneas de texto. Cada línea de texto contiene una marca de tiempo, la distancia estimada hasta la pelota y el ángulo estimado hasta la misma. A continuación se observa un fragmento de una captura de un fichero de *log*.

```
0 BALL-POS 1000 -1850
10 GTRUTH 0 0 -1.5707963
308 ODOM-LPS 5.76 0 0 0 0 0  2314 0.47 0.77 56430  424 -3.14 0 0  424 -3.14 0 0  424 -3.14 0 0  424 -3.14 0 0
402 ODOM-LPS 1.92 0 0 0 0 0  2311 0.47 0.76 56430  424 -3.14 0 0  424 -3.14 0 0  424 -3.14 0 0  424 -3.14 0 0
508 ODOM-LPS 2.56 0 0 0 0 0  2307 0.47 0.76 56430  425 -3.14 0 0  425 -3.14 0 0  425 -3.14 0 0  425 -3.14 0 0
604 ODOM-LPS 1.92 0 0 0 0 0  2304 0.47 0.75 56430  425 -3.14 0 0  425 -3.14 0 0  425 -3.14 0 0  425 -3.14 0 0
```

La activación del *log* se hace desde el programa *ChaosManager*, de manera que podemos activar y desactivar la captura de información, sin necesidad de intervenir físicamente sobre el robot (podríamos crear sombras en las imágenes o algún pequeño movimiento al presionar los botones). En la figura 3.10 se observa de manera resaltada la pestaña para activar/desactivar la captura de *logs* en un robot.

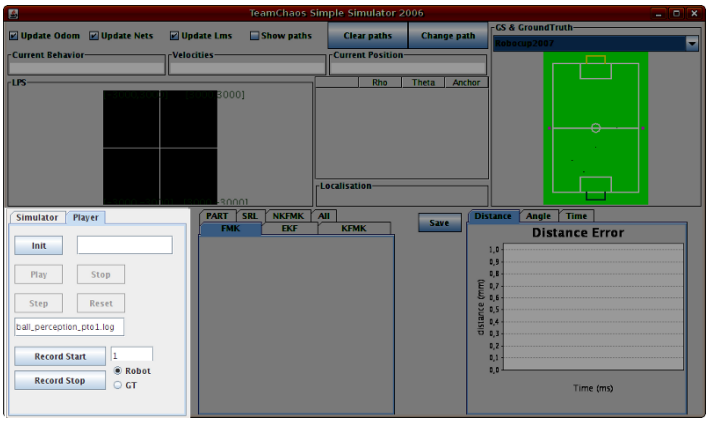


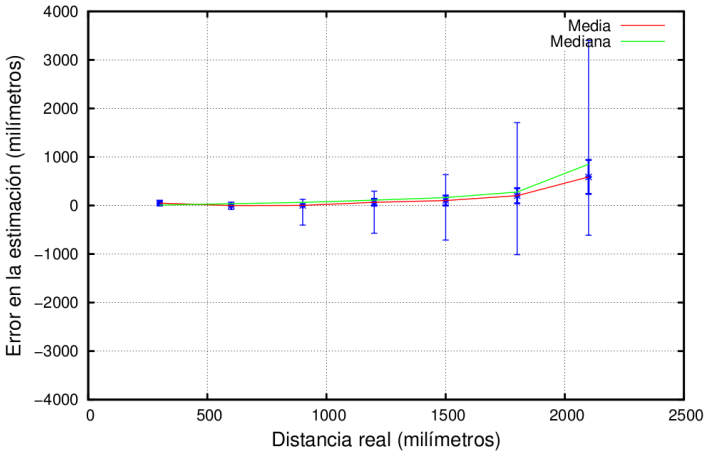
Figura 3.10: Sistema de *log* incluido en ChaosManager

Una vez tenemos preparada toda la infraestructura comenzamos con la captura de información. Situando el robot en la posición especificada en la imagen 3.9, comenzamos situando la pelota en el punto 1. A partir de ahí lanzamos el proceso de *log* durante aproximadamente 20 segundos y, transcurrido este tiempo, detenemos el *log*. Este procedimiento es iterado por cada uno de los 20 puntos de control señalados en el terreno de juego.

Resultados obtenidos con el robot aiBo

Las figuras 3.11 y 3.12 muestran el error (medido en distancia euclídea) en distancia y ángulo respectivamente, a diferentes distancias y ángulos de la pelota. A la vista de la gráfica de error en distancia 3.11 podemos apreciar cómo el error aumenta a partir de distancias superiores a 1.5 metros. Esto indica un error sistemático que debe ser corregido. Además, se observa que la desviación típica del error aumenta a medida que la distancia se amplía. Igualmente se advierte que a distancias elevadas se obtiene algún valor muy alejado de la realidad (díscolo). También es posible consultar un resumen de los valores numéricos obtenidos en la tabla 3.1.

Figura 3.11: Error en la estimación de distancia a la pelota

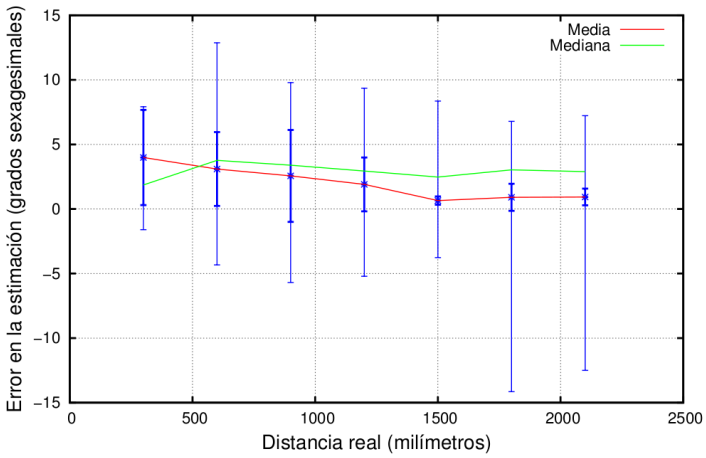


Distancia (mm.)	**Error medio (mm.)**	**Mediana (mm.)**	**Desv. típica (mm.)**	**Valor mín. (mm.)**	**Valor máx. (mm.)**
0-300	49.6769	49.0	8.2350	26.0	65.0
301-600	-1.9644	8.6875	36.3125	-81.3124	69.6875
601-900	2.8396	4.3458	62.4776	-404.1541	126.9670
901-1200	65.6725	70.5595	109.3807	-571.6180	296.3820
1201-1500	101.7489	102.4702	163.7456	-713.0298	633.9702
1501-1800	201.1203	156.6232	277.5007	-1015.0	1707.0
1801-2100	588.4460	350.0259	851.2892	-612.9740	3397.026

Cuadro 3.1: Error en distancia obtenido

La estimación de ángulo hasta la pelota tiene una interpretación algo diferente a la luz de los datos obtenidos. Como advertimos en la figura 3.12 el error del *software* actual se mantiene ligeramente por encima del cero, lo que indica un error sistemático que deberá ser subsanado. Sin embargo, la desviación típica del error se mantiene aproximadamente constante. El resumen de los datos obtenidos se puede consultar en la tabla 3.2.

Figura 3.12: Error en la estimación de ángulo a la pelota



Distancia (mm.)	**Error medio (grados)**	**Mediana (grados)**	**Desv. típica (grados)**	**Valor mín. (grados)**	**Valor máx. (grados)**
0-300	3.9877	3.6898	1.8567	-1.6042	7.9182
301-600	3.0940	2.8590	3.7684	-4.3289	12.8711
601-900	2.5629	3.5623	3.3904	-5.6908	9.7875
901-1200	1.9040	2.0855	2.9426	-5.2083	9.3508
1201-1500	0.6493	0.3237	2.4708	-3.7615	8.3565
1501-1800	0.9057	1.0485	3.0335	-14.1520	6.7895
1801-2100	0.9292	0.6446	2.8899	-12.4991	7.2279

Cuadro 3.2: Error en ángulo obtenido

Modelo de percepción en el robot Nao

Como sucede con el robot aiBo, la cámara es el sensor fundamental del robot Nao. Es capaz de capturar imágenes a 30 Hz., a una resolución de 640x480 y trabaja en espacio de color YUV. Las fases por las que atraviesa una imagen desde que es obtenida por la cámara hasta que se obtiene una estimación de distancia, ángulo e incertidumbres de la pelota son las siguientes:

Transformación a HSV y filtrado de color La imagen original en formato YUV422 es transformada al espacio HSV. Con el fin de acelerar esta conversión se ha utilizado una tabla de consulta o *LUT (LookUp Table*. En la fase de inicialización de la arquitectura se completa esta tabla iterando sobre todos los posibles valores de un píxel en el espacio YUV. Por cada uno de estos valores se realiza la transformación a HSV y se verifica si el citado punto corresponde a un píxel naranja, verde, amarillo, azul o blanco. Cada posición de la tabla permite almacenar un byte de información. A su vez, se ha reservado un bit para establecer si un determinado píxel en YUV pasa el filtro de cada color. El primer bit se dedica al color naranja, el segundo al verde, etc. De esta manera, ante un posible valor en YUV se indexa en la tabla y se obtiene un byte, al que se le aplica convenientemente una máscara para verificar si el citado píxel supera el filtro de naranja, verde, amarillo, azul o blanco. Con el fin de evitar un consumo excesivo de memoria se ha reducido la resolución de la tabla a un tamaño de 64 entradas por canal. Esta técnica ahorra sustancialmente el tiempo de transformación a HSV, pues únicamente con un acceso a memoria y la aplicación de varias máscaras se realiza su filtrado de color.

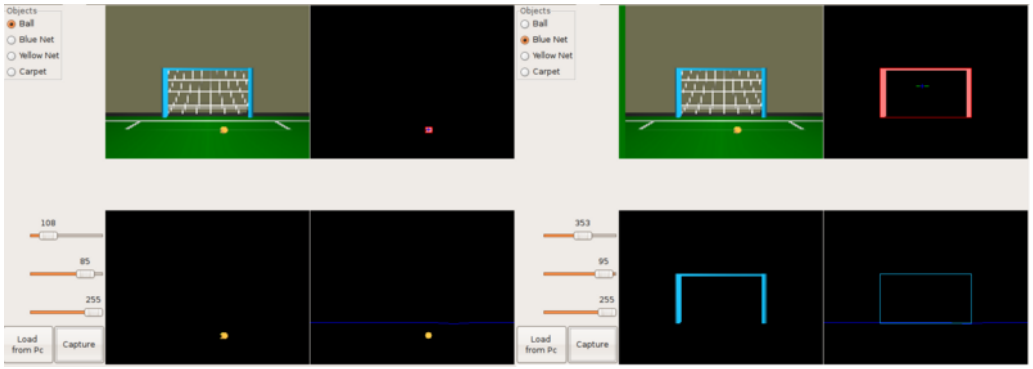
Extracción de *blobs*: Utilizando el principio de vecindad y homogeneidad de color, establecemos una lista de *blobs*, que se corresponden con los objetos potenciales a identificar. Se ha utilizado la biblioteca *cvBlob* [CL09] para extraer de la imagen filtrada por color el conjunto de *blobs*.

Reconocimiento de objetos: Atendiendo al color de referencia de cada *blob* se comprueban las sucesivas restricciones de tamaño, forma, posición respecto del horizonte visual, etc. para descartar o aceptar como pelota o portería cada uno de

los *blobs* potenciales. En caso de que varios *blobs* hayan atravesado la fase de reconocimiento y hayan sido identificados como el mismo objeto, únicamente uno de ellos será finalmente elegido atendiendo a varios criterios. La imagen 3.13 resume las etapas de procesamiento visual para el caso de la pelota y una portería en el simulador Webots. La imagen superior izquierda muestra la imagen original, la inferior izquierda muestra los píxeles que pasan el filtro de naranja (para el caso de la pelota) y el filtro de azul (para el caso de la portería azul), la superior derecha muestra los *blobs* obtenidos y la imagen derecha muestra el objeto reconocido junto con la línea del horizonte.

Estimación de distancia y ángulo al objeto: La fase anterior nos proporciona una región de la imagen donde se encuentra la pelota o las porterías que estamos tratando de estimar. Conociendo determinados píxeles representativos de la portería (esquinas) o el píxel central de la pelota, junto con la odometría interna del robot (las posiciones exactas de sus articulaciones) y asumiendo que los objetos se encuentran siempre sobre el plano del suelo, es posible retroproyectar estos píxeles representativos al espacio tridimensional del mundo real. Una vez establecidos estos puntos relativos a la posición del robot es fácil extraer la distancia y ángulo hasta los mismos. En el anexo B se puede encontrar información ampliada sobre esta técnica.

Figura 3.13: Diferentes etapas del procesamiento visual realizado sobre una pelota y una portería en el simulador Webots

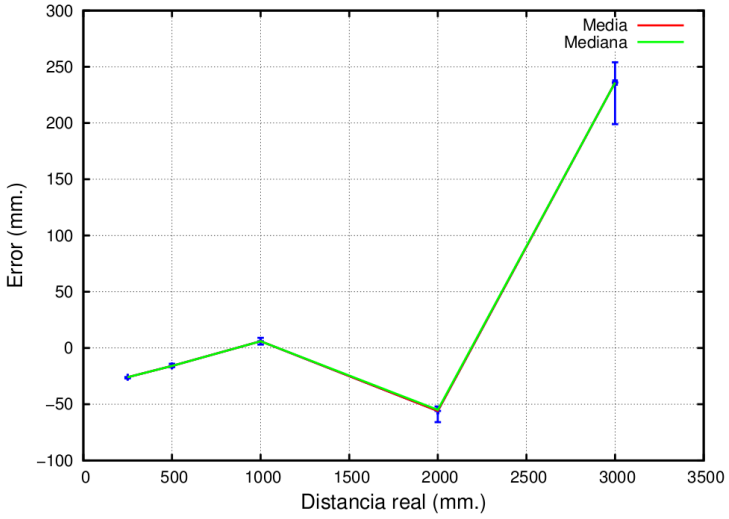


Con el fin de caracterizar el algoritmo de estimación de la pelota y de crear la tabla de incertidumbres, procedemos de manera similar a como se ha hecho para el robot aiBo. Se han marcado en el campo de juego varios puntos de control en posiciones conocidas y se ha realizado un análisis estadístico del *log* obtenido. A continuación se muestra un resumen de los datos obtenidos.

Resultados obtenidos con el robot Nao

Las figuras 3.14 y 3.15 muestran los resultados obtenidos al realizar diferentes estimaciones sobre posiciones de pelota a diferentes distancias. Los datos nos muestran que la estimación es bastante más precisa que la conseguida en el robot aiBo. Se aprecia un error sistemático de hasta 5cm. en distancias inferiores a 2m., para luego aumentar hasta casi 25 cm. a 3 metros. Es posible que este aumento considerable del error medio se produzca debido a un posicionamiento inexacto del robot, pues una desviación de apenas 1º provoca un error de 10-20 cm. a distancias tan grandes. Las oscilaciones son prácticamente despreciables para distancias inferiores a 2 metros y del orden de 1 cm. a una distancia de 3 metros.

Figura 3.14: Error en la estimación de distancia a la pelota

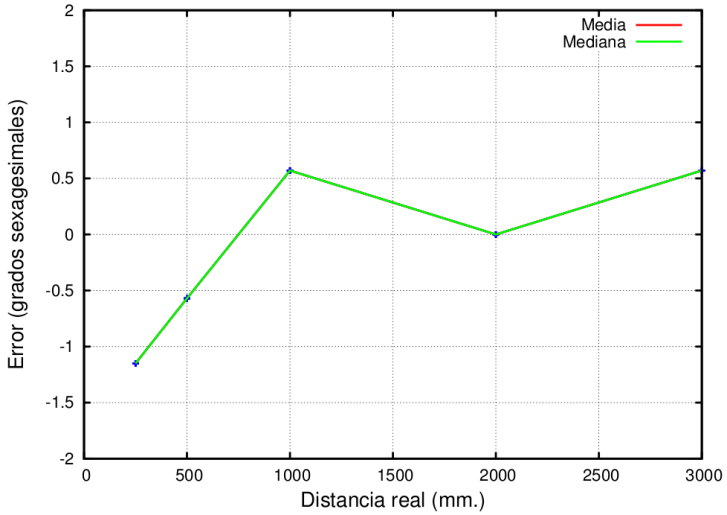


En cuanto al ángulo las medidas tomadas muestran una gran precisión angular, no excediendo nunca de 2 grados de error. Las oscilaciones son prácticamente des-

Distancia (mm.)	**Error medio (mm.)**	**Mediana (mm.)**	**Desv. típica (mm.)**	**Valor mín. (mm.)**	**Valor máx. (mm.)**
0-250	-26.75	-26,74	0.3	-27.68	-26.07
251-500	-16.38	-16,52	0.87	-17.91	-14.44
501-1000	6.32	6,57	1.34	-3.47	9.04
1001-2000	-56.11	-55.51	2.74	-66.34	-52.83
2001-3000	236.55	236.72	10.49	199.96	254.06

Cuadro 3.3: Error en distancia obtenido

Figura 3.15: Error en la estimación de ángulo a la pelota



preciables, lo que demuestra la gran calidad de la técnica empleada. Estos datos se utilizarán para caracterizar las FDPs individuales relativas en cada robot.

Distancia (mm.)	**Error medio (grados)**	**Mediana (grados)**	**Desv. típica (grados)**	**Valor mín. (grados)**	**Valor máx. (grados)**
0-250	-1.15	-1.15	0	-1.15	-1.15
251-500	-0.57	-0.57	0	-0.57	-0.57
501-1000	0.57	0.57	0	0.57	0.57
1001-2000	0	0	0	0	0
2001-3000	0.57	0.57	0.57	0.57	0.57

Cuadro 3.4: Error en ángulo obtenido

Una vez explicado el proceso de extracción de información visual para estimar la distancia, ángulo e incertidumbres, las siguientes secciones detallan las diferentes técnicas que se ha seguido para el resto de pasos del algoritmo de percepción distribuida. Cada técnica concreta deberá crear una FDP individual relativa utilizando los parámetros aquí calculados, deberá obtener la FDP de la posición del observador y finalmente deberá combinar las dos FDPs anteriores para componer la FDP individual absoluta que expresará la estimación instantánea de la pelota de uno de los robots. A continuación, cada algoritmo tiene su estrategia concreta de comunicación de esta FDP para que finalmente sean combinadas entre sí atendiendo a cada técnica específica.

3.2.2. Fusión analítica

La base de este método es modelar las observaciones locales como distribuciones Gaussianas bidimensionales. El primer paso del proceso es la obtención de la observación por parte de cada robot. Como se ha descrito en la sección anterior, ésta observación queda definida como una estimación local en coordenadas polares, es decir, distancia y ángulo desde la posición del robot, junto con una serie de incertidumbres asociadas.

$$C_m = \begin{pmatrix} \theta_r^2 & 0 \\ 0 & \theta_\phi^2 \end{pmatrix} = \begin{pmatrix} (\alpha r)^2 & 0 \\ 0 & \beta^2 \end{pmatrix} \tag{3.24}$$

A continuación modelaremos esa observación con una función de densidad de probabilidad Gaussiana asociada a la estimación anterior incorporándole incertidumbre radial y angular, según el modelo de observación previamente calculado. La incertidumbre será representada como una matriz de covarianza similar a la mostrada en la ecuación 3.24 , donde (α, β) representan las incertidumbres en distancia y ángulo respectivamente. Nuestros experimentos sobre el modelo de observación han revelado que el error en distancia es lineal $(\alpha * r)$ y el error en ángulo puede ser considerado constante (β). La ecuación 3.25 muestra la fórmula de la FDP bidimensional para una distribución gaussiana, donde C' es la matriz de covarianzas, $\bar{\mu_1}$ es la matriz de medias y S la matriz de estados (x, y).

Figura 3.16: FDP Gaussiana correspondiente a la observación relativa al robot

$$\rho(Obs) = \frac{1}{2\pi\sqrt{|C'|}} e^{-\frac{1}{2}(S-\bar{\mu_1})^T C'^{-1}(S-\bar{\mu_1})} \tag{3.25}$$

$$C' = \text{matriz de covarianzas de la obs. } P(pelota|robot)$$
$$S = \begin{pmatrix} x \\ y \end{pmatrix}$$
$$\mu_1 = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \tag{3.26}$$

El siguiente paso del algoritmo consiste en construir la FDP asociada a la posición del observador. La fuente de información aquí es el algoritmo de auto-localización presente en el robot. La salida de este algoritmo es una posición (x, y, θ) y una matriz de covarianza para modelar la incertidumbre en las tres dimensiones. La ecuación 3.27 determina la FDP para nuestra estimación de la posición del observador, donde C" es la matriz de covarianzas, $\bar{\mu_2}$ es la matriz de medias y M la matriz de estados (R_x, Ry, R_θ) para representar la posición y orientación del robot.

$$\rho(Loc) = \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|C''|}} e^{-\frac{1}{2}(M-\bar{\mu_2})^T C''^{-1}(M-\bar{\mu_2})} \tag{3.27}$$

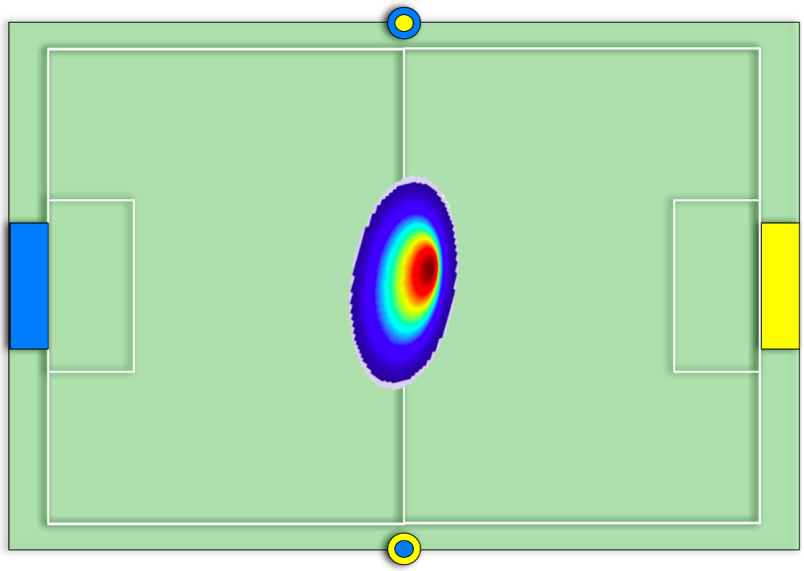


Figura 3.17: FDP Gaussiana correspondiente a la posición absoluta del robot

$$C'' = \text{matriz de covarianzas de la pos. del robot } P(robot)$$

$$M = \begin{pmatrix} R_x \\ R_y \\ R_\theta \end{pmatrix}$$

$$\mu_2 = \begin{pmatrix} \mu_{R_x} \\ \mu_{R_y} \\ \mu_{R_\theta} \end{pmatrix} \tag{3.28}$$

El siguiente paso del algoritmo persigue transformar a forma canónica la función de densidad de probabilidad de la observación (que expresa la pelota en coordenadas relativas) e incorporar al observador (que expresa la posición del robot en coordenadas absolutas). Con el fin de combinar las estimaciones procedentes de otros robots, necesitamos tener todas las distribuciones de probabilidad en un marco común, esto es, en un sistema de referencia global.

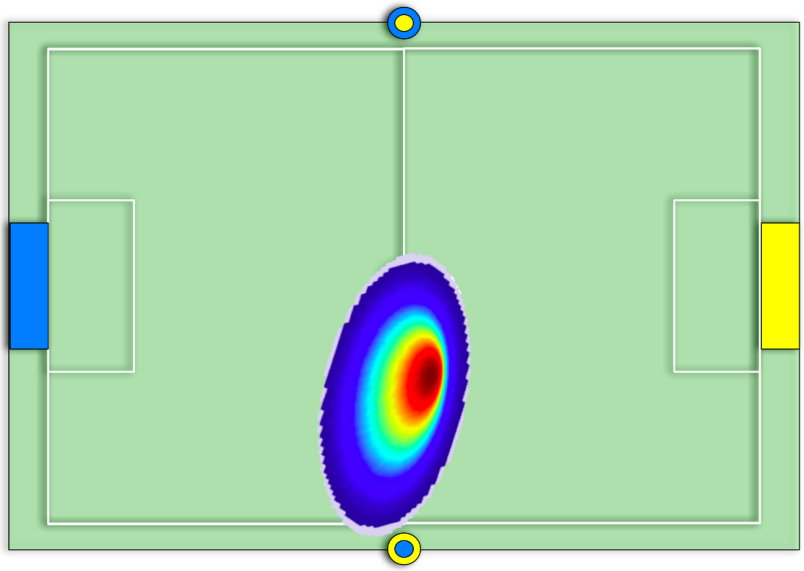


Figura 3.18: FDP Gaussiana correspondiente a la estimación individual absoluta de la pelota

La media de esta FDP absoluta individual se obtiene multiplicando la matriz de medias de la auto-localización por una matriz de rotación-traslación RT. También es necesario parametrizar la incertidumbre de la FDP individual absoluta. Para ello hay que combinar las dos fuentes de incertidumbre presentes: la de la observación y la de posición del observador. Sin embargo, hemos encontrado que esto no tiene una formulación cerrada, pues desde la ecuación 3.29 no podemos llegar fácilmente a la fórmula de otra distribución Gaussiana.

$$\rho(pelota_{xy}) = \int_{R_x}\int_{R_y}\int_{R_\theta} P(pelota_{xy}|robot_{R_xR_yR_\theta})P(robot_{R_xR_yR_\theta})dR_\theta dR_y dR_x \tag{3.29}$$

En su lugar hemos desarrollado una solución aproximada que funciona razonablemente bien para incertidumbres medias-bajas. Esta aproximación busca el englobante con forma Gaussiana que contiene las distribuciones Gaussianas resultantes para cada ángulo del robot. Es decir, la incertidumbre en la posición del robot viene dada por $(\sigma_x, \sigma_y, \sigma_\theta)$. Si suponemos una posición (x, y, θ) del robot, podemos obtener

la función Gaussiana resultante de la estimación de la pelota suponiendo que el robot está en ese punto. Si ahora vamos variando θ a lo largo de su desviación típica σ_θ, obtendremos varias funciones Gaussianas. La aproximación utilizada engloba todas estas funciones en una única función Gaussiana. En la figura 3.25 de la sección 3.2.4 se puede ver un ejemplo gráfico de esta aproximación.

Finalmente abordamos la última etapa del algoritmo: la fusión entre varios robots. La estimación compartida de la pelota es el resultado de combinar dos matrices de covarianza (incertidumbre de cada estimación) y dos estimaciones en posición, según muestran las ecuaciones 3.30 y 3.31 para el caso de dos robots. Se ha utilizado un enfoque probabilístico basado en la regla de Bayes para la fusión de las estimaciones. Con las matrices de covarianza y posición obtenidas, podremos combinar a su vez estimaciones procedentes de otros robots en el mismo instante temporal, gracias a la propiedad asociativa presente en esta formulación. El ángulo de la Gaussiana obtenida puede calcularse como muestra la ecuación 3.32, siendo A, B y D los componentes superior izquierdo, superior derecho/inferior izquierdo e inferior derecho respectivamente de la matriz de covarianza.

La estimación compartida incluye tanto el estado de la pelota como su incertidumbre. El estado almacenado es la posición (x, y) en el campo de juego y su incertidumbre nos da una idea de la precisión en la estimación. Las pruebas experimentales mostrarán cómo de efectiva es esta estimación comparada con las estimaciones locales individuales de cada robot.

$$C = C_1 - C_1[C_1 + C_2]^{-1}C_1 \tag{3.30}$$

$$\hat{X} = \hat{X}_1 + C_1[C_1 + C_2]^{-1}(\hat{X}_2 - \hat{X}_1) \tag{3.31}$$

$$\theta' = \frac{1}{2} \cdot tan^{-1}[\frac{2B}{A - D}] \tag{3.32}$$

La figura 3.20 muestra varios ejemplos de la estimación distribuida generada según las observaciones de tres robots. En la figura las observaciones locales de cada robot están representadas por elipses naranjas. El tamaño de esa elipse representa

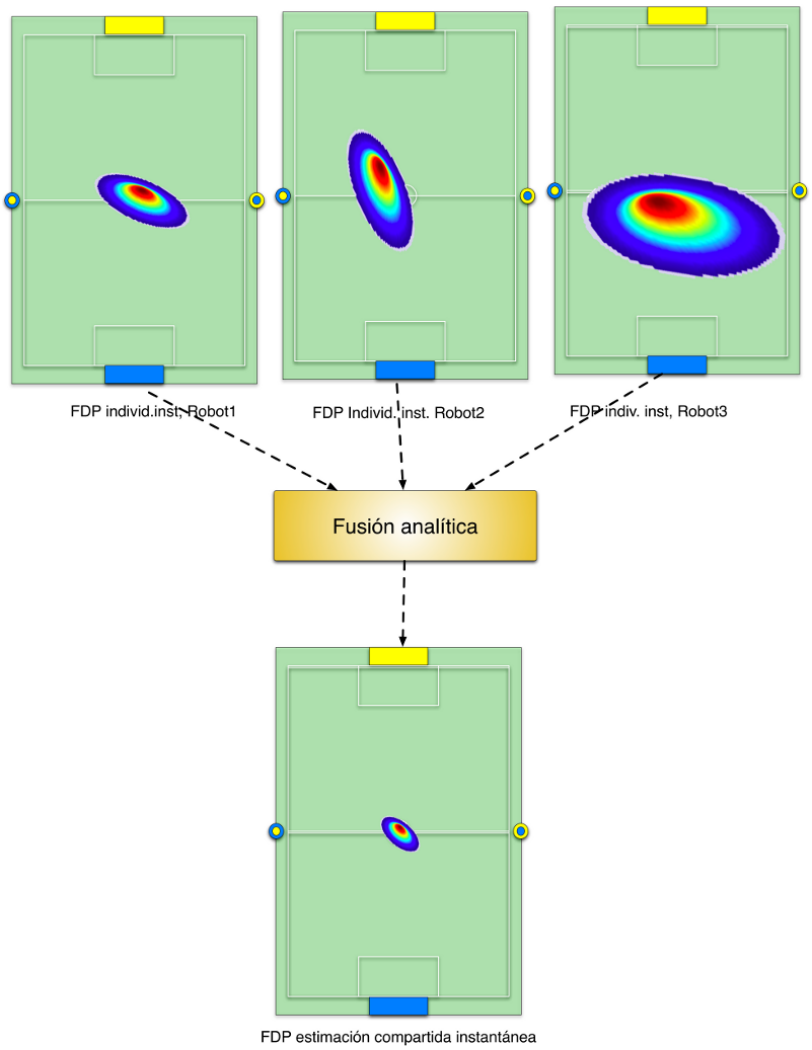


Figura 3.19: Esquema de funcionamiento de la variante analítica de percepción distribuida

la incertidumbre de la observación. La elipse de color azul es el resultado de la fusión de estimaciones locales. Los puntos negros son puntos de control y representan posiciones reales de la pelota en el campo.

El resultado obtenido de fusionar dos estimaciones individuales diferentes con esta técnica es una estimación compartida. Esta estimación compartida pondera de manera dispar las estimaciones que utiliza de entrada. Aquellas estimaciones que gocen de buena confianza, es decir, que dispongan de baja incertidumbre serán teni-

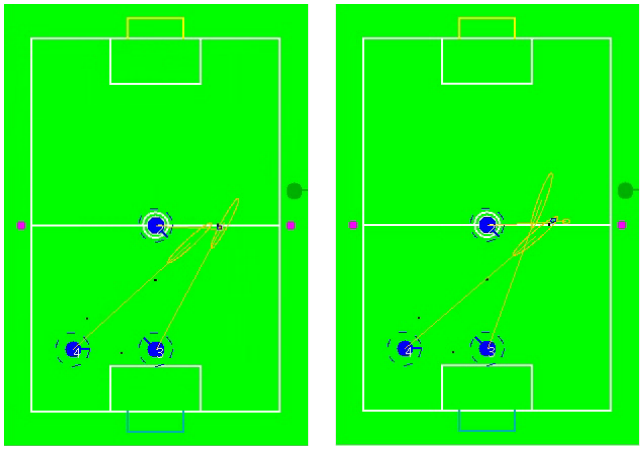


Figura 3.20: Fusión (elipse azul) procedente de tres estimaciones locales (elipse naranja)

das en cuenta en mayor medida que las estimaciones de menor confianza o mayor incertidumbre. Este fenómeno viene determinado por la manera de combinar las incertidumbres en la ecuación 3.30, método basado en la ganancia de Kalman del filtro que lleva su nombre.

Es importante resaltar que el carácter de esta estimación distribuida es completamente instantáneo. La fusión obtenida combina dos estimaciones instantáneas en otra compartida del mismo carácter. En ningún momento se acumulan las estimaciones para disponer de un estimador continuo. Este hecho, como veremos en los experimentos, provocará una mala defensa a los falsos positivos esporádicos.

Por otro lado, disponemos de una estimación distribuida basada en una Gaussiana bidimensional cuyos parámetros son obtenidos utilizando álgebra matricial de una manera muy eficiente. Incluso integrando las dos causas principales de incertidumbre obtenemos esta estimación en tiempos que, como veremos en la sección de experimentos, son insuperables por el resto de técnicas que aquí presentaremos.

3.2.3. Fusión de rejillas probabilísticas

Este segundo método basado en rejillas requiere que discreticemos el campo de juego en celdas de igual tamaño. En vez de manejar las FDPs analíticamente lo vamos a hacer extensivamente discretizando el dominio. En nuestro caso hemos con-

figurado el tamaño de celda a 15cm. x 15cm. Una de las primeras consecuencias que provoca esta técnica es la posible utilización de FDPs tanto de observación como de auto-localización más variadas y heterogéneas que las basadas en distribuciones Gaussianas. A su vez, puesto que cada robot calculará su estimación instantánea individual de la pelota y todas éstas serán comunicadas y combinadas de alguna manera, el intercambio de FDPs individuales será más costoso, pues será necesario enviar la rejilla al completo.

La primera fase del algoritmo es similar al método anterior: Obtención de la percepción local en coordenadas polares y generación de la matriz de covarianzas para modelar la incertidumbre gracias al modelo de observación. Básicamente nos apoyamos en los algoritmos perceptivos que nos devuelven todos estos parámetros por cada imagen que captura la cámara.

El siguiente paso consiste en construir la rejilla de probabilidad de la observación, es decir, la FDP de la observación en coordenadas relativas al robot. La ecuación 3.25 muestra la fórmula de la función de densidad de probabilidad para una distribución Gaussiana de dos dimensiones. En nuestro caso asumiremos esta formulación debido a que nuestro modelo de observación aproxima la incertidumbre de la observación a una distribución Gaussiana. Para cada celda de nuestra rejilla *obs* calcularemos el valor de la *FDP* en ese punto obteniendo un resultado similar al de la figura 3.21.

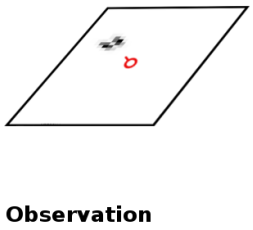


Figura 3.21: Rejilla probabilística asociada a una posición de pelota

A continuación necesitamos modelar la FDP de la posición del observador. Para ello utilizaremos un cubo de probabilidad *loc*, donde cada plano horizontal estará asociado con una posible orientación del robot. El algoritmo de auto-localización proporcionará una terna (x, y, θ), junto con una matriz de covarianzas para expresar la

incertidumbre en (x, y). Para nuestras pruebas hemos considerado un error en orientación modelado como una distribución Gaussiana de desviación típica constante. De manera similar al paso anterior, obtenemos todos los parámetros de la FDP asociada a la auto-localización y evaluamos esta función en el centro de cada celda de la rejilla *loc* obteniendo el resultado mostrado en la figura 3.22.

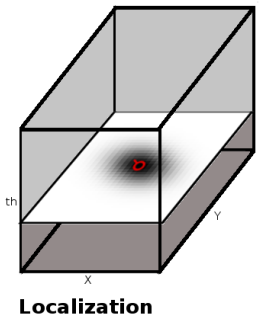


Figura 3.22: Rejilla probabilística asociada a la posición del observador (x, y, θ)

La situación actual es que disponemos de dos rejillas: *obs* mantiene la FDP relativa de la observación y *loc* contiene la FDP en coordenadas globales de la posición del observador. Con la operación de convolución 3.33 podemos combinar estas dos rejillas para crear una tercera y definitiva denominada *ball_indiv_abs*, que contiene la PDF de la observación teniendo en cuenta las incertidumbres en la posición del observador como la de la propia observación. La figura 3.23 ilustra el proceso de convolución entre las rejillas *obs* y *loc*. Esta fórmula está derivada de la ecuación 3.29, pero ahora particularizada para el entorno discretizado que nos ocupa. Puesto que esta FDP utiliza el marco de referencia global y común del campo de juego puede ser combinada con FDPs del mismo carácter pero procedente de otros robots.

$$\rho(pelota_{xy}) = \sum_{i=1}^{x'} \sum_{j=1}^{y'} \sum_{k=1}^{\theta'} P(pelota_{xy}|robot_{ijk}) P(robot_{ijk}) \tag{3.33}$$

Cada robot deberá actualizar periódicamente todas sus rejillas y comunicar al resto *ball_indiv_abs*. El proceso de fusión consiste en aplicar la regla de Bayes para

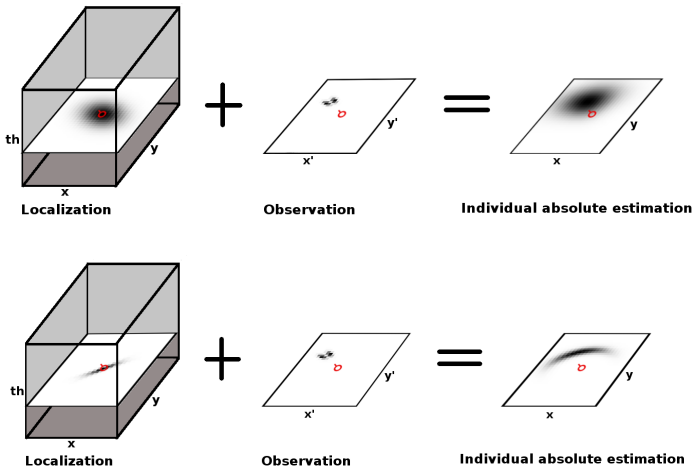


Figura 3.23: Proceso de fusión entre la FDP con la observación relativa y la FDP de auto-localización

combinar las rejillas procedentes del resto de miembros del equipo con la generada localmente. La ecuación 3.34 muestra el desarrollo matemático de la combinación Bayesiana de dos observaciones independientes. Se pretende calcular el valor de c dados las observaciones a y b desde robots diferentes. En las operaciones indicadas con * y ** se asume independencia de las observaciones. El resultado final demuestra que basta con multiplicar las dos observaciones y dividir entre $P(c)$, que corresponde con la probabilidad a priori.

$$\begin{aligned} P(c \mid a,b) &= \frac{P(c \mid a)P(b \mid c,a)}{P(b \mid a)} \\ P(b \mid c,a) = P(b \mid c)^{*} &= \frac{P(c \mid b)P(b)}{P(c)} \\ P(c \mid a,b) = \frac{P(c \mid a)P(c \mid b)P(b)^{**}}{P(b \mid a)^{**}P(c)} &= \frac{P(c \mid a)P(c \mid b)}{P(c)} \end{aligned} \tag{3.34}$$

Utilizando el anterior desarrollo matemático actualizaremos la rejilla *ball_shared* con la estimación distribuida. Para cada celda de *ball_shared* obtendremos su valor

multiplicando esa misma celda en todas las *ball_indiv_abs* recibidas y dividiendo por una constante.

En la figura 3.24 podemos observar un ejemplo de rejillas probabilísticas correspondientes a la estimación de la pelota. En la imagen se puede advertir la rejilla resultante de combinar dos rejillas individuales absolutas.

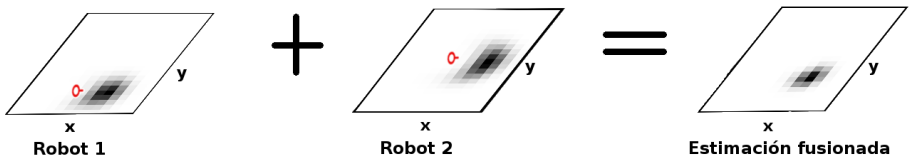


Figura 3.24: Combinación de dos rejillas de probabilidad con la estimación de la pelota

Como sucedía con la variante analítica el carácter temporal de este estimador es completamente instantáneo. Las FDPs recibidas del resto de miembros del equipo se corresponden con estimaciones individuales instantáneas y, puesto que no se realiza ninguna acumulación temporal, el resultado de la combinación de estimaciones continúa siendo instantáneo. En este sentido los experimentos volverán a poner en evidencia el mal comportamiento ante falsos positivos esporádicos, que sí serán amortiguados por las soluciones basadas en filtros que acumulan evidencias.

Puesto que el *software* perceptivo de ambas plataformas utilizadas extrae una medida de distancia y ángulo hasta la pelota, nos encontramos con un modelo de observación monomodal. La técnica de fusión basada en rejillas probabilísticas contempla FDPs multimodales. Debido a las limitaciones perceptivas y del algoritmo de auto-localización nunca tendremos FDPs multimodales en nuestro caso, sin embargo, las rejillas de probabilidad permitirían un funcionamiento correcto del algoritmo si reemplazásemos alguno de estos componentes por variantes multimodales.

En la sección de experimentos se podrán ver valores numéricos de los tiempos de ejecución de esta técnica. Como ya se puede intuir si la resolución se ajusta para valores centimétricos la cantidad de iteraciones necesarias para actualizar todas las rejillas vuelve inabordable el empleo de esta técnica. Esta es la desventaja fundamental de éste y todos los métodos que discretizan el entorno: cuando el número de dimensiones del espacio de estados o su resolución aumenta, el consumo de CPU crece en exceso.

3.2.4. Fusión basada en filtro de Kalman

En esta técnica se emplea un estimador basado en filtro de Kalman para mantener una estimación continua de la pelota. Cada una de las observaciones procedentes del resto de robots son combinadas en una estimación compartida instantánea, que espués se utilizada para alimentar el paso de corrección del filtro.

Con el fin de comprender mejor los pasos del algoritmo de fusión basado en filtro de Kalman, se va a describir el diseño realizado.

La clase *Estimation* permite almacenar toda la información procedente de algún miembro del equipo. Entre esta información se encuentra la posición (x, y, θ) del robot, su observación z de la forma $(distancia, ángulo)$, su incertidumbre en la posición $(Incert_x, Incert_y, Incert_{theta})$ y la incertidumbre en la observación $(Incert_{dist}, Incert_{ang})$.

Una vez recibida la información, la clase *Estimation* calcula cuál es la posición de la pelota en coordenadas globales (coordenadas en el campo de juego) y su incertidumbre asociada, es decir, la FDP individual absoluta de la pelota. Este cálculo se realiza de manera exactamente igual que como se hace en la variante analítica, pues ambos métodos crean las FDP individuales absolutas de la misma manera.

Como se ha mencionado anteriormente, para posicionar la pelota en coordenadas globales es necesario tener en cuenta los dos tipos de incertidumbres posibles (observador y observación). Aunque de manera independiente puedan modelarse como Gaussianas, al realizar la operación de convolución para combinar ambas, no obtenemos una Gaussiana. Puesto que esta técnica asume que la incertidumbre debe modelarse mediante esta distribución, es necesario aproximar la FDP obtenida tras la convolución a una Gaussiana.

La figura 3.25 muestra cuatro ejemplos de estimaciones con diferentes incertidumbres de orientación en el robot observador. Este robot se encuentra situado en la posición $(0, 0, \frac{\pi}{2})$ y su incertidumbre en la orientación de su posición es 30º, 60º, 90º y 120º respectivamente. La elipse blanca representa la desviación típica de la Gaussiana que aproxima la estimación del robot teniendo en cuenta todas las incertidumbres.

Una vez comunicadas y recibidas todas las FDP individuales absolutas de la pelota existe un paso intermedio que busca crear una FDP compartida entre todos

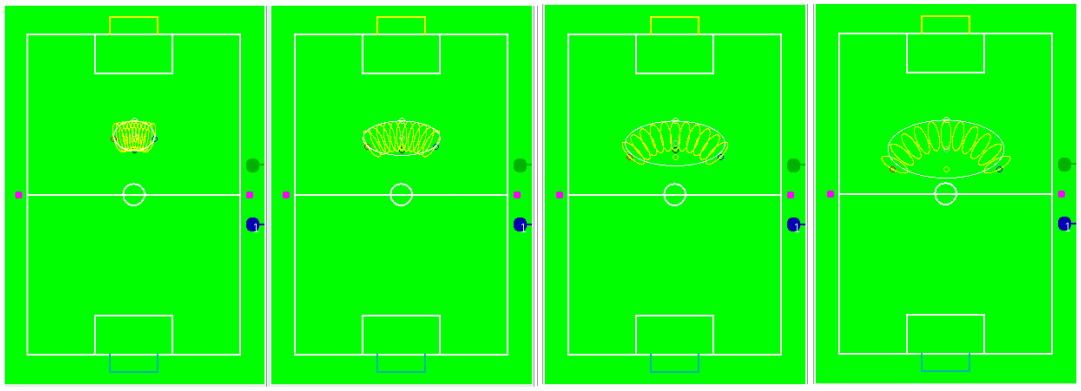


Figura 3.25: Diferentes aproximaciones con 30º, 60º, 90º y 120º de incertidumbre de orientación en el robot observador

los miembros del grupo, que es interpretada como la estimación instantánea de la pelota por parte del equipo. La figura 3.26 ilustra este proceso. Para realizar esta fase del algoritmo hemos empleado la técnica de fusión analítica de la sección 3.2.2, ponderando de manera diferente cada estimación individual absoluta de la pelota. Las FDPs con menos incertidumbre *pesarán* más en la decisión que las FDPs con más incertidumbre.

Para poder considerar las observaciones independientes, los robots únicamente envían el resultado de sus observaciones si se cumple algunas de estas condiciones:

1. Que la observación sea lo suficientemente diferente de la anterior. Para ello se utiliza un constante *Z_MIN_DISTANCE* que evalúa la distancia euclídea entre la posición de la pelota en el instante actual y en la observación anterior. Sólo en caso de que la nueva posición esté lo suficientemente alejada de la posición anterior se considerará una nueva observación.

2. En caso de que el robot se desplace por el campo. La constante *O_MIN_DISTANCE* evalúa la distancia recorrida según la odometría del robot. Sólo en caso de que la diferencia entre las posiciones ocupadas por el robot en el instante actual y el anterior supere este umbral se considerará una nueva observación.

3. En caso de que transcurra una cantidad de tiempo *T_MIN_INTERVAL* entre la última observación enviada y el instante actual.

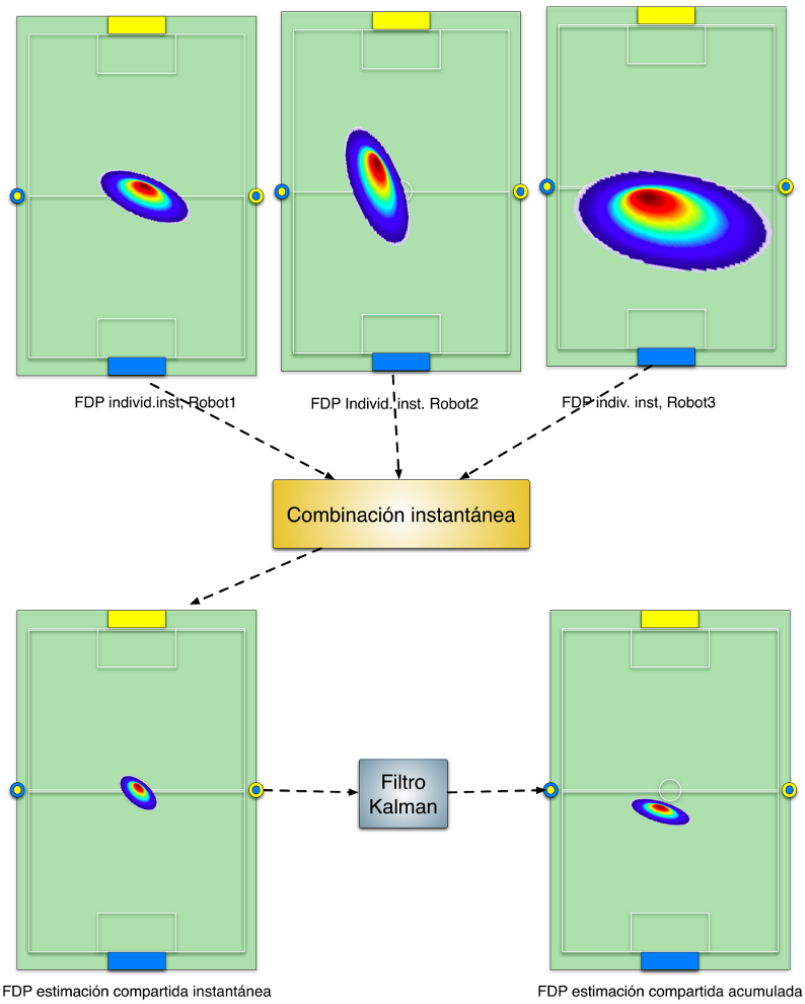


Figura 3.26: Esquema de funcionamiento del estimador distribuido basado en filtro de Kalman

Esta FDP intermedia se utiliza como nueva observación para un filtro de Kalman que estima la posición de la pelota. Cada vez que se recibe una observación, después de realizar la aproximación citada anteriormente, se actualiza el filtro de Kalman con un paso de corrección. Puesto que el tiempo entre sucesivas iteraciones del filtro de Kalman va a ser muy reducido, se ha incluido una dinámica del sistema sencilla basada en una función Gaussiana. La dinámica del sistema expresa cómo se modela el posible movimiento de la pelota dentro del estimador. Esta función Gaussiana predice cuál es el estado de la pelota después del último paso de corrección y, en nuestro

caso, actuará haciendo perder algo de credibilidad a la última corrección, es decir, aumentando ligeramente la incertidumbre asociada a la estimación.

La salida del filtro de Kalman, correspondiente con la posición de la pelota es enviada periódicamente a la aplicación ChaosManager. De esta manera, podemos visualizar en la herramienta de depuración, tanto la posición de la pelota según este algoritmo, como la incertidumbre asociada a la estimación.

Otra variante estudiada para esta técnica de fusión consistió en conectar directamente el filtro de Kalman con las estimaciones individuales absolutas procedentes de cada robot. De esta manera, estas FDPs actúan como observaciones independientes de la pelota y el filtro va corrigiendo su estado según se reciben nuevas observaciones. Esta alternativa nos ofreció peores resultados que la variante explicada anteriormente y que añade una fase intermedia de combinación instantánea. El motivo fundamental es que la fusión instantánea intermedia dota de mayor estabilidad a la observación que corrige el filtro de Kalman y se obtienen menos fluctuaciones en el estado final.

3.2.5. Fusión basada en filtros de partículas

La última variante desarrollada para estimar la posición de la pelota de manera compartida utiliza un filtro de partículas. Anteriormente hemos manejado las FDPs de manera analítica y de manera discretizada. En este caso vamos a representar todas las FDPs como muestras ponderadas aplicando métodos de Monte Carlo como aproximación de las FDPs ideales. Aplicando el operador de convolución obtendremos la estimación individual instantánea absoluta, que también estará compuesta por muestras. Esta FDP es la que se comunica entre todos los miembros del equipo y, en cada robot, se combinan para mantener una hipótesis compartida de la pelota. A continuación detallaremos los pasos del algoritmo con más profundidad.

El primer paso consiste en modelar la localización del robot mediante una FDP muestreada, donde cada una de las muestras representa una hipótesis (x, y, θ) y contiene un factor de importancia. Cada una de estas hipótesis la denominaremos partícula. Puesto que el algoritmo de auto-localización implementado en TeamChaos está basado en filtro de Kalman, la salida de éste es una Gaussiana tridimensional, caracterizada por su matriz de medias y su matriz de covarianza. La matriz de medias

establecerá la posición del campo de juego más probable y la matriz de covarianza nos indicará cómo de fiable es esa medida, es decir, la incertidumbre asociada.

Con el fin de obtener una serie de muestras de esa distribución, empleamos el método *sample_normal,_distribution (double dt)*, que atiende a la ecuación 3.35, para muestrear un valor sobre una distribución normal de medio cero y desviación típica dt.

$$\textbf{Algoritmo sample\_normal\_distribution (double dt):} \quad return\ \frac{1}{2}\sum_{i=1}^{12} rand(-dt, dt) \tag{3.35}$$

Llamando a este método por cada componente x, y y θ y sumando el valor obtenido a la media de cada componente obtendremos las tres coordenadas de la muestra. La figura 3.27 muestra un ejemplo de cómo quedarían distribuidas las partículas después de esta fase de muestreo.

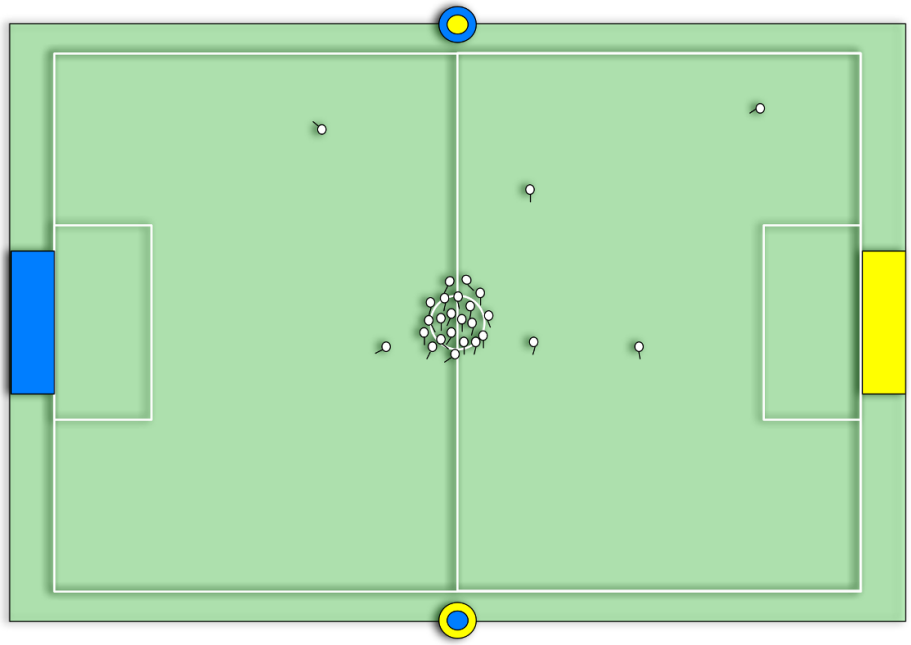


Figura 3.27: Ejemplo simplificado de varias partículas 3D representando posibles posiciones del observador

En caso de disponer de otro algoritmo de auto-localización no basado en filtro de Kalman, como por ejemplo algoritmos Markovianos o algoritmos con un carácter

multimodal basados en filtros de partículas, etc. habría que muestrear la FDP de la posición del observador sobre la salida de esos algoritmos. Si directamente ofrecen una salida en forma de FDP muestreada, ésta podría ser usada directamente por nuestro algoritmo sin ninguna etapa intermedia de adaptación.

El siguiente paso del algoritmo de fusión consiste en obtener la FDP muestreada correspondiente a la observación relativa, es decir, la observación desde el sistema de referencia del robot. El modelo de observación visual implementado en ambas plataformas genera una tupla (ρ, θ) con la distancia y ángulo estimada hasta la pelota. En función de la distancia ρ y ángulo θ se asigna una incertidumbre, tanto en x como en y, a la posición de la pelota en coordenadas relativas. Combinando toda esta información se modela la posición de la pelota como una FDP Gaussiana centrada en la posición calculada y con mayor o menor dispersión según la incertidumbre asociada.

Como sucedía anteriormente con el problema de la auto-localización, utilizamos la función *sample_normal_distribution()* para muestrear de la FDP de la observación y obtener la aproximación muestreada. Sustituyendo la posición de la muestra en la función de densidad de probabilidad de la Gaussiana calculada anteriormente obtendremos el factor de importancia de la partícula. La imagen 3.28 ilustra esta etapa, donde se pueden observar las diferentes hipótesis para la posición de la pelota respecto al observador.

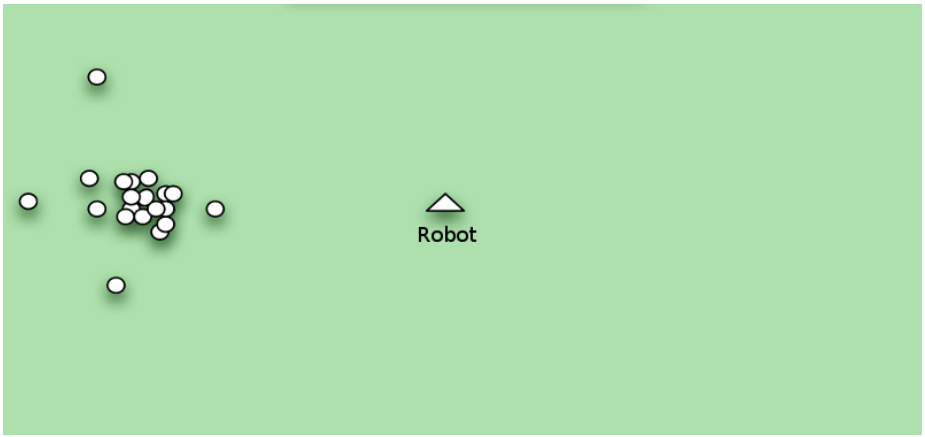


Figura 3.28: Ejemplo simplificado de FDP muestreada 2D representando la observación relativa de la pelota

La etapa final para obtener la estimación individual instantánea de la pelota consiste en combinar la posición del observador y la observación relativa para estimar la posición de la pelota en coordenadas globales. Es necesario trasladar las dos fuentes de incertidumbre (auto-localización y observación) a la estimación final. La operación de convolución realiza esta tarea. Puesto que se emplea un enfoque muestreado, para crear las n muestras de la FDP que buscamos, se necesita muestrear en las dos distribuciones que hemos creado anteriormente: $FDP_{auto-loc.}$ y $FDP_{observ.}$. Por cada pareja de muestras, que representa una posible hipótesis para la posición del observador y una posible observación local, realizamos un cambio de base para obtener la posición de la pelota en el campo de juego y aplicamos la regla de Bayes para combinar las incertidumbres. La figura 3.29 muestra un ejemplo de FDP muestreada 2D para modelar la posición de la pelota en coordenadas globales, usando las 2 FDP's anteriores (auto-localización del observador y observación).

$$\begin{aligned} Muestra^{i}_{indiv.} &= Muestra^{i}_{FDP_{auto-loc.}} * Muestra^{i}_{FDP_{obs.}} \\ &= (Muestra_{indiv.x}, Muestra_{indiv.y}, Muestra_{indiv.importancia}) \end{aligned} \tag{3.36}$$

Una vez calculada la estimación individual instantánea y absoluta de la pelota por parte de cada robot, es necesario mezclar todas estas fuentes de información para converger hacia una única estimación compartida obtenida aplicando sucesivas observaciones a un filtro de partículas.

Periódicamente, cada robot envía su FDP completa con la estimación individual instantánea de la pelota al resto de compañeros. A su vez, también recibe y almacena las FDP's muestreadas que le envían los demás. En caso de que un robot no vea la pelota, su FDP instantánea y absoluta a comunicar será una FDP con las muestras repartidas de modo uniforme. El motor principal del algoritmo de fusión sensorial para mezclar las estimaciones de cada robot es un filtro de partículas, que mantendrá una acumulación temporal y continua de evidencias. La figura 3.30 ilustra el esquema principal de esta técnica.

En el paso de corrección del filtro de partículas es donde se utilizan las últimas estimaciones individuales absolutas de la pelota, comportándose éstas como obser-

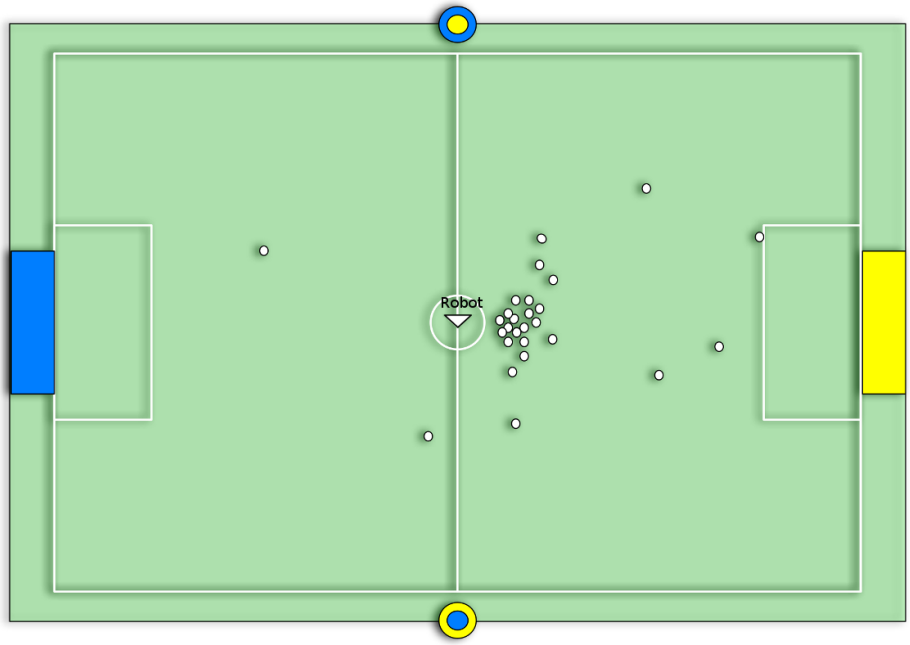


Figura 3.29: Ejemplo simplificado de FDP muestreada 2D representando la posición de la pelota en coordenadas globales

vaciones que alimentan el mencionado filtro. El paso de corrección evalúa cada una de las hipótesis (partículas) según la observación recibida. En este ejemplo contamos con tres observaciones, uno por cada robot compañero. La imagen 3.31 muestra las hipótesis que maneja el filtro de partículas (en color blanco) y cada una de las hipótesis recibidas por el resto de compañeros (en color azul, naranja y rosa).

Por cada una de las hipótesis de nuestro filtro P_i^{shared} se calcula la distancia euclídea hasta cada una de las partículas recibidas por el compañero uno P_i^1. Aquellas cuya distancia se mantenga dentro de cierto umbral, serán emparejadas con la partícula P_i y, por tanto, acumularemos su factor de importancia en $DensityPS1$. La figura 3.32 ilustra un instante del paso de corrección. Una de las hipótesis del filtro de partículas (color blanco) verifica las partículas procedentes de otros robots (colores azul, rosa y naranja). Aquellas que se encuentran dentro del área delimitada por la circunferencia coloreada en blanco afectarán al nuevo factor de importancia de la hipótesis.

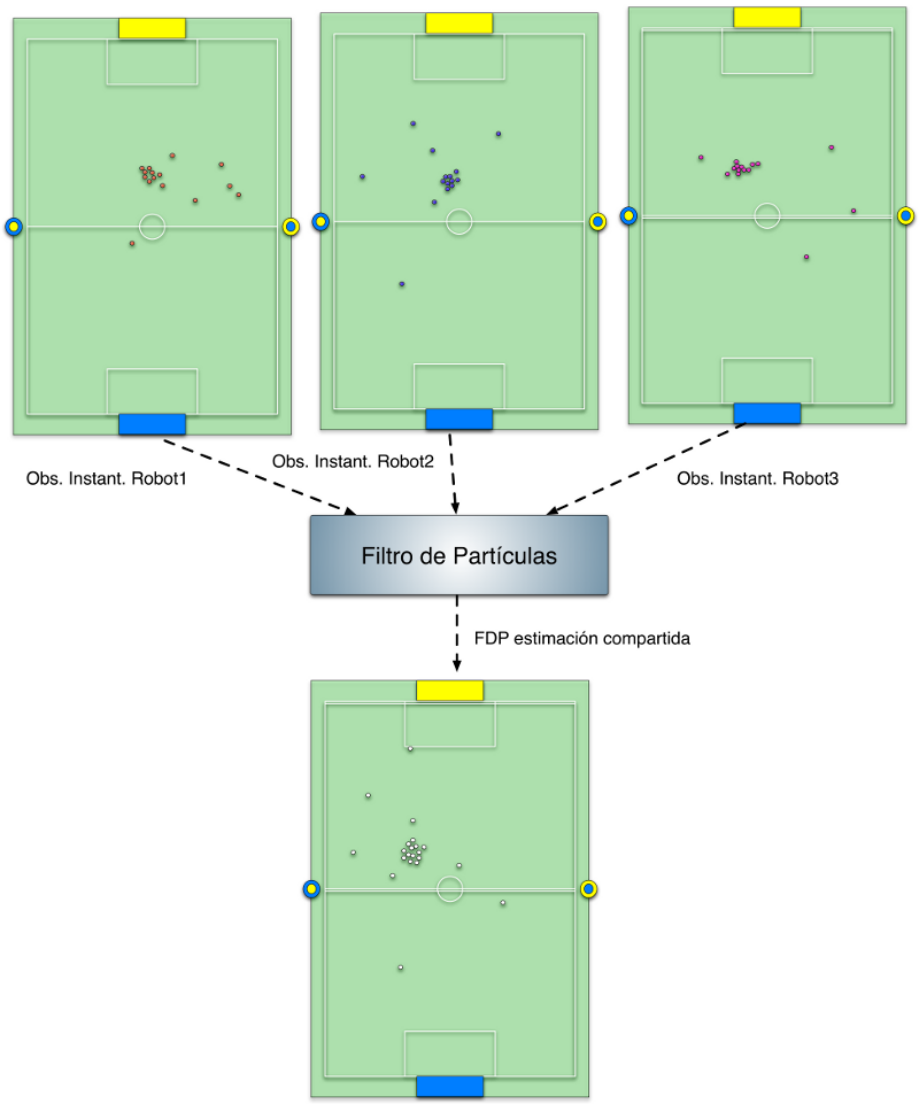


Figura 3.30: Esquema del algoritmo de fusión sensorial basado en filtro de partículas

La misma operación será realizada entre P_i^{shared} y las hipótesis recibidas por los compañero dos P_i^2 y tres P_i^3, calculándose el valor de $DensityPS2$ y $DensityPS3$ para la partícula i.

Finalmente, resta calcular el nuevo factor de importancia de la partícula i, en función de $DensityPS1$, $DensityPS2$ y $DensityPS3$.

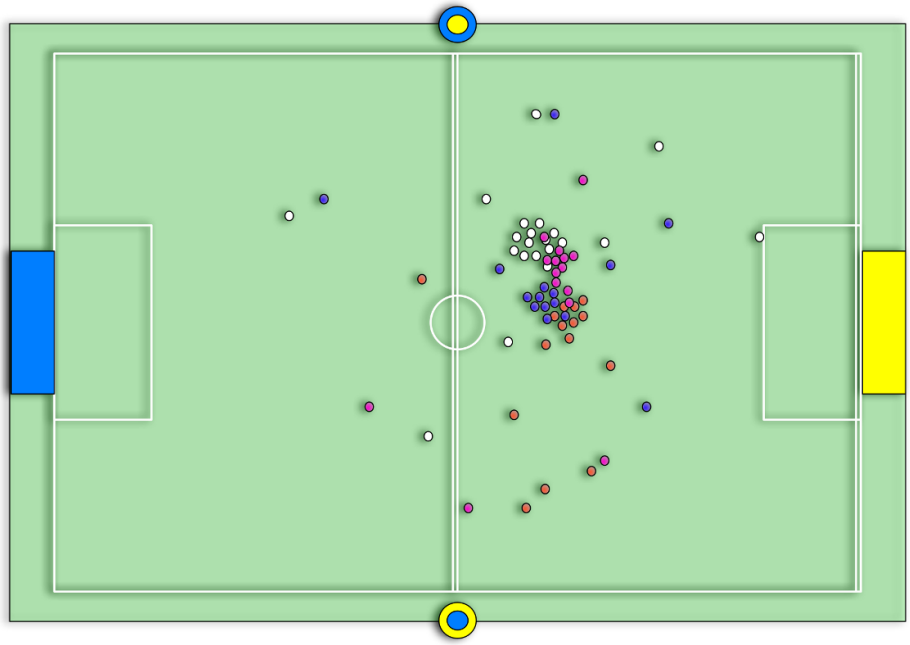


Figura 3.31: Esquema con las hipótesis del filtro de partículas (blanco) e hipótesis recibidas del resto de compañeros (azul, naranja y rosa)

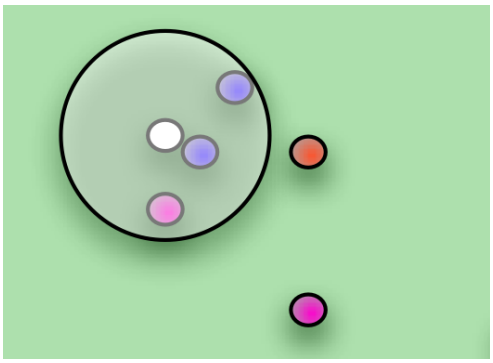


Figura 3.32: Detalle del paso de corrección para evaluar el nuevo factor de importancia de una hipótesis del filtro de partículas

$$P_i^{shared}.importance = DensityPS1 \cdot DensityPS2 \cdot DensityPS3 \cdot n^2 \quad (3.37)$$

siendo n el número de partículas o hipótesis utilizadas en cada filtro.

Las fases de predicción y remuestreo se realizan como se expuso en la sección 3.1.4. Puesto que el filtro de partículas no calcula una posición concreta para la pelota, sino que mantiene una FDP de la misma, tenemos que calcular cuál es la posición más probable donde encontrarla. Para ello realizamos una media ponderada por los factores de importancia de cada hipótesis. Las coordenadas (x, y) obtenidas delimitarán cuál es la estimación concreta de nuestro algoritmo. De igual manera, se obtienen los valores de incertidumbre, para evaluar cómo de fiable o precisa es la estimación.

3.3. Experimentos

Los experimentos que vamos a describir en esta sección permiten caracterizar y comparar experimentalmente las cuatro maneras que hemos presentado de combinar información local para estimar la pelota, así como comprobar si merece la pena incorporar un algoritmo de percepción distribuida. Las pruebas están organizadas en varios apartados, según su objetivo y los parámetros a medir. Los objetivos principales serán obtener medidas de precisión, estabilidad y comportamiento ante situaciones anómalas como oclusiones o falsos positivos en un equipo de varios robots. Para cada prueba se expondrán los resultados obtenidos con todas las implementaciones realizadas y se extraerán las conclusiones oportunas. Las observaciones visuales obtenidas en cada experimento son diferentes, aunque todas ellas del mismo orden. Todos los experimentos se han realizado teniendo en cuenta incertidumbre en la posición del observador. Se ha configurado el algoritmo de auto-localización con 200mm. de incertidumbre en el eje X, 200mm. de incertidumbre en el eje y y 18º de incertidumbre angular. En consecuencia, los experimentos integran las dos fuentes posibles de error en una situación real.

Como muestra la figura 3.33, se han realizado experimentos tanto en entornos simulados como en escenarios reales y utilizando un equipo de tres robots aiBo y dos robots Nao. Todos los cálculos efectuados por los algoritmos están realizados a bordo de los robots. Las herramientas externas utilizadas, únicamente tienen como fin recibir la información de depuración procedente de los robots para visualizar ciertos aspectos de los algoritmos o activar/desactivar los mecanismos de registro de datos en los robots (*log*).

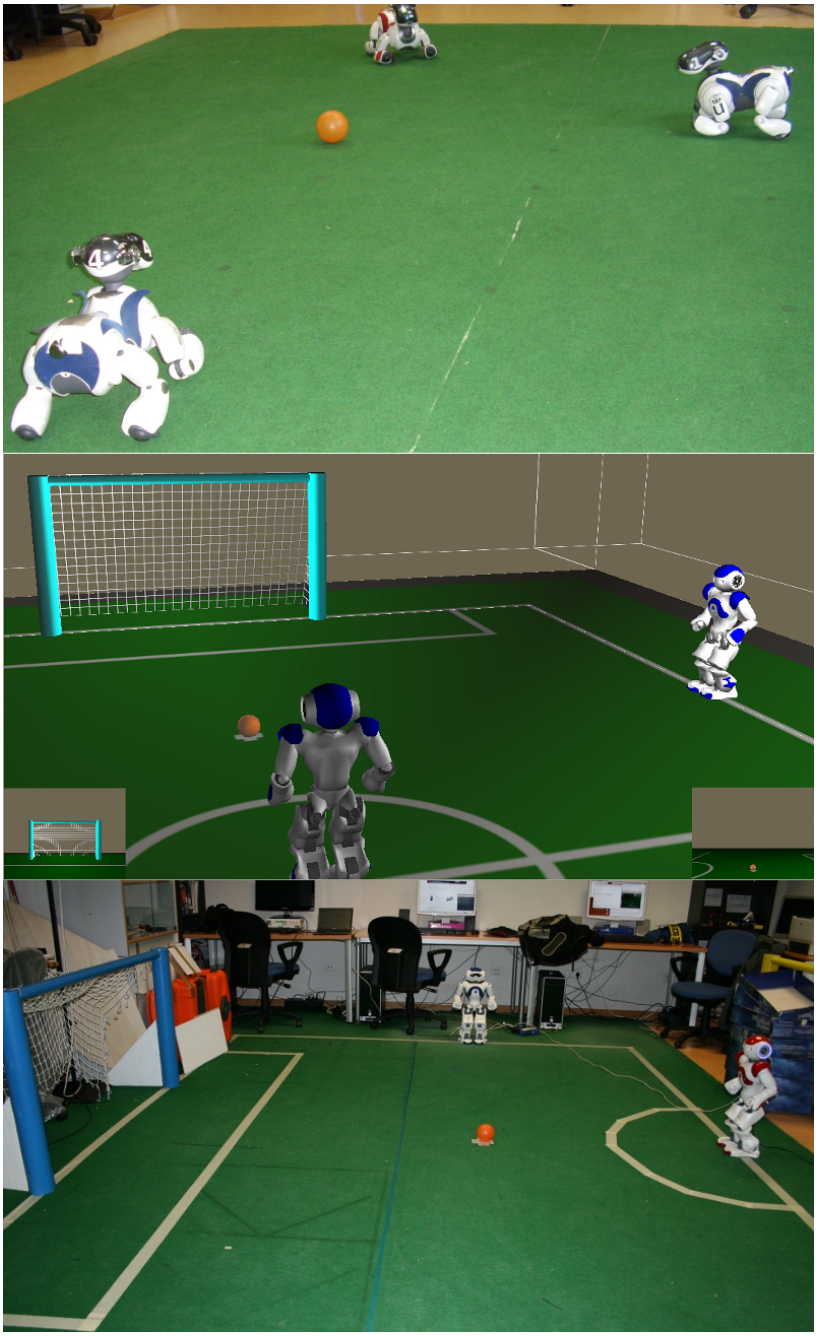


Figura 3.33: Diversos instantes de los experimentos de percepción distribuida tanto en simulación como en entorno real con las dos plataformas utilizadas: Robot aiBo y robot Nao

3.3.1. Precisión

Un primer parámetro a medir consiste en comparar los dos tipos de estimaciones, las individuales y la compartida, para evaluar si merece la pena intercambiar y compartir información. La percepción distribuida será única a cada instante, mientras que estimaciones individuales habrá tantas como robots diseminados por el entorno. Para conseguir una comparación justa, se ha contrastado la media del error en todas las estimaciones individuales con la estimación distribuida.

El montaje del experimento con los robots aiBo es el mostrado en la figura 3.34, donde podemos observar tres robots que perciben la pelota en una posición concreta del campo. Durante el experimento las posiciones de los robots permanecen invariables, mientras que la pelota se coloca en cuatro puntos diferentes, que diferentes zonas del campo. En la figura 3.34 el campo visual de los robots aparece sombreado y la pelota es el objeto de color naranja. En un color naranja más suave se indican las otras tres posiciones que ha ocupado la pelota en el experimento. Las gráficas que se muestran a continuación reflejan la media de los datos acumulados durante las estimaciones en las cuatro posiciones de pelota. Para el caso de los robots Nao el experimento es exactamente igual pero empleando una pareja de robots.

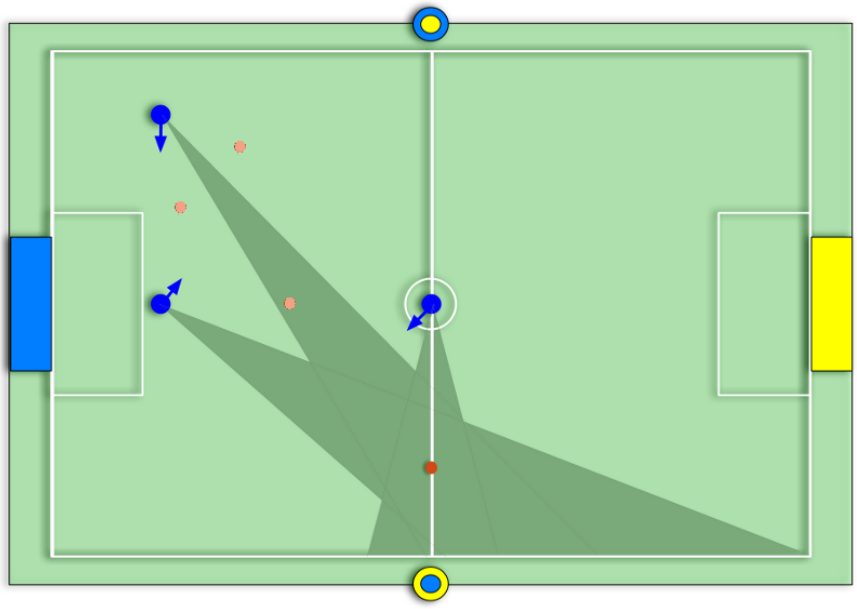


Figura 3.34: Montaje experimental para pruebas de precisión

Resultados obtenidos con fusión analítica

En la figura 3.35 se muestran los resultados que resumen este experimento empleando el algoritmo de fusión analítica. Las líneas discontinuas corresponden a las estimaciones individuales de cada robot. La línea roja representa la media del error de todas las estimaciones individuales. A su vez, la línea de color turquesa representa el error de la estimación compartida. Cuando hablamos de error nos referimos a la distancia euclídea en milímetros entre la estimación propuesta y la posición real de la pelota.

Del análisis de esta gráfica, más que fijarnos en el valor concreto del error, que depende de las posiciones relativas de pelota y robots, interesa ver la tendencia entre la estimación distribuida y las estimaciones individuales. Si eligiéramos otro conjunto de puntos para la posición de la pelota los valores del error diferirían de los mostrados en la gráfica, pues a mayores distancias entre pelota y robot se acentúan las fluctuaciones provocadas por cambios de iluminación, pequeñas vibraciones de los robots, etc. Sin embargo, la tendencia entre la estimación compartida y las indivi-

duales es la misma. Este experimento es, por tanto, significativo del comportamiento entre estimaciones individuales y compartida.

Como se puede ver en la figura, la estimación distribuida es incluso mejor que la mejor estimación local, cometiendo durante todo el transcurso del experimento menor error que la media de las estimaciones individuales. Como hemos analizado durante la fase de obtención del modelo de observación, la incertidumbre individual de la estimación es mayor en distancia que en ángulo. El hecho de que el error en ángulo se tan bajo provoca que cuando disponemos de un equipo de varios robots percibiendo la pelota desde ángulos distintos, la estimación distribuida sea incluso más fiable que cualquier estimación individual.

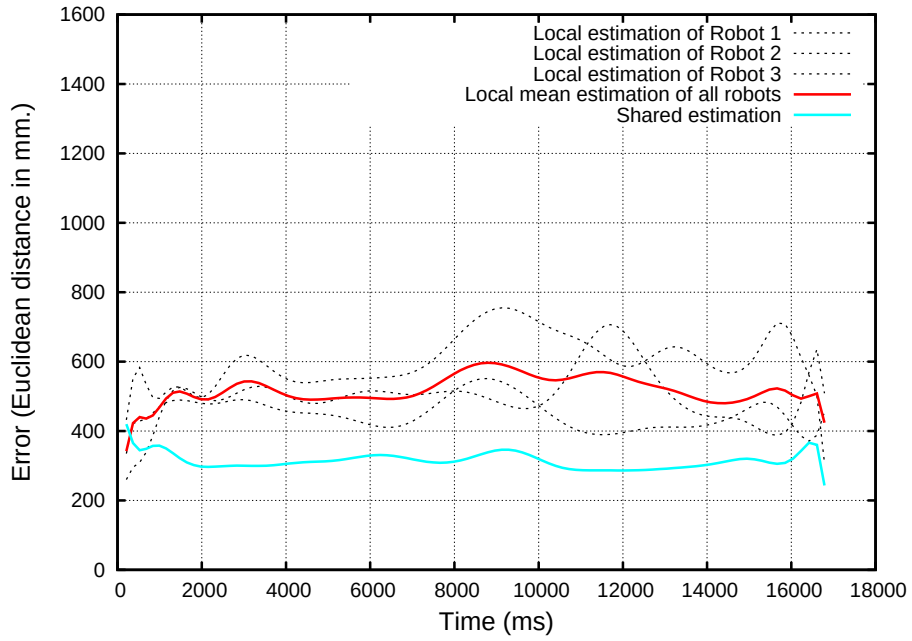


Figura 3.35: Comparación del error cometido en la estimación local y la estimación compartida mediante fusión analítica

Resultados obtenidos con fusión de rejillas probabilísticas

Para este experimento se ha configurado un tamaño de celda de 20 cm. Los resultados demuestran que la percepción distribuida mediante rejillas probabilísticas es beneficiosa. Los datos reflejan que el error de la estimación compartida es menor que la media del error individual durante prácticamente todo el experimento. La media del error local se encuentra aproximadamente en 550mm, mientras que la media del error en la estimación distribuida es de 450mm., un 20 % menos.

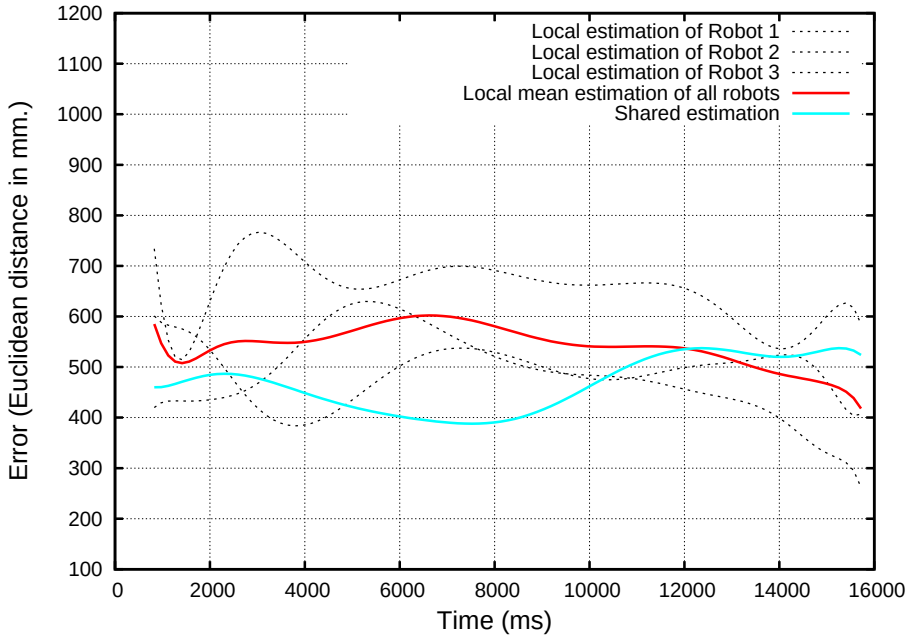


Figura 3.36: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en rejilla probabilística

Resultados obtenidos con fusión basada en filtro de Kalman

La figura 3.37 muestra los resultados obtenidos durante la percepción de la pelota en sus diferentes posiciones del campo. A la vista de los resultados podemos inferir que el error medio cometido por la percepción local es de aproximadamente 500mm. La aproximación de combinación empleando Kalman comete un error medio de aproximadamente 300mm., es decir, 20 cm. menos de error. La estabilidad de la estimación es relativamente buena y exenta de las fluctuaciones que sí tienen algunas estimaciones locales. El motivo de este comportamiento es la incercia y continuidad temporal del filtro de Kalman que amortigua las pequeñas desviaciones de la media manteniendo la estimación más cerca de la posición ideal.

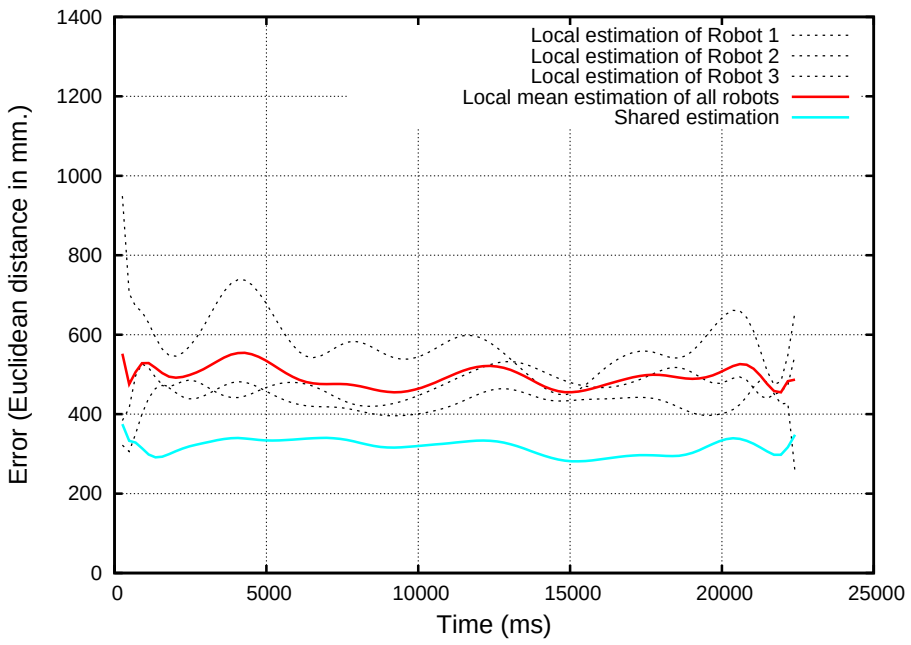


Figura 3.37: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman

Resultados obtenidos con fusión de filtros de partículas

La imagen 3.38 resume los resultados de esta prueba. El error medio de aproximadamente 500mm. en las estimaciones individuales se vuelve a repetir. La media del error utilizando esta técnica de combinación se encuentra alrededor de 200mm., unos 30cm. de mejora media respecto a la alternativa local. Se han utilizado 200 partículas en cada filtro que ha intervenido en el experimento.

Los datos muestran resultados parecidos a los obtenidos con la técnica basada en filtro de Kalman. El carácter acumulativo del filtro provoca una mejor estimación de la pelota, pues observaciones que puntualmente están alejadas de la media de la estimación son amortiguadas y inducen a menores errores. La precisión es aún mejor que la variante basada en Kalman y la razón fundamental es el comportamiento de los díscolos. La observación en la técnica de Kalman se compone ponderando las estimaciones individuales de cada robot. Aunque las estimaciones individuales de mayor incertidumbre influyan menos en la estimación compartida instantánea que las de baja incertidumbre, algo siempre arrastran la media. En la alternativa basada en filtro de partículas la mayor masa crítica que se obtienen de hipótesis procedentes de robots que aciertan en sus estimaciones inhibe en mayor medida la de los robots que opinan de manera diferente.

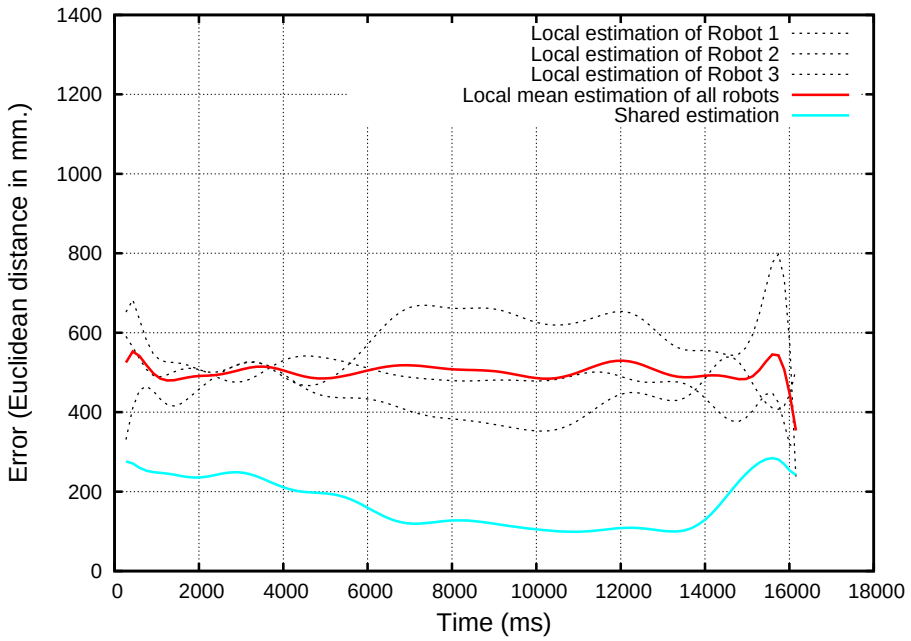


Figura 3.38: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de partículas

Resultados obtenidos con el robot Nao

Por cuestiones de implementación, en el robot Nao si ha sido posible ejecutar en paralelo varias técnicas de estimación distribuida. De esta manera podemos comparar los resultados obtenidos sobre los mismos datos de entrada. Con esta plataforma se han realizado los mismos experimentos que con el robot aiBo con las variantes basadas en filtro de Kalman y filtro de partículas por ser éstas las más prometedoras.

Mas allá de los valores concretos de error, los datos de la gráfica 3.39 no hacen sino corroborar las impresiones que teníamos durante los experimentos con el robot aiBo. En el simulador las dos técnicas de estimación distribuida mejoran en precisión los números obtenidos por los estimadores individuales y la variante basada en Monte Carlo es la alternativa que más se acerca a la posición ideal de la pelota.

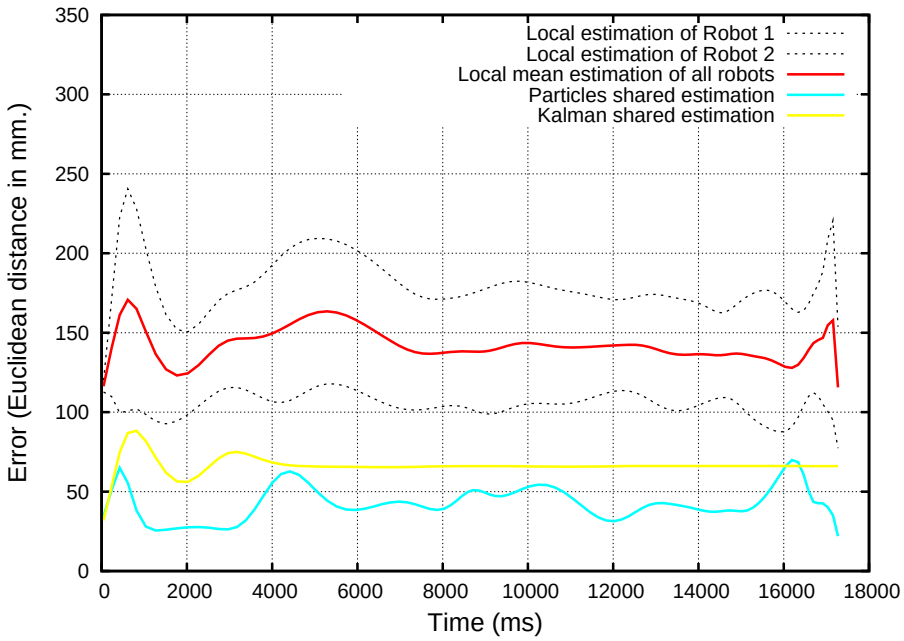


Figura 3.39: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

Los resultados obtenidos con las pruebas sobre los Nao reales (gráfica 3.40) vuelven a confirmar la tónica mostrada durante los experimentos anteriores de precisión: combinar compensa pues se obtiene mayor precisión en la estimación con ambos métodos que con las variantes individuales. Las implementaciones de los estimadores en el robot Nao arrojan las mismas conclusiones que en el robot aiBo: la variante que utiliza filtro de partículas continúa por encima de la que emplea filtros de Kalman en cuanto a precisión se refiere.

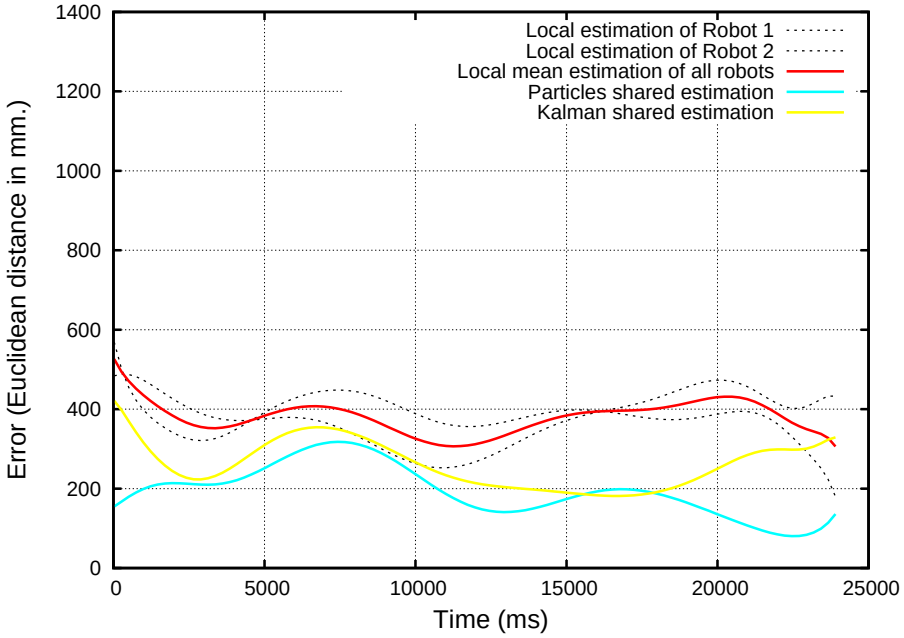


Figura 3.40: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

3.3.2. Mediciones de eficiencia computacional

Otro parámetro interesante a la hora de caracterizar cada una de las técnicas de fusión distribuida desarrolladas es la eficiencia computacional. De nada sirve un algoritmo muy preciso si requiere un tiempo de cómputo inabordable. La tabla 3.5 resume el tiempo que emplea cada algoritmo en completar una iteración completa, es decir, generar una nueva estimación. Estos valores corresponden a tiempos medios calculados tras muchas iteraciones. Puesto que en el robot aiBo han sido implementados los cuatro algoritmos completos, los tiempos que se muestran son los obtenidos en esta plataforma.

Los resultados muestran que el métodos analítico puro y el basado en filtro de Kalman son significativamente los más rápidos. La simplificación que asumen ambos estimadores supone una ventaja notable en cuanto a tiempo de cómputo. La alterna-

Técnica de fusión	**Tiempo por iteración**
Analítica	0.25 ms.
Rejilla probabilística (celdas de 200 x 200 mm.)	700 ms.
Filtro de Kalman	0.35 ms.
Filtro de partículas (200 partículas)	25 ms.

Cuadro 3.5: Tiempos de cómputo por iteración en cada una de las técnicas de fusión distribuida desarrolladas

tiva discretizada muestra los peores números con tamaños de celda medio-altos. Es cierto que esos números podrían mejorarse a costa de perder resolución en la rejilla, pero entonces dispondríamos de una escasa precisión en la estimación. En un puesto intermedio encontramos la técnica que apuesta por un filtro de partículas para fusionar la información. Sus tiempos alcanzados, aunque convierten en usable la técnica, están dos órdenes de magnitud por encima de las variantes analíticas.

La figura 3.41 muestra cómo es el reparto de tiempo de procesador dentro del algoritmo de combinación basado en filtro de partículas. La etapa más costosa es la fase de combinación que consume el 59 % del tiempo de cada iteración. La razón principal es su complejidad $O(3 \cdot n^2)$ puesto que por cada hipótesis se necesitan evaluar los conjuntos de partículas recibidos por los tres miembros del equipo.

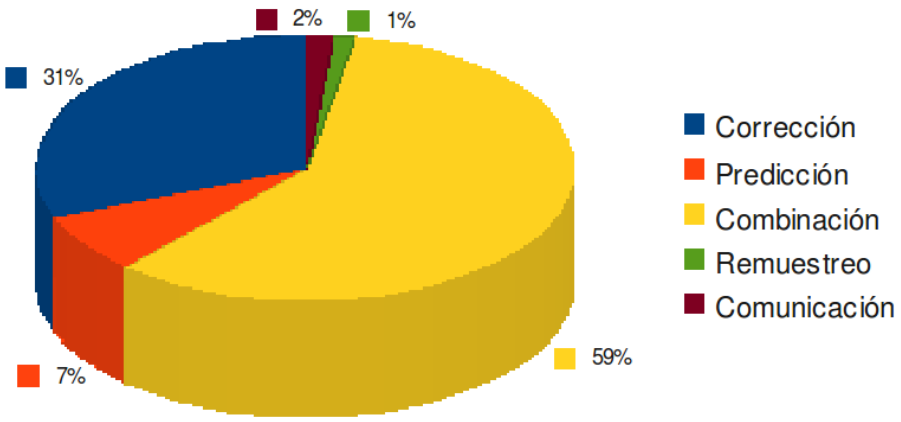


Figura 3.41: Distribución del consumo de CPU dentro de cada iteración

3.3.3. Comportamiento ante oclusiones en uno o varios robots

El objetivo de este experimento es provocar oclusiones en uno o varios robots, para comprobar su impacto en la estimación compartida de la pelota. Es habitual durante los partidos en la RoboCup que un jugador contrario se cruce entre un robot y la pelota, obstaculizando la visión de la misma. Es deseable que las técnicas de percepción distribuida fueran robustas a estas situaciones tolerando el mayor número de oclusiones posible.

Por cada técnica se han realizado dos experimentos: La primera de ellas limitando el rango perceptivo de uno de los robots y la segunda dejando que únicamente un robot perciba la pelota.

La figura 3.42 muestra la configuración de los experimentos, donde los círculos negros representan barreras que impiden a los robots percibir la pelota.

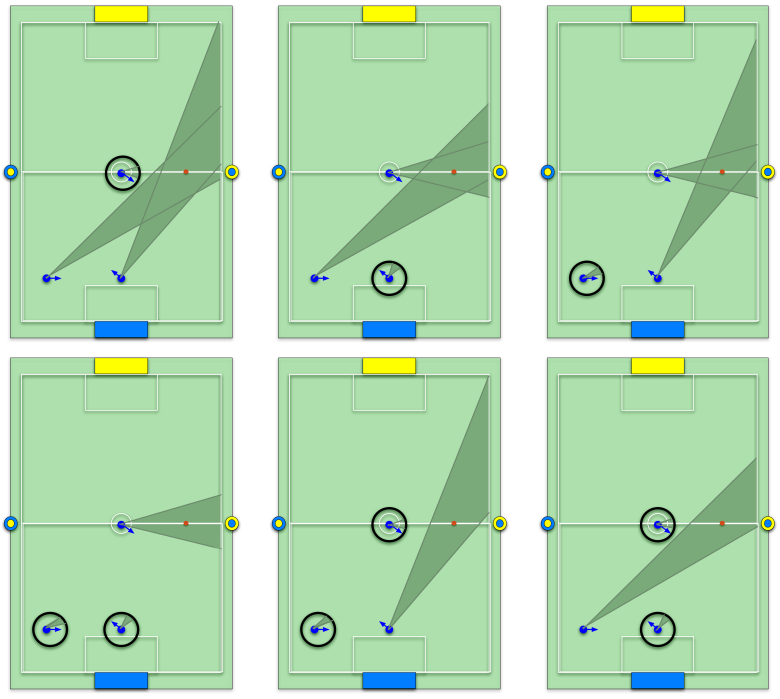


Figura 3.42: Montaje para pruebas de oclusiones en las observaciones de los robot

Resultados obtenidos con fusión analítica

Los datos muestran que el impacto de la oclusión en la estimación compartida es apenas apreciable. Puesto que siempre hay al menos un robot que percibe la pelota con calidad, la estimación distribuida toma prácticamente la misma hipótesis que la estimación individual de dicho robot. La incertidumbre del robot que no detecta la pelota es infinita y esto provoca que apenas influya en la estimación combinada.

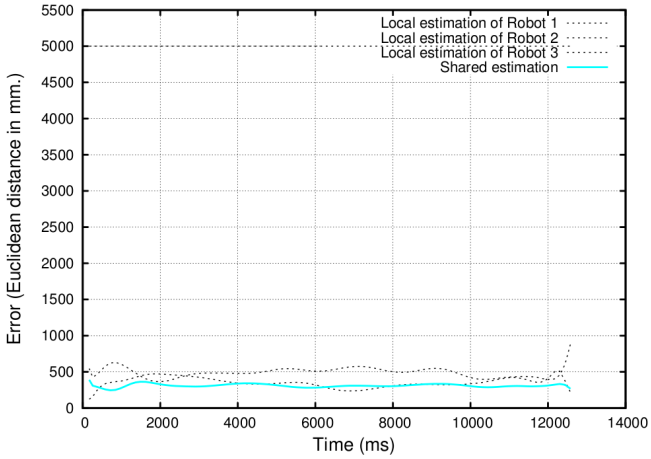


Figura 3.43: Resultados obtenidos ante oclusiones en un robot

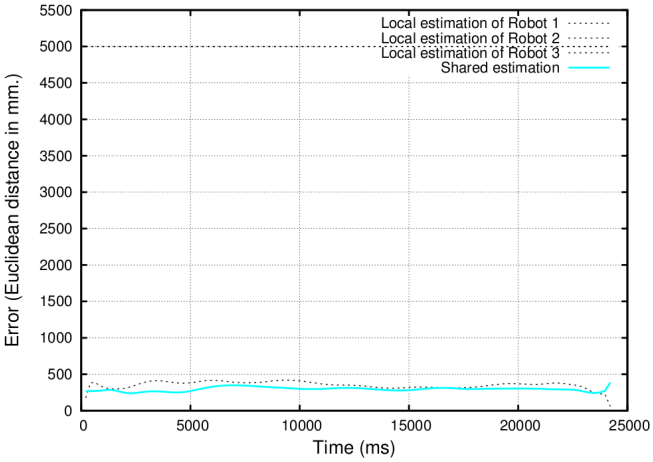


Figura 3.44: Resultados obtenidos ante oclusiones en dos robots

Resultados obtenidos con fusión de rejillas probabilísticas

Los resultados con esta técnica son similares a los obtenidos anteriormente. Puesto que siempre existe uno o dos robots que perciben la pelota, la estimación compartida pondera estas observaciones con mucha más fuerza que la de los robots que no perciben la pelota. La estimación distribuida, por tanto, tolera bien este tipo de situaciones siempre que exista en el equipo al menos una estimación local de la pelota.

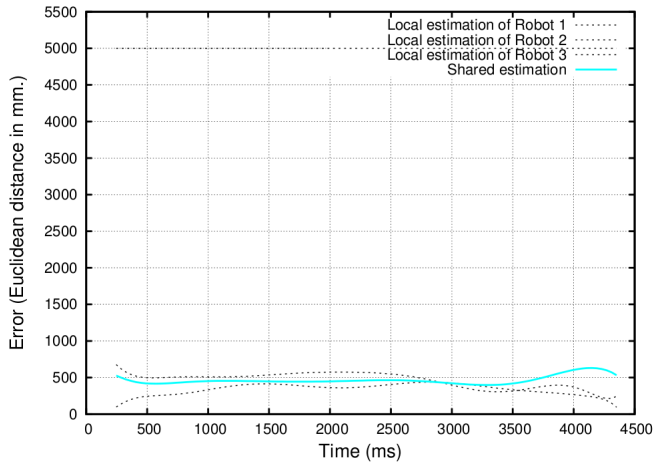


Figura 3.45: Resultados obtenidos ante oclusiones en un robot usando rejillas

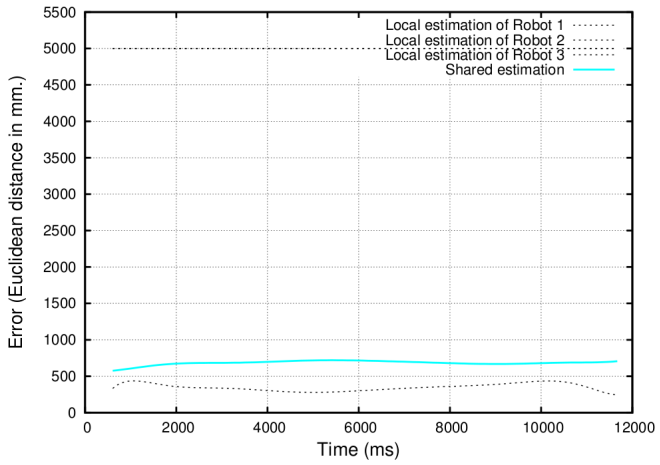


Figura 3.46: Resultados obtenidos ante oclusiones en dos robots usando rejillas

Resultados obtenidos con fusión basada en filtro de Kalman

De nuevo, los resultados experimentales muestran la buena tolerancia a los falsos negativos de la fusión basada en filtro de Kalman. Una o dos estimaciones con alta incertidumbre no afectan a la estimación compartida, siempre que ésta reciba observaciones de calidad de alguno de los robots que sí percibe la pelota.

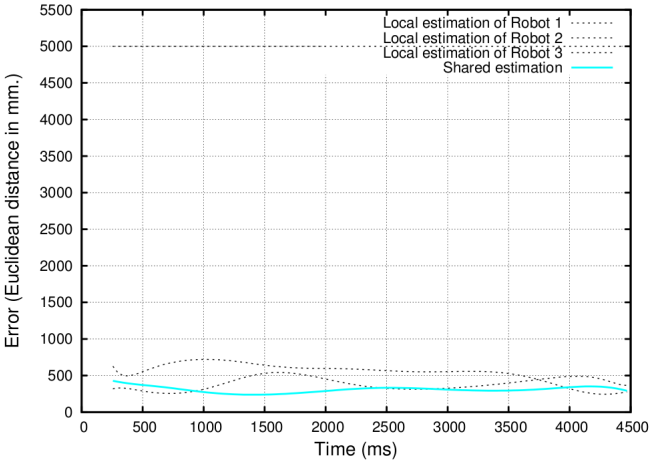


Figura 3.47: Resultados obtenidos ante oclusiones en un robot utilizando filtro de Kalman

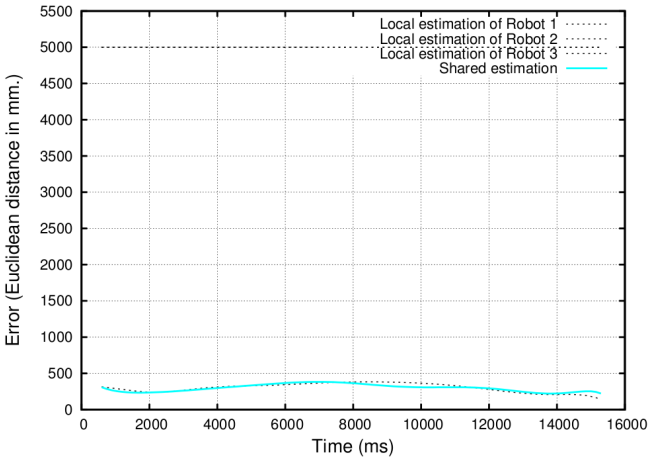


Figura 3.48: Resultados obtenidos ante oclusiones en dos robots utilizando filtro de Kalman

Los experimentos nos muestran que no hay diferencias significativas entre que uno, dos o un grupo n de robots no localicen la pelota, siempre que haya al menos uno que si lo haga.

Resultados obtenidos con fusión basada en filtro de partículas

Las pruebas demuestran que ante oclusiones existe una gran ventaja utilizando el enfoque colaborativo: los robots que no perciben directamente la pelota, ahora sí son capaces de mantener una estimación de ella. Este aporte supone un gran avance en estos robots y queda patente en los resultados obtenidos, al igual que en el resto de técnicas implementadas.

Los robots que no perciben la pelota comunican una FDP individual con las muestras aproximando una distribución uniforme. En el paso de corrección del filtro de partículas distribuido se comprueba la densidad de partículas recibidas de los robots compañeros dentro de un área determinado alrededor de cada hipótesis. Aquellos robots que no perciban la pelota raramente acumularán masa crítica en alguna zona concreta y su influencia estará *diluida* por todo el espacio de búsqueda. Son los robots que sí vean la pelota los que tienen una acumulación de partículas en alguna zona del campo de juego y potencian que ganen importancia las partículas del filtro distribuido que se encuentran en posiciones cercanas.

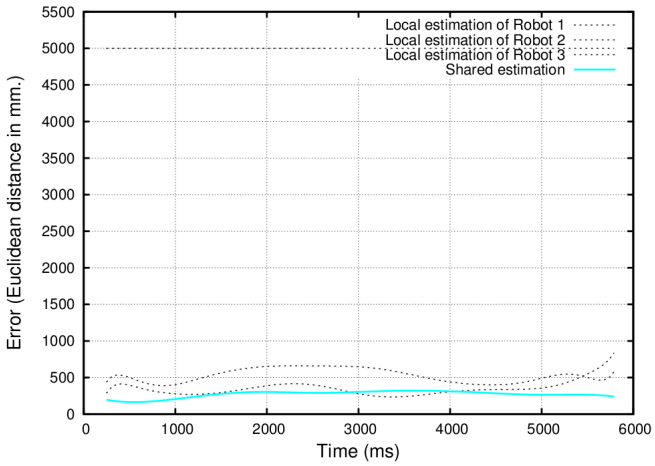


Figura 3.49: Resultados obtenidos ante oclusiones en un robot utilizando filtro de partículas

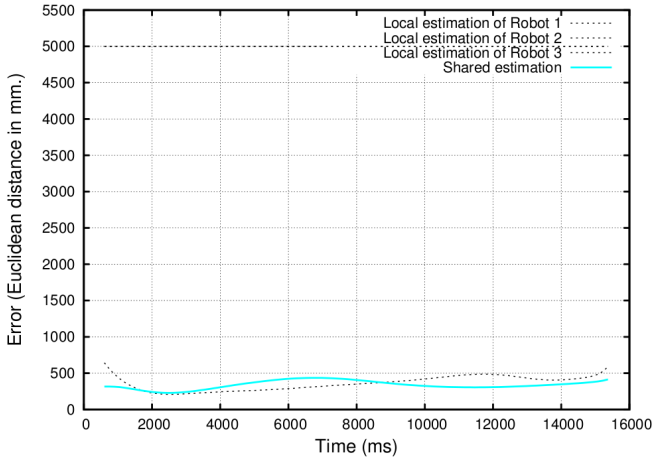


Figura 3.50: Resultados obtenidos ante oclusiones en dos robots utilizando filtro de partículas

Resultados obtenidos con el robot Nao

Las gráficas obtenidas tanto en simulación (imagen 3.51) como con el robot real (imagen 3.52) revalidan de nuevo la hipótesis de que un único robot es suficiente para mantener una estimación de la pelota. Recordemos que en los experimentos con los robots Nao únicamente hay una pareja de robots y en este caso sólo uno percibe la pelota.

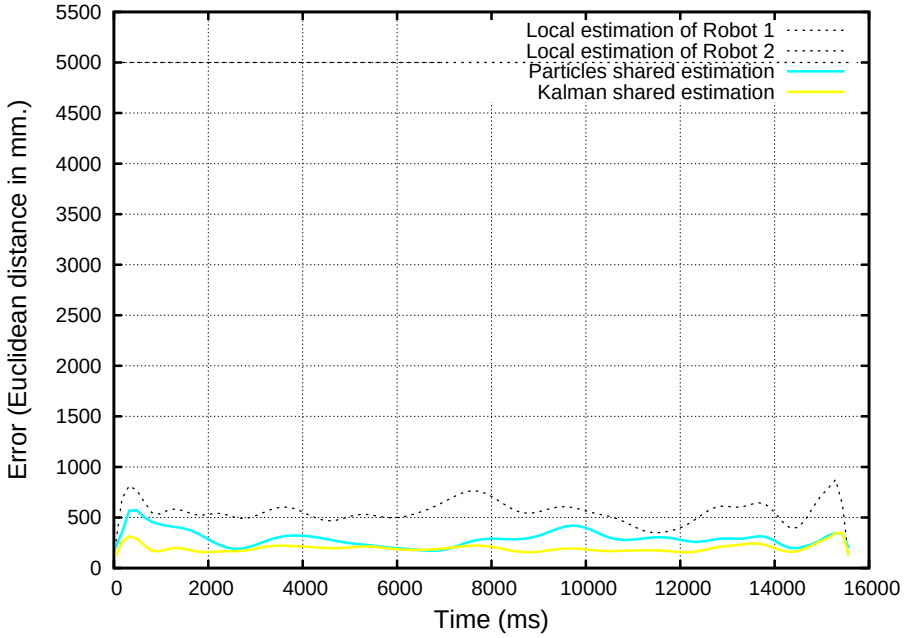


Figura 3.51: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

3.3.4. Comportamiento ante falsos positivos

El objetivo de este experimento es medir cómo afectan los falsos positivos a la estimación compartida. Los falsos positivos son estimaciones erróneas de la pelota provocadas por cambios de iluminación, objetos situados cerca del campo de juego y de apariencia parecida a la pelota, etc.

Por cada implementación disponible hemos realizado cuatro experimentos. El primero de ellos persigue comprobar cuál es el impacto en la estimación compartida ante un falso positivo con baja incertidumbre. La segunda prueba también simula un falso positivo pero ahora con mayor incertidumbre. El tercer experimento provoca no uno sino dos falsos positivos de incertidumbre media-baja. Todos estos falsos positivos son persistentes en el tiempo, es decir, durante todo el experimento alguno de los robots mantiene su estimación errónea. El último experimento de esta sección busca

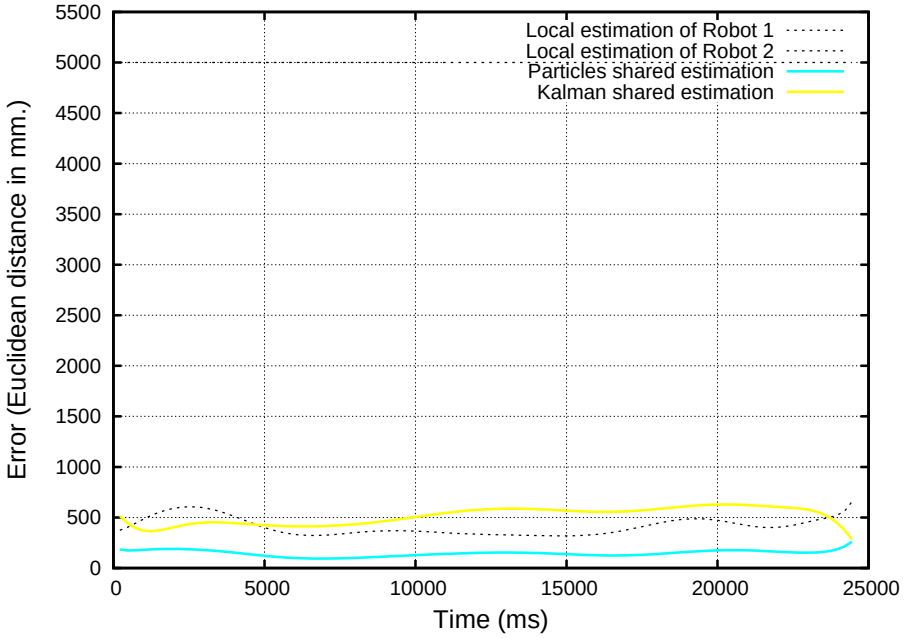


Figura 3.52: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

caracterizar el comportamiento de la estimación distribuida ante un falso positivo esporádico, es decir, puntual y no persistente en el tiempo. Los falsos positivos han sido recreados con pelotas de juego adicionales.

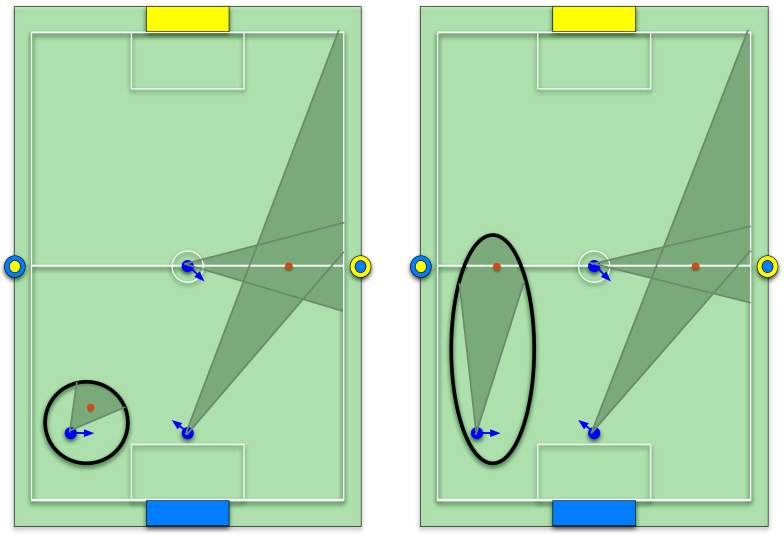


Figura 3.53: Montaje experimental para pruebas de falsos positivos

Resultados obtenidos con fusión analítica

Los resultados extraídos de las pruebas de comportamiento ante un falso positivo demuestran que este tipo de errores no son bien tolerados por la técnica analítica. Un falso positivo de baja incertidumbre arrastra drásticamente la estimación compartida aumentando considerablemente el error y situándole por encima del error medio local. A su vez, si la incertidumbre del falso positivo es mayor, su menor peso en la estimación distribuida provoca que exista menor error, situándose éste ligeramente por debajo del error medio local.

Si el equipo cuenta con dos robots percibiendo falsos positivos, por la misma razón anterior la estimación compartida de la pelota pierde mucha precisión y su error aumenta considerablemente. En el experimento llega a alcanzar aproximadamente 1400mm. de error, es decir, una estimación muy alejada de la posición real de la pelota.

El comportamiento ante falsos positivos esporádicos tampoco es muy favorable, pues se observa cómo la estimación compartida registra picos de mayor error coincidiendo con las observaciones erróneas. La razón principal de este comportamiento es la naturaleza instantánea de este mecanismo de fusión.

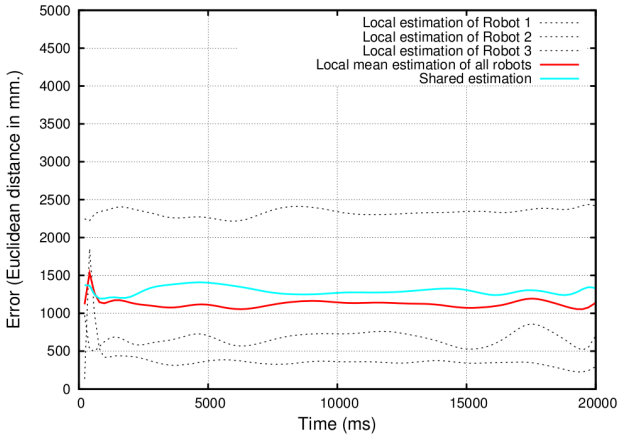


Figura 3.54: Resultados obtenidos ante un falso positivo con baja incertidumbre

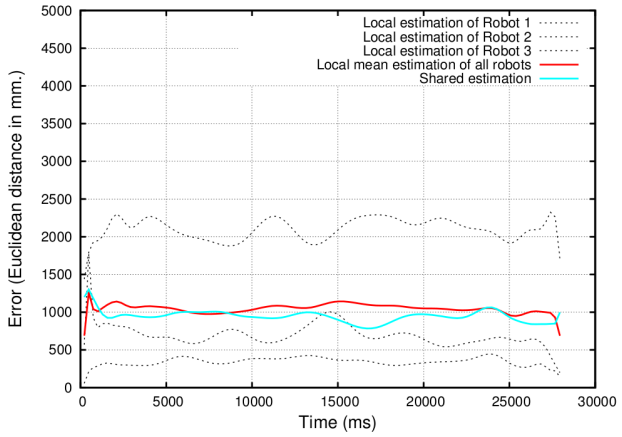


Figura 3.55: Resultados obtenidos ante un falso positivo con alta incertidumbre

Resultados obtenidos con fusión de rejillas de probabilidad

El comportamiento registrado por la técnica de fusión de rejillas de probabilidad muestra resultados similares a la técnica analítica. El falso positivo de menor incertidumbre es más dañino para la estimación compartida que el de mayor incertidumbre. Ambos son mal tolerados por el algoritmo, pues no es capaz de diferenciar un falso positivo de una pelota de verdad.

El análisis de la prueba con dos falsos positivos ratifica que ante varias estimaciones no compatibles de la pelota, el algoritmo resuelve una posición intermedia

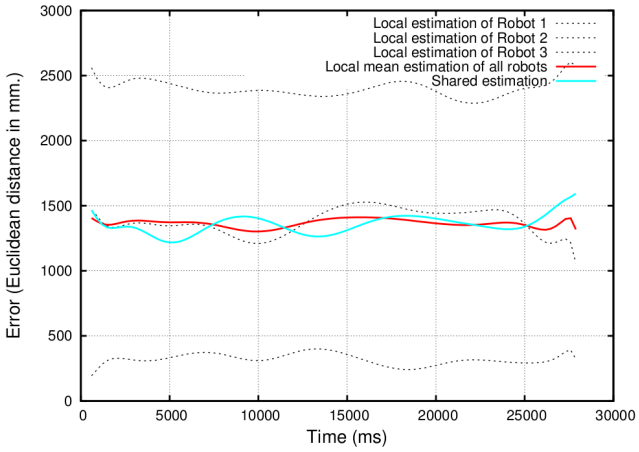


Figura 3.56: Resultados obtenidos ante dos falsos positivos con incertidumbre media

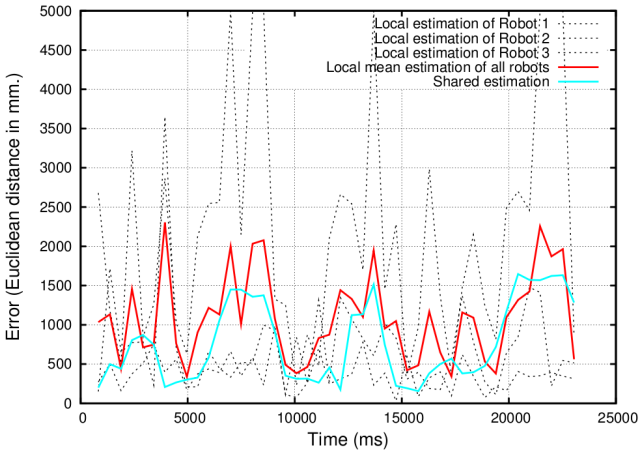


Figura 3.57: Resultados obtenidos ante cuatro falsos positivos puntuales

siempre que las estimaciones sean de incertidumbres parecidas, pues se realiza una media ponderada entre todas las celdas para obtener la posición de la pelota. Por esta razón el error es considerable y los falsos positivos hacen mucho daño a la estimación distribuida de la pelota. Puesto que esta técnica también combina de manera instantánea las observaciones procedentes de cada robot, no es capaz de amortiguar los picos de error producidos por los falsos positivos, como muestra la última gráfica.

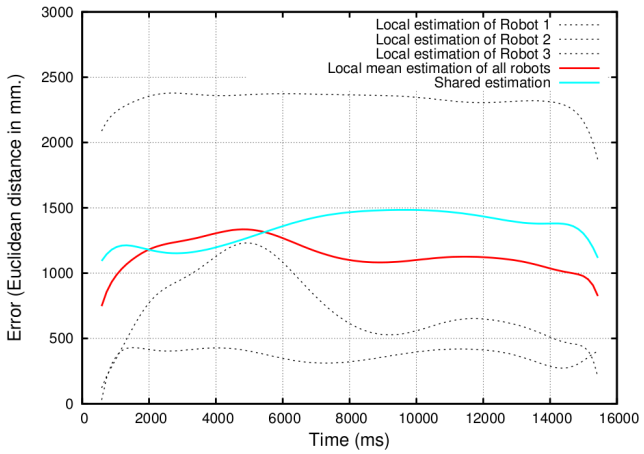


Figura 3.58: Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con baja incertidumbre utilizando rejillas probabilísticas

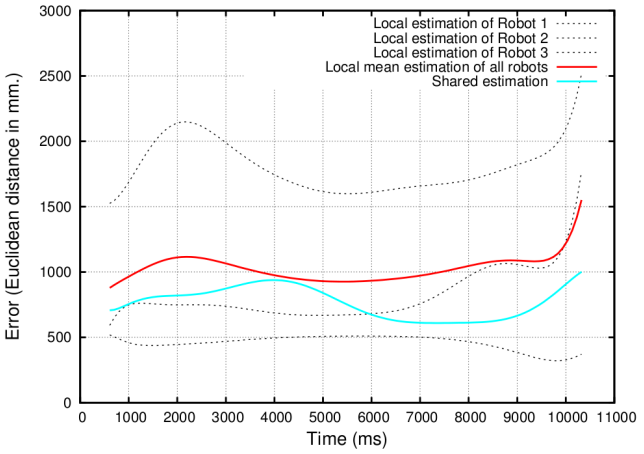


Figura 3.59: Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con alta incertidumbre utilizando rejillas probabilísticas

Resultados obtenidos con fusión basada en filtro de Kalman

De nuevo observamos que este algoritmo de fusión tampoco admite bien los falsos positivos en uno de sus robots. Se repite el fenómeno visto hasta el momento por el que un falso positivo con escasa incertidumbre provoca mayor error en la estimación distribuida que uno de mayor incertidumbre.

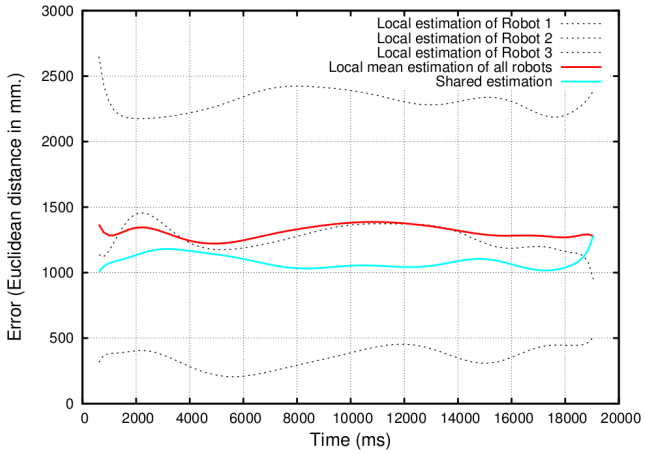


Figura 3.60: Resultados obtenidos ante dos falsos positivos con incertidumbre media

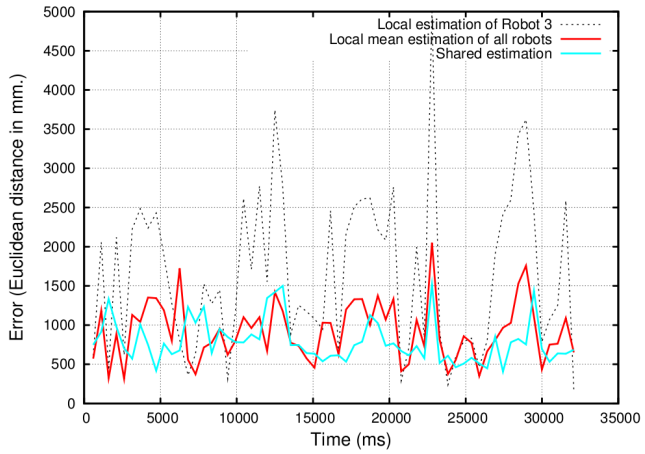


Figura 3.61: Resultados obtenidos ante un falso positivo puntual

Si examinamos la gráfica ante dos falsos positivos tampoco obtenemos ninguna sorpresa. El error resultante es incluso mayor que la media del error local y certifica el daño que hacen los falsos positivos a la fusión basada en filtro de Kalman.

En el caso de un falso positivo esporádico, este algoritmo sí se comporta de manera adecuada aplacando la estimación tan diferente procedente de alguno de los robots. La acumulación temporal de evidencias que realiza el filtro es la clave para protegerse de este fenómeno, bastante habitual en la RoboCup

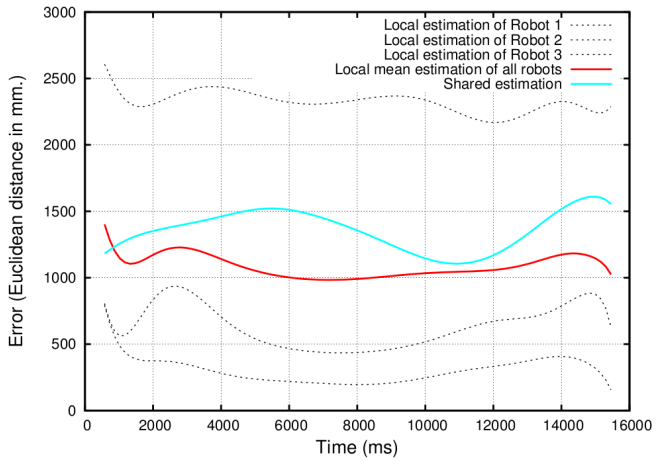


Figura 3.62: Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con baja incertidumbre utilizando un filtro de Kalman

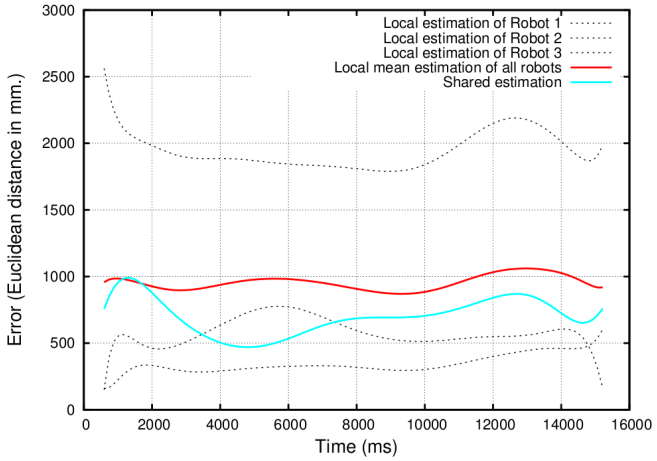


Figura 3.63: Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con alta incertidumbre utilizando un filtro de Kalman

Resultados obtenidos con fusión basada en filtros de partículas

La variante apoyada en filtro de partículas sí responde adecuadamente a un falso positivo independientemente de su incertidumbre. Su error se mantiene en márgenes razonables y siempre por debajo de la media del error local. La razón principal de este comportamiento es que la observación que alimenta el filtro contiene las partículas de cada una de las estimaciones locales. Puesto que existe mayoría de hipótesis en la

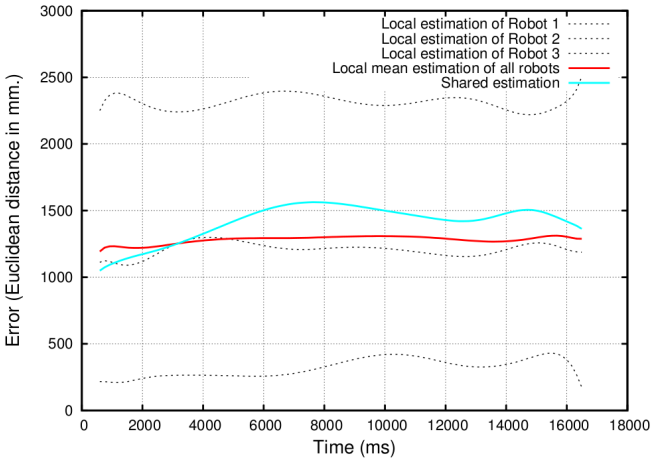


Figura 3.64: Resultados obtenidos ante dos falsos positivos con incertidumbre media

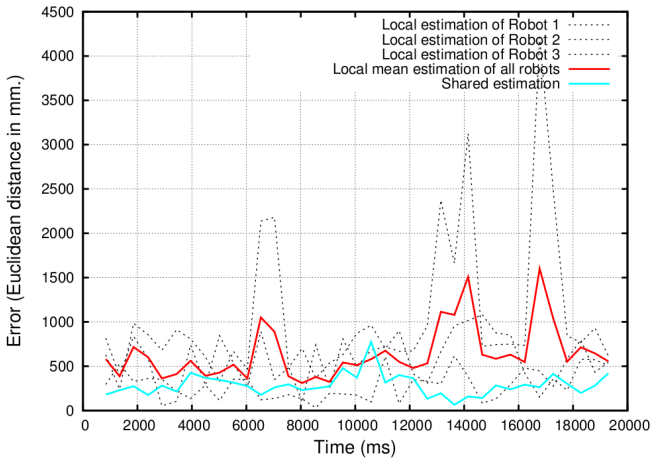


Figura 3.65: Resultados obtenidos ante un falso positivo puntual

zona de influencia de la pelota real, el falso positivo no arrastra la población de partículas del filtro distribuido. El carácter convergente del filtro de partículas empleado no permite que se mantengan dos hipótesis en régimen permanente, prevaleciendo la más *votada*. Esta es una ventaja sustancial de esta manera de combinar frente a las otras tres propuestas.

Ante dos falsos positivos el comportamiento deja de ser eficaz. Ahora no hay mayoría de partículas en una zona, sino que coexisten tres grandes focos donde se concentran las hipótesis de cada robot. La gráfica 3.68 muestra cómo el error de la

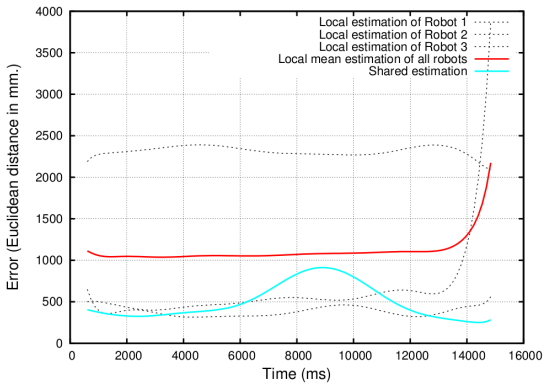


Figura 3.66: Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con baja incertidumbre utilizando filtros de partículas

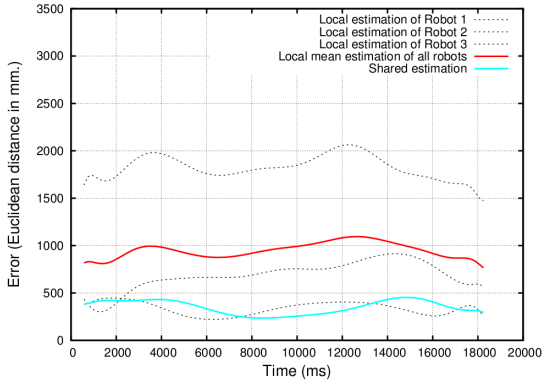


Figura 3.67: Comparación del error cometido entre las estimaciones locales y la estimación compartida ante un falso positivo con alta incertidumbre utilizando filtros de partículas

estimación compartida se acerca primero al de uno de los robots y después al error de otro. Esto es debido a que ambos robots, que sufren falsos positivos, cuentan con incertidumbres semejantes. Durante el experimento se produce un movimiento de partículas desde la estimación del primero a la posición estimada por el segundo. El robot que cuenta con la estimación verdadera no arrastra a las partículas de la estimación distribuida en ningún momento aunque su incertidumbre es menor que la de sus dos compañeros.

El comportamiento ante falsos positivos esporádicos es apropiado y lógico, pues el filtro de partículas también tiene una naturaleza acumulativa en su funcionamiento y es robusto a observaciones esporádicas erróneas.

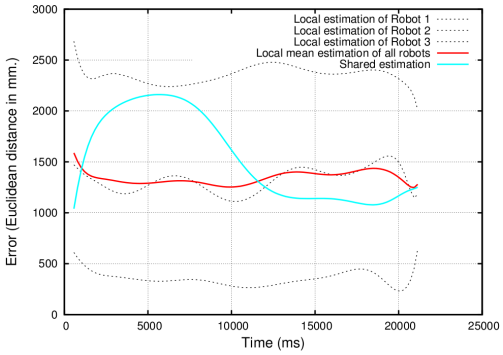


Figura 3.68: Resultados obtenidos ante dos falsos positivos con incertidumbre media

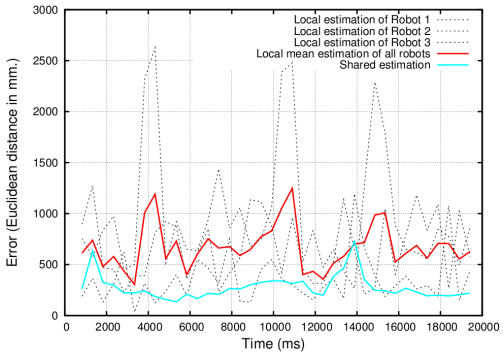


Figura 3.69: Resultados obtenidos ante tres falsos positivos puntuales

Resultados obtenidos con el robot Nao

Los datos extraidos del experimento provocando un falso positivo en un equipo de robots Nao son los mostrados en las figuras 3.70 y 3.71, siendo la primera de ellas la resultante de las pruebas en simulación y la segunda sobre el entorno real.

El falso positivo genera una estimación de incertidumbre media-baja en el robot que lo percibe y esa es la razón del acusado error en la estimación individual de uno de los robots. A pesar de tener menor incertidumbre la hipótesis del robot que percibe la pelota *auténtica*, ésta hipótesis no arrastra completamente la estimación distribuida en la variante basada en filtro de Kalman, sino que ésta también *escucha* al robot que se está equivocando, aunque sea en menor medida.

Tanto en simulación como con los robots reales la implementación que utiliza filtros de partículas es más *elitista* y se deja influenciar en mayor medida por las hi-

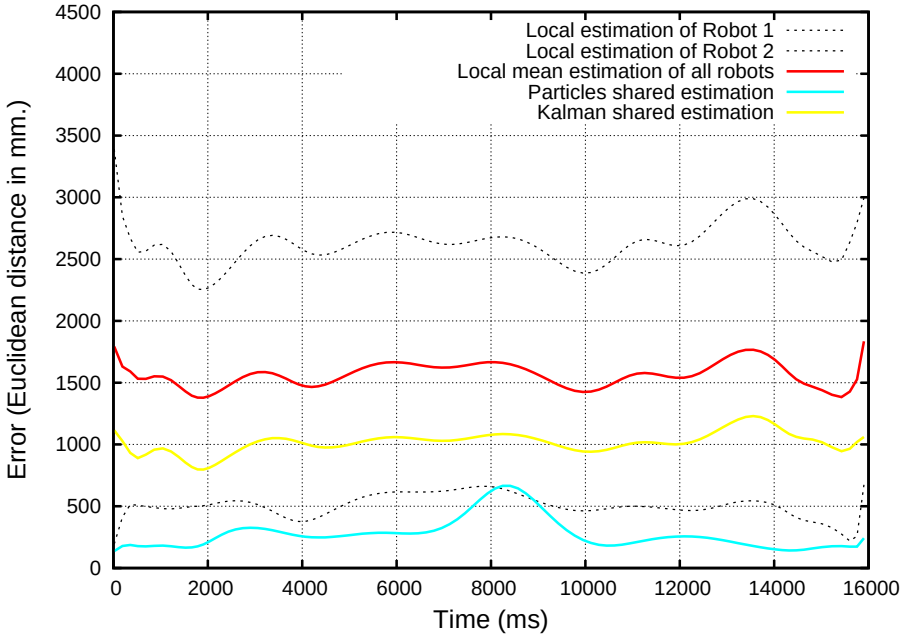


Figura 3.70: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

pótesis que tienen menos incertidumbre, descartando en mayor medida las opiniones de los demás. Recordemos que los filtros de partículas tienen a converger hacia una hipótesis y el *modo* hacia el que lo hacen es hacia el más probable. Es por esto que la hipótesis menos verosímil apenas modifica la creencia del filtro.

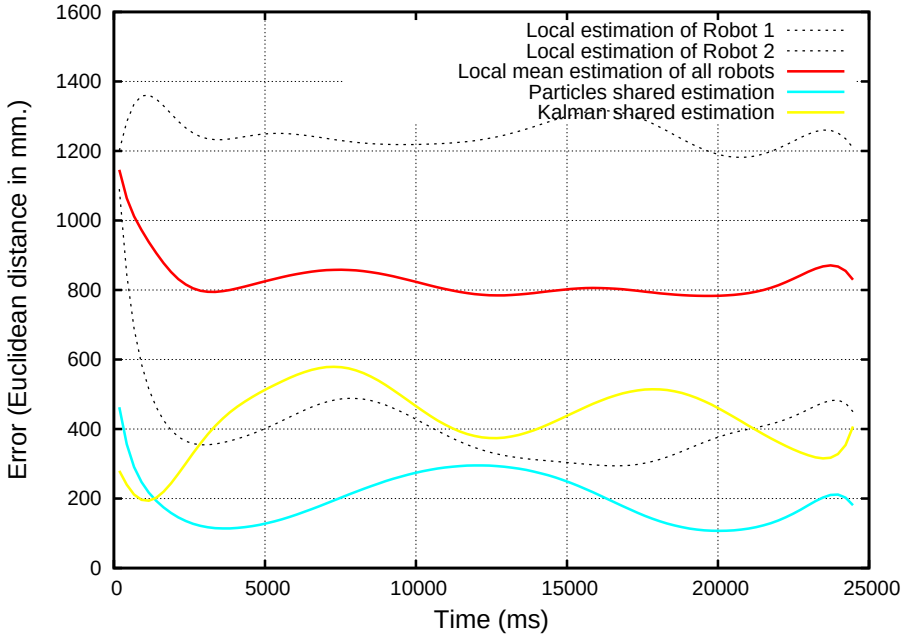


Figura 3.71: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

3.3.5. Comportamiento ante alta incertidumbre en la posición del observador

El objetivo de este experimento es verificar la tolerancia de los algoritmos de fusión sensorial a otra situación común en un partido de fútbol robótico: Que alguno de los robots no se encuentre localizado en el campo. Si esto sucede, la estimación local de la pelota contará con muy baja incertidumbre, pues se ve influida tanto por la incertidumbre de la propia observación, como por la incertidumbre en la posición del observador.

El experimento realizado es similar al llevado a cabo previamente para evaluar el comportamiento ante oclusiones. La diferencia radica en que ahora todos los robots perciben la pelota, pero uno o dos de ellos no están localizados en el campo.

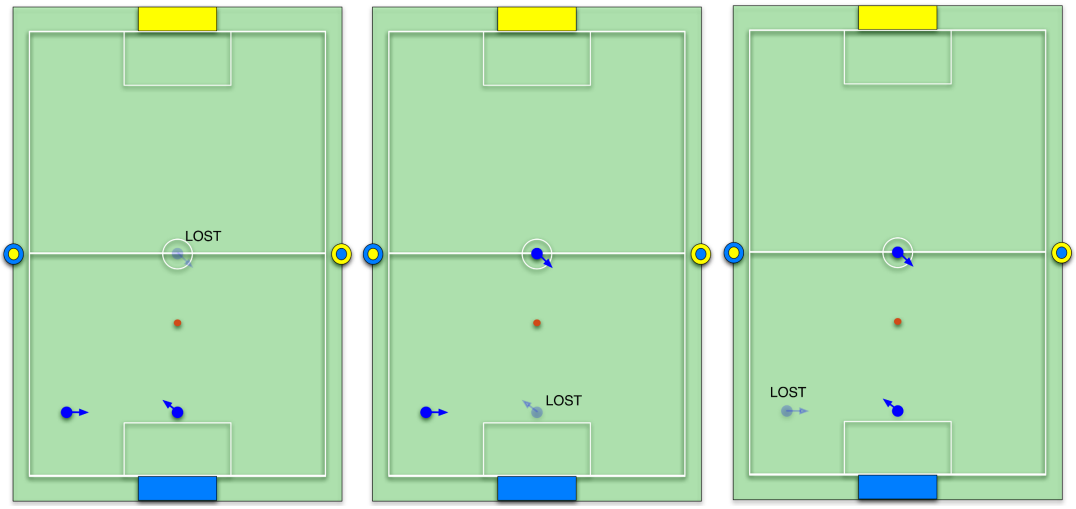


Figura 3.72: Escenario para experimentos con elevada incertidumbre en la posición del observador

Resultados obtenidos con fusión analítica

Las gráficas muestran un comportamiento similar al obtenido en las pruebas de oclusiones. Una alta incertidumbre del observador provoca una alta incertidumbre en su estimación individual absoluta de la pelota y, en consecuencia, que esa observación no tenga peso en la estimación distribuida.

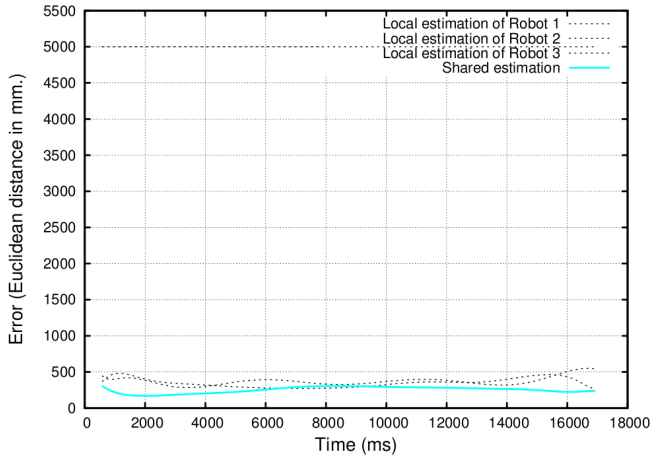


Figura 3.73: Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots

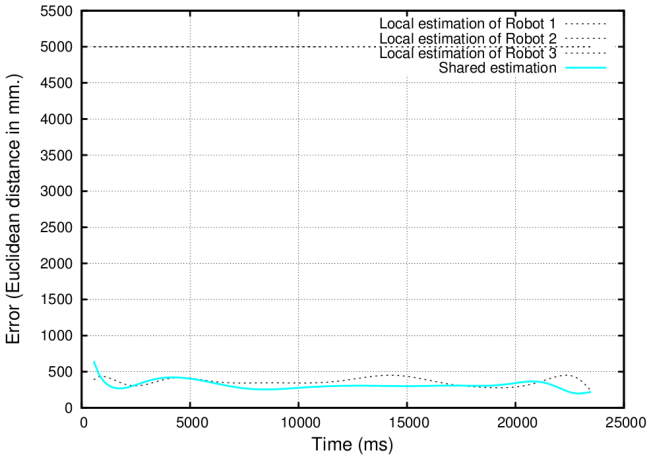


Figura 3.74: Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots

Resultados obtenidos con fusión de rejillas probabilística

El análisis de los experimentos con rejillas probabilísticas también arroja unos resultados positivos ante elevada incertidumbre en la posición del observador. Incluso en el caso de que existan dos robots completamente perdidos, sus malas estimaciones no afectan a la percepción compartida de la pelota.

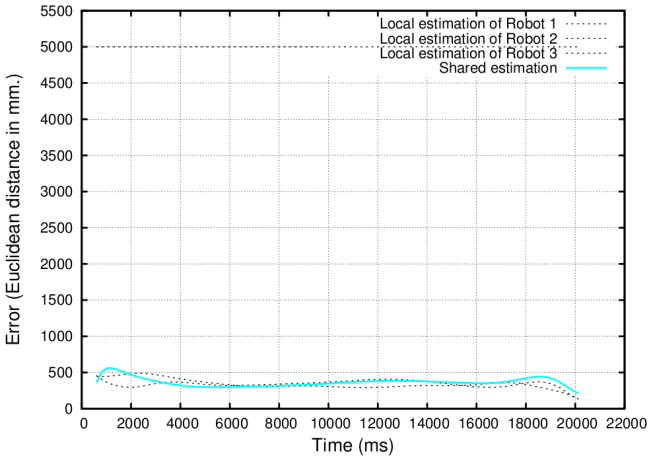


Figura 3.75: Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots

Las primeras conclusiones que podemos extraer de este experimento es que no hay diferencia entre que uno, dos o n robots no estén bien localizados, con tal de que haya al menos uno que lo esté y perciba la pelota.

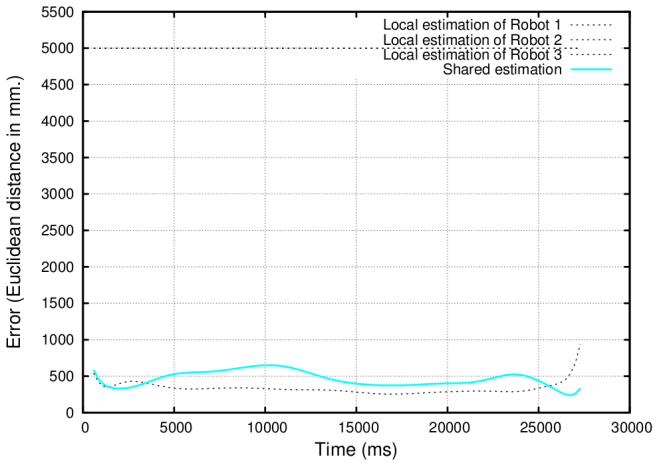


Figura 3.76: Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots

Resultados obtenidos con fusión basada en filtro de Kalman

Puesto que en esta técnica las observaciones instantáneas que alimentan el filtro de Kalman están ponderadas por la incertidumbre de cada estimación individual, la resultante de todas ellas siempre es una observación similar a la de los robots que si están bien localizados. Por esta razón, esta técnica se defiende bien ante robots no localizados o con escasa calidad de auto-localización, como muestran las gráficas.

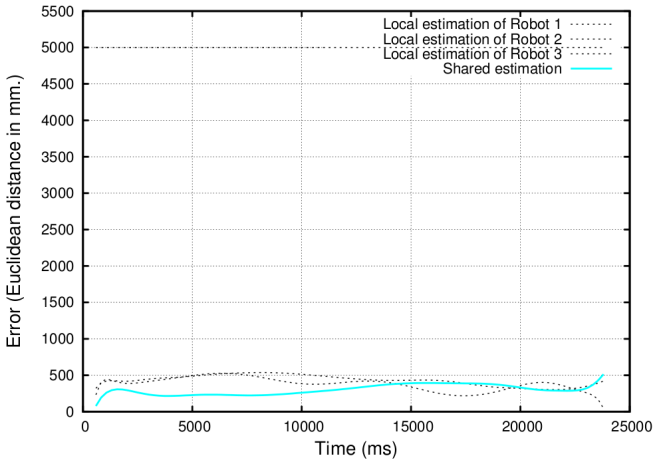


Figura 3.77: Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots

¿Qué sucede si un robot se encuentra mal localizado pera él piensa que conoce correctamente su posición? Esta situación es similar a la provocada por un falso positivo. Entre las FDPs comunicadas existe una que posiciona la pelota en una zona incompatible geométricamente con la de los demás. Para el caso de los métodos de fusión analítica, basados en rejillas probabilísticas y filtros de Kalman, el resultado será dependiente de la incertidumbre de la estimación del robot mal localizado. Si su FDP individual instantánea cuenta con baja incertidumbre arrastrará más la percepción distribuida hacia su propuesta, en caso contrario no influenciará demasiado la estimación distribuida.

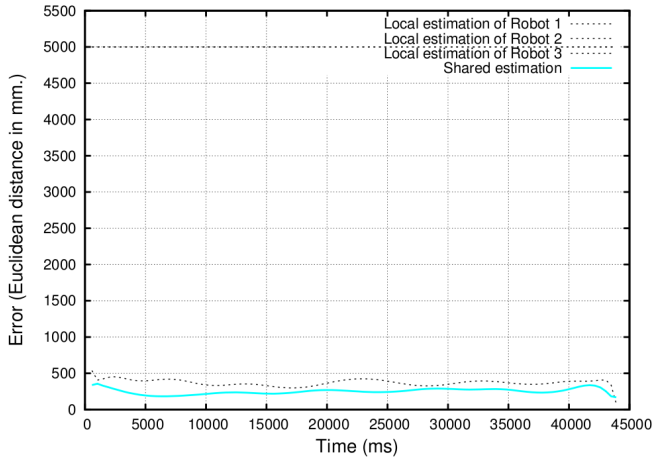


Figura 3.78: Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots

Resultados obtenidos con fusión basada en filtro de partículas

El algoritmo de fusión sensorial basado en filtro de partículas también soporta de manera adecuada la degradación de las estimaciones provocadas por problemas en la auto-localización de algún miembro del equipo. Las partículas recibidas por el robot que dispone de una estimación de calidad son suficientes para asegurar la convergencia del filtro y mantener una estimación precisa y sin oscilaciones.

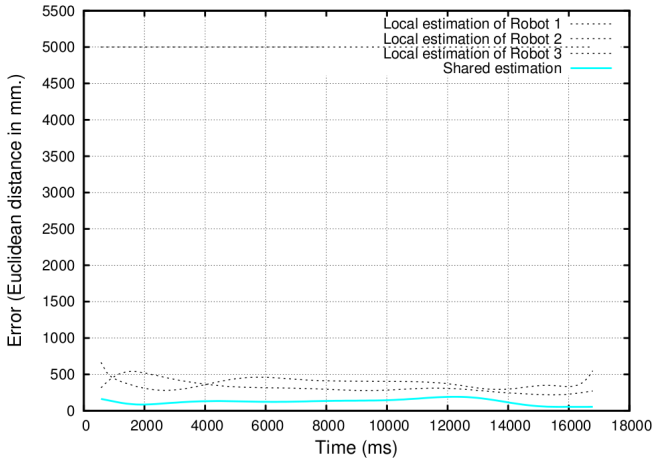


Figura 3.79: Resultados obtenidos ante alta incertidumbre de auto-localización en uno de los robots

Si se volvemos a encontrarnos con una situación en la que un robot incorrectamente localizado no es consciente de su situación, la variante apoyada en filtros de partículas se defenderá mejor que el resto e técnicas, debido a la mayoría de partículas que se concentra en los alrededores de la posición de pelota correcta. Esta mayoría no permite que se produzca un salto en la estimación de la zona correcta de la pelota a la posición errónea.

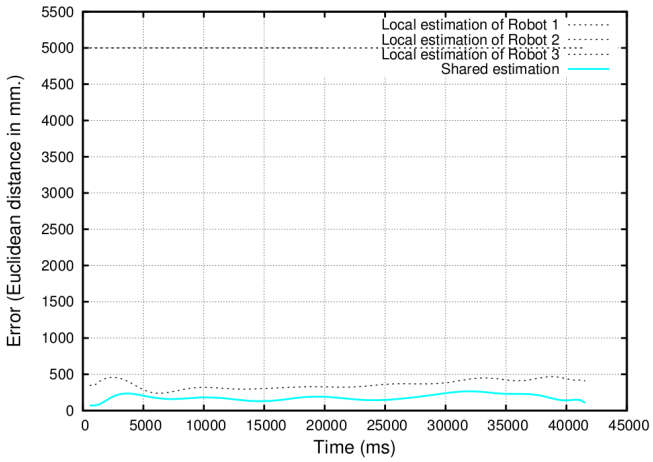


Figura 3.80: Resultados obtenidos ante alta incertidumbre de auto-localización en dos robots

Resultados obtenidos con el robot Nao

Las pruebas con los robot Nao tanto en simulación como en entornos reales certifican más aún las conclusiones extraídas sobre el comportamiento ante problemas de auto-localización en algún robot. En ambos escenarios siempre contamos con uno de los robots razonablemente bien auto-localizado, de ahí la buena calidad de las estimaciones distribuidas.

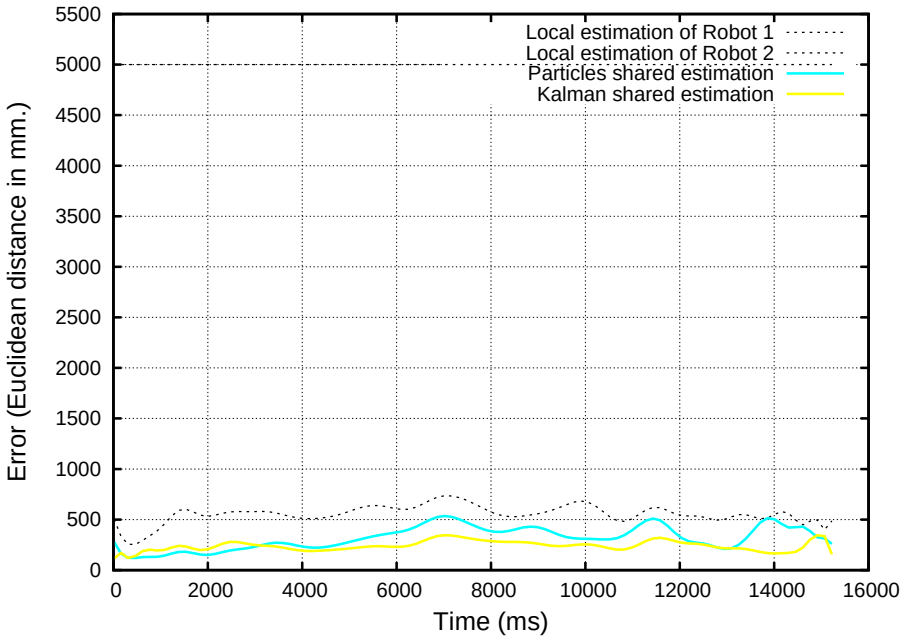


Figura 3.81: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

La mejor precisión de la alternativas de percepción distribuida están motivadas por las virtudes que ofrecen los mecanismos de acumulación de evidencias típicas de los filtros. Su incorporación continua de observaciones permite que no se acentúen los errores provocados por los errores de posicionamiento y observación de la pelota.

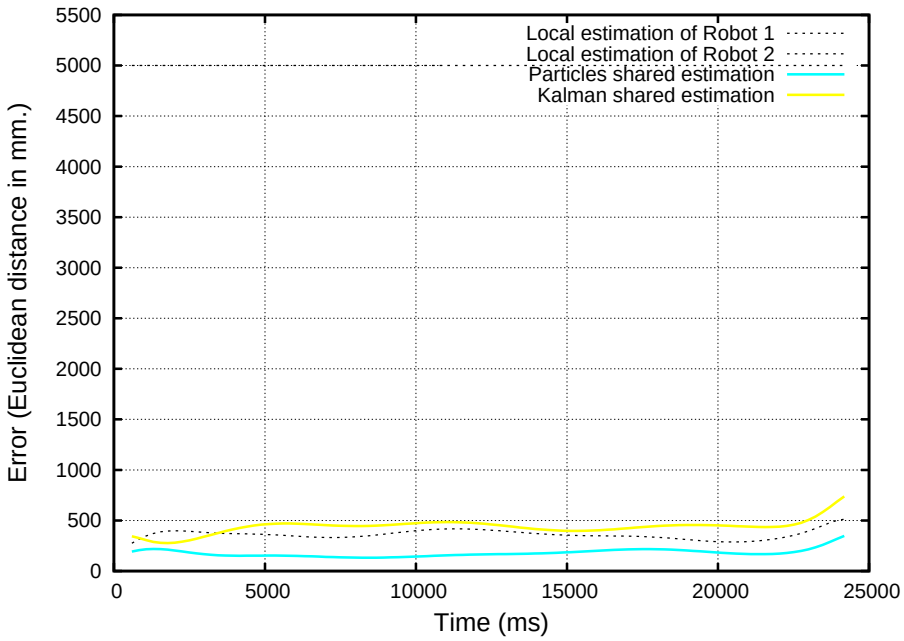


Figura 3.82: Comparación del error cometido entre las estimaciones locales y la estimación compartida mediante fusión basada en filtro de Kalman y filtro de partículas

3.3.6. Comportamiento ante objetos dinámicos

El objetivo de este experimento es verificar cómo se comportan los diferentes algoritmos de estimación a cambios rápidos en el estado del objeto a percibir, por ejemplo porque se está moviendo por el entorno.

En este experimento hemos colocado una guía que une dos puntos del campo. Esta guía permite que la pelota se desplace por ella en línea recta siguiendo una trayectoria conocida. Para este experimento hemos almacenado cada una de las estimaciones de la pelota a lo largo del tiempo y las hemos contrastado con la trayectoria ideal. De esta manera podemos constatar lo alejado o aproximado que se encuentra cada estimación de la trayectoria ideal, así como la vivacidad de la estimación.

Además de las gráficas con las estimaciones en 2 dimensiones, tanto de los estimadores individuales como los compartidos, por cada prueba hemos calculado la

distancia desde cada hipótesis (ya sea individual o distribuida) hasta el segmento delimitado por los puntos A$(900, 0)$ y B$(-1000, -1350)$. De esta manera conseguimos una aproximación al error medio cometido en todos los puntos respecto de la trayectoria ideal que sigue la pelota desde el punto A al punto B.

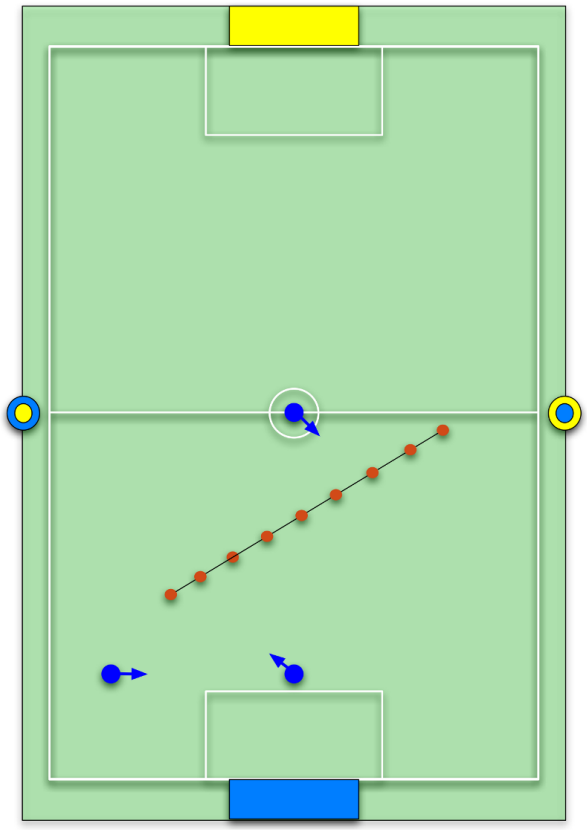


Figura 3.83: Escenario para experimentos con la pelota en movimiento

Resultados obtenidos con fusión analítica

El experimento demuestra que la técnica de fusión analítica es lo suficientemente vivaz como para seguir a un objeto en movimiento. El número de percepciones distribuidas diferentes a lo largo del experimento es elevado y garantiza un correcto seguimiento en todos los instantes de la prueba. Como muestra la gráfica, a pesar del movimiento el error en la estimación permanece más acotado que el error individual, pues las estimaciones individuales se sitúan más alejadas de la trayectoria ideal.

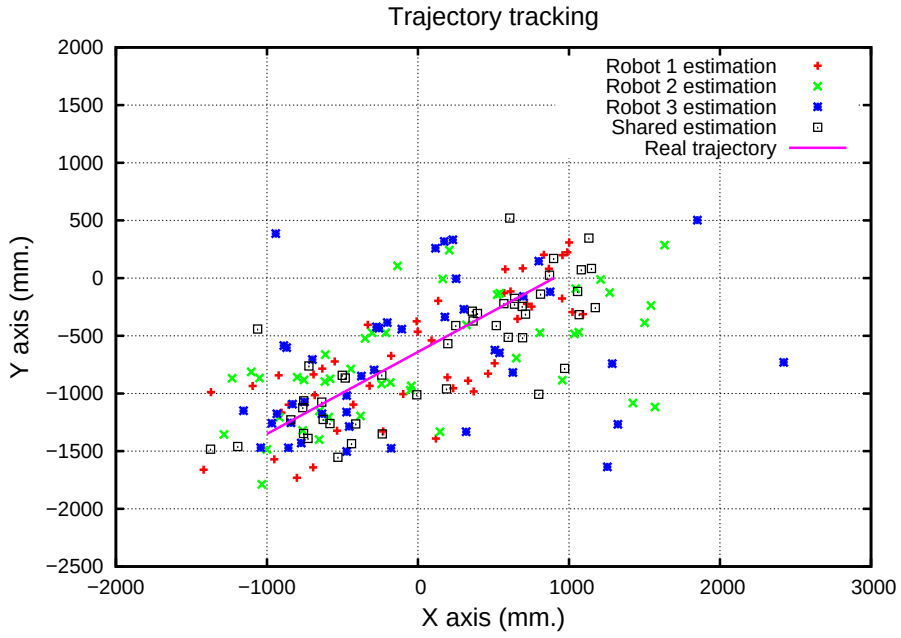


Figura 3.84: Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método analítico de fusión

Númericamente también se refrenda el hecho de que individualmente los robots son menos precisos. En media, las estimaciones individuales cometen un error de 375 mm., mientras que la técnica compartida se equivoca en 222 mm. a lo largo del recorrido en movimiento de la pelota, es decir, un 40 % menos de error.

Error robot1 (mm.)	**Error robot2 (mm.)**	**Error robot3 (mm.)**	**Error percep. distribuida (mm.)**
287.99	341.00	496.27	222.04

Cuadro 3.6: Error medio de cada estimación respecto de la trayectoria ideal empleando fusión analítica

Resultados obtenidos con fusión de rejillas de probabilidad

Esta técnica también consigue mantener una estimación de la pelota con un error menor que las variantes locales. Sin embargo se observa un menor número de estimaciones debido a la pobre eficiencia temporal que obtiene este método de estimación. Como hemos visto en la sección de eficiencia temporal este algoritmo necesita iterar en un conjunto elevado de celdas y esto provoca que el tiempo de cómputo se dispare, perjudicando la vivacidad del seguimiento.

Existe por tanto una baja cadencia de estimaciones pero la precisión que obtienen es sustancialmente mejor que la media de las individuales. Éstas cometen un error medio de 465 mm., mientras que la variante basada en fusión de rejillas obtiene 190 mm. de error medio, es decir aproximadamente un 60 % de mejora.

Error robot1 (mm.)	**Error robot2 (mm.)**	**Error robot3 (mm.)**	**Error percep. distribuida (mm.)**
290.30	452.14	652.32	190.30

Cuadro 3.7: Error medio de cada estimación respecto de la trayectoria ideal empleando rejillas de probabilidad

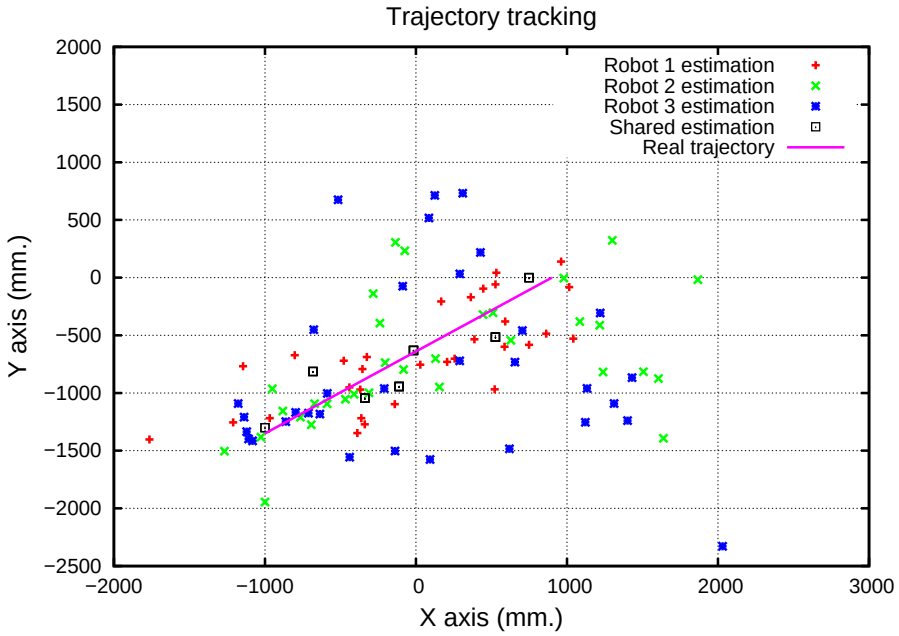


Figura 3.85: Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método de fusión de rejillas probabilísticas

Resultados obtenidos con fusión basada en filtro de Kalman

El algoritmo basado en filtro de Kalman obtiene buenos resultados en cuanto a número de estimaciones a lo largo del experimento. Su eficiencia computacional asegura una buena cadencia de estimaciones y no ralentiza el seguimiento de la pelota. A su vez la precisión es razonablemente buena ante la trayectoria que ha descrito la pelota mientras se movía.

Los números registrados muestran un error medio realizado por las estimaciones individuales de 403 mm, frente a los 174 mm. aproximadamente de la técnica basada en fusión con filtros de Kalman. De nuevo obtenemos una mejora en la precisión compartiendo información entre los robots del 57 %.

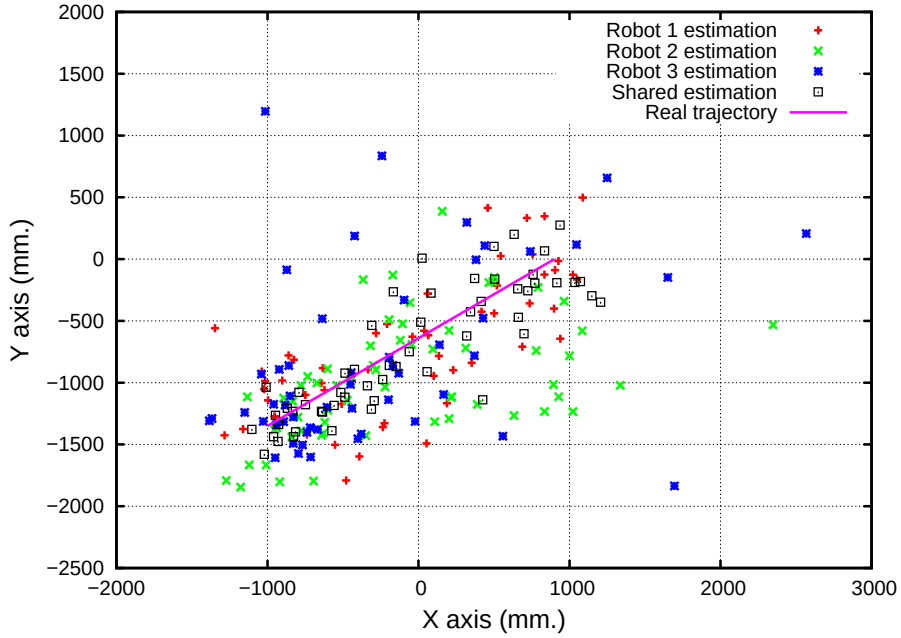


Figura 3.86: Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de Kalman

Resultados obtenidos con fusión basada en filtro de partículas

El estimador distribuido basado en filtro de partículas también se comporta eficazmente ante la estimación de un objeto moviéndose. El flujo de estimaciones no llega al ritmo de las variantes analíticas o basadas en Kalman pero sin embargo sus estimaciones permanecen más cerca de la trayectoria ideal demostrando mejor precisión.

Individualmente los robots cometen de media 432 mm. de error sin utilizar ningún tipo de fusión basada en intercambio de información. Utilizando nuestra propuesta basada en filtros de partículas el error se reduce hasta los 142 mm. aproximadamente. La mejora en precisión ha sido de un 67 %.

Error robot1 (mm.)	**Error robot2 (mm.)**	**Error robot3 (mm.)**	**Error percep. distribuida (mm.)**
307.47	352.66	549.97	173.90

Cuadro 3.8: Error medio de cada estimación respecto de la trayectoria ideal empleando filtros de Kalman

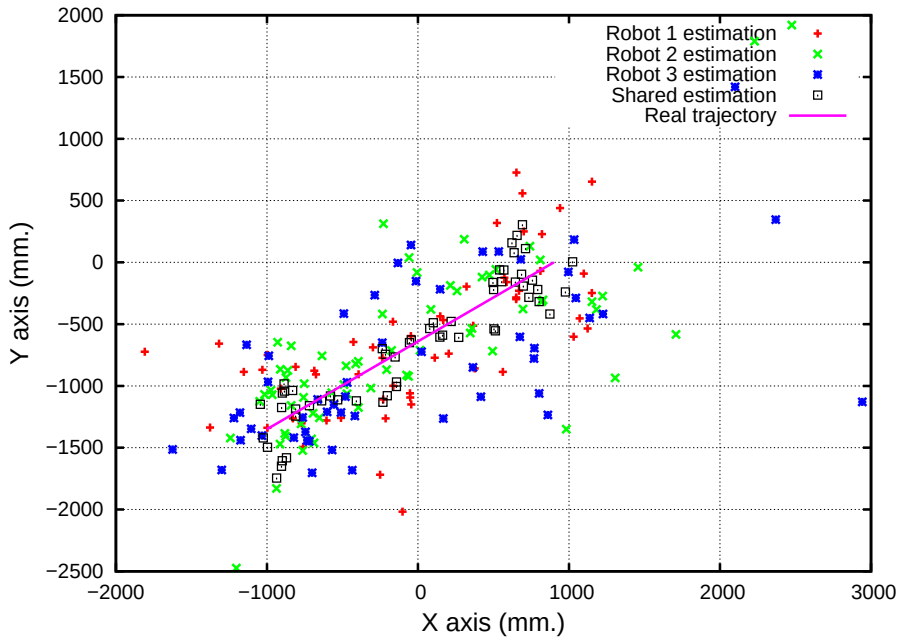


Figura 3.87: Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de partículas

Resultados obtenidos con el robot Nao

Este experimento se ha llevado a cabo con dos robots Nao reales. Hemos utilizado la misma guía que en la serie de pruebas anterior para que la pelota describa una trayectoria desde el punto $(900, 0)$ al punto $(-1000, -1350)$. Ambos robots están ejecutando sendas instancias del algoritmo de percepción distribuida basado en filtro de Kalman y filtro de partículas. A continuación se muestran los resultados obtenidos.

Error robot1 (mm.)	**Error robot2 (mm.)**	**Error robot3 (mm.)**	**Error percep. distribuida (mm.)**
314.03	368.83	613.58	141.65

Cuadro 3.9: Error medio de cada estimación respecto de la trayectoria ideal empleando filtros de partículas

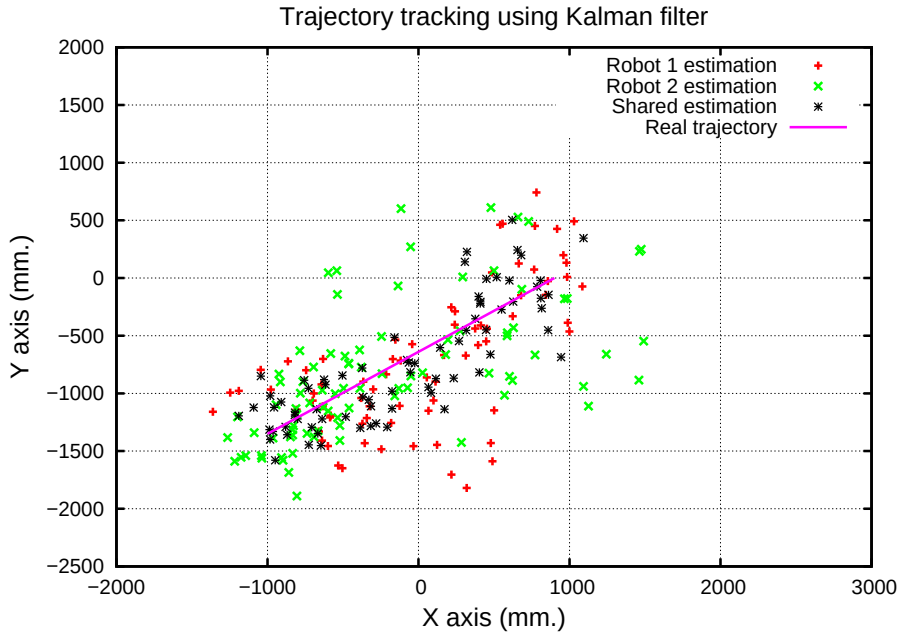


Figura 3.88: Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de Kalman

Con la técnica apoyada en Kalman se consigue mayor densidad de puntos a lo largo de la trayectoria. Esto corrobora la mejor eficiencia temporal del algoritmo. Además, los puntos negros correspondientes a las diferentes hipótesis del estimador distribuido se mantienen en un rango de distancias respecto a la trayectoria ideal más que aceptable, mejorando notablemente las estimaciones individuales de los dos robots (dibujadas en rojo y verde).

Error robot1 (mm.)	**Error robot2 (mm.)**	**Error percep. distribuida (mm.)**
263.72	325.77	192.64

Cuadro 3.10: Error medio de cada estimación respecto de la trayectoria ideal empleando filtros de Kalman

La tabla de errores medios de cada estimación arroja resultados que cuantifican las bondades de la variante distribuida, pues el error cometido es aproximadamente un 35 % menor que el cometido por las alternativas individuales.

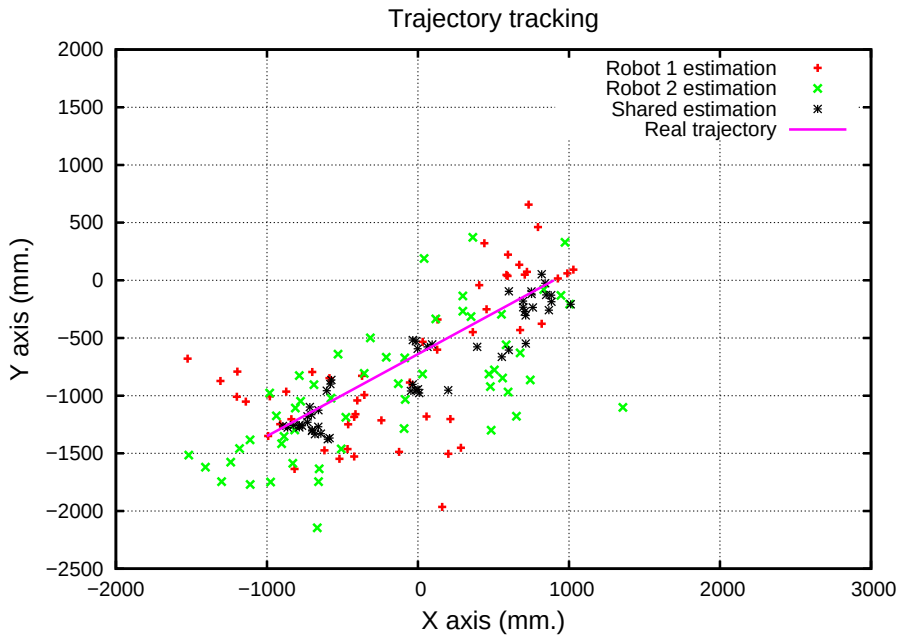


Figura 3.89: Comportamiento de las estimaciones locales y la estimación compartida ante el movimiento de la pelota mediante el método basado en filtro de partículas

En cuanto a la técnica apoyada en filtros de partículas también podemos concluir que se comporta bien con el dinamismo de la pelota. La cantidad de hipótesis por unidad de tiempo no es tan grande como con filtros de Kalman, pero es suficiente

para realizar un seguimiento correcto de la pelota. La precisión es aún mejor que en el experimento anterior, pues se consigue una mejora de aproximadamente un 65 % respecto a los estimadores individuales.

Error robot1 (mm.)	**Error robot2 (mm.)**	**Error percep. distribuida (mm.)**
304.19	368.13	126.58

Cuadro 3.11: Error medio de cada estimación respecto de la trayectoria ideal empleando filtros de partículas

La razón fundamental de la mejora con nuestras técnicas distribuidas se produce por el aislamiento que se consigue de los díscolos. Por un lado, éstos son amortiguados debido al proceso de combinación de información, que premia muy poco a las estimaciones que no se refrendan en otros robots. Por otro lado, las técnicas basadas en filtros de partículas y filtros de Kalman tienen una inercia interna que proporciona otra protección extra frente a los espurios. Sin embargo y como muestran los datos, en situaciones de movimiento de la pelota este efecto sigue asegurando la movilidad de las estimaciones.

Las pruebas de dinamismo ofrecen resultados discretos para la alternativa basada en rejillas probabilísticas. Su elevado tiempo de cómputo requerido hace inabordable el seguimiento de un objeto cuyo estado es tan cambiante como el de la pelota. La alternativa basada en filtro de partículas muestra que las estimaciones se encuentran muy cercanas a la trayectoria ideal. Idealmente, no deberían existir demasiados huecos libres en el eje X durante el experimento. Ese hecho manifestaría un vacío de estimaciones durante un tramo del movimiento de la pelota. En la variante basada en filtros de partículas existe algún hueco, que aunque no demasiado grande, no llega a la fluidez que consiguen las técnicas analíticas.

Las alternativas basadas en filtro de Kalman y métodos analíticos puros no consiguen una precisión tan buena como su homóloga apoyada en partículas, pero sin embargo consigue una vivacidad muy alta, similar a la lograda por las estimaciones individuales.

3.4. Discusión

Los resultados de los experimentos de precisión arrojan datos favorables para las estimaciones compartidas confirmando las expectativas sobre las ventajas de fusionar. Podemos afirmar que la fusión sensorial es ventajosa en todas las implementaciones y produce una reducción del error en la estimación de la pelota. Además, la estimación es más estable y exenta de fluctuaciones como suele ocurrir con algunas estimaciones individuales. Los robots que perciben la pelota a largas distancias son los principales beneficiados de la nueva estimación, pues su precisión e incertidumbre se ven mejoradas drásticamente.

Las pruebas de eficiencia revelan que las implementaciones más eficientes son las variantes de combinación basadas en filtros de Kalman y métodos analíticos puros. Su tiempo medio aproximado por iteración se sitúa entre 0,25 y 0,35 ms., frente a los 25 ms. de la versión muestreada. Como ya hemos mencionado anteriormente la versión que utiliza rejillas de probabilidad para discretizar el entorno obtiene unos tiempos de aproximadamente 700 ms. por iteración, duración muy elevada para los requisitos de este escenario.

Los experimentos de oclusiones y comportamiento ante robots mal localizados han sido muy prometedores. Los resultados han demostrado que todas las técnicas toleran estas situaciones obteniendo gran beneficio en el efecto global del equipo. Cualquier robot que no vea la pelota o esté mal localizado puede disponer de una estimación aceptable de la pelota. Sin duda la ganancia es elevada.

Las pruebas de falsos positivos demuestran que éstos son el peor enemigo de las técnicas Bayesianas. El resultado de combinar dos estimaciones geométricamente no compatibles es una tercera situada a medio camino de las dos anteriores. Éste es el peor escenario posible que se puede dar y el error aumenta a medida que disminuye la incertidumbre asociada al falso positivo. Todas las técnicas salvo la variante basada en filtros de partículas han sucumbido a este tipo de situaciones. En general, esta última variante es capaz de tolerar falsos positivos siempre y cuando éstos supongan una minoría frente al número de robots que estiman correctamente la pelota.

Los falsos positivos instantáneos son adecuadamente amortiguados por los estimadores basados en filtros de Kalman y partículas pero no por las otras dos técnicas.

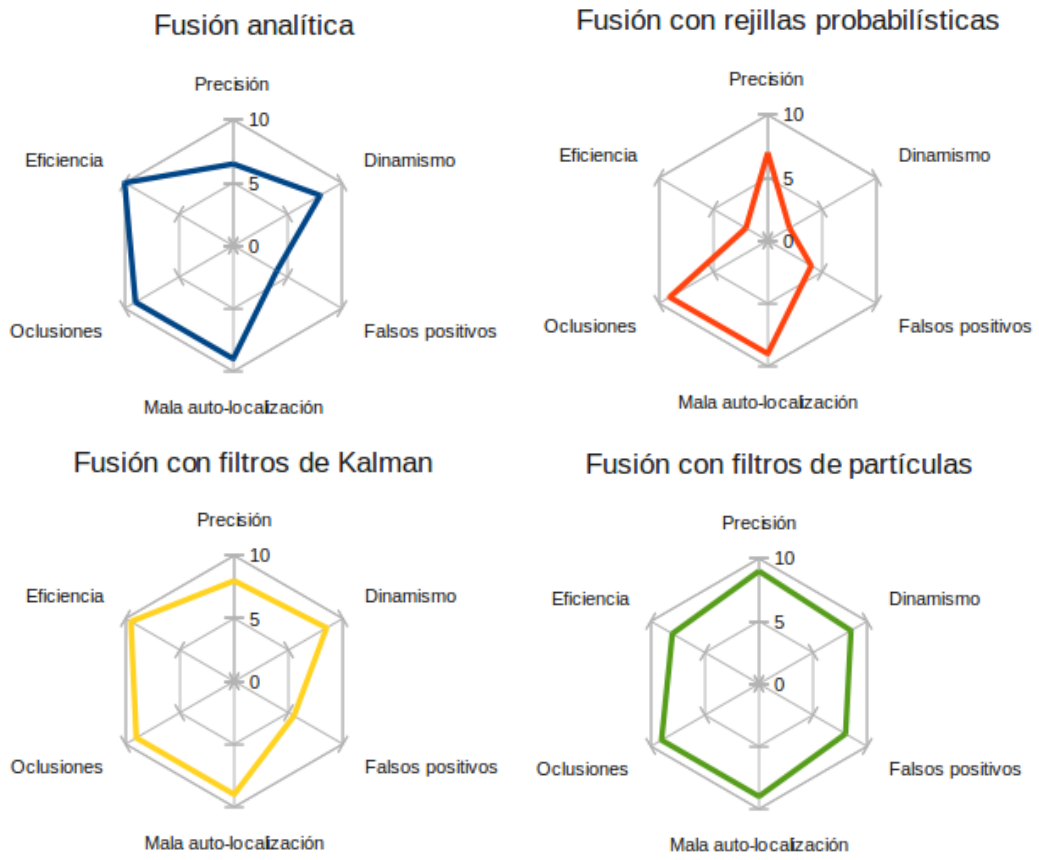


Figura 3.90: Diagrama comparativo entre las técnicas propuestas de percepción distribuida

La incorporación de persistencia temporal hace más robusta su estimación a estas observaciones incorrectas y díscolas. El resto de algoritmos sufre picos en el error cometido debido al carácter instantáneo de su fusión debido a los falsos positivos.

Las pruebas de comportamiento ante movimiento de la pelota han demostrado que todos los algoritmos mejoran la precisión comparado con alternativas individuales que no fusionan información. Como anticipamos cuando estudiamos el parámetro de coste computacional, las rejillas probabilísticas son muy costosas de mantener y su elevado tiempo de iteración tiene un fuerte impacto en este tipo de situaciones. El algoritmo no es suficientemente vivaz como para garantizar un correcto seguimiento. El resto de alternativas si garantizan unas condiciones de funcionamiento, siendo la técnica basada en filtro de partículas la que mejor precisión ha conseguido, seguida

de la variante basada en filtro de Kalman. Tras ellos se encuentran la fusión con rejillas de probabilidad y el método de fusión analítico puro. A pesar de que la fusión mediante filtros de partículas es la que mejores números ha conseguido en precisión, su ritmo de estimaciones no llega al conseguido por las técnicas paramétricas basadas en Kalman o métodos analíticos puros.

Otro aspecto interesante a discutir tiene que ver con la monomodalidad o multimodalidad de la estimación. Puesto que el *software* perceptivo que hemos empleado en ambas arquitecturas extrae únicamente distancia y ángulo hasta la pelota, no es posible mantener múltiples hipótesis de la misma. Si este proceso permitiera generar múltiples hipótesis, algunas de las variantes de fusión sensorial podrían sacar partido a esta característica y propagar esta multimodalidad hasta la estimación distribuida. La auto-localización también es un proceso que puede provocar naturalmente múltiples hipótesis ante simetrías o escaso conocimiento del entorno. En general, siempre que las FDPs involucradas sean multimodales, algunos mecanismos de combinación las toleran mejor que otros. Por ejemplo, las técnicas basadas en rejillas probabilísticas y filtros de partículas son buenos aliados de la multimodalidad. Los métodos analíticos y basados en Kalman, debido a su carácter paramétrico no permiten manejar múltiples hipótesis careciendo de la flexibilidad de las otras técnicas.

CAPÍTULO 4

Asignación dinámica de roles

It is absurd to divide people into good and bad. People are either charming or tedious.

– Oscar Wilde

En el capítulo 2 realizamos una revisión bibliográfica de los trabajos donde equipos de robots desempeñaban diferentes tareas. Además de centrarnos en aspectos generales de esta rama de la robótica, ahondamos en los dos aspectos fundamentales que soportan esta tesis: Percepción distribuida y reparto de tareas. El capítulo 3 ha servido para desgranar los detalles de la propuesta de percepción colaborativa utilizada para mantener una mejor estimación del objeto más importante en el entorno de la RoboCup: la pelota. Este capítulo aborda la descripción de la segunda pata de soporte de esta tesis: el reparto de tareas en un sistema multi-robot.

Cuando pensamos en completar cualquier tarea de nuestra vida, habitualmente aplicamos la frase célebre de Julio César *divide et vinces* o *divide y vencerás*. Si somos capaces de fragmentar el trabajo en unidades más sencillas, podremos completar poco a poco nuestro trabajo sin afrontar de una vez toda la complejidad contenida en la tarea global. Este es el principio básico que se aplica en las empresas, ingeniería, robótica o incluso la vida cotidiana.

Cuando nos circunscribimos a problemas de robótica, encontramos tareas que son realmente difíciles o incluso imposibles para un único robot. La exploración de grandes territorios llevaría demasiado tiempo empleando un único robot. La realización de tareas heterogéneas, por ejemplo, realizar un seguimiento por agua, tierra y mar exigiría que un robot fuera capaz de desenvolverse en todos estos medios, situación que raramente encontramos actualmente. Sin embargo, si pudiéramos disponer de cientos de robots de exploración, o un equipo de robots formado por un robot submarino, un robot terrestre y un robot aéreo posiblemente las probabilidades de completar con éxito nuestra misión ganarían muchos enteros.

El entorno de pruebas que hemos elegido para esta tesis es idóneo para labores de coordinación. La tarea fundamental de los robots es jugar al fútbol. Para expresar las ventajas del reparto de tareas vamos a establecer un símil entre un partido habitual de niños de primaria y un partido entre jugadores profesionales. En el primer caso advertiríamos rápidamente que todos los niños correrían hacia el balón. El resultado neto sería que se entorpecerían unos a otros y dejarían la mayor parte del campo sin cubrir. En un partido de fútbol profesional, cada jugador tiene una posición de partida según sus aptitudes previas y sólo el jugador más apropiado se desplaza hacia el balón. El resto modifican sus posiciones según la posición del balón, del resto de jugadores, del estado del partido, etc. Un detalle interesante es que los delanteros no son los únicos jugadores con posibilidades potenciales de marcar goles. Si un defensa se encuentra con una situación favorable, podría llegar hasta la portería contraria y marcar gol. Las posiciones de los jugadores son orientativas pero ante situaciones oportunistas o favorables, los jugadores tienen la flexibilidad de cambiar sus posiciones.

El enfoque propuesto en esta tesis trata en parte de imitar el comportamiento humano descrito anteriormente. En una primera aproximación, para evitar la situación de *patio de colegio*, se establecen roles de juego. Cada rol tiene asociado un objetivo, así como una estrategia de posicionamiento ideal. La siguiente evolución ha sido permitir el intercambio de roles entre los miembros del equipo. Switch! es el nombre elegido para la técnica que presentamos. Su nombre hace referencia a este concepto, a la posibilidad de cambiar de rol si las condiciones del juego lo requieren.

El capítulo comienza con la descripción de los roles que se han considerado apropiados para liga de plataforma estándar de la RoboCup. A continuación intro-

duciremos el concepto de utilidad, pieza clave para medir el grado de adecuación de cada robot a los roles disponibles. Ligado a este concepto presentaremos una serie de funciones heurísticas para calcular las utilidades en cada robot. Finalmente especificaremos la estrategia de posicionamiento que hemos desarrollado para cada robot.

La sección 4.3 está dedicada a exponer los experimentos que hemos seguido para validar el diseño propuesto. Hemos utilizado tanto un simulador como robots reales: los robots aiBo y Nao. Además de las clásicas pruebas en laboratorio, se ha asistido a varias pruebas en competiciones internacionales donde se han puesto en práctica las técnicas aquí presentadas en un entorno competitivo y real como es el de la RoboCup.

La sección 4.4 cierra este capítulo y hace un análisis de los resultados obtenidos. El análisis de los datos, la experiencia acumulada en las competiciones y los resultados obtenidos son los parámetros que se evalúan para validar una de las hipótesis fundamentales de esta tesis: La cooperación es positiva en un equipo de robots.

4.1. Fundamentos teóricos

El problema de la asignación dinámica de roles en sistemas multi-robot toma inspiración de la teoría de investigación de operaciones. La investigación operativa utiliza herramientas estadísticas para analizar la toma de decisiones en función de los parámetros disponibles (recursos, objetivos, costes, etc.).

4.1.1. Asignación óptima

Uno de los problemas más habituales dentro de este campo es el denominado *problema de la asignación óptima u OAP (Optimal Assignment problem)*. Una definición general de este problema podría ser la siguiente: Existe un número de agentes y un número de tareas a realizar. Cada agente puede desempeñar cualquier tarea con un determinado coste. Se necesita cubrir todas las tareas disponibles asignando a cada una de ellas un agente, de manera que el coste total sea mínimo.

Explicaremos este concepto con un ejemplo: Supongamos que disponemos de n trabajadores (cada uno en busca de empleo) y n trabajos disponibles (cada uno requiere un trabajador). Los empleos no tienen todos la misma prioridad, es decir,

algunos de ellos requieren ser cubiertos a la mayor brevedad posible. Consideremos también que cada aspirante ha sido evaluado por el departamento de recursos humanos, obteniendo una nota no negativa para cada puesto disponible. Estas notas miden su grado de competencia para cada cargo. También podría darse el caso de encontrar algún empleado no capacitado para algún puesto concreto. Su nota para ese trabajo sería 0. El problema de la asignación óptima en este entorno consiste en cómo asignar trabajadores a cada trabajo maximizando el rendimiento obtenido (en este caso la competencia total del equipo). Para ello habrá que tener en cuenta las prioridades de los trabajos y las aptitudes de los candidatos. En [Gal60] se estudia con profundidad este problema.

Si aplicamos este problema al reparto de tareas en equipos de robots, tendremos n robots, n roles que deberán ser asignados de manera priorizada y deberemos encontrar algún parámetro para estimar la competencia de cada robot para cada tarea. El objetivo deberá ser maximizar la competencia global del equipo en todas las tareas. Puesto que el entorno de la RoboCup es altamente dinámico, una solución que asigne estáticamente a cada robot una tarea no es lo más adecuado. La asignación deberá ser dinámica para adaptar el equipo a la situación del juego. Tampoco es deseable que se produzca un intercambio de tareas excesivamente frecuente que provoque oscilaciones e imprecisiones en los comportamientos. El hecho de realizar un intercambio de roles producirá un coste extra, por lo que *Switch!* deberá tener en cuenta este aspecto para encontrar el mejor momento para intercambiar tareas.

4.1.2. Utilidad

El término utilidad se emplea en campos como la investigación de operaciones, teoría de juegos, economía y en coordinación multi-robot, como queda descrito en [GM04]. La idea subyacente del concepto de utilidad es disponer de un parámetro para estimar el coste de ejecutar una acción. En ocasiones el término es conocido como *fitness*, salud o coste.

En el ámbito de la RoboCup lo habitual es que la utilidad se calcule en función de varios parámetros que tienen que ver con la estrategia del equipo, la posición de los objetos relevantes, el marcador, etc.

En nuestro caso, *Switch!* emplea el valor de la utilidad para evaluar el grado de aptitud de un robot a un determinado rol. Por supuesto, el cálculo de la utilidad no está exento de incertidumbre. Puesto que depende de los algoritmos de auto-localización, de percepción distribuida, etc. los valores obtenidos no reflejarán exactamente la realidad, será una estimación de la misma. Por tanto, la optimalidad de la asignación de tareas, será siempre asumida como una asignación óptima a la luz de los recursos disponibles.

La utilidad es un indicador muy flexible para medir el coste, pues se pueden incluir o extraer de su cálculo numerosos parámetros de manera sencilla. La única restricción es que su valor sea un escalar, para favorecer la rápida comparación entre los candidatos a una tarea o rol. La inclusión de todos los aspectos relevantes del estado del robot y del entorno en el cálculo de utilidades favorecerá al mejor resultado de la asignación de tareas.

4.1.3. Roles y mecanismos de comunicación

Cuando los miembros de un equipo tienen los mismos intereses, éstos pueden organizarse en un equipo. Como ocurre en cualquier organización, cada individuo juega un rol diferente dentro del equipo. Sin embargo, es necesario que alguien establezca estos papeles dentro de la agrupación. En caso de que cada individuo cuente con características propias muy diferentes y específicas, la asociación entre miembro-tarea puede ser muy sencilla. El problema aparece cuando el equipo está formado por miembros con las mismas características. En este caso, los individuos son más flexibles y son capaces de adaptarse a diferentes roles. Esta situación dificulta el mecanismo de asignación de roles.

Switch! emplea roles para dividir la tarea de jugar al fútbol en el entorno de la RoboCup. Una de las tareas comunes a todo robot que forma parte de un equipo cooperante basado en comunicaciones es la de comunicar al resto de miembros determinada información. En el capítulo 3 ya describimos cómo se combina esta información para enriquecer la percepción de la pelota. Ahora, esta información se utiliza para que *Switch!* disponga de elementos de juicio para evaluar cuál es el rol más adecuado para ejecutar.

El tipo de información que se intercambia para mantener la técnica de intercambio de roles operativa puede llevarse a cabo de diversas maneras. Un modelo concreto es el de negociaciones, que toman inspiración de las acuerdos entre personas que se llevan a cabo dentro del mundo empresarial. En este caso cada agente se preocupa de sus objetivos, sin importarle demasiado los de los demás. Los agentes realizan ofertas y aceptan las de otros miembros del equipo. Este mercado figurado regula la ocupación de los recursos, que en este caso serían los roles de juego.

Otra alternativa es el empleo de acuerdos y desacuerdos (*commitment/Decommitment*). La comunicación establece un acuerdo de cooperación en una tarea concreta durante un periodo determinado de tiempo. Los miembros del equipo acuerdan estas ayudas para perseguir diferentes objetivos. A mayor escala, es incluso posible realizar acuerdos entre grupos de robots, formando coaliciones para abordar problemas que requieren mayor cantidad de individuos.

La variante que emplea *Switch!* establece una comunicación del estado de cada individuo. De esta manera cada robot conoce cuáles son las creencias de cada compañero y cuáles son las suyas. Esta información es suficiente para evaluar el grado de adaptación de cada robot a cada rol en nuestro entorno, pues los roles están muy ligados a la percepción de la pelota y la posición de cada robot. Dada la naturaleza del problema, la velocidad de respuesta es mucho mayor que ante alternativas basadas en negociaciones o acuerdos, que implican mayor cantidad de mensajes intercambiados para seleccionar roles.

4.2. *Switch!:* Asignación dinámica de roles en la RoboCup

4.2.1. Especificación de roles para el juego

Varios equipos [RLB+05], [Lab05] participantes de la RoboCup han utilizado un juego de roles asignado a cada robot en lugar de emplear un único comportamiento para todos los jugadores del mismo equipo. Esta colección de roles permite mantener

una mejor y más ordenada estructura de equipo, que contribuye sin duda a resolver con mayor éxito la tarea a la que se destina el grupo.

Los roles pueden asignarse de manera estática o dinámica. Los beneficios de la asignación dinámica son superiores, pues las situaciones que involucran cambios de rol debido a rápidas transiciones de la pelota, penalizaciones o fallos en jugadores (los jugadores penalizados deben abandonar el campo durante 30 segundos), provocarían configuraciones de juego poco recomendables si no se produjeran cambios de roles.

En nuestro caso hemos definido cuatro roles para el entorno de la RoboCup: portero o *Goal-keeper*, chutador o *striker*, defensa o *defender* y atacante o *supporter*. El jugador con el comportamiento de portero tiene asignado este rol de manera estática. La principal razón para mantener un jugador con un rol fijo es que la normativa no permite que jugadores de campo entren en su propio área (similar a la regla del balonmano). El resto de roles pueden ser intercambiados por los jugadores de campo según las condiciones del juego.

A continuación vamos a describir los objetivos o tareas de cada rol, así como las ventajas asociadas al uso del intercambio dinámico:

- ***Portero:*** Su función principal es proteger su propia portería de los disparos de otros jugadores. La defensa de la portería implica permanecer dentro del área o sus inmediaciones la mayor parte del tiempo.

- ***Chutador:*** Su mayor interés es conseguir la pelota, bien para conducirla hasta la portería contraria o para chutarla en esa dirección. En caso de que la pelota esté en poder del equipo contrario, el chutador trata de recuperarla activamente, es decir, interceptando el camino del jugador contrario y acosando al robot que conduce la bola.

  Ninguno de lo otros roles del equipo tiene como misión alcanzar la pelota. Esta aproximación proporciona una ventaja implícita: Evita que varios jugadores propios se golpeen entre sí al tratar de alcanzar la pelota simultáneamente. El reglamento de la RoboCup sanciona esta situación conocida como *pushing* con una exclusión del campo para los jugadores. En [AMMM06] se presenta un mecanismo de reserva de pelota para conseguir el mismo objetivo.

- ***Defensa:*** Su función es la de proteger su propia portería de los tiros a puerta del equipo contrario. Su posición debe favorecer el bloqueo de estos disparos. Además, también tiene una labor de ocultación de la portería, de manera que dificulte su localización para el robot que pretende disparar en su dirección.

  Otra consecuencia implícita del rol de defensa es la de sacar partido a disparos del equipo contrario que sitúan la bola en los alrededores del campo propio. El hecho de mantener un jugador permanentemente en tu propio campo o incluso cerca de la portería garantiza que estarás cerca de la pelota después de un tiro hacia tu portería.

- ***Atacante:*** La misión de este rol es auxiliar la labor del chutador en labores de ataque. Por un lado evitará obstaculizar cualquier tiro del chutador y procurará situarse en una posición que maximice el espacio cubierto en el campo de ataque.

  Los mayores beneficios de este rol al equipo son la capacidad de recuperación de la pelota, en caso de que el chutador no dispare con una dirección precisa, además de mantener una buena posición para futuros pases.

4.2.2. Funciones heurísticas para el cálculo de utilidad

A continuación introduciremos algunos conceptos necesarios para explicar cómo los roles son intercambiados. Nuestro algoritmo de asociación de roles está basado en funciones heurísticas. Estas funciones evalúan algunos parámetros como distancia hasta la pelota o posición en el campo para obtener el valor de utilidad. Las utilidades se calculan periódicamente y los roles son emparejados con cada robot de manera ordenada según los parámetros de utilidad obtenidos.

Nuestra propuesta calculará individualmente la utilidad como la suma ponderada de varios parámetros. Con el fin de describir estos factores usaremos el formalismo descrito en [GM04]:

- Sea $I_1, ..., I_n$ el conjunto de n robots.
- Sea $J_1, ..., J_n$ el conjunto de n roles ordenados por prioridad y $w_1, ..., w_n$ sus pesos relativos.

- Sea U_{ij} el valor de utilidad no negativo del robot I_i para ejecutar el rol J_j, $1 \leq i, j \leq n$.

Cada robot I_i siempre debe tener asociado un rol, y cada rol J_j debe ser asignado a un robot atendiendo a su prioridad. Si durante un periodo de tiempo las comunicaciones resultan inaccesibles, algunos roles no se asignarán. En este caso todos los robots elegirán el rol más prioritario, es decir, en nuestro entorno el rol de chutador. Esta solución garantiza que al menos uno de los robots siempre tratará de ir a por la pelota, principal objetivo del equipo. Cuando las comunicaciones se restauran, el mecanismo de asignación de roles comienza su trabajo de nuevo y todos los roles se asocian a algún robot. *Switch!* no es una solución centralizada, sino que cada robot cuenta con una instancia y ésta es la que se encarga de elegir el rol más adecuado. Es una técnica, por tanto, distribuida por todo el equipo de robots. A continuación se muestra la sucesión de pasos que calcula cada robot para seleccionar su rol:

1. Actualizar utilidades
2. Buscar el robot con mayor utilidad para desempeñar el rol de chutador.
3. Buscar el robot con mayor utilidad para desempeñar el rol de defensa.
4. Buscar el robot con mayor utilidad para desempeñar el rol de atacante.

La tarea de actualizar utilidades procesa una matriz con todas las combinaciones de robots y roles según las funciones heurísticas que describiremos a continuación. La información de entrada para calcular la utilidad es enviada por *broadcast* periódicamente por cada robot. Esta información consiste en la posición del robot en el campo y la distancia euclídea a la pelota.

A continuación se muestran las funciones heurísticas utilizadas para calcular cada rol. La prioridad de los roles es chutador - defensa - atacante.

- $U_{i,Chutador} = D_{I_i,Pelota} + \alpha \cdot (\theta_{I_i - Portera\_contraria} - \theta_{Pelota - Portera\_contraria}) + REC$
- $U_{i,Defensa} = D_{I_i,Portera\_propia} + REC$
- $U_{i,Atacante} = D_{I_i,Portera\_contraria} + REC$

$D_{I_i,Pelota}$, $D_{I_i,Portera\_propia}$ y $D_{I_i,Portera\_contraria}$ representan la distancia desde un determinado robot a los objetos de interés del campo (pelota, portería propia y portería contraria). $D_{I_i,Pelota}$ expresa la distancia entre el robot i y la pelota. $D_{I_i,Portera\_propia}$ expresa la distancia entre el robot I_i y su propia portería. $D_{I_i,Portera\_contraria}$ declara la distancia entre el robot I_i y la portería contraria. $\theta_{I_i-Portera_contraria}$ declara la diferencia en orientación entre la orientación actual del robot y la orientación ideal para encarar a la portería ajena. $\theta_{Pelota-Portera\_contraria}$ expresa el ángulo entre la pelota y el centro de la portería contraria. Una pelota situada en el centro del campo se encuentra a 0º del centro de la portería. El ángulo será mayor a medida que se vaya acercando a la banda y la propia portería. La orientación ideal para el jugador que quiere golpear la pelota contra la portería contraria debería coincidir con este último parámetro.

REC es el acrónimo de *Role Exchange Cost* o coste de intercambio de rol. Este parámetro se añade a la función para penalizar el intercambio a un nuevo rol y así prevenir un excesivo trasiego ante cambios menores en el juego. Este factor proporciona histéresis al sistema. Esta idea ha sido utilizada en otros trabajos, por ejemplo en [CA07], donde se ha utilizado como parámetro de auto-excitación para asegurar que un comportamiento no sea interrumpido por otros que compiten con él por la ejecución.

A continuación se describen las fórmulas para seleccionar el robot más apropiado para cada rol (teniendo en cuenta la prioridad chutador-defensa-atacante).

1. $Utility_{chutador} = min(U_{i,chutador})\ , \forall i \in (1..n)$

2. $Utility_{defensa} = min(U_{i,defensa})\ , \forall i \in (1..n) \wedge i \neq Robot_{chutador}$

3. $Utility_{atacante} = min(U_{i,atacante})\ , \forall i \in (1..n) \wedge i \neq Robot_{chutador}, Robot_{defensa}$

La prioridad entre el defensa y el atacante puede intercambiarse según el marcador del partido para obtener una estrategia más ofensiva o defensiva. Por ejemplo, en caso de que el partido no esté a nuestro favor y decidamos pasar al ataque, podríamos ofrecer más prioridad al rol de atacante que al de defensa. De esta manera, la política

de asignación de roles dispondría de dos robots para el rol de atacante en lugar de uno.

Cada robot es responsable de actualizar sus utilidades periódicamente y enviar por inundación su información al resto de compañeros. Utilizaremos el término *información de coordinación* para referirnos a esta información. Los datos enviados (auto-localización y distancia hasta la pelota) son enviados a un ritmo de 5 Hz.

Cada vez que un robot recibe un nuevo mensaje con información de coordinación, actualizará los datos asociados al correspondiente robot en su modelo global. Este modelo almacena la posición de cada uno de sus compañeros, que junto con la posición de la pelota forman la información utilizada para la utilidad.

Un detalle interesante a resaltar es que cada robot envía su propia estimación de la distancia a la pelota. Esta distancia puede ser calculada de dos formas: En origen o en destino. En una primera aproximación cada robot enviaba únicamente su posición en el campo. Cuando otro robot debía actualizar su tabla de utilidades, usaba su estimación local de la pelota junto con la estimación de su posición en el campo para componer la posición de la pelota en un marco de coordenadas globales. Teniendo la pelota situada en el campo ahora podía evaluar la distancia hasta cada uno de los otros robots, pues había recibido de ellos su información de posición, es decir, la distancia hasta la bola se calculaba en destino. Esta opción tiene el inconveniente de que ante malas estimaciones en la auto-localización se falsea la medida de distancia entre robots y pelota.

El método de cálculo de la distancia en origen obliga a que cada robot incluya en la información de coordinación su propia estimación local de distancia hasta la pelota. Ahora este valor es independiente la auto-localización. Los experimentos demuestran mejores valores de utilidad cuando las distancias hasta la pelota son calculadas en origen.

4.2.3. Posicionamiento de cada rol de juego

Una vez se han explicado los mecanismos de selección de rol, otro aspecto interesante es qué comportamiento debería desplegar un robot si está ejecutando el rol de chutador, defensa o atacante. Además de las funciones descritas anteriormente, en

nuestra propuesta tanto el defensa como el atacante deben calcular y dirigirse a una determinada posición del campo. A continuación se describe cómo se obtiene esta posición *ideal* dependiendo del rol que se tenga asignado.

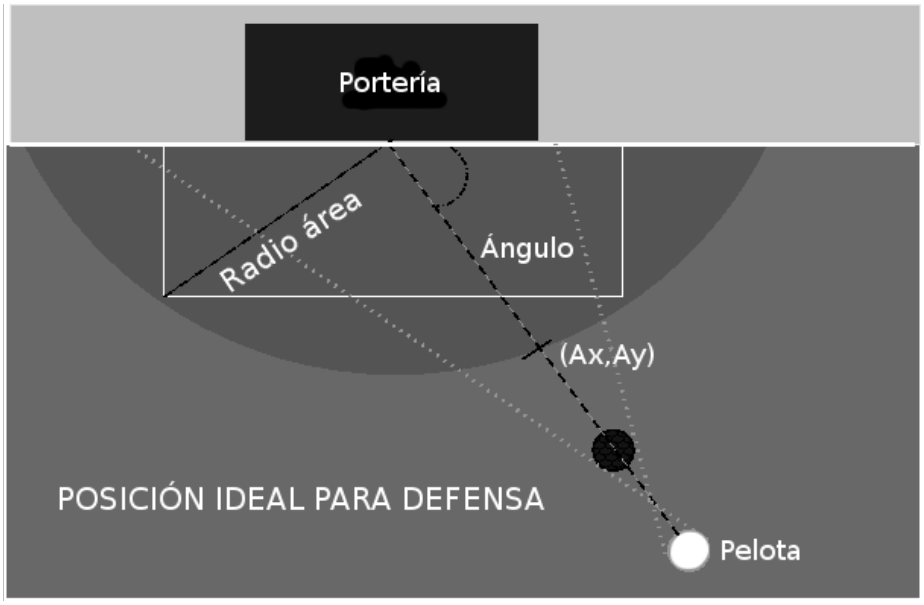


Figura 4.1: Puntos de interés para el cálculo de la posición óptima de defensa

Posicionamiento del rol defensa El defensa debería permanecer en una posición cercana a su propia portería con el fin de realizar labores defensivas. Nuestra aproximación sitúa al robot con el rol de defensa en un punto de la línea recta imaginaria que une la pelota y el centro de la portería. Si un jugador contraria dispara contra la portería a defender, el defensa tendrá muchas posibilidades de bloquear el lanzamiento al estar en la trayectoria de la pelota.

El ángulo formado por la línea que une pelota-portería respecto de la línea perpendicular a la línea de meta es calculado periódicamente. El defensa ajusta su movimiento a lo largo de la línea virtual que une pelota y portería constantemente. Este punto de destino puede observarse en el ejemplo de la figura 4.1. Cuando la pelota está cerca de la portería a defender, el defensa debería permanecer en una posición más cercana a la pelota. Sin embargo, en situaciones donde la pelota esté en una zona

más alejada de la portería, la posición idónea para defender estaría más cerca de la portería y, por tanto, más alejado de la pelota.

Esta estrategia de posicionamiento proporciona mayor seguridad al equipo, pues durante las labores defensivas el equipo dispone de más miembros cerca de la portería para su guarda. En caso de que el equipo esté atacando, el defensa cubre más superficie al adelantar su posición y permanecer más alejado de la pelota.

$$\begin{aligned} ángulo &= arctan(Portera\_propia_y - pelota_y, Portera\_propia_x - pelota_x) \\ Ax &= -cos(ángulo) * radioÁrea \\ Ay &= \frac{LongitudCampo}{2} - sin(ángulo) * radioÁrea \\ pelota2área &= distancia(A, Pelota) \end{aligned}$$

La regla de *defensa ilegal* debe ser tenida en cuenta cuando el equipo se encuentra en tareas defensivas. Esta penalización se aplica cuando robot de campo entra en su propio área. Con el fin de evitar este tipo de sanción en el defensa, hemos aproximado el área a una semi-circunferencia. El radio de esta semi-circunferencia (*radioÁrea*) se corresponde con la distancia entre cualquiera de las dos esquinas del área y el centro de la portería, como se muestra en la figura 4.1. A continuación se calcula el punto auxiliar *A* (A_x, A_y). Este punto es el resultado de intersecar la semi-circunferencia que aproxima el área y la línea que une la pelota con el centro de la portería.

La posición de destino seleccionada para el defensa se corresponde con el punto medio del segmento delimitado por la pelota y el punto A. Esta posición (DEF_x,DEF_y) es calculada de la siguiente manera:

$$\begin{aligned} DEF_x &= -cos(ang) * (radioÁrea + \frac{pelota2Área}{2}) \\ DEF_y &= \frac{LongitudCampo}{2} - sin(ang) * (radioÁrea + \frac{pelota2Área}{2}) \end{aligned}$$

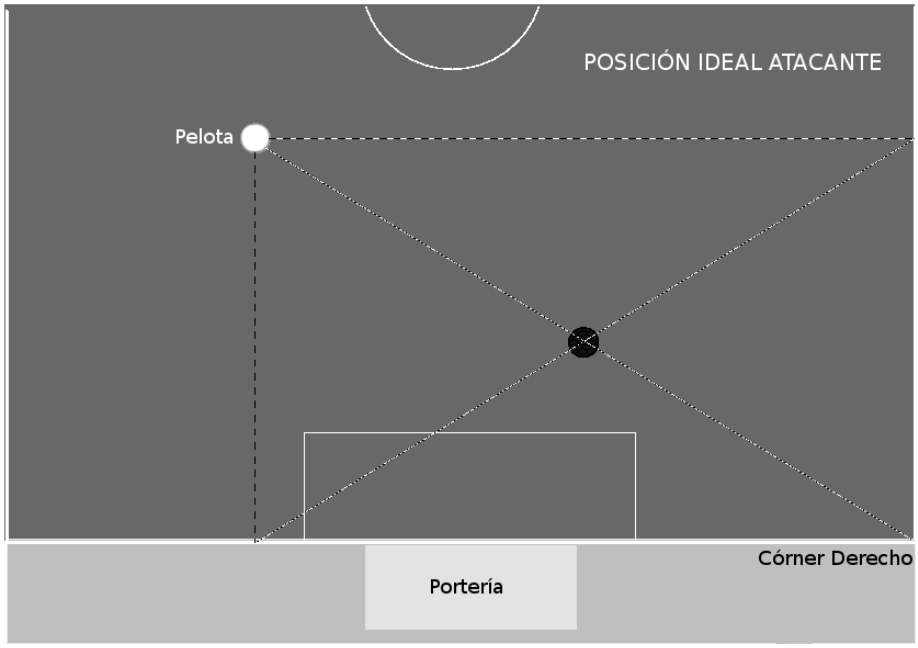


Figura 4.2: Puntos de interés para el cálculo de la posición óptima de atacante

Posicionamiento del rol atacante La posición destino para el jugador que tiene asociado el rol de atacante se corresponde con el punto medio del cuadrante delimitado por la pelota y el córner más alejado del campo de ataque. La figura 4.2 ilustra la posición de destino y el cuadrante. El cálculo de este punto se realiza atendiendo a la siguiente fórmula:

$$D_{lateral} = Max(dist(pelota_x, lat_{izq}), dist(pelota_x, lat_{der}))$$

$$D_{líneaFondo} = \frac{longitudCampo}{2} + pelota_y$$

$$ATACANTE_x = pelota_x \pm \frac{D_{lateral}}{2}$$

$$ATACANTE_y = pelota_y - \frac{D_{líneaFondo}}{2}$$

Esta posición ofrece una relativa garantía de que el atacante nunca estará situado en la trayectoria de disparo del chutador. Además ofrece un buen apoyo al chutador cubriendo gran parte del campo de ataque, que puede ser utilizado para recuperar la pelota de manera rápida ante un disparo fallido o para favorecer la recepción de un pase.

4.3. Experimentos

Como veremos en el anexo A la infraestructura del código del equipo *TeamChaos* [Tea05] está compuesta por el código fuente del programa que ejecuta en el robot, así como de un conjunto de herramientas para facilitar la depuración y visualización de determinados aspectos del código. Una de estas herramientas es el simulador de equipo. Este simulador permite verificar cada uno de los comportamientos de los robots, chequear las comunicaciones, algoritmos de auto-localización, etc. Este programa que ejecuta en un PC convencional simula un entorno similar al de la competición RoboCup. Como parte de la evaluación de *Switch!* hemos comprobado las diferencias en el juego utilizando este mecanismo de coordinación y sin utilizarlo.

Hemos evaluado el juego del equipo en un simulador sin contar con el equipo oponente. El objetivo de esta decisión es aislar al máximo nuestros comportamientos de cualquier otro elemento externo para evaluar el impacto de la cooperación en el juego del equipo.

4.3.1. Coordinación con *Switch!* en el simulador

En este experimento hemos utilizado un simulador multi-robot básico creado originariamente dentro de *TeamChaos* y modificado para esta tesis. El objetivo de la prueba es verificar el funcionamiento de *Switch!* aislado de los problemas clásicos de la percepción y localización en robots móviles. En este entorno cada robot conoce perfectamente su posición, la de sus compañeros y la de la pelota. En el simulador los robots están ejecutando *Switch!* y los comportamientos de posicionamiento de

juego especificados con anterioridad. El vídeo completo de este experimento puede visualizarse en *http://gsyc.es/∼caguero/tesis/videos/switchSimulador.mov*.

A simple vista el juego utilizando *Switch!* es mucho más ordenado y dinámico. Es más ordenado porque los jugadores están mejor repartidos por el campo y no acuden a la vez a por la pelota. Es más dinámico porque cuando la pelota se traslada de un sitio a otro, el tiempo que transcurre hasta que alguien la alcanza es mucho menor que antes. Sin coordinación todos los jugadores se concentran cerca de la bola y, tras un disparo, todos acuden a la vez de nuevo a por la pelota.

Se ha comprobado en el simulador la correcta transición de roles cuando la pelota viaja de lado a lado del campo. El jugador mejor situado cambia su rol y se convierte en chutador para dirigirse cuanto antes a por ella. La pelota es alcanzada en mucho menor tiempo utilizando *Switch!*.

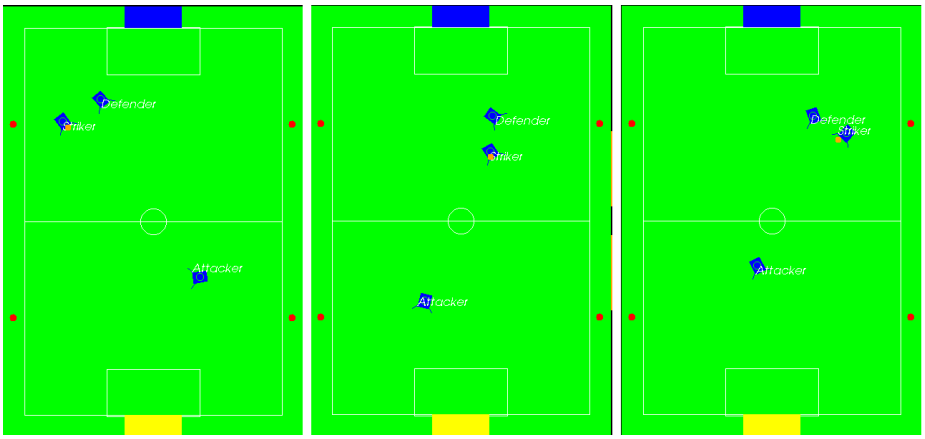


Figura 4.3: Posición del defensa ante diferentes situaciones de juego

La figura 4.3 ilustra el posicionamiento del defensa cuando la pelota se sitúa en diferentes zonas del propio campo. Se puede advertir cómo el defensa trata de acomodar su posición para interponerse entre una hipotética trayectoria desde la posición de origen de la pelota hasta la portería. Tanto el chutador como el defensa se encuentran cerca de la pelota. Debido a que el chutador no siempre consigue recuperar la pelota, es una buena estrategia mantener un segundo jugador (defensa) cerca del área de influencia de la pelota, pues ésta se encuentra en un área peligroso para los intereses del equipo que defiende.

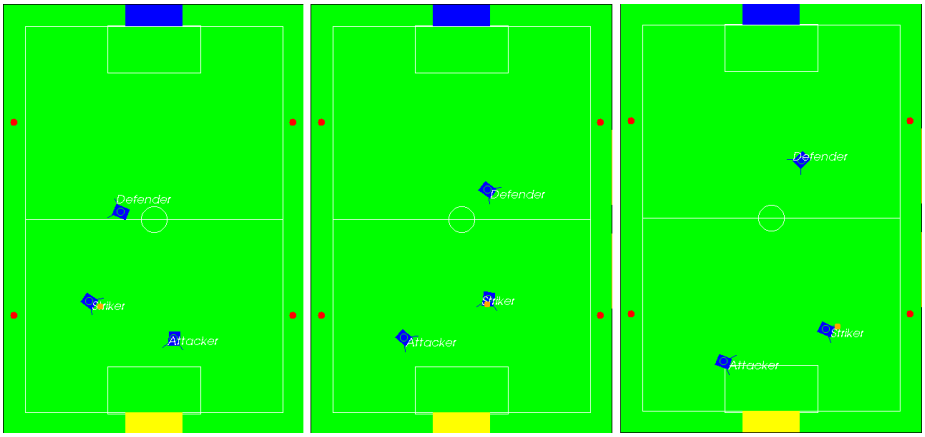


Figura 4.4: Posición del atacante ante diferentes situaciones de juego

Partido #	Goles con *Switch*	Goles sin *Switch*
1º	3	2
2º	4	3
3º	4	3

Cuadro 4.1: Resultado de los partidos sin oponentes

La imagen 4.4 muestra cómo el robot atacante se distribuye por el campo para cubrir la mayor parte de la zona de ataque. El atacante siempre se posiciona en el lado contrario al lado que ocupa el chutador, favoreciendo la distribución de jugadores y el bloque de la pelota ante un tiro del compañero chutador.

Con el fin de obtener unos parámetros más objetivos de la incidencia del mecanismo de asignación dinámica de roles en el juego, se han jugado seis partidos diferentes en el simulador. Tres de ellos se han jugado empleando *Switch!* como mecanismo de coordinación y otros tres sin utilizar cooperación alguna. Cada vez que algún jugador marcaba un gol, la pelota era posicionada en el centro del campo de manera similar a como se hace durante un partido real. La tabla 4.3.1 muestra los resultados de los partidos simulados.

El equipo que no utiliza ningún tipo de coordinación tiene un problema fácilmente perceptible: Todos los jugadores se dirigen a la vez a por la pelota. De este modo no existe ninguna ventaja entre jugar con un único jugador o emplear varios.

En el caso de emplear un equipo de robots, cuantos más miembros formen el grupo, más obstrucciones e interferencias se producen entre ellos.

El juego coordinado soluciona el problema anterior. Al existir menos obstrucciones y que los robots no permanecen tan juntos, existen más posibilidades de percibir la pelota y chutarla a portería. Las sanciones debidas a *pushing* no se producen y las recuperaciones de pelota son mucho más rápidas.

A pesar de todo la mejora en goles no es excesivamente alta. El comportamiento de los robots es mucho más equilibrado, pues cada robot desempeña su papel y el juego es mejor pero el resultado siempre depende del grado final de acierto del robot a la hora de chutar, tarea ésta de elevada dificultad en robots cuadrúpedos.

4.3.2. Coordinación con *Switch!* en robots aiBo reales

Además de emplear un simulador hemos evaluado la eficacia del mecanismo de intercambio dinámico de roles en un equipo de aiBo reales. Hemos utilizado tres robots del modelo ERS-7 cargados con el software del equipo *TeamChaos* y hemos efectuado varios experimentos.

Para el primero de ellos hemos modificado ligeramente los roles para que únicamente el chutador se mueva y se dirija hacia la pelota. El resto de roles en esta prueba mantendrán al robot parado. De esta manera queremos observar la velocidad de los cambios de rol y que únicamente uno de los robots se haga cargo del rol de chutador. El vídeo completo de este experimento puede visualizarse en *http://gsyc.es/∼caguero/tesis/videos/switchBallBooking.mov*.

En este experimento hemos apoyado los robots sobre cajas como puede apreciarse en la figura 4.5. De esta manera mantenemos a los robots controlados pero observamos cuál de ellos ejerce de chutador debido al movimiento de sus patas (tratando de dirigir al robot hasta la pelota). El objetivo del experimento es mostrar que sólo el robot con mejor valor de utilidad tratará de moverse hasta la pelota. Si situamos la bola en frente de uno de ellos, sólo éste trata de ir a por ella. Si a continuación desplazamos la pelota hasta las inmediaciones de otro robot, éste se alza con el papel de chutador y sus patas comienzan a moverse indicando que trataría de alcanzarla. A

Figura 4.5: Experimento de asignación del rol de chutador

su vez, el robot anterior se detiene al producirse un cambio de rol y dejar de ejercer de chutador.

El siguiente experimento pretende evaluar el comportamiento de *Switch!* en el entorno de juego pero sin contrarios. El fenómeno a evitar es el mostrado en la figura 4.6, donde todos los robots tratan de conseguir la pelota sin preocuparse de los demás. La consecuencia fulminante en la RoboCup es la penalización de uno o varios de los miembros del equipo, afectando enormemente al rendimiento grupal.

En una primera prueba hemos configurado *Switch!* para que no dispare ningún intercambio de roles, evaluando de esta manera los comportamientos individuales de cada robot. La imagen 4.7 resume un instante de la prueba, donde podemos advertir cómo el robot chutador se dirige hacia la pelota hasta que logra alcanzarla. Paralelamente, el robot más retrasado se mantiene en una posición defensiva pero corrigiendo su posición para mantenerse en línea entre la pelota y su propia portería.

El robot que ejerce de atacante se mantiene ligeramente distanciado del robot chutador para cubrir la mayor superficie posible. Tras el golpeo del chutador la pelota es enviada a la otra parte del campo y se puede apreciar cómo el atacante comienza

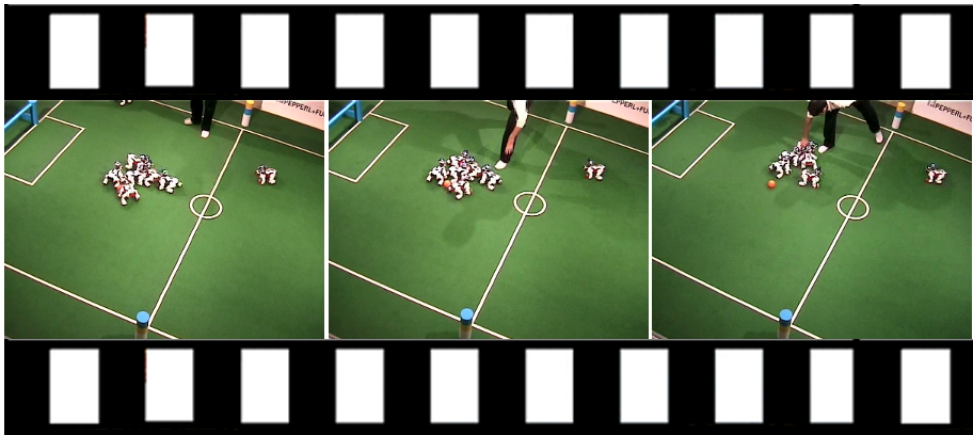


Figura 4.6: Secuencia de fotogramas donde se aprecia una situación clásica de *patio de colegio*, donde todos los robots tratan de ir a por la pelota sin ninguna coordinación entre sí

a corregir su posición para cambiar de banda y no entorpecer la labor del chutador, además de cubrir esa zona de juego.

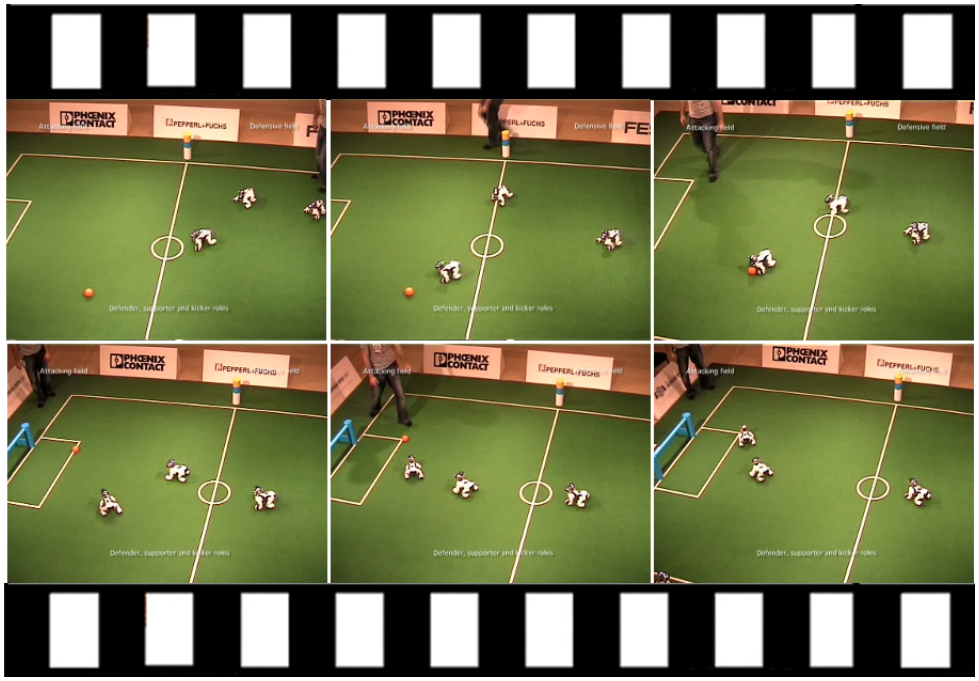


Figura 4.7: Secuencia de fotogramas de un instante del experimento de *Switch!* utilizando roles fijos

La figura 4.8 resume un segundo experimento con *Switch!* ya configurado en modo normal para que sí realice intercambio de roles. Los tres primeros fotogramas muestran cómo se configura inicialmente el equipo con uno de los robots dirigiéndose hasta la pelota, el robot atacante modificando su trayectoria para ocupar la zona del campo contraria a la posición de la pelota y el robot defensa manteniendo su posición retrasada y alineada con la pelota.

Tras un primer golpeo del robot chutador la pelota se acerca a la línea de fondo y éste se acerca a por ella de nuevo hasta que golpeamos premeditadamente la pelota hasta el campo contrario. En ese momento la maquinaria de *Switch!* decide realizar un intercambio de roles y el robot que antes ejercía de defensa ahora se convierte en chutador. En las imágenes se observa cómo camina hasta la pelota y consigue atraparla. Entre tanto, el robot que anteriormente apoyaba al chutador se ha convertido en defensa y debe desplazarse hasta la zona del campo de su responsabilidad, esto es, su propio campo y en una posición en línea entre la pelota y su portería. Finalmente el robot que comenzó su labor como chutador se mantiene en el campo rival y comienza su desplazamiento hasta la banda contraria sin perder de vista la pelota.

Como se puede ver en las fotografías de los experimentos realizados, la organización conseguida con el empleo de *Switch!* es mucho mejor que cuando no hay ningún tipo de cooperación. Los fotogramas muestran que los robots cubren mucho mejor el campo de juego y sus posiciones ocupan lugares estratégicamente mejor definidos que antes. El vídeo completo con los experimentos en el entorno real puede visualizarse en *http://gsyc.es/∼caguero/tesis/videos/switchJuego.mov*.

4.3.3. Reto del pase en la RoboCup

La edición del 2006 de la RoboCup estrenó por primera vez el reto del pase (*passing technical challenge*). Este desafío fue creado para promover el desarrollo de técnicas de coordinación y comportamientos de pase entre los participantes. La normativa establece que cada equipo utilice tres robots aiBo. La organización es responsable de dibujar tres círculos sobre el campo de juego y depositar un robot en cada círculo, marcando las posiciones de inicio del reto. Estas posiciones son desconocidas de antemano y se proporcionaban en forma de fichero de texto conteniendo los

Figura 4.8: Secuencia de fotogramas de un instante del experimento de *Switch!* utilizando roles dinámicos

tres puntos al inicio del reto. De este modo, los robots conocían su posición de partida y la de sus compañeros, pero se evitaba el desarrollo de pases predefinidos pues estas posiciones no se conocían con anterioridad al inicio de la prueba. La imagen 4.9 muestra la posición de inicio de este reto.

Figura 4.9: Instantánea del entorno de pruebas recreando el desafío del pase de la RoboCup 2006

Tras el comienzo del desafío los robots disponen de 15 segundos para percibir la pelota, detectar a los compañeros, auto-localizarse, etc. Tras este periodo de tiempo el equipo dispone de dos minutos para realizar algún pase. Un pase se considera exitoso si se chuta la pelota desde el interior de un círculo y se detiene por otro robot en el interior de su respectivo círculo.

La arquitectura de comunicaciones desarrollada para el código de *TeamChaos* y el mecanismo de coordinación *Switch!* son los elementos principales empleados para abordar el desarrollo de este desafío. Este reto sirve de experimento en esta tesis, pues afronta una prueba con robots reales y necesidades de coordinación específicas, sin duda un buen test para evaluar el comportamiento de *Switch!*.

Disponemos de una base para implementar el comportamiento cooperante requerido pero también algunos puntos débiles. La mayor restricción es que el código de *TeamChaos* no contempla la posibilidad de avanzar conduciendo la pelota. Esta

limitación fuerza a mantener la pelota controlada en el interior del círculo, pues de otro modo no seremos capaces de traerla de vuelta y disparar el pase.

Como explicamos en la sección 4.2, el mecanismo de intercambio de roles propuesto necesita ajustar una serie de parámetros. A continuación describimos la configuración concreta de *Switch!* para abordar el reto.

1. **Conjunto de roles:** Se han definido los roles de *chutador, receptor1, and receptor2*. El *chutador* debería ir hacia la pelota, apuntar hacia un compañero y chutar la pelota para completar el pase. Los *receptores* deberían orientarse mirando hacia la pelota y ser capaces de detenerla cuando el *chutador* emita el pase.

$$J_1 = Chutador,\ J_2 = Receptor1,\ J_3 = Receptor2$$

2. **Prioridad de la asignación de roles:** La política de prioridades diseñada para los roles dota al rol de chutador de la mayor prioridad, seguida de los roles de receptor. Esto es así para favorecer que el robot mejor situado alcance la pelota. Por tanto, el peso del rol de chutador es mayor que el resto de roles.

$$w_1 = 0{,}50,\ w_2 = 0{,}25,\ w_3 = 0{,}25$$

3. **Funciones de *utilidad*:** Como hemos descrito anteriormente hemos diseñado dos roles diferentes para desplegar el comportamiento cooperativo: Chutador y receptor. Sólo uno de los robots del equipo alcanzará el rol de chutador y, los otros dos, se asignarán los roles de receptor. Por tanto, únicamente se ha declarado una función de utilidad para evaluar quién es el robot más apropiado para la labor de chutador. En caso de que *Switch!* considere que un determinado robot no es el chutador (porque existirá otro compañero mejor situado para esta tarea), automáticamente se le será asociado el rol de receptor.

   La función heurística seleccionada para el rol de chutador está basada en la estimación local de distancia hasta la pelota ($D_{I_i,Pelota}$). Es importante reseñar que menor valor de utilidad será interpretado como mejor grado de adaptación al rol. El coste de intercambio de rol también penaliza a la hora de calcular la utilidad para evitar rápidas fluctuaciones entre los roles.

$$U_{i,Striker} = D_{I_i,Ball} + Costedeintercambioderol$$

4. **Unidad de intercambio de información:** Puesto que la heurística seleccionada para esta prueba únicamente utiliza la estimación local de la distancia hasta la pelota, ésta es la única información que se intercambia entre los robots.

Una vez configurados los parámetros de *Switch!*, necesitamos crear los comportamientos para cada rol. Una de las herramientas de las que dispone *ChaosManager* es la denominada HFSM (Hierarchical Finite State Machine), que permite la creación de jerarquías de máquinas de estados y reutilización de comportamientos de bajo nivel. Se ha utilizado esta herramienta para diseñar un comportamiento común que ejecutarán todos los robots con la estructura de bloques descrita en la figura 4.10.

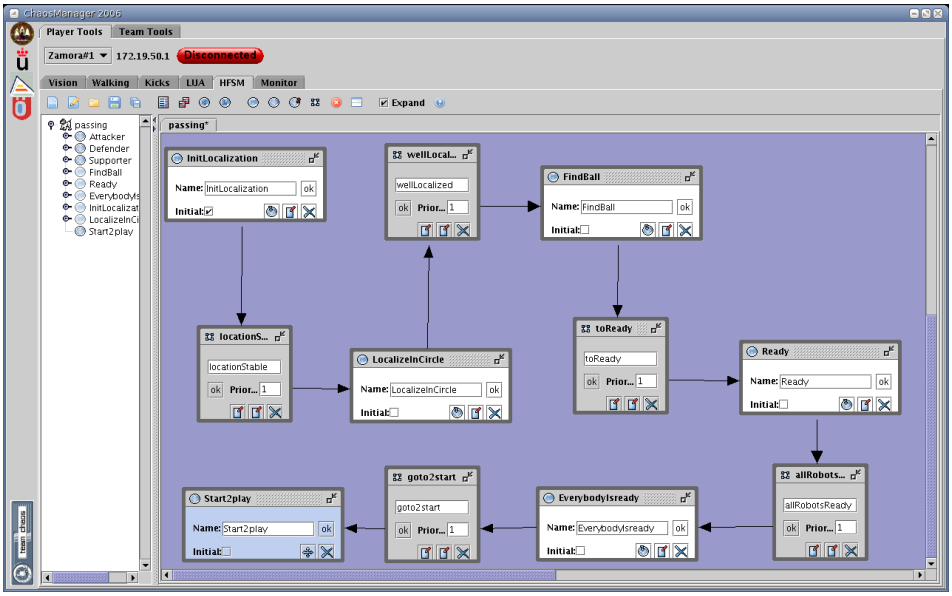


Figura 4.10: Captura de la herramienta HFSM que muestra los estados y transiciones que conforman el diagrama de comportamientos para el desafío del pase

- **Inicialización de la localización (initialize localization):** En este estado los robots rotan sobre sí mismos en busca de información visual útil para alimentar el algoritmo de auto-localización y lograr una auto-localización estable. La duración de este estado es fija, es decir, tras unos segundos se transita al siguiente estado y nunca más se retorna.

- **Asociación de un círculo (localize in circle):** El objetivo de esta fase es conocer en qué círculo se encuentra cada robot, pues éstos no saben si están posicionados en el primero, segundo o tercero. En función de la estimación generado por el algoritmo de auto-localización, cada robot calcula la distancia desde su supuesta posición (una vez que la estimación proporciona suficientes garantías) hasta cada uno de los centros de los tres círculos conocidos. El círculo que permanezca a menor distancia de su hipotética posición será el que elija el primer robot como su posición de partida.

  Este robot compondrá un mensaje para informar al resto de compañeros de que su círculo ya está ocupado, con el fin de reducir el número de opciones posibles al resto de miembros del equipo. Si alguno de los robots sufre problemas para localizarse, este mecanismo le permite obtener su posición por descarte. Se han incluido mensajes de asentimiento en el protocolo de esta fase con el fin de garantizar que todos los robots escuchan los mensajes.

- **Búsqueda de la pelota (find ball):** Una vez conocida la posición de cada robot, todos empiezan a rotar sobre sí mismos en busca de la pelota. En esta fase se ha reutilizado el movimiento de escaneo de cabeza del jugador habitual. Cuando el robot está correctamente orientado puede pasar al siguiente estado pues está preparado para comenzar el pase.

- **Preparado (ready):** Este es un estado de sincronización en el que el robot permanece quieto, esperando que el resto de compañeros terminen sus fases previas. De empezar en este momento el pase, podría darse el caso de que el mejor robot para chutar no estuviera percibiendo la pelota y, por tanto, otro robot actuara de chutador, etc. Necesitamos que todos los robots estén preparados para comenzar el pase en las mejores condiciones que garanticen que el rol de chutador será asignado al robot mejor situado.

  En este estado cada robot envía mediante *broadcast* un mensaje denominado *ready message* a sus compañeros. Este mensaje debe ser confirmado por todos ellos mediante un asentimiento. Se han incluido retransmisiones en el protocolo de sincronización por si fuera necesario.

- **Equipo listo (everybody is ready):** Cuando un robot ha enviado su paquete *ready message*, ha recibido la confirmación de todos sus compañeros y además ha recibido el *ready message* del resto de miembros del grupo, asume que todos los robots están preparados.

- **Comienzo del pase (start to play):** Este estado está siempre guiado por *Switch!*, que selecciona el comportamiento correcto para cada robot. Si a un robot determinado se le ha asignado el rol de chutador, se ejecutará un comportamiento de bajo nivel para ir hacia la pelota. A continuación, el robot apuntará hacia el centro del círculo más cercano utilizando su información de auto-localización y la posición de destino conocida. En ese momento el chutador enviará mediante *broadcast* un paquete *mensaje de pase inminente* para avisar a los compañeros de que está a punto de golpear la pelota.

  Los receptores estarán ejecutando un comportamiento denominado *face ball*, cuya misión es orientar el robot hacia la pelota. En el momento de recibir un *mensaje de pase inminente* o en caso de que la distancia hasta la pelota se reduzca drásticamente, se ejecutará un comportamiento de recepción de pase para tratar de detener la pelota.

  En ese momento, *Switch!* generará un nuevo cambio de roles entre el chutador y el receptor que tiene la pelota. Los papeles se invertirán y comenzarán los comportamientos para realizar otro nuevo pase.

Durante la RoboCup celebrada en 2006 en Bremen sólo cuatro equipos consiguieron realizar algún pase. *TeamChaos* fue uno de ellos y terminó clasificado en tercer lugar, sin duda un éxito que demuestra la robustez y flexibilidad de *Switch!* para diversas tareas que implican cooperación a nivel de equipo.

4.3.4. Competición *German Open* 2007

Durante el año 2007 decidimos participar en la competición robótica *German Open* que se celebró en Hannover durante el mes de abril. Esta competición es la ante-sala de la RoboCup para los equipos europeos, pues acuden a ella muchos de los equipos que tienen pretensiones de victoria en la competición RoboCup que se

celebra en verano. Las aspiraciones del equipo *TeamChaos* no eran las de ganar pero sí acudir para realizar experimentos de primera mano sobre el desarrollo realizado hasta la fecha.

Nuestro objetivo fue acudir para probar en situaciones de juego real un nuevo algoritmo de auto-localización y *Switch!*. Los experimentos en el laboratorio eran prometedores pero un test real es, sin duda, el mejor experimento para comprobar qué efecto tiene sobre el comportamiento del juego el mecanismo de cooperación presentado aquí.

Asistieron diez equipos con intención de participar en la liga de plataforma de estándar con los robots aiBo. Los equipos procedentes de Alemania, Holanda, Turquía, Grecia, Irán, Estados Unidos, Australia y España se enfrentaron entre sí en una liguilla de dos grupos, de los cuáles se clasificaban para la fase final 8 equipos. En esta fase previa se ganaron dos partidos (1-0) y (4-0), se empató un partido (0-0) y se perdió el restante (0-8) con los actuales campeones mundiales. Ocupamos el segundo puesto del grupo, por lo que accedimos a la fase final eliminatoria. En el partido de cuartos de final fuimos eliminados (3-8) por el equipo turco *Cerberus*. Atendiendo a los goles encajados y marcados, ocupamos el quinto puesto de la clasificación general.

El juego desplegado y los resultados cosechados fueron sin duda alguna los mejores logrados hasta la fecha. El equipo logró marcar 8 goles en partidos oficiales, más otros tantos en partidos amistosos. Lamentablemente lo habitual en las competiciones previas era volver sin haber conseguido marcar un sólo gol. El juego mucho mejor organizado, la carencia de penalizaciones por *pushing* y el mejor posicionamiento de los robots en el campo desencadenó la mejora. Por supuesto, *Switch!* no podría haber sido tan eficiente sin la ayuda de un eficiente y preciso algoritmo de auto-localización, que también funcionó de manera excelente. No sólo el carácter ofensivo se vio beneficiado de la incorporación de *Switch!*. El número de goles encajados también se redujo drásticamente comparado con las competiciones anteriores.

La imagen 4.11 muestra una secuencia de fotogramas de una grabación real de un partido en la competición German Open 2007. Nuestros robots pertenecientes al equipo rojo atrapan la pelota durante los tres primeros fotogramas. Se puede apreciar que sólo uno de los robots (el chutador) avanza hasta la pelota, mientras que el resto

Figura 4.11: Secuencia de fotogramas de un instante de juego real en la competición German Open 2007 donde se aprecia el funcionamiento de *Switch!*

se posicionan en sus zonas defensivas y de atacante. El rol de defensa está siendo ejecutado por el robot que vemos en la parte derecha de las imágenes y que permanece en nuestro campo alineándose con la pelota y la portería. El rol de atacante se materializa con el robot que permanece en la parte superior de las imágenes. En todo momento está orientado con la pelota para favorecer su detección.

Durante los fotogramas 4, 5 y 6 la pelota es golpeada y viaja hasta la otra banda. En ese momento se produce un intercambio de roles entre chutador y atacante y ahora se intercambian los papeles. Se observa cómo el nuevo chutador se aproxima hasta

la pelota y la golpea de nuevo. Obsérvese también como el robot defensor corrige su posición adelantándola debido a que la pelota se sitúa prácticamente en la portería contraria. El vídeo completo de este experimento en la competición RoboCup puede visualizarse en *http://gsyc.es/∼caguero/tesis/videos/switchGO2007.mov*.

4.4. Conclusiones

En este capítulo hemos descrito *Switch!*, el mecanismo de intercambio dinámico de roles para equipos de robots móviles. El objetivo fundamental de *Switch!*, presente en cada robot, es evaluar cuál es el rol más apropiado para ser ejecutado por cada robot.

Estos roles han surgido ante la flexibilidad existente en equipos de robots homogéneos, pues todos disponen de los mismos elementos *hardware*. La única diferencia radica en el estado de cada individuo, y ahí es donde entra en juego *Switch!* para maximizar el beneficio del equipo. Se ha diseñado una estrategia de posicionamiento dinámico para cada robot, que establece un punto óptimo de posicionamiento según el objetivo a completar (bloquear disparos del contrario, alcanzar la pelota o apoyar al atacante).

El grado de conveniencia de cada robot a cada rol ha sido calculado utilizando diferentes heurísticas. Todas ellas emplean diferentes parámetros del estado de los robots para su evaluación. Nos hemos valido del concepto de utilidad para unificar los costes asociados a cada rol y tener un indicador para poder comparar las diferentes opciones.

El interfaz ofrecido por *Switch!* es sencillo, pues una vez definidos los roles y sus parámetros de funcionamiento, todos los detalles de la comunicación de información quedan ocultos. Los comportamientos encargados de la toma de decisiones pueden consultar cuál es el rol de juego más apropiado según los criterios especificados.

Switch! ha demostrado ser lo suficientemente flexible como para poder usarse en diferentes aplicaciones. En los experimentos hemos diseñado un conjunto de roles para el juego del fútbol y otro para el reto del pase de la competición RoboCup. Además, se han utilizado entornos tanto simulados como reales y en ambos ha sido

posible programar satisfactoriamente comportamientos cooperativos de asignación de tareas.

En cuanto a los resultados obtenidos, las pruebas de simulación han sido útiles para contrastar diferentes técnicas de posicionamiento y realizar labores de depuración. Las simulaciones de partidos realizadas demostraron el juego con *Switch!* genera un juego más ordenado y más goles que la variante de juego sin cooperación.

Las pruebas de laboratorio han ayudado a verificar el correcto intercambio de roles entre los robots, así como la velocidad y estabilidad de los mismos. Los resultados demuestran que *Switch!* puede usarse en tiempo real, pues el intercambio de roles es prácticamente instantáneo y raramente se producen interferencias entre robots (rápidos intercambios continuados de roles, asignación del mismo rol, etc.).

La RoboCup también ha sido objeto de experimentos para *Switch!*, experimentando un gran crecimiento en cuanto a número de goles obtenidos en la competición. Es difícil medir cuantitativamente el impacto de *Switch!* en el juego, pero la calidad del comportamiento de equipo es claramente superior utilizando un enfoque colaborativo. El equipo defiende mejor debido al papel del robot defensa, dedicado en exclusiva a esta labor. En la fase previa de la RoboCup donde se experimentó con *Switch!*, únicamente con el equipo campeón del mundo de ese año se encajaron goles. El carácter ofensivo del equipo también se vio reforzado, pues se marcaron 8 goles en toda la competición, cuando el año anterior únicamente se logró un único gol.

La imbricación del mecanismo de percepción distribuida dentro de *Switch!* ha aportado algunas ventajas. La mayor estabilidad de la estimación de la pelota ha provocado un menor número de oscilaciones entre robots equidistantes a la pelota. Además, el rol de defensa se beneficia en gran medida, pues es muy habitual que desde su posición no sea capaz de percibir directamente la pelota. La gran cantidad de robots que puede haber entre la pelota y él mismo dificultan esta labor, sin embargo si que es posible corregir su posición continuamente en base a la estimación distribuida.

CAPÍTULO 5

Conclusiones

Last words are for people who haven't said anything in life.

– Karl Marx

En los capítulos anteriores hemos introducido el problema que esta tesis aborda, se han repasado y organizado las referencias bibliográficas más representativas y hemos descrito las técnicas propuestas al hilo de la percepción compartida e intercambio dinámico de roles. Este capítulo resume las respuestas que ofrece esta tesis a las preguntas que nos planteamos en el capítulo 1. Se realizará un repaso a los objetivos planteados al comienzo del documento, verificando el grado de satisfacción de los mismos y analizando los principales aportes de esta tesis. El final del capítulo está dedicado a las líneas de investigación abiertas y trabajos futuros asociados al trabajo aquí realizado.

5.1. Percepción distribuida aplicada a la asignación dinámica de roles en equipos de robots móviles

La adición de un comportamiento de cooperación a un equipo de robots móviles es el objetivo principal de esta tesis. De entre los diversos enfoques de coordinación posibles, el aquí presentado conjunta percepción y acción. La pieza encargada de combinar la perpepción grupal es el mecanismo Bayesiano de percepción distribuida. El componente que modifica las acciones del equipo es *Switch!*, la técnica de intercambio dinámico de roles aquí presentada.

Una de las preguntas planteadas en este trabajo es si es posible mejorar la estimación de un objeto, que un robot realiza de manera individual. La literatura de fusión sensorial estudiada ofrece un marco de trabajo sólido y con garantías para dar una respuesta positiva a esta pregunta. Sin embargo, el objeto a estimar en nuestro caso es el eje central sobre el se disputan los partidos de fútbol robótico en la RoboCup y presenta algunas dificultades añadidas: por un lado todos los robots pretenden alcanzar la pelota, por lo que es muy probable que existan varios robots permanentemente cerca de la bola causando sombras, oclusiones a otros robots, etc. Además la pelota es un objeto móvil y puede desplazarse a gran velocidad si es golpeada por algún robot. El seguimiento de un objeto en movimiento es una tarea mucho más complicada que la de la estimación de un objeto estático. Existe otra dificultad añadida que complica la percepción de la pelota: lo poco aislado que se encuentra el entorno ante variaciones de iluminación, público rodeando el campo de juego, etc.

Los diseños que hemos realizado abarcan métodos analíticos, soluciones que discretizan el entorno y estimadores basados en filtros Bayesianos que acumulan evidencias, tienen en cuenta la historia pasada, el presente e incluso predicen cuál puede ser el estado en el futuro. En los experimentos hemos recreado múltiples situaciones características de los encuentros en la RoboCup para caracterizar las diferentes implementaciones. Las soluciones basadas en filtro de Kalman y filtros de partículas han demostrado ser las soluciones más sólidas y apropiadas para este problema.

Estas dos técnicas compartidas, a la vista de los experimentos realizados, demuestran mejorar en media la precisión de las estimaciones individuales que realizan los miembros del equipo. Las razones de esta ventaja que ofrece la variante percepti-

va distribuida son múltiples: Por un lado, debido a los filtros empleados los díscolos son amortiguados y se evitan falsas observaciones que deterioran la calidad de la estimación. Por otro lado, las observaciones son combinadas de manera ponderada y así, robots que aportan buenas observaciones tienen más peso en las decisiones, frente a robots con mayor incertidumbre ya sea en un su posición u observación.

Las oclusiones son un denominador común en entornos altamente dinámicos. El problema de la RoboCup, ámbito principal de esta tesis, es un claro ejemplo donde los robots sufren numerosas oclusiones durante las competiciones. Las técnicas de fusión de información distribuida mejoran sustancialmente este punto. Un robot que no percibe la pelota no tiene posibilidad de situarla en el campo. Sin embargo, gracias al mecanismo distribuido es posible conocer siempre, con mayor o menor grado de incertidumbre, la posición de la pelota. En la competición de la RoboCup, el portero es uno de los grandes beneficiados de esta característica. Es muy habitual que la pelota esté lejos de su radio de acción y, en consecuencia, no sea posible percibirla directamente. Sin embargo, es habitual disparar a puerta desde el medio campo o incluso desde el campo propio si la situación es favorable. Si el portero puede estar previamente posicionado, dispone de muchas más opciones de bloquear la pelota.

Los sistemas de percepción están muy lejos de ser altamente fiables y exentos de errores. Los falsos positivos también ocurren con relativa frecuencia y es deseable que la percepción distribuida ayude en este sentido. Los experimentos propuestos revelan que posiblemente, este tipo de situaciones sean el peor aliado de las técnicas Bayesianas. Un falso positivo con baja incertidumbre modifica en gran medida la estimación compartida y, por tanto, disminuye la calidad de la estimación de la pelota. Los resultados demuestran que la técnica basada en filtro de partículas tolera mucho mejor este tipo de problemas. La razón principal es que, a pesar de su carácter multimodal, la tendencia es a que las hipótesis convergan hacia las zonas de mayor densidad de probabilidad. Por esta razón, si las partículas de dos robots se sitúan en la zona del campo donde se encuentra la pelota, éstas serán mayoría frente a las partículas que se acumulen en la zona del falso positivo y no se dejarán dominar en exceso. La variante basada en Kalman usa como observación la combinación instantánea de la última observación de cada robot. Si esta observación combinada instantánea ya

está contaminada, el filtro de Kalman modificará su estimación magnificando el error cometido.

La robustez en la estimación o estabilidad es otro de los requisitos exigidos al mecanismo de percepción distribuida. Como han demostrado las pruebas realizadas, la incertidumbre asociada a la estimación del equipo es menor que la homóloga individual en media. Este hecho demuestra que las oscilaciones son mucho menores. Las decisiones de actuación están muy ligadas a la posición de la pelota. Esta mayor estabilidad en la estimación repercute positivamente en el comportamiento de los robots, pues sus acciones son más limpias y carentes de correcciones y oscilaciones constantes.

El estudio del problema de la percepción distribuida ha desembocado en cuatro implementaciones realizadas: La primera de ellas emplea un método analítico basado en la ganacia de Kalman para combinar estimaciones instantáneas procedentes de miembros del equipo. La ventaja fundamental de esta técnica frente al resto de implementaciones es la eficiencia. Sin embargo, su mayor desventaja es la ausencia de acumulación temporal, información muy valiosa para el mantenimiento de una estimación estable.

La variante basada en rejilla probabilística da respuesta al enfoque basado en dividir el espacio de estados contínuo en celdas discretas. La gran cantidad de celdas existentes cuando la precisión es centimétrica convierte en inusable por completo esta técnica. Se obtienen tiempos de cómputo por iteración muy elevados (del orden de minutos si aumentamos la resolución de la rejilla) y, aunque los resultados son satisfactorios, los requerimientos de vivacidad descartan por completo esta técnica.

El mecanismo de percepción distribuida basado en filtro de Kalman ha demostrado ser una alternativa muy buena para este problema. Su carácter analítico requiere de alguna aproximación para integrar las dos fuentes de incertidumbre en una única distribución Gaussiana. Una vez resuelto este problema, los experimentos han demostrado que su precisión es buena y su eficiencia es muy elevada. Como desventajas principales podríamos citar su peor comportamiento ante elevada incertidumbre angular en la posición del observador, debido a la aproximación utilizada para integrar posición y observación. Su precisión también es menor ante falsos positivos comparado con la alternativa muestreada.

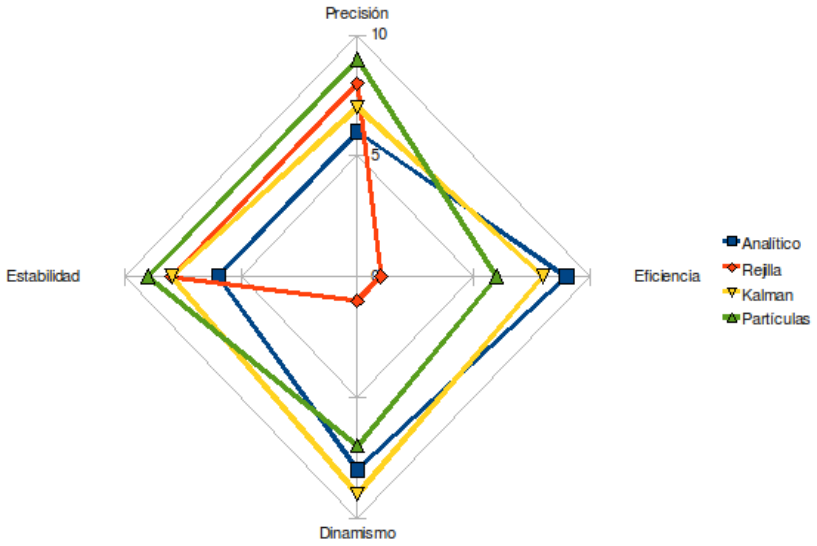


Figura 5.1: Diagrama comparativo entre las técnicas propuestas de percepción distribuida

Switch! es el segundo argumento que defiende esta tesis para mejorar el comportamiento de un equipo de robots móviles. Los experimentos y resultados experimentales obtenidos en competiciones reales han demostrado que la coordinación mediante roles mejora notablemente la organización del grupo. La técnica implementada es lo suficiente genérica como para utilizarse en diferentes situaciones. Por ejemplo ha sido la base de los roles de juego en los partidos de fútbol y también de los roles para el desafío técnico del pase en la RoboCup.

El juego en los partidos de fútbol robótico ha sufrido un notable incremento de calidad. La ausencia de coordinación causaba constantes penalizaciones debido al empuje repetido de unos robots con otros. Además, provocaba un descubrimiento de las zonas del campo donde no se encontraba la pelota, dificultando la recogida de la misma después de un disparo o pase. El número de goles en competiciones oficiales ha demostrado el impacto real de *Switch!* en el juego.

Además de la flexibilidad demostrada también ha demostrado gran robustez. La selección de roles responde de manera precisa a los cambios del entorno y no se producen apenas errores en las asignaciones. Sin duda, en la calidad de la robustez y estabilidad influye el empleo del sistema de percepción distribuida. La obtención

de estimaciones de calidad radica en una selección de acción mucho más acertada y libre de fallos.

Aunque la juventud de la arquitectura desarrollada para el robot Nao no ha permitido realizar unos experimentos tan extensos como con el robot aiBo, los resultados preliminares obtenidos refuerzan las sensaciones obtenidas con el grupo de robots aiBo. Sin duda, combinar compensa y *Switch!* supone una capa de abstracción muy grande a la hora de diseñar y programar roles en un equipo de robots.

5.2. Aportes realizados

El objetivo principal de esta tesis ha sido el diseño, implementación y validación de mecanismos de percepción y acción cooperativos para un equipo de robots móviles. Como hemos detallado a lo largo de este documento, se ha realizado una implementación real y funcional del mecanismo de percepción distribuida en dos plataformas robóticas diferentes, así como una implementación completa en el robot aiBo que abarca la percepción compartida y el intercambio dinámico de roles. Se han utilizado simuladores durante la fase de diseño, pero un aporte importante de este trabajo es la consecución final de un bloque software completamente operativo que satisface los requisitos planteados y opera con robots reales en un entorno difícil y competitivo como el de la RoboCup.

La arquitectura completa BICA, así como el módulo de comunicaciones TCM de la arquitectura TeamChaos también son aportes directos derivados del trabajo de esta tesis. No son el objeto fundamental de la misma, pero sí ha sido necesaria una labor de estudio, desarrollo y pruebas para integrar las técnicas que aquí nos ocupan dentro de robots reales.

La fase de estudio bibliográfico también aporta un repaso, caracterización y organización del estado del arte en cuanto a trabajos muy relacionados con esta tesis. Se han barrido los estudios de numerosos equipos participantes en la RoboCup, así como gran cantidad de documentos relacionados con fusión sensorial, percepción distribuida e intercambio dinámico de roles y reparto de tareas en grupos de robots. El capítulo 2 ofrece una visión actual del panorama cooperativo en los dos pilares fundamentales que aquí se tratan, especialmente dentro del entorno de la RoboCup.

Es mérito de este trabajo la labor de clasificación de este campo relativamente joven de la robótica, muy heterogéneo y en ocasiones escasamente sistemático.

El capítulo 3 presenta varias soluciones al problema de la estimación compartida, demostrando que suponen una ventaja para la mejora de las capacidades perceptivas dentro de un equipo de robots. Este estudio aporta un paso más hacia la mejora y apuesta por sistemas distribuidos frente al enfoque clásico individual. En este capítulo se ha efectuado una caracterización de las alternativas diseñadas, así como una comparativa entre ellas. Esta comparativa resulta de elevado interés, pues establece algunas pautas para elegir la técnica mejor adaptada a un determinado problema. No existe una técnica ideal, todas ellas disponen de ventajas e inconvenientes y es necesario buscar el mejor valor de compromiso según la aplicación donde intervengan.

El mantenimiento de una estimación completa combinando las dos fuentes principales de incertidumbre (posición del robot y observación) es una contribución importante de este trabajo. Existen varios trabajos que utilizan mecanismos de combinación de observaciones pero, sin embargo, no es habitual en la literatura encontrar desarrollos que combinen las dos fuentes de incertidumbre. En este sentido, la aproximación basada en suma de englobantes para las variantes analítica y basada en filtro de Kalman supone una primera aproximación al problema. Sin embargo, las alternativas basadas en rejillas probabilísticas y filtro partículas abordan el problema empleando el operador de convolución obteniendo unos resultados muy satisfactorios. Los algoritmos de auto-localización están ampliamente estudiados y caracterizados por la comunidad robótica. Estos algoritmos permiten posicionar a los robots dentro de un mapa del entorno pero siempre estarán rodeados de errores e incertidumbre, pues las observaciones así lo son. Es por tanto indispensable tener en cuenta este aspecto para estimar la posición de un objeto en un marco global, pues de otra manera, la estimación realizada estará muy lejos de ser real.

El reparto de tareas en equipos de robots también ha sido estudiado en trabajos anteriores. Sin embargo, la alternativa aquí presentada supone un aporte extra, pues incluye en la estrategia de reparto de roles un parámetro extra obtenido de manera cooperativa. Este hecho dota a *Switch!* de una mayor estabilidad y robustez de funcionamiento. En cierta medida se obtiene un beneficio añadido debido a la realimentación mutua del sistema de percepción distribuida y *Switch!*. Si se percibe mejor

la pelota, se decidirá mejor el rol más adecuado para cada robot y su posición óptima. Si se decide mejor el papel de cada robot en el campo, la percepción también se beneficiará de ello, pues se abarcará mayor superficie para percibir los objetos.

5.3. Líneas futuras

Una vez descrito todo el trabajo, aportes y limitaciones principales de las técnicas aquí presentadas, esta sección aborda las líneas de investigación futuras que han surgido alrededor de esta tesis.

Una primera posibilidad de ampliación de este trabajo es la generalización a otros estímulos fijos (porterías, líneas del campo, etc.). La diferencia fundamental entre estos estímulos y la pelota es su carácter estático. Sería interesante estudiar el comportamiento de las técnicas aquí presentadas ante objetos no móviles y si existen diferencias notorias frente a objetos dinámicos.

Los mecanismos de cooperación descritos suponen una ventaja o ayuda para el sistema perceptivo y la decisión de acción. La auto-localización es otra gran aplicación que puede sacar gran provecho del empleo de un equipo coordinado de robots. Si pensamos en la pelota como un objeto situado en una posición del campo y visible por los miembros del equipo, podemos imaginarla como una baliza visual cuya posición puede variar a lo largo del tiempo. Si además el sistema perceptivo es capaz de identificar y estimar la posición de otros robots, toda esta información puede emplearse para corregir y mejorar la calidad de la auto-localización.

La base probabilística y de acumulación de evidencias sobre la que se sustenta gran parte del trabajo de esta tesis es común al problema de la auto-localización y, por tanto, está íntimamente relacionada con las cuestiones aquí abordadas. Además, al igual que ocurre con la posición de la pelota, la posición de cada robot es un factor muy importante para *Switch!*, pues de ello depende el cálculo de utilidades y por ende la mejor calidad en la selección de roles y acciones.

La mejora del algoritmo de fusión sensorial multi-robot también puede seguir depurándose para mejorar sus limitaciones. Una de ellas es el mayor ancho de banda consumido en el algoritmo basado en filtro de partículas, en relación a los métodos basados en FDPs parametrizadas. Una posible solución para paliar esta carencia po-

dría pasar por enviar resúmenes de las hipótesis existentes en cada observador. Estos resúmenes podrían continuar siendo partículas en menor número, conjuntos de distribuciones Gaussianas o, en general, cualquier otro tipo de distribución que se asemeje a la original pero de menor coste comunicacional.

APÉNDICE A

Software del equipo Team Chaos para el robot aiBo

Este apéndice describe y profundiza en algunos aspectos del software utilizado y desarrollado en esta tesis para participar en la competición RoboCup con los robots aiBo. Inicialmente el equipo TeamChaos tomó parte en la liga de robots de cuatro patas, que desde 2008 pasó a denominarse liga de plataforma estándar (de cuatro patas). Los aspectos aquí detallados se corresponden con el estado del código fuente durante el periodo de 2006 a 2008. A partir de esta fecha los esfuerzos de desarrollo al hilo de la RoboCup se han centrado en la liga de plataforma estándar, que se describirá en el siguiente apéndice.

Como parte de esta tesis se han diseñado e integrado todas las técnicas y herramientas de estimación cooperativa y *Switch!* dentro de la plataforma de TeamChaos para los robot aiBo. Esta arquitectura ha sido la base para realizar varios experimentos en esta tesis. Esto hecho ha permitido una mayor velocidad de desarrollo y mejor capacidad de experimentación, pues los módulos básicos de locomoción, percepción, etc. no han tenido que ser programados. La capa de comunicaciones ha supuesto una excepción, pues siendo una pieza básica tuvo que reescribirse. Este módulo se describe la sección A.2 con mayor profundidad.

TeamChaos es un equipo formado por varias universidades y con una larga trayectoria de participación en la RoboCup. Originariamente fue creado por Alessandro Saffiotti (actualmente en la Universidad de Örebro) con el nombre original de *Team Sweden*. Una extensa lista de investigadores han aportado sus contribuciones en mayor o menor medida hasta el estado actual del equipo y de su *software*. Actualmente el núcleo central está formado por un consorcio de grupos de investigación procedentes de la Universidad de Murcia, Universidad Carlos III, Universidad de León, Universidad Rovila i Virgili y Universidad Rey Juan Carlos. Humberto Martínez Barberá (Universidad de Murcia) es el coordinador central del equipo, así como el *Team Leader* en la competición RoboCup. La Universidad de Alicante también participó activamente en el desarrollo de TeamChaos hasta el año 2006.

La incorporación de la Universidad Rey Juan Carlos se produjo en el año 2004. Desde entonces el equipo TeamChaos ha participado en numerosas competiciones y eventos tanto nacionales como internacionales. Las más importantes han sido: RoboCup 2005 (Osaka, Japón), RoboCup 2006 (Bremen, Alemania), RoboCup 2008 (Suzhou, China) y RoboCup 2009 (Graz, Austria). Asimismo ha acudido a competiciones internacionales de reconocido prestigio: German Open 2005 (Paderborn, Alemania), European Open 2006 (Eindhoven, Holanda), German Open 2007 (Hannover, Alemania), Campeonato Latinoamericano 2007 (Santiago, Chile), German Open 2009 (Hannover, Alemania). Finalmente TeamChaos también ha acudido a eventos nacionales para mostrar sus progresos en las citas de mayor interés para la comunidad robótica nacional.

A.1. Arquitectura general

A continuación se describen los principales aspectos de la arquitectura que utiliza TeamChaos, que ha sido el eje central sobre la que se ha realizado la experimentación con los robots aiBo de las técnicas presentadas en esta tesis.

El software que comúnmente denominamos TeamChaos está compuesto por dos bloques principales: El código que ejecuta en el propio robot y el conjunto de herramientas, denominado ChaosManager, que ejecuta en un PC convencional. En esta

sección nos centraremos en el código dedicado al robot y la sección A.3 describirá los detalles de la herramientas de monitorización y depuración ChaosManager.

La arquitectura *software* marca cómo se organiza todo el código que ejecuta el robot, establece cómo es el ciclo de control, cuáles son sus módulos principales, cómo se intercambia información entre cada módulo, etc. Es sin duda el armazón de todo el software de un robot y su diseño condiciona muchas de las características del funcionamiento del mismo.

En el caso que nos ocupa la arquitectura de TeamChaos es una evolución de la arquitectura *ThinkingCap* [MS01], que se puede observar en la figura A.1. El entorno de programación proporcionado por Sony fuerza a que el diseño se haga utilizando objetos *OPEN-R*. Estos objetos son las unidades de ejecución, actuando cada objeto como un proceso independiente. Bajo estos objetos *OPEN-R* se encapsulan los módulos de TeamChaos. Un módulo está formado por una o varias clases C++ que engloban una funcionalidad determinada. De esta manera tendremos el módulo encargado de la locomoción, el módulo responsable de la percepción, etc. Estos módulos se organizan dentro de cada objeto *OPEN-R* según la arquitectura implementada.

El código del robot aiBo se compone de tres objetos *OPEN-R*: ORHRobot, ORLRobot y ORTCM, que pasamos a describir a continuación.

El objeto *ORHRobot* tiene como misión aglutinar todos los módulos no dependientes de la plataforma. Los módulos que contiene se denominan *Gm (Global Map)* y *Ctrl (ConTRoLler module)*. El primero de ellos recibe la información sensorial procedente del objeto *ORLRobot* en forma de LPS (*Local Perception System*) y odometría. LPS permite estimar la posición de los objetos relevantes desde el sistema de referencia solidario al robot. Con esta información, *Gm* mantiene un mapa del entorno, así como la posición actualizada del robot en él. El cerebro de este módulo es el algoritmo probabilístico de auto-localización, que en sus diversas implementaciones puede utilizar técnicas basadas en filtro de Kalman, lógica borrosa o Markov.

El módulo *Ctrl* es el encargado de decidir qué comportamientos de movimiento debe desplegar el robot. Se emplea un control en velocidad para gobernar la locomoción del robot aiBo. La información sensorial, de auto-localización e incluso estratégica son las entradas de las que dispone este módulo para decidir un comando de movimiento o incluso un golpeo de la pelota.

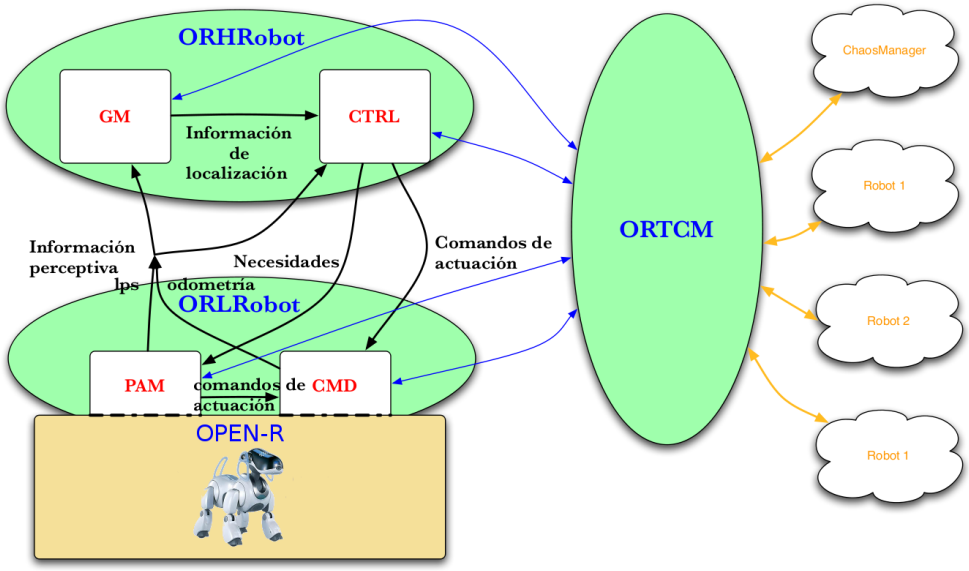


Figura A.1: Arquitectura del TeamChaos para el robot aiBo

Figura A.2: Entradas y salidas del módulo GM

Debido a la complejidad de los posibles comportamientos requeridos, este módulo está subdividido en dos niveles. La capa superior o *HFSM (Hierarchical Finite State Machine)* utiliza un conjunto de máquinas de estados jerárquicas para transitar entre estados del juego. Cada estado define un comportamiento del robot y los diferentes cambios en el juego se corresponden con las transiciones entre estados.

Una vez establecido el estado actual del robot, se ejecutará un comportamiento de bajo nivel. Estos comportamientos corresponden a la capa inferior del módulo *Ctrl*. Están escritos en lenguaje LUA, a diferencia de los de la capa superior que están escritos en C++. Estos comportamientos son directamente ejecutados por un intérprete de LUA en tiempo de ejecución. Cada comportamiento LUA encapsula una funcionalidad básica como buscar la pelota, ir a una determinada posición del campo, etc.

Las ventajas de emplear código LUA es la fácil depuración de los comportamientos básicos al vuelo, sin necesidad de detener y arrancar de nuevo el robot.

El segundo objeto *OPEN-R* que vertebra la estructura del código del robot es el objeto *ORLRobot*. La letra *L* de su nombre expresa el bajo nivel (*Low*) que tiene como cometido principal dentro de la arquitectura. Se encarga de tareas dependientes del hardware del robot, como son el acceso a los sensores y el control directo de los actuadores (patas, cabeza, etc.). Este objeto contiene dos módulos denominados CMD (*CoMmanDer*) y PAM (*Perceptual Anchoring Module*).

El módulo CMD es capaz de convertir los comandos de velocidad recibidos en secuencias de posiciones para las articulaciones de las patas. Este módulo es el que implementa la locomoción y las diferentes maneras de moverse, ya sea de forma lateral, hacia delante o atrás, rotando sobre sí mismo o combinando todos estos movimientos. A su vez es el encargado de generar las secuencias de movimientos específicas para realizar los golpeos de pelota, paradas en caso del portero o cualquier movimiento predefinido que se quiera ejecutar. Otra de sus misiones es recibir comandos de movimiento para el cuello procedentes del módulo PAM. CMD convierte estos comandos en movimiento de la cabeza para implementar una percepción activa del entorno.

PAM (*Perception Anchoring Module*) contiene todo lo relacionado con la percepción. Es un módulo que comienza capturando una imagen procedente de la cámara del aiBo y tras sucesivas etapas de filtrado, cálculo del horizonte visual, segmentación y reconocimiento de objetos, consigue identificar los objetos relevantes del entorno. La distancia y ángulo hasta esos objetos también es estimada y se almacena un modelo del entorno que rodea la robot. PAM también contiene un mecanismo de visión activa o selectiva en función de las necesidades del juego. Cada vez que se percibe un objeto la *memoria* de ese objeto se refresca, es decir, la creencia de la posición de ese objeto se refuerza. A medida que el tiempo transcurre y un objeto deja de ser percibido, su posición en el entorno va perdiendo credibilidad y se olvida. El mecanismo de atención basado en necesidades permite refrescar selectivamente aquellos objetos que necesitamos ver, porque de otra manera cae su credibilidad. La manera de conseguir que la necesidad disminuya es mirando a ese objeto. Ese es otro de los cometidos del PAM, enviar secuencias de movimiento a la cabeza (a través del CMD), para que los

objetos de interés se perciban cada cierto tiempo. En [SL00] se pueden consultar los detalles de esta técnica.

El último objeto *OPEN-R* que completa la arquitectura de TeamChaos para el robot aiBo es el objeto *ORTcm*. Debido a que es especialmente interesante para el contenido de esta tesis vamos a dedicar la siguiente sección a describir este objeto, cuya misión fundamental es la de servir de pasarela a cualquier módulo que quiera comunicarse con otro robot o con un PC. Este módulo ha sido desarrollado por completo para el desarrollo de este trabajo.

A.2. Módulo de comunicaciones

ORTcm es un objeto *OPEN-R* que gestiona las comunicaciones entre un robot aiBo y cualquier proceso externo (ya sea otro robot aiBo, un PC, etc.). Contiene el módulo TCM (*Team Communication Module*), que implementa la funcionalidad de *centro de comunicaciones*, es decir, módulo centralizado que utilizan el resto de módulos para comunicarse con el exterior.

Se ha desarrollado un protocolo de comunicaciones sin estado, no existen establecimientos ni finalización de conexión. Otra característica importante es que las transmisiones no garantizan fiabilidad. Debido a las restricciones de tiempo real requeridas en este entorno, la comunicación no utiliza asentimientos ni retransmisiones. El empleo de asentimientos ralentizaría la comunicación y además, en este tipo de entorno, no tiene sentido el recibir un dato retrasado pues seguramente ya no resulte útil. Es preferible perder determinada información en un instante determinado a retransmitir numerosas veces la información y recibirla cuando ya es vieja.

OPEN-R ofrece soporte para utilizar tanto TCP como UDP como nivel de transporte. Tcm se ha desarrollado empleando UDP para garantizar la ligereza del protocolo y velocidad para garantizar comunicaciones en tiempo real.

La clase *ExternalMessage* implementa el protocolo de comunicaciones y es la unidad de intercambio de información entra las entidades que quieren comunicar. La figura A.3 muestra el conjunto de campos que incluyen los mensajes: Origen, destino, módulo de origen, módulo de destino, longitud de los datos, datos, etc.

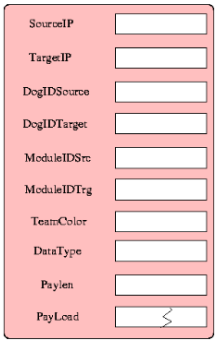


Figura A.3: Objeto de tipo ExternalMessage con sus campos relevantes

Cada robot dispone de una única dirección IP que se asocia al identificador del jugador mediante el fichero de configuración *TCM.ini*. Si algún robot envía un mensaje a un robot concreto (usando su identificador de juego), la consecuencia será que TCM generará un mensaje *unicast* hacia la dirección IP asociada a ese robot. TCM también implementa envíos *broadcast*. Esta característica es especialmente útil para enviar información de depuración (que será analizada y visualizada por ChaosManager) o enviar información de cooperación entre todos los miembros del equipo.

La clase *UDPConn* se utiliza para proporcionar una capa de abstracción mayor que la que ofrece OPEN-R para acceder a la pila de comunicaciones IP.

Otro de los aspectos interesantes de TCM es su capacidad para enviar/recibir cualquier tipo de información. TCM obliga al módulo que quiere usar sus servicios, a que el tipo de datos a enviar/recibir herede del interfaz *iSerializable*. Este interfaz fuerza a implementar los métodos para serializar y componer ese tipo de datos concreto. Garantizando que el objeto a enviar/recibir cumple este requisito, TCM hace el resto y resuelve la tarea de comunicarlo con la entidad solicitada.

El módulo de comunicaciones incorpora también un mecanismo de *callbacks*, que permite a un módulo registrar un determinado método asociado a un tipo de datos. Cuando este tipo de datos concreto es recibido en el robot y dirigido a ese módulo, el método previamente registrado es invocado de manera automática.

TCM también aborda el problema del troceado/composición de mensajes de tamaño superior a lo permitido en un datagrama UDP. El módulo es capaz de trocear los mensajes que así lo requieran y componerlos de nuevo en el destino. Esta caracte-

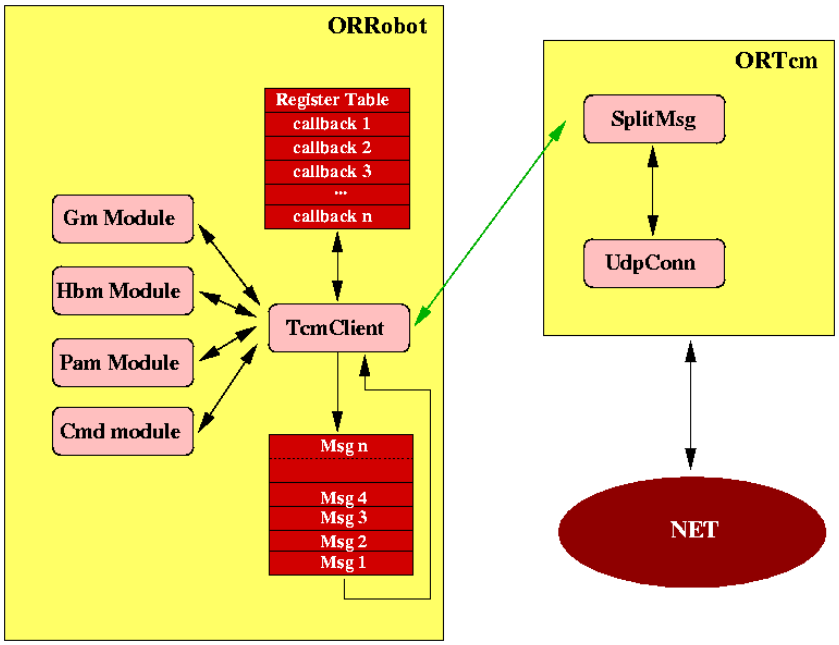


Figura A.4: Esquema de comunicación entre los módulos y el TCM

rística es muy útil para enviar grandes ficheros de configuración o imágenes, que no suelen caber en un único datagrama.

Además, se ha creado la clase *TcmClient* para ocultar los detalles de la comunicación entre objetos OPEN-R a los módulos que quieren utilizar la funcionalidad de TCM. *TcmClient* actúa como proxy o representante ofreciendo un sencillo interfaz para enviar o recibir información. Se ha utilizado el patrón de diseño *Singleton* para mantener una única instancia de *TcmClient* compartida por todos los módulos, protegida debidamente para evitar condiciones de carrera. *TcmClient* dispone de colas de almacenamiento para guardar y regular el flujo de información que se transfiere al módulo TCM. La figura A.4 muestra el esquema de funcionamiento de las comunicaciones.

A.3. La aplicación *ChaosManager*

La aplicación ChaosManager es una herramienta fundamental en el desarrollo del software del equipo TeamChaos. Tan importante es desarrollar buenos algorit-

mos como tener las herramientas adecuadas para depurar, visualizar y automatizar las tareas más habituales a realizar. ChaosManager persigue este objetivo y puede conectarse a un robot para verificar su funcionamiento, generar los *memory sticks* con el código, etc.

Esta aplicación escrita en Java permite ejecutarse en cualquier ordenador, ofreciendo soporte para multitud de sistemas operativos.

ChaosManager posee una interfaz gráfica dividida en pestañas que organizan las diferentes herramientas con las que cuenta la aplicación. En una primera pestaña se encuentran las funciones que atañen a un único jugador (*player tools*) y en otra las que repercuten a todo el equipo (*team tools*).

Entre las herramientas de equipo podemos encontrar un generador automático de *memory sticks*, donde es posible configurar algunos parámetros de red para cada robot, especificar algoritmos concretos de auto-localización, seleccionar la máquina de estados principal a ejecutar por el módulo CTRL, etc.

Otra herramienta de equipo disponible es un visor de información de depuración emitida por cada robot. Es posible comprobar la posición de cada robot en el campo según su propio algoritmo de auto-localización, verificar la estimación de la pelota de cada robot, etc.

Finalmente también es posible manipular el árbitro electrónico o *Game Controller* integrado en la pestaña de herramientas de equipo. Esta herramienta es similar a la utilizada en los encuentros de RoboCup y permite llevar el tiempo del partido, sancionar a los robots o especificar el color de cada equipo, entre otras muchas funciones.

La pestaña de herramientas para un jugador concreto está a su vez compuesta por varias herramientas. Es requisito indispensable conectarse a un robot concreto antes de utilizar cualquier herramienta de esta sección.

La cámara es el sensor principal del robot aiBo y ChaosManager cuenta con una pestaña específica para calibrar los parámetros de la cámara, así como configurar las tablas de color para las fases de filtrado. A su vez es posible configurar diferentes aspectos de los algoritmos de segmentación, cálculo del horizonte visual y reconocimiento de objetos.

La locomoción también tiene una herramienta específica para su configuración y depuración. Con ella podemos calibrar el modo de caminar atendiendo al suelo sobre el que gatean los robots, para obtener unos parámetros de odometría lo más precisos posible.

Además de la locomoción el robot también debe generar secuencias de movimientos fijas para chutar la pelota, realizar pases, atrapar la pelota o incluso tratar de despejarla (para el caso del portero). ChaosManager dispone en su sección de herramientas para jugador de otra pestaña dedicada a generar secuencias de movimiento predefinidas. Estas secuencias funcionan de manera parecida a como lo hacen los fotogramas en las películas. Se establecen los valores de todas las articulaciones para un instante determinado, después para el siguiente, y así sucesivamente. Ejecutando de una sola vez la secuencia de fotogramas obtenemos el movimiento predefinido en el robot.

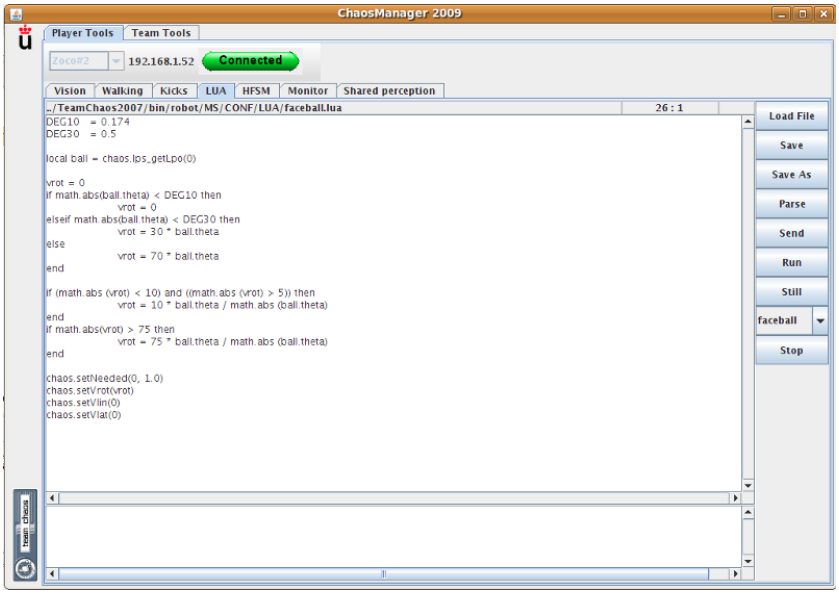


Figura A.5: Editor de lenguaje LUA para escribir y depurar comportamientos básicos

La siguiente utilidad es un editor de comportamientos escritos en LUA (figura A.5). Como describimos en la sección anterior, los comportamientos básicos del robot aiBo están escritos en LUA y son interpretados al vuelo en el robot. Con este editor

podemos modificar cada uno de estos comportamientos e incluso actualizarlos en el robot para comprobar su efecto sobre él.

La herramienta HFSM permite crear máquinas de estado de manera gráfica, especificando la secuencia de acciones para cada estado, así como las condiciones para cada transición. Esta herramienta permite generar el código del autómata en lenguaje LUA para que sea ejecutado por el robot.

El *Monitor* es un visualizador de los objetos relevantes alrededor del robot. Con esta herramienta podemos conocer el estado de las estimaciones de la pelota o las porterías desde un robot determinado.

Hemos extendido la funcionalidad implementada en la pestaña *Monitor* para visualizar gráficamente la información procedente de nuestro algoritmo de percepción distribuida. La figura A.6 muestra la apariencia de la pestaña desarrollada. Se ha incluido la posibilidad de visualizar cualquiera de las técnicas de fusión implementadas. En la imagen puede apreciarse la rejilla probabilística en un instante de las pruebas, así como la posición y estimación basada en filtro de Kalman.

Figura A.6: Pestaña integrada en la aplicación ChaosManager para visualizar información sobre la estimación compartida

APÉNDICE B

Software BICA para el robot NAO

Este apéndice detalla la arquitectura basada en comportamientos (BICA), que ha sido la base para varios de los experimentos de esta tesis con el robot Nao. Esta arquitectura *software* para el jugador robótico de la RoboCup está en pleno desarrollo y ésta es la razón fundamental por la que los experimentos aún no cubren tantos cosas como los diseñados para el robot aiBo.

Debido al carácter comercial del robot, todos los esfuerzos para desarrollar y mejorar las aplicaciones tienen que concentrarse en el *software*, pues el *hardware* es fijo. Aldebaran Robotics, su fabricante, proporciona un API de programación para acceder a los sensores y actuadores del robot: naoQi.

La aplicación de fútbol robótico emplea algunas de las funcionalidades ofrecidas por esta capa de *software* denominada NaoQi[1]. NaoQi permite programar aplicaciones sobre ella escritas en C++ o Python.

NaoQi es un *framework* de objetos distribuidos, que permite que varios binarios sean ejecutados simultáneamente, cada uno de ellos conteniendo un módulo de *software* diferente y con capacidades de inter-comunicación entre los objetos. El acceso a los motores y sensores también está encapsulado en módulos *software*. Para conseguir el valor de un determinado sensor o modificar el valor de algún motor sólo será

[1] www.aldebaran-robotics.com

necesario conectarse al módulo responsable de estas tareas e invocar sus métodos apropiados.

Cada binario, también llamado *broker*, ejecuta de manera independiente y permanece atado a una dirección IP y a un puerto específico. Estos *brokers* pueden ejecutar tanto en el robot (utilizando un compilador cruzado) como en un ordenador convencional. De esta manera, es posible diseñar una aplicación dividida en varios *brokers*, algunos de ellos ejecutando en el robot y otros en un ordenador externo. Esta funcionalidad es especialmente útil para desacoplar del robot procesos que requieren mucha capacidad de cómputo.

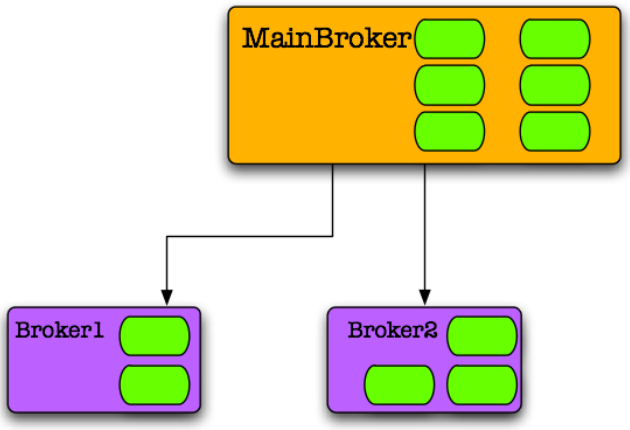


Figura B.1: Ejemplos de árbol de *brokers*

La funcionalidad de los *brokers* se implementa a través de uno o varios módulos. Cada *broker* se encarga de repartir y gestionar los mensajes de comunicación entre los módulos. Todos los *brokers* se estructuran de manera jerárquica como podemos ver en la figura B.1. En el nodo raíz de la jerarquía se encuentra el *MainBroker*, que contiene los módulos para acceder a los sensores y actuadores. La figura B.2 muestra estos módulos.

NaoCam: Este módulo proporciona la mayor fuente de información para nuestra aplicación. A través del driver *video4linux* las imágenes procedentes de las cámaras del robot se ponen a disposición de cualquier módulo que cree un *proxy* a NaoCam. Este *proxy* puede ser obtenido de manera local o remota. En

Figura B.2: Módulos incluidos en el *MainBroker*

caso de creación local del *proxy* se obtiene directamente un puntero a los datos. Si el *proxy* es remoto, la imagen es encapsulada en un mensaje SOAP y enviada por la red.

ALMotion: Este es el nombre del módulo que proporciona NaoQi para mover las articulaciones del robot. ALMotion ofrece varios niveles de abstracción para gestionar y modificar los motores. El interfaz ofrecido por este módulo permite modificar los actuadores tanto a nivel individual como a nivel de cadena de actuadores, e incluso a nivel de cuerpo entero (enviando un comando de movimiento a cada motor del robot). El acceso de más bajo nivel para realizar un control en posición o velocidad de un actuador está permitido, pero también se incluyen comandos de más alto nivel, para permitir comandos de locomoción más elaborados. En esta arquitectura se emplean ambas aproximaciones y, por ejemplo, para el control del cuello se realiza un acceso de bajo nivel a los motores, mientras que para desplazar el robot se utilizan los comandos de alto nivel *walk*, *turn* o *walk sideways* para caminar, rotar o realizar pasos laterales.

ALMemory: NaoQi facilita este módulo para intercambiar información entre el resto de módulos. *ALMemory* garantiza la exclusión mutua y evita la aparición de condiciones de carrera al realizar operaciones de lectura y escritura entre varios hilos de ejecución. Además de permitir la escritura, este módulo permite la subscripción de *callbacks* para que sean llamados de manera automática cuando determinadas variables de *ALMemory* se modifiquen.

Adicionalmente este módulo, que puede verse como una pizarra compartida, ofrece una serie de variables predeterminadas para reflejar el valor de todos los sensores del robot.

Cuando se diseña una aplicación compuesta por varios módulos, ésta puede compilarse como una biblioteca dinámica o como un ejecutable (broker). Si se opta por la compilación como biblioteca dinámica, sus módulos pueden ser cargados por el broker principal (*Main Broker*). Esta alternativa acelera notablemente la velocidad de comunicación entre módulos. Sin embargo, en caso de fallo en alguno de los módulos, el broker principal también terminará su ejecución abruptamente y provocando que el robot se caiga. La compilación como ejecutable es más segura desde este punto de vista, pues en caso de fallo no se detendrán el resto de módulos.

El diseño *hardware* del robot Nao impone algunas restricciones. Una de ellas atañe a las dos cámaras que incorpora el robot. Estas cámaras no forman un par estéreo, están colocadas una encima de la otra y sus campos visuales no están solapados. La imagen B.3 muestra la disposición de las mismas en la cabeza del robot.

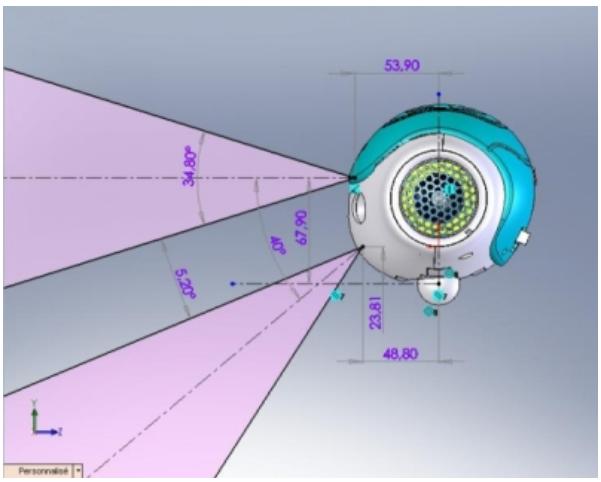


Figura B.3: Dimensiones de la cabeza del robot Nao y disposición de sus cámaras

Debido a esta restricción no podremos realizar estimaciones 3D de objetos basadas en triangulación estéreo empleando una imagen desde cada cámara. Para lograr estas estimaciones se deberán asumir algunas restricciones en cuanto a tamaños conocidos de objetos, etc. Además, las dos cámaras no pueden usarse simultáneamente.

Una restricción *hardware* obliga a seleccionar una de ellas para obtener imágenes. El tiempo de cambio de cámara es considerable, dedicando 70 ms. a esta tarea. Esta restricción ha condicionado el diseño de esta arquitectura.

La aplicación programada sobre NaoQi puede ejecutarse tanto en el robot real como en un simulador. En nuestro caso utilizamos Webots (Microsoft Robotics Studio también está soportado) para depurar el *software* antes de ejecutarlo en el robot real. Este proceso acelera el desarrollo del código y protege el *hardware* del robot, que ya de por sí es frágil.

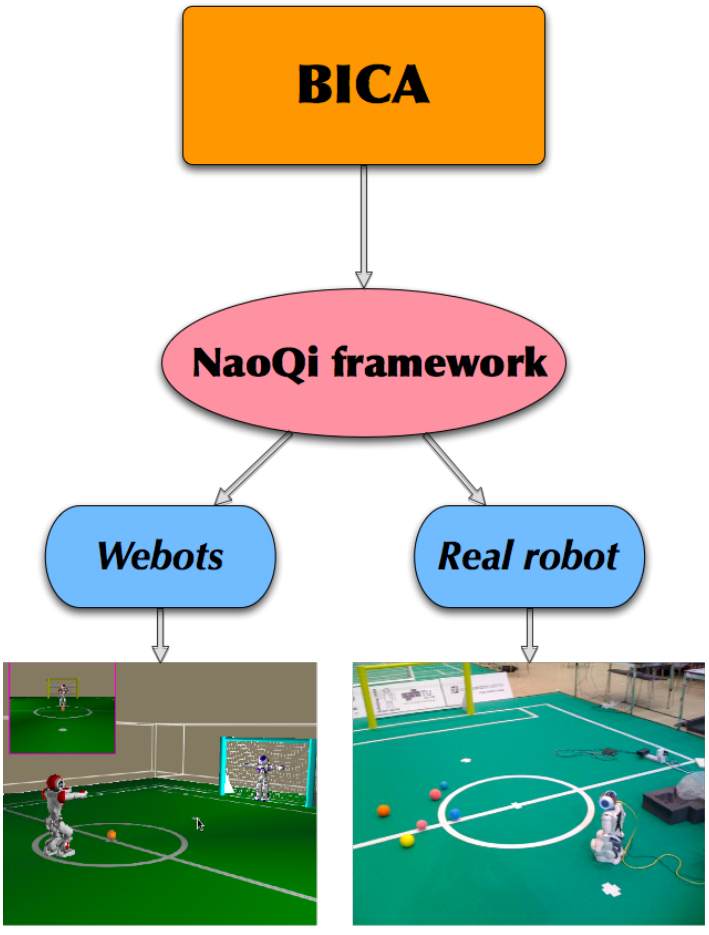


Figura B.4: Simulador Webots y robot real

B.1. Arquitectura basada en comportamientos

El *framework* presentado anteriormente ofrece una funcionalidad para el desarrollo de una arquitectura *software* que permite a un robot realizar diferentes tareas. Como hemos descrito, es posible descomponer la aplicación en módulos que se comunican entre sí. NaoQi oculta muchas de las dificultades de bajo nivel que aparecen asociadas al desarrollo de software en este tipo de robots: Generación de movimiento, acceso a los sensores y actuadores.

Aunque es posible el desarrollo de comportamientos básicos utilizando únicamente NaoQi, no es suficiente para nuestras necesidades. Necesitamos una arquitectura que permite activar y desactivar componentes y más cercana a la organización cognitiva de un sistema basado en comportamientos.

En esta sección se describen los conceptos más interesantes de la arquitectura desarrollada. Abordaremos aspectos como la interacción con NaoQi, cuáles de sus funciones se usan y cuáles no, cuáles son los elementos que conforman nuestra arquitectura, cómo están organizados y detallaremos aspectos sobre temporización. Una especificación más extensa de esta arquitectura puede consultarse en [MAC09].

El principal elemento de la arquitectura *BICA* es el componente. Es la unidad básica que implementa una funcionalidad concreta. En un instante determinado, cada componente puede estar activo o inactivo. Esta propiedad se logra con el interfaz *start / stop*, como puede observarse en la figura B.5. Cuando el componente está activo, éste ejecuta iterativamente su método *step()* persiguiendo un objetivo concreto. En caso de encontrarse inactivo, está detenido y no consume tiempo de proceso. Todo componente permite una modulación a su actuación y proporciona información como salida de la tarea que está realizando.

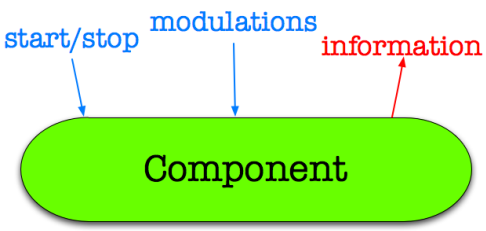


Figura B.5: Entradas y salidas de la entidad componente

Cuando un componente se activa, éste puede activar a su vez a otros componentes para lograr su objetivo. Este es un concepto clave en esta arquitectura y permite la descomposición funcional en diversos componentes que operan juntos. Una aplicación está formada por un conjunto de componentes, algunos de ellos activos y otros inactivos. El conjunto de componentes activos y sus relaciones de activación son denominadas árbol de activación. Dos componentes diferentes pueden activar el mismo componente hijo. Esta propiedad permite que dos componentes obtengan la misma información de un tercero. Cualquier de los componentes padre puede modular al hijo y los cambios afectarán a ambos componentes padre. La figura B.1 muestra dos ejemplos de árboles de activación que ilustran estos conceptos.

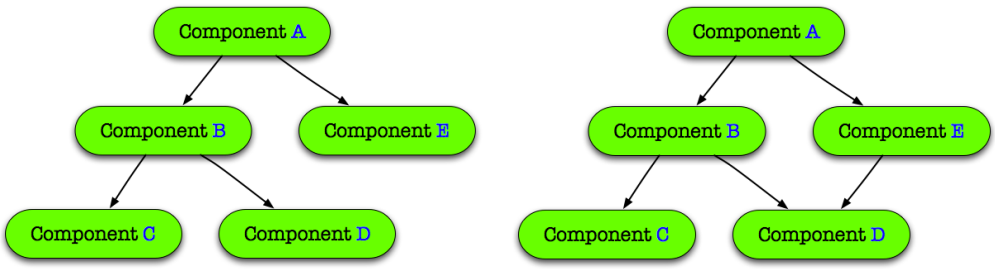


Figura B.6: Árbol de activación con varios niveles de componentes (izquierda) y árbol con un componente utilizado por dos componentes *padre* (derecha)

El árbol de activación no es fijo durante la ejecución de una aplicación. Éste varía dinámicamente en función de varios factores: objetivo principal, posición de los elementos del entorno, interacción con robots o humanos, cambios en el entorno, errores o caídas, etc. El robot debe adaptarse a los cambios en estos parámetros modulando los componentes de bajo nivel o activando y desactivando componentes, pudiendo alterar el árbol de activación en cada momento.

Utilizando esta aproximación se ha diseñado e implementado esta arquitectura sobre un módulo NaoQi. Los componentes se han materializado como clases C++ basadas en el patrón de diseño *Singleton*. La comunicación entre cada módulo se realiza mediante llamadas a sus métodos, evitando el intercambio de mensajes sobre *SOAP* y la sobrecarga asociada.

Cuando el módulo NaoQi se crea, arranca una hebra que iterativamente invoca el método *step()* del componente raíz del árbol de activación. El contenido del método *step()* de todo componente tiene siempre la misma estructura:

1. Llamadas a los métodos *step()* de los componentes perceptivos de los que depende del nivel inferior y pertenecientes a la rama del componente actual. Es necesario realizar estas llamadas porque posteriormente se necesitará la información proporcionada por estos componentes.

2. Procesado de las entradas disponibles para la consecución del objetivo del componente. Esta etapa podría incluir llamadas a otros componentes de niveles inferiores para obtención de información o modulación de su comportamiento.

3. Llamadas a los métodos *step()* de los componentes de actuación del nivel inferior y pertenecientes a la rama del componente actual. De esta manera se materializarán las modulaciones realizadas.

Cada componente ejecuta iterativamente a una frecuencia específica y configurable. No tiene sentido que todos los componentes ejecuten a la misma frecuencia, unos necesitarán refrescar su información a la máxima frecuencia disponible y otros, a su vez, bastará con que ejecuten, por ejemplo, una o dos veces por segundo. Cada vez que un método *step()* es invocado se comprueba el tiempo transcurrido desde la última ejecución completa. Si este intervalo es igual o superior a la frecuencia configurada, el componente ejecutará los pasos 1, 2 y 3 de la estructura de componente descrita anteriormente. En caso negativo, únicamente se ejecutarán los pasos 1 y 3. Habitualmente, los componentes de alto nivel son configurados con frecuencias menores que los componentes de niveles inferiores, como podemos observar en la figura B.7.

B.2. Diseño del jugador de fútbol robótico

Los conceptos presentados en la sección anterior resumen los conceptos clave de esta arquitectura. Se ha presentado el concepto de componente, cómo pueden

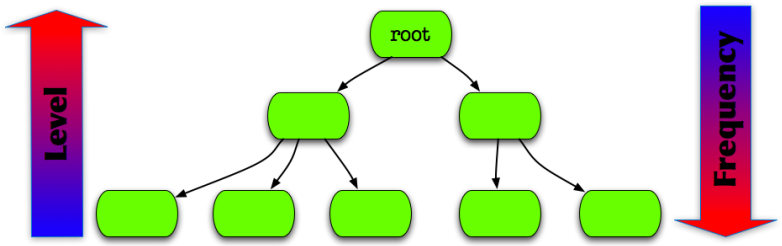


Figura B.7: Árbol de activación con el componente *root* en el nivel más alto. Típicamente a mayor nivel de abstracción de un módulo, menor es su frecuencia

activarse o desactivarse conformando un árbol de activación y cómo ejecutan. Esta arquitectura está enfocada al desarrollo de aplicaciones robóticas utilizando una aproximación basada en comportamientos. A continuación presentaremos cómo esta arquitectura logra dar respuesta al problema que se quiere abordar: El desarrollo de un jugador para jugar al fútbol en la competición RoboCup.

La implementación del jugador de fútbol robótico queda definida por un conjunto de árboles de activación y una política de modulación entre componentes. A continuación resumiremos cada uno de los componentes que se han definido en el diseño del jugador.

B.2.1. Percepción del jugador de fútbol

Esta tarea es llevada a cabo por el componente *Perception*. Su objetivo principal es obtener una imagen, procesarla y ofrecer cierta información a cualquier componente de nivel superior interesado. Además, es capaz de calcular la posición 3D de determinados objetos en el entorno. Por último, se ha incluido un novedoso aporte para la detección de porterías, obteniendo simultáneamente la posición del robot en el campo de juego.

Los objetos relevantes de este entorno son la pelota, el suelo de moqueta verde, las dos porterías, las líneas del campo y los otros robots. La detección de los elementos se realiza atendiendo a su color, forma, dimensiones y posición con respecto al borde del terreno de juego enmoquetado.

Este componente puede ser modulado por otros componentes para ajustar algunos de sus parámetros:

- Selección de cámara. Únicamente una de las cámaras puede estar activa en cada momento.

- Selección del objeto de interés. Con el fin de reducir la carga de CPU durante las fases de procesado de la imagen, únicamente se buscarán las características propias de los objetos de interés especificados, no de todos.

Pelota y porterías en la imagen

La detección de estos estímulos se lleva a cabo en el método *step()* de este componente. El primer paso a realizar es filtrar la imagen por el color característico de los objetos de interés. Para acelerar este proceso se ha utilizado una tabla de 64 entradas por cada canal. A continuación se efectúa un segmentado de la imagen para agrupar aquellos pixels conectados o agrupados. Por cada uno de estos *blobs* se verifican una serie de condiciones propias de cada objeto. Estas condiciones tienen que ver con restricciones de tamaño, densidad de color, posición del centro de masas respecto al horizonte (frontera entre moqueta y el resto de objetos), etc.

Pelota en coordenadas locales respecto del robot

La información de la pelota en 2D del plano imagen no siempre es suficiente para realizar una determinada tarea. Por ejemplo, si el robot quiere alinearse con la pelota necesitamos conocer con precisión cuál es su posición en el campo respecto del sistema de referencia solidario al robot.

La obtención de la posición 3D de un elemento no es tarea sencilla, y es aún más difícil en el caso de un robot humanoide que se balancea al caminar y captura imágenes con una única cámara. Para calcular la posición 3D de la pelota comenzamos obteniendo la posición 2D del centro del objeto en el plano imagen. Empleando el modelo *pinhole* podemos situar este punto a lo largo de la recta que une el centro de la cámara y la posición de la pelota respecto al sistema de referencia de la cámara. Una vez obtenido este punto, es necesario realizar un cambio de sistema de referencia para expresarlo respecto del origen situado entre sus dos pies. Se ha utilizado NaoQi para realizar las transformaciones de un sistema de referencia a otro.

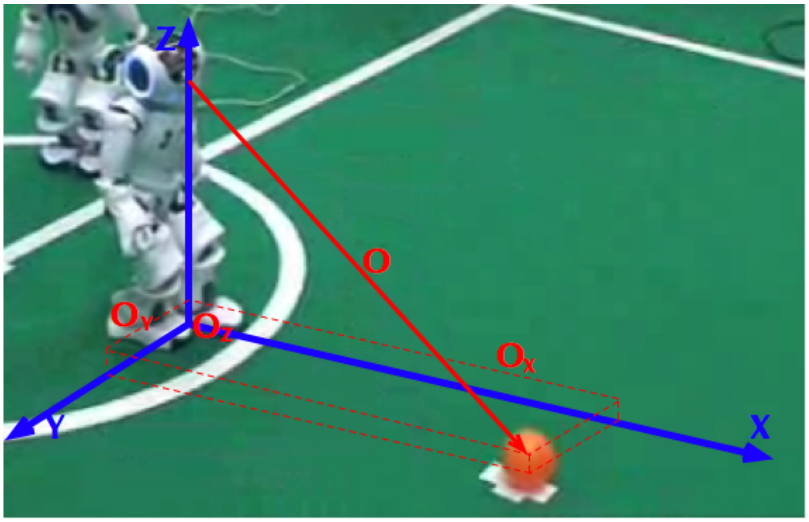


Figura B.8: Posición 3D de la pelota y su sistema de referencia

Porterías en coordenadas locales respecto del robot

Una vez que la portería ha sido correctamente identificada en la imagen, es posible extraer información espacial de ella utilizando geometría 3D. Dados *pix1*, *pix2*, *pix3* y *pix4*, que corresponden a las esquinas de la portería en la imagen B.9, es posible extraer la posición y orientación de la misma respecto del sistema de referencia solidario a la cámara. Es decir, podríamos transformar los pixels de las esquinas a los puntos 3D *P1*, *P2*, *P3* y *P4*. Puesto que las posiciones reales de la portería son conocidas *(AP1, AP2, AP3, AP4)*, esta información puede invertirse para obtener la posición de la cámara respecto de la portería y, por tanto, la posición absoluta de la cámara (y del robot) en el campo. Con el fin de realizar estas transformaciones geométricas 3D la cámara del robot debe estar calibrada. En [CPPG09] se puede consultar una descripción extendida de este algoritmo.

B.2.2. Movimientos básicos

La gestión de la actuación de un robot con patas no es trivial. En robots bípedos es aún más complicado. El movimiento se lleva a cabo asegurando que la proyección

Figura B.9: Extracción de características visuales de la portería

del centro de masas sobre el suelo (*ZMP*) se mantenga dentro del polígono de soporte. Esta tarea involucra que exista una buena coordinación entre todos los actuadores.

NaoQi ofrece algunas funciones de alto nivel para resolver el problema de la locomoción en el robot Nao. Existen funciones para caminar (recto o lateral) para rotar, caminar realizando un movimiento en arco e incluso para mover articulaciones de manera individual. Nuestra arquitectura emplea esta funcionalidad para controlar los actuadores del robot y hacer que se desplace por el campo de juego. Para mantener un orden en los comandos de actuación, hemos repartido su funcionalidad en varios componentes, cada uno de ellos con su interfaz. Así, hemos diseñado los componentes *Body*, *Head* y *FixMove*.

Componente Body

Este componente gestiona los desplazamientos del robot. Su modulación consta de dos parámetros: Velocidad lineal (v) y velocidad angular (w). Cada parámetro acepta un valor normalizado entre $[-1, 1]$. Desafortunadamente, en este momento ambos parámetros no pueden ser combinados y sólo uno de ellos puede estar activo en cada momento.

El componente Body no envía directamente los comandos de movimiento. Su función es activar y modular dos componentes de nivel inferior: *GoStraight* y *Turn*, como se muestra en la figura B.10.

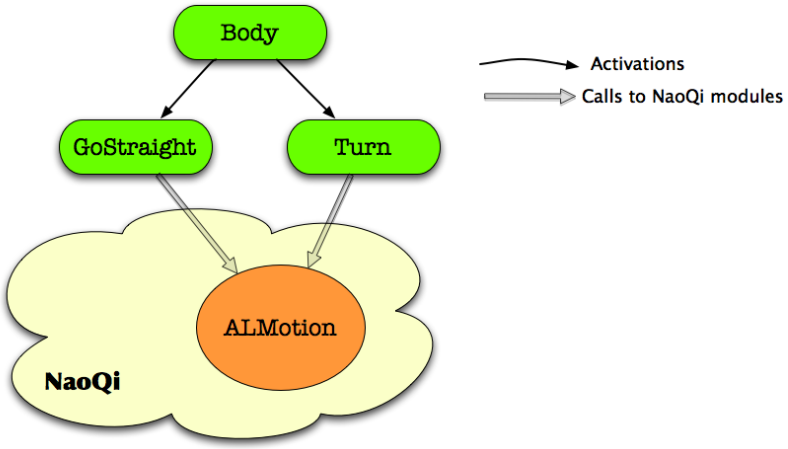


Figura B.10: Componente *Body* y sus componentes de nivel inferior que utilizan NaoQi para desplazar el robot

Componente Head

El componente Body se encarga de todo el movimiento del robot exceptuando su cabeza. La cabeza del robot está muy ligada con su percepción y proceso de atención y puede ser controlada de manera independiente a través de las dos articulaciones del cuello del robot Nao. La cabeza es gobernada a través del componente *Head*.

Este componente, una vez activo, puede ser modulado tanto en posición como en velocidad para controlar el movimiento del cuello. Mientras que el control en posición es relativamente sencillo (únicamente tiene que enviar la posición al módulo ALMotion que materializará la orden), el control en velocidad necesita de un control más sofisticado. Se ha desarrollado un controlador *PID* para ajustar debidamente el movimiento en velocidad. La modulación se realiza con dos parámetros normalizados entre $[-1, 1]$ para cada articulación del cuello. Estos dos parámetros son la entrada del controlador.

Comportamiento para realizar movimientos predefinidos

El último elemento involucrado en la actuación es el componente *FixMove*. En ocasiones se requiere que el robot lleve a cabo un movimiento relativamente complicado pero de trayectorias fijas y conocidas de antemano. Estos movimientos están

compuestos por secuencias completas de movimientos por cada articulación durante determinado tiempo. Por ejemplo, si queremos realizar un golpeo de la pelota necesitamos realizar un movimiento que involucra a todas las articulaciones del cuerpo y necesita un determinado tiempo para completarse. Estos movimientos están codificados en ficheros (uno archivo por cada movimiento completo) con un formato concreto. Veamos un ejemplo:

```
Movement_name
name_joint_1 name _joint_2 name _joint_3 ... name _joint_n
angle_1_joint_1 angle_2_joint_1 angle_3_joint_1 ... angle_m1_joint_1
angle_1_joint_2 angle_2_joint_2 angle_3_joint_2 ... angle_m2_joint_2
angle_1_joint_3 angle_2_joint_3 angle_3_joint_3 ... angle_m3_joint_3
...
angle_1_joint_n angle_2_joint_n angle_3_joint_n ... angle_mn_joint_n
time_1_joint_1 time_2_joint_1 time_3_joint_1 ... time_m1_joint_1
time_1_joint_2 time_2_joint_2 time_3_joint_2 ... time_m2_joint_2
time_1_joint_3 time_2_joint_3 time_3_joint_3 ... time_m3_joint_3
...
time_1_joint_n time_2_joint_n time_3_joint_n ... time_mn_joint_n
```

Además del movimiento definido en el propio fichero, es posible modular el comportamiento con dos parámetros que indican el desplazamiento longitudinal y lateral que realizará el robot antes de comenzar el movimiento. Esto es muy útil para alinear el robot con la pelota antes de realizar un golpeo, por ejemplo.

B.2.3. Comportamiento para seguir la pelota con la mirada

El componente *FaceBall* busca centrar la pelota en la imagen tomada por la cámara. Para alcanzar este objetivo se apoya en los componentes *Head* y *Perception*, como muestra la figura B.11.

En su método *step()* toma la salida del componente *Perception* y utiliza este valor directamente como entrada para el componente *Head*. A continuación mostramos el fragmento de código del método *step()*, donde se puede apreciar cómo de una manera sencilla es posible realizar este componente.

```
void
```

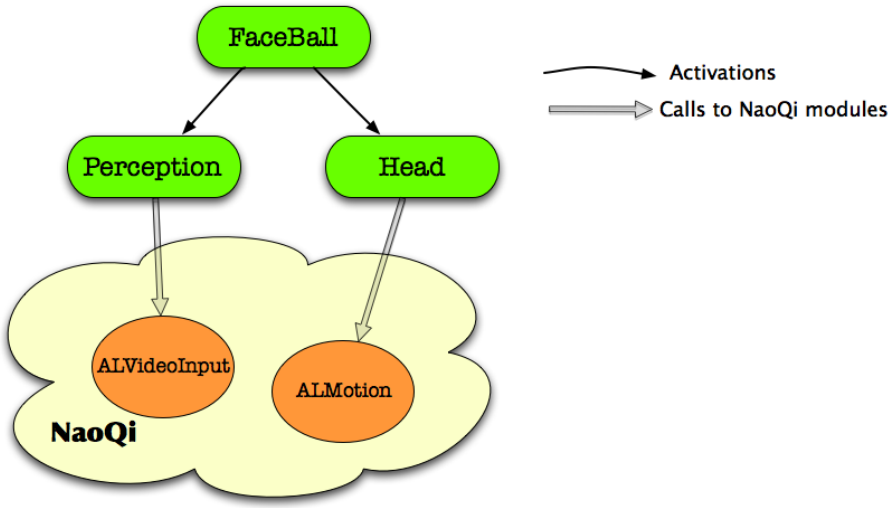


Figura B.11: Árbol de activación del componente *FaceBall*

```
FaceBall::step(void)
{
        perception->step();

        if (isTime2Run())
        {
                head->setPan( perception->getBallX());
                head->setTilt( perception->getBallY() );
        }

        head->step();
}
```

B.2.4. Comportamiento para seguir la pelota con el cuerpo

La principal función del componente *FollowBall* es que el robot avance hasta la pelota cuando ésta es percibida. Este componente activa los componentes de menor nivel *FaceBall* y *Body*, como se puede apreciar en la figura B.12. La modulación para el componente *Body* se obtiene de la posición de la cabeza mientras se realiza el seguimiento de la pelota. Si la cabeza está mirando a la izquierda, la modulación enviada al componente *Body* implicará rotar para alinearse con la pelota. Una vez

alineado, el ángulo de cabeceo del cuello nos indicará aproximadamente a qué distancia se encuentra la pelota del robot y, por tanto, será utilizado para modular el componente *Body* con el fin de acercarse o detenerse.

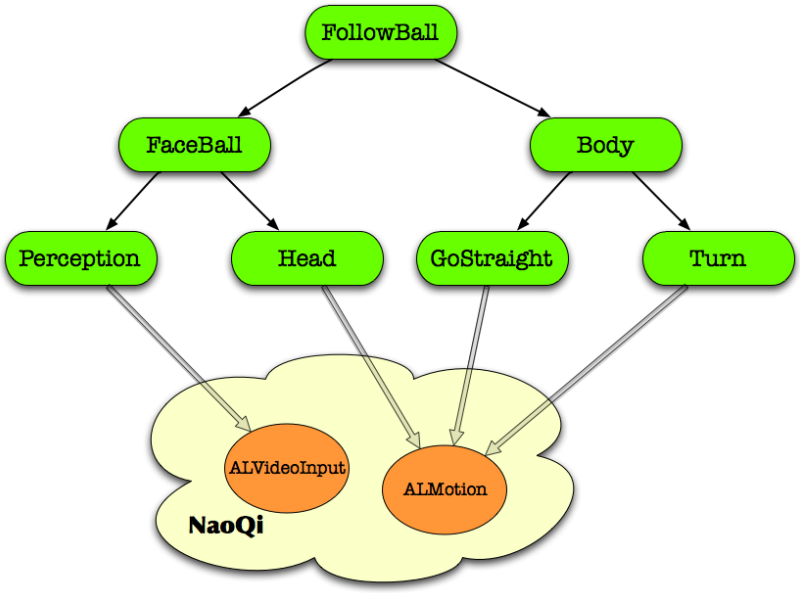


Figura B.12: Árbol de activación del componente *FollowBall*

B.2.5. Comportamiento de búsqueda de la pelota

El componente *SearchBall* busca la pelota cuando el robot no la percibe. Este elemento introduce el concepto de máquina de estado a nivel de componente. Cuando *SearchBall* está activo, puede encontrarse en dos estados posibles: *HeadSearch* o *BodySearch*. Inicialmente el estado de arranque es *HeadSearch*, cuya misión principal es realizar movimientos de cuello para buscar la pelota delante del robot. Cuando ha recorrido todo el espacio y no ha encontrado la pelota, se transita al estado *BodySearch*, que provoca un movimiento extra de rotación en el robot para buscar la pelota a los lados y detrás del robot Nao. Paralelamente, este componente modula a *Perception* para ir alternando de cámara periódicamente en busca de la pelota.

Dependiendo del estado en el que se encuentre *SearchBall*, el árbol de activación será diferente (ver figura B.13).

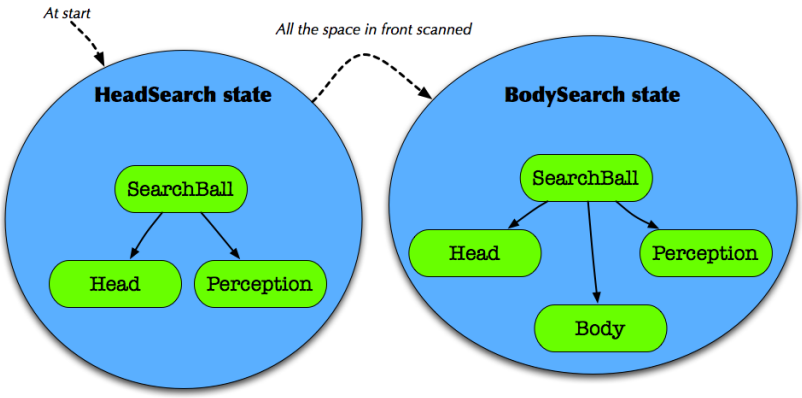


Figura B.13: Árbol de activación en cada posible estado del componente *SearchBall*

Este componente no utiliza *Perception* para obtener información sobre la presencia o no de la pelota. Únicamente gestiona el movimiento del robot y la selección de cámara. Es necesario que otro componente de nivel superior active *SearchBall* y *Perception* y detenga *SearchBall* cuando detecta que se ha percibido la pelota. Este componente se denomina *Player*, y es el de más alto nivel de la jerarquía de componentes.

B.2.6. Comportamiento de búsqueda de porterías

Este comportamiento implementado por el componente *SearchNet* se utiliza para buscar la portería hacia la que tiene que dirigir el golpeo de pelota el jugador. Activará los componentes *Head* y *Perception* y su estado se dividirá en *buscando o scanning* y *recuperando o recovering*.

Cuando arranca el proceso de búsqueda se guarda la posición del cuello, que típicamente estará apuntando a la pelota. El robot modulará ahora el componente *Perception* para que el objeto de interés sea la portería y no la pelota. Durante el tiempo que se permanece en este estado se modula el componente *Head* para que barra el horizonte visual en busca de porterías. Si el movimiento de barrido visual se ha completado o se ha detectado la portería, se transita al estado de recuperación.

En este estado se vuelve a cambiar el objeto de interés por el de la pelota y se recupera la posición que tenía el cuello cuando arrancó el proceso de búsqueda.

B.2.7. Comportamiento de jugador de campo

Player es el componente raíz de la jerarquía de comportamientos. Su funcionalidad está dividida en cinco estados: *Búsqueda de pelota*, *aproximación*, *búsqueda de portería*, *caída* y *disparo a portería*. Estos cinco componen el comportamiento completo que provoca que el robot juegue al fútbol.

En el estado *buscar pelota* se activan los componentes *SearchBall* y *Perception*, como se observa en la figura B.14. Por razones de claridad, es esta figura no se muestran los árboles de activación, sino que únicamente se muestran los componentes que activa cada estado de *Player*.

Cuando *Player* percibe que la pelota ha sido detectada, se transita al estado *aproximación*. Se desactiva el componente *SearchBall* y se supone que la pelota está en el campo visual del robot. En este estado se activa el componente *FollowBall* para que el robot se acerque a la pelota.

Una vez que la pelota está relativamente cerca del robot, éste se encuentra en condiciones de golpear la pelota. Como el golpeo debe ser dirigido a la portería apropiada, *Player* realiza una nueva transición al estado *búsqueda de portería* activando el componente *SearchNet*.

Cuando la portería ha sido identificada el robot debe golpear la pelota en la dirección correcta. El componente *Player* toma esta decisión en el estado *disparo a portería*. Justo antes de activar el componente *FixMove*, pregunta a *Perception* cuál es la posición 3D exacta de la pelota respecto a su sistema de referencia. Con esta información, *Player* puede calcular la distancia necesaria para la fase de alineación del golpeo, así como el movimiento más apropiado en función de la posición de la portería. Una vez activado el componente *FixMove* el robot realiza el disparo de la pelota. En este estado, el componente también activa *FaceBall* para realizar un seguimiento de la pelota durante la ejecución del disparo.

El estado restante se denomina *caída*. *Player* transita a este estado cuando el robot se cae al suelo por cualquier circunstancia inesperada. En ese caso se activa el componente *FixMove* modulado con el movimiento más apropiado para recuperarse de la caída y levantarse de nuevo.

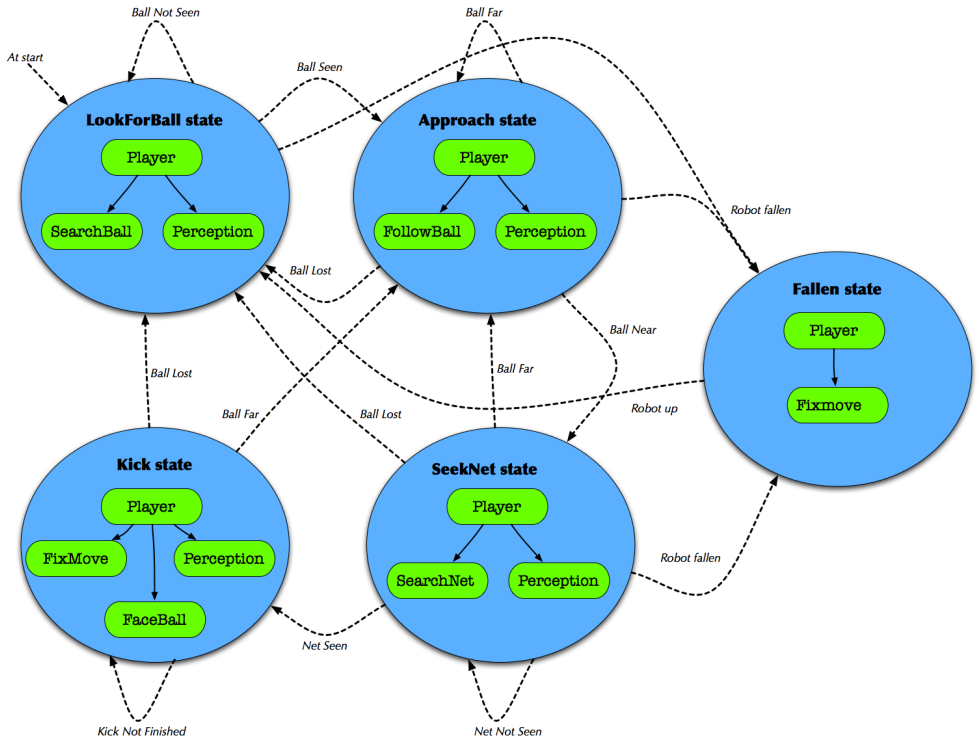


Figura B.14: Máquina de estados del componente *Player* con su correspondiente árbol de activación

B.3. *Manager*

En la sección anterior se ha descrito el *software* que ejecuta el robot Nao. Este *software* es necesario para que el robot pueda realizar cualquier tarea. De hecho es el único programa que se permite ejecutar en una competición oficial de la liga de plataforma estándar de la RoboCup. Sin embargo, siempre es necesario realizar una fase de calibración y depuración previa a una competición.

Se han desarrollado un conjunto de herramientas para ayudar en el desarrollo del robot jugador de fútbol. La aplicación *Manager* contiene todas ellas integradas en un único programa que ejecuta en un PC y se conecta al robot mediante un cable *ethernet* o mediante comunicación inalámbrica.

Con el fin de comunicar el programa del robot y la aplicación *Manager* se ha usado el protocolo SOAP que proporciona NaoQi. Este hecho simplifica enormemente las comunicaciones, puesto que no es necesario diseñar un sistema de comunicación

basado en *sockets* o alternativas similares. Únicamente es necesario crear un *proxy* al módulo que contiene todo nuestro *software* y realizar las llamadas a métodos oportunas para enviar y recibir información.

A continuación se describirán las principales herramientas integradas en *Manager* para depurar cualquier componente, calibrar el sistema perceptivo y los movimientos del robot.

B.3.1. Herramienta de depuración de componentes

A través de la herramienta *Manager* es posible depurar cualquier componente de manera individual. Es posible activarlo, modularlo y cambiar su frecuencia de ejecución. Además también disponemos de opciones para visualizar su consumo de CPU.

En la imagen B.15 podemos observar la apariencia gráfica de esta herramienta mientras se depura el componente *Turn*.

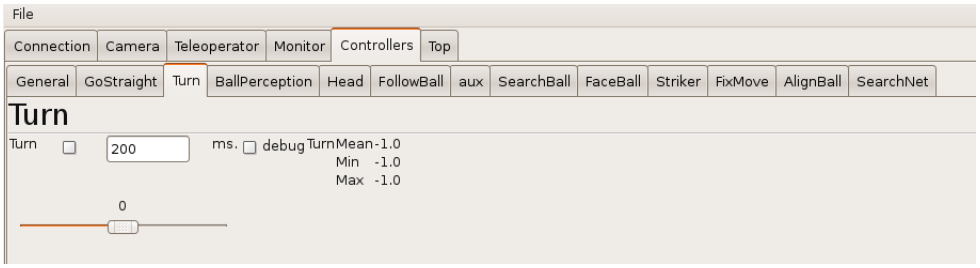


Figura B.15: Herramienta de depuración de componentes trabajando sobre el componente *Turn*

B.3.2. Herramienta de calibración del sistema de percepción visual

Las condiciones del entorno no son iguales en todos los lugares donde el robot debe trabajar. Incluso en el mismo lugar, las condiciones de luz pueden cambiar a lo largo del día. Estos cambios de iluminación obligan a tener una herramienta desde la cuál calibrar los diferentes parámetros de las dos cámaras del robot Nao, así como los parámetros de los filtros de color empleados en la etapa perceptiva.

La figura B.16 muestra la apariencia de la herramienta de calibración para modificar el brillo, contraste, saturación, etc. de cada cámara.

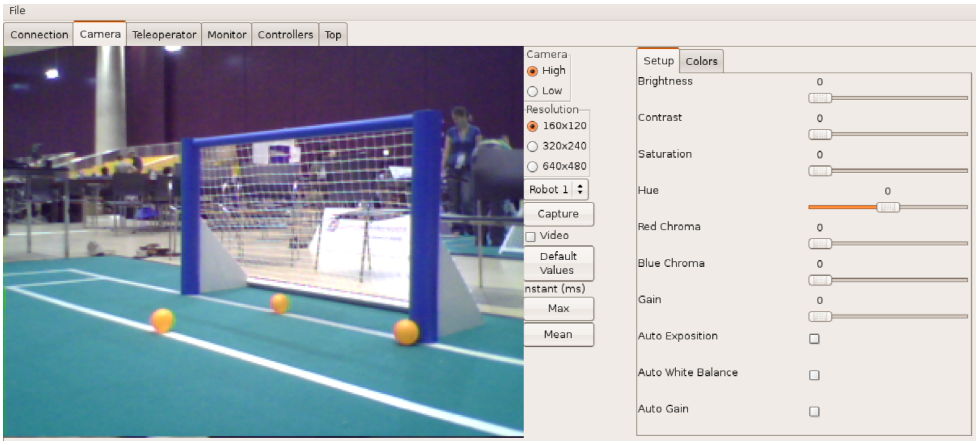


Figura B.16: Ajuste de los parámetros de una de las cámaras del robot Nao

B.3.3. Herramienta de generación de movimientos predefinidos

Como hemos descrito en la sección anterior, en ocasiones el robot debe ejecutar un patrón de movimiento predefinido. El golpeo de la pelota o la maniobra para levantarse son algunos ejemplos que involucran secuencias de movimientos de las articulaciones. Es una labor muy tediosa crear estos movimientos sin una herramienta que ayude en este trabajo.

Para generar un patrón nuevo necesitamos especificar las articulaciones involucradas, los ángulos que cada articulación debe ir tomando y los tiempos que cada paso debe ocupar por articulación. El objetivo de esta herramienta es generar un fichero con el formato adecuado de manera visual, conteniendo todos estos elementos.

La metodología de creación de patrones exige que se vayan creando movimientos paso a paso. En cada paso, se define la duración del mismo y podemos variar los ángulos de cada articulación utilizando *sliders* o liberando la articulación y posicionándola manualmente en el punto deseado. Además es posible desplazarse gráficamente por cada uno de los pasos, ejecutar ese paso en el robot, enviar la secuencia

de movimientos completa, añadir nuevos pasos intermedios o eliminar algunos ya existentes.

Con esta herramienta se han creado por ejemplo varios movimientos de golpeo de la pelota según la dirección de destino. Una característica adicional incluida es la posibilidad de generar secuencias simétricas a una ya existente. Por ejemplo, si ya disponemos de un patrón que golpea la pelota con la pierna derecha, con una única pulsación en uno de los botones de la herramienta podemos generar la secuencia completa de golpeo con la pierna izquierda.

BIBLIOGRAFÍA

[AB97] Ronald C. Arkin and Tucker Balch. AuRA: Principles and Practice in Review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:175–189, 1997.

[AMCG06] Carlos E. Agüero, Vicente Matellán, José M. Cañas, and Víctor Gómez. Switch! dynamic roles exchange among cooperative robots. In *Proceedings of the International Workshop on Multi-Agent Robotics Systems*, 2006.

[AMdlHC03] C. Agüero, V. Matellán, P. de las Heras, and J. M. Cañas. PERA: Routing protocol for mobile robotics. In *International Conference on Advanced Robotics*, 2003.

[AMGC06] Carlos E. Agüero, Vicente Matellán, Víctor Gómez, and J.M. Cañas. Switch! Dynamic roles exchange among cooperative robots. In *Proceedings of the 2nd International Workshop on Multi-Agent Robotic Systems - MARS 2006*, pages 99–105. INSTICC Press, 2006.

[AMI89] H. Asama, A. Matsumoto, and Y. Ishida. Design of an autonomous and distributed robot system: ACTRESS. In *IEEE/RSJ IROS*, pages 283–290, 1989.

[AMMM06] Carlos E. Agüero, Francisco Martín, Humberto Martínez, and Vicente Matellán. Communications and basic coordination of robots in TeamChaos. In *Actas VII Workshop de Agentes Físicos*, pages 3–9, 2006.

[Ark92] R. C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[Ark98] Ronald C. Arkin. *Behavior-Based Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, May 1998.

[Arm88] M. Armada. Climbing and Walking from research applications. In *International Conference on Climbing and Walking Robots*, pages 39–48, 1988.

[AS08a] A. Saffiotti A. Sanfeliu, N. Hagita, editor. *Special Issue on Network Robot Systems of Robotics and Autonomous Systems*, volume 56. Elsevier, 2008.

[AS08b] Mazda Ahmadi and Peter Stone. Instance-based action models for fast action planning. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup-2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Artificial Intelligence*, pages 1–16. Springer Verlag, Berlin, 2008.

[Bal97] Tucker Balch. Social Entropy: A New Metric for Learning Multi-robot Teams. In *In Proc. 10th International FLAIRS Conference (FLAIRS-97*, pages 272–277, 1997.

[BBG+08] D. Becker, J. Brose, D. Göhring, M. Jüngel, M. Risler, and T. Röfer. GermanTeam 2008. In *RoboCup 2008: Robot Soccer World Cup XII Preproceedings*. RoboCup Federation, 2008.

[Ben88] G. Beni. The concept of cellular robotic system. In *IEEE International Symposium on Intelligent Control*, pages 57–62, 1988.

[BG88] Alan H. Bond and Les Gasser. An Analysis of Problems and Research in DAI. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, 1988.

[BMB90] R. M. C. Bodduluri, J. M. McCarthy, and J. E. Bobrow. Planning movement for two puma manipulators holding the same object. In *The First International Symposium on Experimental Robotics I*, pages 579–593, London, UK, 1990. Springer-Verlag.

[BMF+00] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative Multi-Robot Exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[BMR01] A. Bonarini, M. Matteucci, and M. Restelli. Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem, 2001.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[BP02] Tucker Balch and Lynne E. Parker, editors. *Robot Teams: From Diversity to Polymorphism*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[CA07] Sonia Chernova and Ronald C. Arkin. From deliberative to routine behaviors: A cognitively inspired action-selection mechanism for routine behavior capture. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 15(2):199–216, 2007.

[CCL+90] P. Caloud, W. Choi, J. C. Latombe, C. Le Pape, and M. Yin. Indoor automation with many mobile robots. In *IEEE/RSJ IROS*, pages 67–72, 90.

[CFK97] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots*, 4(1):7–23, March 1997.

[CHI+09] Nak Young Chong, Norihiro Hagita, Volkan Isler, Klaus Schilling, and Dezhen Song, editors. *Special Issue on Networked robots: Serving the Society of Journal of Intelligent Service Robotics*. Springer, 2009.

[CKC04] Luiz Chaimowicz, Vijay Kumar, and M. Campos. A Paradigm for Dynamic Coordination of Multiple Robots. *Autonomous Robots*, 17(1), 2004.

[CL09] Cristobal Carnero Liñán. http://code.google.com/p/cvblob/, cvblob - Blob library for OpenCV, 2009.

[CLS04] J.P. Canovas, K. LeBlanc, and A. Saffiotti. Robust multi-robot object localization using fuzzy logic. In *Proc. of the Int. RoboCup Symposium, Lisbon, PT*, 2004.

[CM08] Miguel Cazorla and Vicente Matellán, editors. *Special Issue on Multi-Robot Systems of Journal of Physical Agents*, volume 2. Open Journal Systems, 2008.

[CMMTS08] Andreu Corominas Murtra, Josep M. Mirats Tur, and Alberto Sanfeliu. Action evaluation for mobile robot global localization in cooperative environments. *Robot. Auton. Syst.*, 56(10):807–818, 2008.

[CMQ06] Stephan K. Chalup, Craig L. Murch, and Michael J. Quinlan. Machine learning with aibo robots in the four-legged league of robocup. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, 2006.

[CPPG09] José M. Cañas, Domenec Puig, Eduardo Perdices, and Tomás González. Visual Goal Detection for the RoboCup Standard Platform League. In *Proceedings of WAF2009 X Physical Agents Workshop*, pages 121–127, Cáceres, Spain, 2009.

[DGB01] Markus Dietl, Jens-steffen Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS01*, pages 1706–1713, 2001.

[DJW96] Gregory Dudek, Michael R. M. Jenkin, and David Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3:375–397, 1996.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[DOK98] Jaydev P. Desai, Jim Ostrowski, and Vijay Kumar. Controlling formations of multiple mobile robots. In *in Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2864–2869, 1998.

[Dor08] Marco Dorigo, editor. *Swarm Intelligence*. Springer, 2008.

[DP80] D. Dubois and H. Prade. *Fuzzy sets and systems - Theory and applications*. Academic press, New York, 1980.

[DS03] M Bernardine Dias and Anthony (Tony) Stentz. A comparative study between centralized, market-based, and behavioral multirobot coordination approaches. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, volume 3, pages 2279 – 2284, October 2003.

[Dur95] Edmund H. Durfee. Blissful Ignorance: Knowing Just Enough to Coordinate Well. In *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*, pages 406–413, 1995.

[DW87] Hugh F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[Esp08] Joel M. Esposito. Distributed grasp synthesis for swarm manipulation with applications to autonomous tugboats. *IEEE Conference on Robotics and Automation*, pages 1489–1494, 2008.

[Eur09] European Robotics Research Network (EURON). http://www.euron.org/, 2009.

[Fer92] Alexis Drogoul Jacques Ferber. From tom thumb to the dockers: Some experiments with foraging robots. In *In Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 451–459. MIT Press, 1992.

[FF] Tim Finin and Richard Fritzson. The Knowledge Query and Manipulation Language (KQML),.

[FFHL05] Alexander Ferrein, Er Ferrein, Lutz Hermanns, and Gerhard Lakemeyer. Comparing Sensor Fusion Techniques for Ball Position Estimation. In *In Proc. of the RoboCup 2005 Symposium*, 2005.

[FN87] T. Fukuda and S. Nakagawa. A dynamically reconfigurable robotic system (concept of a system and optimal configurations). In *Internation Conference on Industrial Electronics, Control and Instrumentation*, pages 588–595, 1987.

[Gal60] David Gale. *The Theory of Linerar Economic Models*. McGraw-Hill, New York, 1960.

[GBFK98] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An Experimental Comparison of Localization Methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[GM02] Brian P. Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*, 2002.

[GM04] Brian Gerkey and Maja Mataric. On role allocation in RoboCup. In *RoboCup 2003: Robot Soccer World Cup VII, 2004*, pages 43–53. Springer-Verlag, 2004.

[Gui08] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *Spectrum, IEEE*, 45(7):26–34, July 2008.

[Har58] H. O. Hartley. Maximum Likelihood Estimation from Incomplete Data. *Biometrics*, 14(2):174–194, June 1958.

[HC01] Jason Hill and David Culler. A wireless embedded sensor architecture for system-level optimization. Technical report, U.C. Berkeley, 2001.

[HKK+04] Kensuke Harada, Shuuji Kajita, Fumio Kanehiro, Kiyoshi Fujiwara, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. Real-time planning of humanoid robot's gait for force controlled manipulation. In *ICRA*, pages 616–622, 2004.

[IB98] Michael Isard and Andrew Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[iRtmad09] iRobot Robots that make a difference. www.irobot.com, 2009.

[JMCB08] Antonio Pineda José M. Cañas and Pablo Barrera. Automatic detection of high risk situations for elder persons care. In *II International Congress on Domotics, Robotics and Remote-assistance for All*, pages 313–317, 2008.

[KAK+95] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *ICJAI-95 - Workshop on Entertainment and AI/ALIFE*, 1995.

[Kal60] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[KFO+04] Kurt Konolige, Dieter Fox, Charlie Ortiz, Andrew Agno, Michael Eriksen, Benson Limketkai, Jonathan Ko, Benoit Morisset, Dirk Schulz, and Regis Vincent. Centibots: Very large scale distributed robotic teams. In *In 9th International Symposium on Experimental Robotics*, pages 18–20. Springer Verlag, 2004.

[KFP] J. M. Kahn, R. H. Katz (ACM Fellow), and K. S. J. Pister. Abstract Next Century Challenges: Mobile Networking for Smart Dust.

[Kiv09] Kiva Systems. www.kivasystems.com, 2009.

[KKR+94] Y. Kuniyoshi, N. Kita, S. Rougeaux, S. Sakane, M. Ishii, and M. Kakikura. Cooperation by Observation - The Framework and Basic Task Patterns. In *IEEE ICRA*, pages 767–774, 1994.

[KRI+94] Y. Kuniyoshi, J. Riekki, M. Ishii, S. Rougeaux, N. Kita, S. Sakane, and M. Kakikura. Vision-based behaviors for multi-robot cooperation, 1994.

[KUS+02] Gerhard K. Kraetzschmar, Hans Utz, Stefan Sablatnög, Stefan Enderle, and Günther Palm. Miro - Middleware for Cooperative Robotics. In *RoboCup 2001: Robot Soccer World Cup V*, pages 411–416, London, UK, 2002. Springer-Verlag.

[Lab05] Newcastle Robotics Laboratory. The NUbot 2005 Team Report. www.robots.newcastle.edu.au, 2005.

[Lab06] Newcastle Robotics Laboratory. The NUbot 2006 Team Report. www.robots.newcastle.edu.au, 2006.

[Lab07] USC Robotics Research Lab. Mezzanine, 2007. http://playerstage.sourceforge.net/mezzanine/.

[LJGM06] Kristina Lerman, Chris V. Jones, Aram Galstyan, and Maja J. Mataric. Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, 25(4):225–242, 2006.

[LL94] T. Lueth and T. Laengle. Task description, decomposition and allocation in a distributed autonomous multi-agent robot system, 1994.

[LLC08] N. Lau, L. S. Lopes, and G. A. Corrente. Cambada: Information sharing and team coordination. In *Robotica 2008, Portugal*, 2008.

[MAC09] F. Martín, C. Agüero, and J. M. Cañas. Follow ball behavior for an humanoid soccer player. In *Proceedings of WAF2009 X Physical Agents Workshop*, pages 128–136, Cáceres, Spain, 2009.

[May79] P. S. Maybeck. *Stochastic models, estimation and control. Volume I.* Press, Academic, 1979.

[MBB+08] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[ME85] H Moravec and A Elfes. High resolution maps from wide angular sensors. In *Proceedings of the IEEE International Conference On Robotics and Automation (ICRA)*, pages 116–121, 1985.

[MGCRJ04] F. Martín, González-Careaga, Cañas R., and V. J.M., Matellán. Programming model based on concurrent objects for the AIBO robot. In *Actas de las XII Jornadas de Concurrencia y Sistemas Distribuídos, JCSD 2004*, pages 367–380, Navas del Marqués, Ávila (España), 2004.

[Min85] Marvin Minsky. *The Society of Mind*. Simon and Schuster, 1985.

[Mon05] Héctor Montes. *Análisis, Diseño y Evaluación de Estrategias de Control de Fuerza en Robots Caminantes*. PhD thesis, Universidad Complutense de Madrid, 2005.

[Moo67] Ramon E. Moore. *Interval Analysis (Automatic Computation S.)*. Prentice Hall, 1967.

[Mor05] Nobuyuki Morioka. Road To RoboCup 2005: Behaviour Module Design and Implementation System Integration. Technical report, University of New Wales, 2005.

[MS01] H. Martínez and A. Saffiotti. ThinkingCap-II Architecture, 2001. En línea en http://ants.dif.um.es/˜humberto/robots/tc2/.

[Mur00] Robin R. Murphy. *Introduction to AI Robotics*. MIT Press, Cambridge, MA, USA, 2000.

[Nab08] S. Nabulsi. *Diseño y control reactivo de robots caminantes sobre terreno natural*. PhD thesis, Universidad Complutense de Madrid, 2008.

[NNM+98] Itsuki Noda, C Fl Itsuki Noda, Hitoshi Matsubara, Hitoshi Matsubara, Kazuo Hiraki, Kazuo Hiraki, Ian Frank, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[Nor93] Fabrice R. Noreils. Toward a robot architecture integrating cooperation between mobile robots: application to indoor environment. *Int. J. Rob. Res.*, 12(1):79–98, 1993.

[oIRsAF09] Federation of International Robo-soccer Association FIRA. www.fira.net, 2009.

[Par94] L. E. Parker. *Heterogeneus Multi-Robot Cooperation*. PhD thesis, MIT EECS Dept., 1994.

[Par96] Lynne E. Parker. On The Design Of Behavior-Based Multi-Robot Teams. *Journal of Advanced Robotics*, 10:547–578, 1996.

[Par98] L. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[Par02] Lynne E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255, 2002.

[Par03] L. E. Parker. Current research in multirobot systems. *Artificial Life and Robotics*, 7:1–5, March 2003.

[Par08] L. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, 2008.

[PL04] Pedro Pinheiro and Pedro Lima. Bayesian sensor fusion for cooperative object localization and world modeling. In *In Proceedings of the 8th Conference on Intelligent Autonomous Systems*, 2004.

[PNDW98] D. Pagac, E. M. Nebot, and H. Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 14(4):623–629, 1998.

[PV08] Michael Phillips and Manuela Veloso. Robust Supporting Role in Coordinated Two-Robot Soccer Attack. In *Proceedings of the RoboCup Symposium*, 2008.

[QC06] Michael J. Quinlan and Stephan K. Chalup. Impact of tactical variations in the RoboCup four-legged league. In *PCAR '06: Proceedings of the 2006 international symposium on Practical cognitive agents and robots*, pages 27–38, New York, NY, USA, 2006. ACM.

[RAE02] Real Academia Española. *Diccionario de la Lengua Espanola de la Real Academia*. French & European Pubns, 21 edition, January 2002.

[RASP00] J. Rabaey, J. Ammer, J. L. Da Silva, and D. Patel. Picoradio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes. In *in Proceedings of the IEEE Computer Society Annual Workshop on VLSI*, pages 9–12, 2000.

[RDM00] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. In *Proceedings of International Conference in Robotics and Automation, San Francisco, CA*, 2000.

[RLB+05] T. Röfer, M. Laue, T.and Weber, H.-D. Burkhard, M. Jüngel, D. Göhring, J. Hoffmann, B. Altmeyer, T. Krause, M. Spranger, O. von Stryk, R. Brunn, M. Dassler, M. Kunz, T. Oberlies, M. Risler, U. Schwiegelshohn, M. Hebbel, W. Nistico, S. Czarnetzki, T. Kerkhof, M. Meyer, C. Rohde, B. Schmitz, M. Wachter, T. Wegner, and C. Zarges. http://www.germanteam.org/, Team report, 2005.

[RLO01] Luis Paulo Reis, Nuno Lau, and Eugenio Costa Oliveira. Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, LNAI 2103*, pages 175–197. Springer Verlag, 2001.

[SFF+03] Gerald Steinbauer, Michael Faschinger, Gordon Fraser, Arndt Mühlenfeld, Stefan Richter, Gernot Wöber, and Jürgen Wolf. Mostly harmless team description. In Daniel Polani, Brett Browning, Andrea Bondarini, and Kazuo Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *lnai*, Padova, Italy, 2003. Springer.

[Six98] Sixth Annual Mobile Robot Competition at Georgia Institute of Technology. http://www.cc.gatech.edu/ai/robot-lab/research/aaai97/, 1998.

[SK08] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[SL00] A. Saffiotti and K. LeBlanc. Active perceptual anchoring of robot behavior in a dynamic environment. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3796–3802, San Francisco, CA, 2000. Online at http://www.aass.oru.se/˜asaffio/.

[SL09] Joao Santos and Pedro Lima. Multi-robot Cooperative Object Localization, Decentralized Bayesian Approach. In *Proc. of the Int. RoboCup Symposium, Graz, Austria*, 2009.

[SMB01a] Ashley Stroupe, Martin C. Martin, and Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *IEEE International Conference on Robotics and Automation, May, 2001*. IEEE, May 2001.

[SMB01b] Ashley Stroupe, Martin C. Martin, and Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *IEEE International Conference on Robotics and Automation, May, 2001*. IEEE, May 2001.

[Son] Sony. Sony Global - AIBO Global Link. http://www.sony.net/Products/aibo/.

[Son06] Sony. Sony AIBO SDE and Open-R SDK. http://openr.aibo.com, 2006.

[SSH94] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to Coordinate without Sharing Information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, 1994.

[SVR99] Peter Stone, Manuela Veloso, and Patrick Riley. CMUnited-98: RoboCup-98 Simulator World Champion Team. In *In Asada and Kitano [1*, pages 61–76. Springer-Verlag, 1999.

[Syc90] Katia Sycara. Negotiation Planning: An AI Approach. *European Journal of Operational Research*, 46(1):216–234, May 1990.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[Tea05] Team Chaos Team Description Paper. http://veo.gsyc.es/dipta/documentacion/index.html, 2005.

[Tea06] TeamChaos. Team report 2005, 2006. Online at www.teamchaos.es.

[The09] The Player Project. http://playerstage.sourceforge.net/, 2009.

[TK93] Brian Tung and Leonard Kleinrock. Distributed Control Methods. In *HPDC*, pages 206–215, 1993.

[TMD+06] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the DARPA Grand Challenge. *Journal of Field Robotics*, 2006.

[TRF] The Robocup Federation. Robocup Four-Legged League. http://www.tzi.de/spl/bin/view/Website/WebHome.

[urb] urban. DARPA Urban Challenge. http://www.darpa.mil/grandchallenge/.

[VD95] José M. Vidal and Edmund H. Durfee. Recursive agent modeling using limited rationality. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95*, pages 376–383. AAAI Press, 1995.

[VRC+04] Manuela Veloso, Paul E. Rybski, Sonia Chernova, Colin Mcmillen, Juan Fasola, and Raquel Ros Espinoza. CMPack04: Team Report. Technical report, Robotics Institute of Carnegie Mellon, 2004.

[WAD+01] Thilo Weigel, Willi Auerbach, Markus Dietl, Burkhard Dümler, Jens-Steffen Gutmann, Kornel Marko, Klaus Müller, Bernhard Nebel, Boris Szerbakowski, Maximilian Thiel, and Optik Elektronik. CS Freiburg: Doing the Right Thing in a Group. In *In RoboCup 2000: Robot Soccer World Cup IV*. Springer-Verlag, 2001.

[WB04] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 2004.

[WD92] G. Werner and M. Dyer. Evolution of herding behavior in artificial animals. In *Simulation of adaptive behavior*, 1992.

[WL04] M. R. Walter and J. J. Leonard. An experimental investigation of cooperative SLAM. In *Proceedings of the Fifth IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal*, July, 2004.