Estado del Arte

RoboCup - Liga Simulada[1]

Proyecto Forrest

Serrana Casella
Pablo Rodriguez
Raúl Canales

Tutor
MSc. Gonzalo Tejera

Instituto de Computación
Facultad de Ingeniería - Universidad de la República
Montevideo - Uruguay

[1] Versión 6.0

Índice general

Índice de figuras

Índice de cuadros

List of Algorithms

Capítulo 1

Introducción

El fútbol de robots puede ser considerado como un sistema de pruebas con características similares a las que debería enfrentar un robot en sociedad (correr, caminar, determinar situaciones de peligro, etc.) para que cualquier tipo de invento, tecnología e integración de las mismas pueda ser usado con tranquilidad hasta que maduren a tal punto que puedan integrarse al mundo real. La característica mas importante de un equipo de fútbol de robots es que cada agente es autónomo y soporta en sí mismo todas las capacidades esenciales de los robots físicos.

Con este propósito, investigadores de todo el mundo se han organizado formando lo que hoy se conocen como las Federaciones de FIRA y RoboCup.

En el marco de la realizaci**ó**n del proyecto del grado *Forrest* se realizó un estudio de campo cuyos dos objetivos mas importantes fueron por un lado, realizar un análisis general de la Federación RoboCup, su organización, sus principales eventos así como un estudio de los problemas más importantes a resolver para formar un equipo capaz de competir en la liga de simulación de fútbol de tal Federación. Esto incluye el ánalisis del modelado de la realidad, tecnología de agentes y sistemas multiagentes, software de simulación oficial, problemas de sincronización, etc. Asimismo se incluye un estudio de los principales equipos de la liga con el fin de presentar los principales avances y desafíos existentes en este contexto, los cuales incluyen los resultados del campeonato 2005.

Por otro lado, este trabajo busca proveer un marco teórico que permita elaborar posibles desarrollos y trabajos futuros en el área de fútbol de robots, específicamente para la Federación RoboCup, liga simulada 2D.

Este trabajo de investigación ha requerido un esfuerzo muy importante por varias razones. La información de referencia para la liga de interés, si bien existe, se encuentra dispersa y en muchas ocasiones no disponible como es el caso de la información de la mayoría de los equipos de relevancia. Por otra parte, las técnicas y áreas relacionadas al tema del proyecto no eran de dominio de los autores, por lo que requierió un esfuerzo extra para relevar, comprender y sintetizar la información disponible.

El trabajo esta organizado de la siguiente manera: el capítulo 1 presenta a la Federación RoboCup, su objetivo, historia, y sus ligas de competición. El capítulo 2 expone el marco teórico acerca de los principales problemas a resolver para la construcción de un equipo de fútbol de robots, enfatizando las características de simulación. Por otra parte el capítulo 3 desarrolla las características específicas de la liga simulada. Esto incluye una breve historia de la liga, los campeonatos existentes, las reglas y el software de simulación oficial de la liga, un importante análisis del problema de sincronización y por último un resumen de las características del área de simulación 3D. El capítulo 4 por su parte, contiene el estudio de los principales y mas destacados equipos de la liga simulada 2D de RoboCup.

1.1. RoboCup

RoboCup es una iniciativa internacional de investigación y educación. Con la finalidad de promover la investigación en robótica y en inteligencia artificial se provee un problema estándar en donde una amplia gama de tecnologías pueden ser integradas y examinadas, así como utilizadas en proyectos reales. Dentro de estas tecnologías se destacan agentes autónomos, colaboración en sistemas multiagentes, razonamiento en tiempo real, robótica y visión entre otras.

Con este propósito, los propulsores de esta iniciativa escogieron el juego de fútbol como dominio de prueba y organizan lo que se conoce como *RoboCup*, un campeonato mundial de fútbol de robots y ciclos de conferencias que se desarrolla año a año con la creciente participación de la comunidad científica y académica, quienes se reúnen para presentar sus principales logros y avances.

Mas allá de que el fútbol es el dominio de prueba por excelencia para esta iniciativa, desde el año 2000 se han incororado las áreas de robots de búsqueda y rescate por su relevancia social. Es por eso que se crean las *Ligas de Rescate* que nada tienen que ver con el dominio del fútbol de robots.

Por otra parte, también fue introducida en el año 2000 la *Liga Junior*, la cual hoy en día se han transformado en parte importante de cualquier evento RoboCup. Su principal objetivo es introducir a la Robótica a jóvenes estudiantes de secundaria e inclusive primaria, incluyendo a estudiantes que todavía no tienen recursos para formar parte de las ligas mayores de RoboCup[AKI].

1.1.1. El Objetivo

La intención de RoboCup es promover la investigación en inteligencia artificial y robótica al utilizar, como dominio de prueba, un juego mundialmente conocido: El fútbol. Una de la formas mas efectivas y motivadoras para promover la investigación en ingeniería, además de los desarrollos orientados a una aplicación especifica, es la de establecer una meta a largo plazo. Además, cuando el cumplimiento de esta meta logra un impacto social significativo, se dice que el proyecto es parte de un gran reto.

El reto que plantea RoboCup es:

"Para mediados del siglo 21, un equipo de robots humanoides totalmente autónomos, jugará un partido de fútbol según las reglas de la FIFA[1] *contra el campeón mundial más reciente y ganará".*

Tal vez pensemos que lograr que un grupo de robots jueguen fútbol no representa un gran impacto social o económico, sin embargo, el conseguirlo ciertamente será un logro importante en el campo de la investigación científica. Y aún si el reto no es logrado, habrá muchos avances tecnológicos. A nueve años de su creación, ya es posible destacar algunos, como por ejemplo, TPOT-RL (Team-Partioned, Opaque-Transition Reinforcement Learning), un método de aprendizaje distribuído donde los agentes tienen información limitada de las transiciones de estados del medio donde habitan, el cual ha encontrado aplicación en el dominio de ruteo de paquetes en las redes de computadores [STOb].

1.1.2. Historia de la RoboCup

Alan Mackworth[2] introdujo por primera vez la idea de usar el fútbol de robots como dominio de prueba para los desarrollos en inteligencia artificial y robótica. Desafortunadamente la idea no tuvo una respuesta apropiada en su momento, hasta que más tarde fue retomada, desarrollada y adaptada por Kitano, Asada y Kuniyoshi, cuando propusieron un programa de investigación japonés llamado *Robot J-League*[3]**.** Durante el otoño de 1993, varios investigadores norteamericanos se interesaron en el *Robot J-League*, haciendo que la iniciativa se presentara como un proyecto internacional. A partir de entonces el nombre del programa cambio a *Robot World Cup Initiative* o *RoboCup* para abreviar. RoboCup es conocido también como *RoboCup challange* o *RoboCup domain*.

[1] Federación Internacional de Fútbol Asociado.
[2] Profesor de *University of British Columbia*, Canadá. Propulsor del proyecto *Dynamo* [BAR]
[3] *J-League* es la liga de fúbol profesional de Japón.

En Agosto de 1995, durante la ”Conferencia Internacional de Inteligencia Artificial”, Kitano anunció el primer juego mundial de fútbol de robots y conferencias de RoboCup [ASA], el cual luego de un profundo estudio de factibilidad tuvo lugar por primera vez en el año 1997 en la ciudad de Nagoya, Japón.

RoboCup fue presentado como plataforma de un nuevo problema estándar de inteligencia artificial (IA) y robótica, lo que se describió en su presentación como un nuevo desafío para IA luego de *DeepBleu*[4]. RoboCup se diferencia de otras investigaciones previas en el área de IA ya que busca una solución distribuida en lugar de una solución centralizada, y además porque desafía no solo a los investigadores de las áreas tradicionales de IA sino también a investigadores de áreas como robótica, sociología, sistemas de tiempo real crítico, etc.

Para coordinar los esfuerzos de todos los investigadores se forma la Federación RoboCup. La misma organiza anualmente un torneo mundial donde todos los investigadores puedan encontrarse, mostrar y demostrar sus avances y resultados. Los miembros de RoboCup hoy en día representan un número importante de Universidades y Compañías en todo el mundo. Dado que el grupo de investigadores es grande y disperso, se han formado comités locales para promover y desarrollar eventos de RoboCup dentro de su propia área geográfica [CHE]. En la sección 3.2 se mencionan las más relevantes.

Desde el primer torneo oficial del año 1997, los siguientes torneos mundiales oficiales se han desarrollado anualmente en París, Estocolmo, Melbourne, Seatle, Fukuoka & Busan, Padua, Lisboa y recientemente, en Julio de 2005 en Osaka, Japón. El próximo torneo 2006 se realizará en la ciudad de Bremen, Alemania.[CROB97, CROB98, CROB99, CROB00, CROB01, CROB02, CROB03, CROB04, CROB05a].

1.1.3. Ligas de RoboCup

El torneo oficial mundial de RoboCup consta de dos actividades principales, el *campeonato* y el *simposio* o ciclo de conferencias. Durante el campeonato, los equipos de las diferentes ligas compiten entre sí ya sea en el juego de fútbol o en las ligas de rescate y junior. Para algunas ligas de robots físicos una competencia consiste en ubicar obstáculos en la cancha de juego y hacer que los robots lleguen al arco contrario evadiendo los mismos. Un comité evaluador asigna puntajes a las habilidades demostradas y gana el equipo que haya obtenido mayor puntaje. Asimismo el campeonato también prevé las actividades de exhibición para todas las ligas, donde se destaca determinado avance o logro.

Inmediatamente después a la finalización del campeonato comienza el ciclo de conferencias, donde el principal objetivo es reunirse para la presentación de contribuciones científicas en áreas de relevancia de la federación. Todas los artículos presentados y expuestos son editados y publicados anualmente en lo que se denomina *RoboCup subseries of the Springer LNAI book series* junto con la descripción general de cada equipo participante de la competencia[AKI]. Estas ediciones son comerciales.

En lo que sigue de esta sección vamos a presentar cada una de las ligas que conforman actualmente la federación RoboCup, mostrando para cada una de ellas sus principales características, logros y desafíos enmarcado en la información disponible de los últimos campeonatos mundiales. Muchas de las características que se detallan respecto a logros y desafíos se refieren al año 2004 debido a que aún no se encuentra disponible este tipo de información del campeonato 2005.

Ligas Oficiales

RoboCup Soccer (fútbol)

- Middle-Size Soccer Robot (F-2000) League
- Small-Size Soccer Robot (F-180) League
- 4-Legged Robot League
- Humanoid League

[4]En referencia a la máquina de IBM *DeepBleu* y la competencia de ajedrez con Kasparov, ver http://www.chess.ibm.com

- Soccer Simulation League
- E-league

RoboCup Rescue (rescate)

- Rescue Robot League
- Rescue Simulation League

RoboCup Junior (jóvenes)

- Soccer
- Rescue
- Dance

La mayoría de estas ligas consisten en grupos de robots o programas cuyo objetivo es cooperar para derrotar a un equipo oponente. Sin embargo, las ligas de rescate y las pruebas de exhibición persiguen otras metas. El objetivo de derrotar a un oponente en la liga de rescate llevaría tal vez, a debates de cuestiones éticas debido a que no se puede asignar utilidades comparables a las vidas humanas y a las construcciones, edificios, etc. Es por esto que el foco en la liga de rescate esta en lograr esfuerzos cooperativos entre agentes heterogéneos en situaciones de desastre. Por otra parte, el objetivo en las pruebas de exhibición es observar y comentar.

1.1.3.1. Middle Size Robot (F-2000) League

En esta liga cada equipo esta compuesto de cuatro robots como máximo. Las restricciones para cada robot son las siguientes:

- 50 cm de diámetro como máximo
- 75 cm de alto como máximo.

El tamaño de la cancha es de aproximadamente 12 x 8 metros[AKI]. Una característica importante de esta liga es que los robots poseen información restringida del mundo real (en contraposición a la visión global) y la comunicación entre robots es soportada por tecnología *wireless*. Las áreas de investigación para esta liga incluyen localización, visión, motores para control de robots, etc.

Figura 1.1: Middle Size Robot League - Campeonato Osaka 2005 [CROB05b].

Posición	Equipo	País
1st	EIGEN Keio Univ (Keio University)	Japón
2nd	FU - Fighters (Freie Univeristat Berlin)	Alemania
3rd	Philips (Philips)	Netherlands

Cuadro 1.1: Middle Size Robot (F-2000) League Osaka 2005 - Los tres mejores equipos [ROBb].

1.1.3.2. Small Size Robot (F-180) League

En esta liga cada equipo se compone de 5 robots donde cada uno debe cumplir las siguientes restricciones:

- 15 cm de diámetro como máximo
- 20 cm de alto como máximo.

Esta categoría posee visión global la cual es provista por una cámara posicionada encima de la cancha. Las áreas de investigación que son importantes para esta liga incluyen el control inteligente de los robots, el procesamiento de imágenes, etc.[CROB05b, CROB05a].

Figura 1.2: Small Size Robot League - Campeonato Osaka 2005 [CROB05b].

Posición	Equipo	País
1st	FU - Fighters (Freie Univeristat Berlin)	Alemania
2nd	Cornell Big Red (Cornell University)	USA
3rd	Field Rangers (Singapore Polytechnic)	Australia

Cuadro 1.2: Small Size Robot (F-180) League Osaka 2005 - Los tres mejores equipos [ROBb].

1.1.3.3. 4-Legged Robot League

En esta liga cada equipo esta formado por tres robots cuadrúpedos Sony (también conocidos como AIBOs). El tamaño de la cancha es similar al de la liga Small Size League. Estos robots no tienen una visión global de la realidad, usan marcas de color que se ubican alrededor del campo de juego de forma de que los ayude a autolocalizarse. Las principales áreas de interés para esta liga son el control inteligente del robot y la interpretación de la información sensorial.

Figura 1.3: Sony Legged Robot League - Campeonato Osaka 2005 [CROB05b].

Posición	Equipo	País
1st	German Team 2005 (Humboldt Universitat Berlin)	Alemania
2nd	NUBots (University of Newcastle)	Australia
3rd	rUNSWift (University of New South Wales and National ICT)	Australia

Cuadro 1.3: 4-Legged Robot League Osaka 2005 - Los tres mejores equipos [ROBb].

1.1.3.4. Humanoid League

Esta liga fue introducida a RoboCup en el campeonato de Fukuoka, Japón 2002. El tamaño de los robots que se construyen en esta liga debe situarse entre 10 cm de alto como mínimo y 200 cm como máximo [ROBa]. La característica distintiva de esta liga respecto a las demás es el aspecto humano de los robots (dos piernas, dos brazos, posición erguida, etc).

Las competencias en esta liga consisten principalmente en la exhibición de habilidades como pararse en una pierna, caminar, ejecutar un tiro penal, expresión libre de habilidades, etc. Los competencias entre robots consisten en partidos 2 contra 2, carreras, etc.[CROB05c]. El actual campeón del campeonato oficial en esta liga es el equipo Team Osaka [CROB05b, CROB05a].

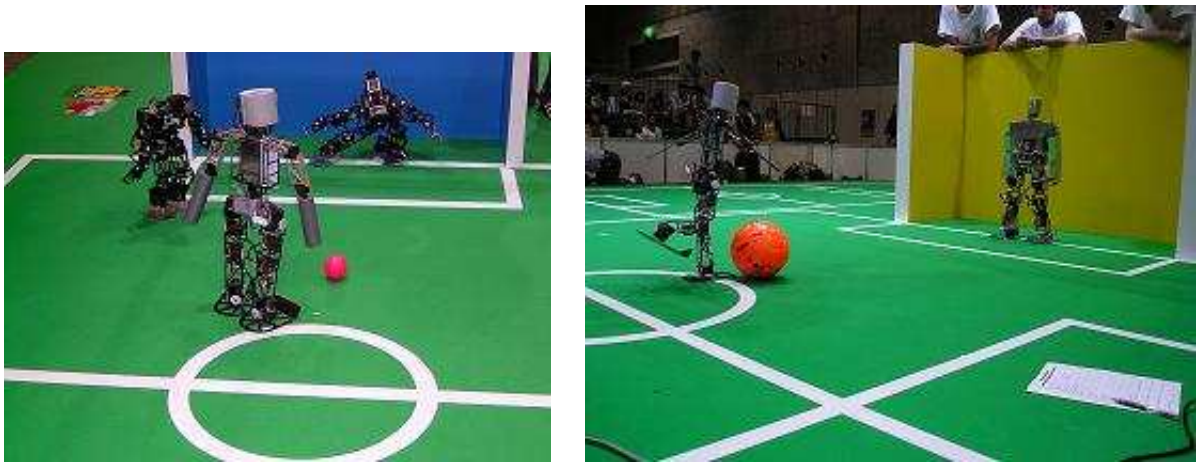

Figura 1.4: Humanoid League - Robot NimbRo, segundo en el campeonato Osaka 2005 [PONIM].

Categoría	Equipo	País
Mejor Humanoide	Team Osaka (Vstone) Co., Ltd.	Japón

Cuadro 1.4: Soccer Humanoide League Osaka 2005 - Mejor Humanoide [ROBb].

Posición	Equipo	País
1st	Team Osaka (Vstone) Co., Ltd.	Japón
2nd	NimbRo (University of Freiburg)	Alemania
3rd	Team Hajime (Hajime Lab.)	Japón

Cuadro 1.5: Soccer Humanoide League Osaka 2005 - Los tres mejores Humanoides en la competencia 2 contra 2 [ROBb].

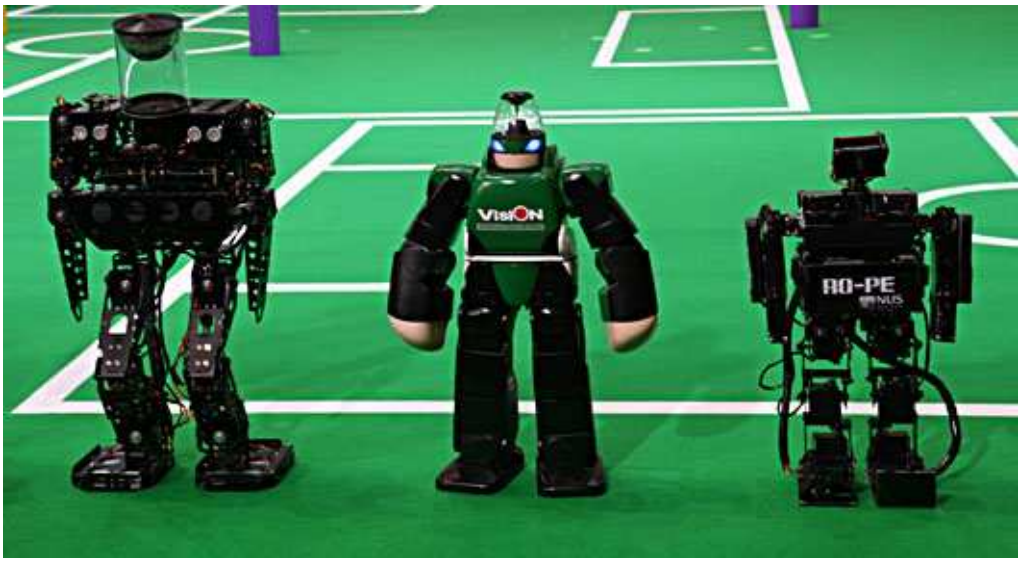

Figura 1.5: Humanoid League. En el medio figura el Humanoide *Visión* del equipo Team Osaka [CROB05c]

Categoría	Equipo	País
Tiro Penal en tamaño medio	NimBro(University of Freiburg)	Alemania
Tiro Penal en tamaño pequeño	Team Osaka (Vstone Co.,Ltd.)	Japón
Desafío técnico	Team Osaka (Vstone Co.,Ltd.)	Japón
	NimBro(University of Freiburg)	Alemania

Cuadro 1.6: Soccer Humanoide League Osaka 2005 - Mejores Humanoides por categoría [ROBb].

1.1.3.5. Soccer Simulation League

Ésta es una de las ligas más antiguas de RoboCup. Consiste en dos equipos que juegan entre ellos a través de un simulador de software que simula el campo de juego y distribuye información sensorial a los agentes conectados. Cada participante conecta al simulador 11 agentes jugadores y opcionalmente un agente entrenador. Los partidos constan de dos tiempos de cinco minutos de duración.

Cada agente funciona como un proceso independiente el cual se comunica con el servidor para comunicarse con otros agentes y para ejecutar acciones que afectan el entorno que lo rodea. Asimismo recibe información (la cual es parcial y afectada por ruido) de la realidad. Los países con mayor cantidad de participantes son Irán, Japón y Alemania.

Actualmente, esta liga consta de tres competiciones:

2D

Posición	Equipo	País
1st	Brainstormers 2D (University of Osnabrueck)	Alemania
2nd	Wright Eagle 2005 (University of Science & Technology of China)	China
3rd	Tokyo Tech SFC (Tokyo Institute of Technology)	Japón

Cuadro 1.7: Soccer Simulation League 2D Osaka 2005 - Los tres mejores equipos de la competencia [ROBb].

Esta competición se caracteriza por usar el simulador descrito en [CHE]. En el campeonato de Lisboa 2004 fue la primera vez que la calificación para participar se realizo a través de la Liga Simulada de Internet [SSIL]. Desde en-

tonces es el procedimiento oficial para calificar para el campeonato mundial (con la consideración que los mejores equipos del campeonato anterior califican automaticamente).

3D

Posición	Equipo	País
1st	Aria (Amirkabir University of Techonology)	Irán
2nd	Brainstormers 3D (University of Osnabrueck)	Alemania
3rd	ZJUBase (Zhejiang University)	China
3rd	Caspian (IUST)	Irán

Cuadro 1.8: Soccer Simulation League 3D Osaka 2005 - Los tres mejores equipos de la competencia [ROBb].

La competición 3D se introdujo en el año 2004 y hace uso de en un nuevo simulador introducido en el simposio de la RoboCup 2003 el cual se basa un middleware llamado SPADES[RILb] introducido en el año 2002. En tal simulador los jugadores son percibidos como esferas en un entorno tridimensional con un modelo físico.

En el primer año de esta competencia (2004), los equipos participantes se destacaron por la habilidad de usar las capacidades básicas de los agentes para moverse en la cancha de juego y trasladar la pelota. Los agentes en el simulador 3D pueden moverse en cualquier dirección, pero debido a la inercia y a los efectos de rozamiento del modelo de comandos, los métodos usados por los jugadores para lograr trasladar la pelota no fueron sencillos. A pesar del corto tiempo de desarrollo y de las dificultades que presentan las tres dimensiones, algunos participantes manejaron no solo la implementación de las habilidades de bajo nivel como interceptar la pelota o el *dribbling*, sino también la implementación de comportamientos a nivel de equipo como el pase y el manejo cooperativo de determinadas situaciones [AKI].

Coach

En esta competición cada participante tiene que proveer un agente entrenador que puede actuar directamente sobre un equipo usando un lenguaje especifico para la comunicación entre los equipos llamado *Clang* [CHE]. Estos agentes entrenadores pueden trabajar analizando logs de juegos previos así como observando y adaptando su comportamiento mientras el juego se desarrolla.

Un agente entrenador es evaluado a través de un partido donde uno de los equipos es entrenado por él y el oponente es un equipo dado por el comité evaluador. Todos los agentes entrenadores juegan contra el mismo equipo y el que obtenga mejores resultados, a juicio del comité, es el ganador.

Posición	Equipo	País
1st	UT Austin Villa (UT Austin)	USA
2nd	Aria (Amirkabir University of Technology)	Irán
3rd	Kasra (OSW Kasra Amirkabir University of Technology)	Irán

Cuadro 1.9: Soccer Simulation League Coach Osaka 2005 - Los tres mejores equipos de la competencia [ROBb].

Las líneas de investigación en esta competencia están dirigidas hacia la adaptación en línea del equipo y en el modelado del oponente.

Logros y desafíos de la liga

En la competición 2D, los enfoques usados por los equipos varían notoriamente y no existe un único método para implementar un equipo exitoso. Los mejores equipos de esta competición llevan un tiempo considerable participando en eventos RoboCup por lo que el nivel de juego es muy avanzado lo cual generalmente dificulta la incorporación de nuevos equipos a las competencias oficiales [AKI].

1.1.3.6. E-League

E-League tiene como principal objetivo priorizar el desarrollo de la inteligencia de los robots por encima del desarrollo de costosos y complejos componentes electrónicos.

Un partido de esta liga se juega en una cancha de 1.52 x 2.74 metros y cada equipo tiene como máximo 4 jugadores entre los cuales se incluye al arquero. Los robots pueden tener diferentes diseños siempre y cuando cada robot satisfaga las siguientes restricciones:

- Un máximo de 22 cm de diámetro
- Un máximo de 15 cm de alto
- Un máximo de 3 motores

La información del mundo real es provista por un sistema de visión global común a ambos equipos. Este sistema incluye una cámara de video conectada a un servidor de software de reconocimiento de imágenes llamado *Doraemon* [AND, DEL] capaz de distinguir los distintos objetos dentro del campo de juego.

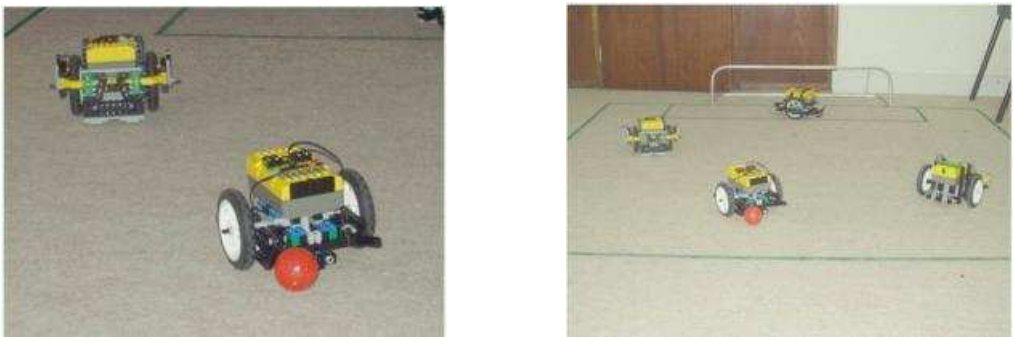

Figura 1.6: E-League - Robots del equipo MateBots, ganadores del campeonato Portugal 2004 [POMAT].

1.1.3.7. RoboCup Rescue

Salvar vidas en situaciones de desastre es una de las tareas más importantes que deben resolver un gran número de agentes con diferentes características interactuando en un ambiente hóstil. El objetivo de la liga *Rescue* es fomentar la investigación y el desarrollo dentro de este campo social incluyendo la coordinación del trabajo en equipo de múltiples agentes (agentes robóticos de búsqueda y rescate, infraestructuras de información, asistentes digitales personales, simulador estándar y sistemas de ayuda a la decisión, pruebas de evaluación para estrategias de rescate y sistemas robóticos que se integrarán en el futuro dentro de un sistema global)[CROB05c]. La liga RoboCupRescue está dividida en las siguientes competencias:

- Rescue Robot League

Posición	Equipo	País
1st	Toin Pelican (Toin University of Yokohama)	Japón
2nd	ROSCUE (KIST)	Korea
3rd	Casualty (Centre for Autonomous Systems)	Australia

Cuadro 1.10: RoboCup Rescue Robot Osaka 2005 - Los tres mejores equipos de la competencia [ROBb].

- Rescue Simulation League

Posición	Equipo	País
1st	Impossibles (Sharif University of Techonology)	Irán
2nd	Caspian (IUST)	Irán
3rd	Kshitij (IIIT-Hyderabad)	India

Cuadro 1.11: RoboCup Rescue Simulation Osaka 2005 - Los tres mejores equipos de la competencia [ROBb].

Figura 1.7: Rescue Real Robot League - Trabajo del actual campeón Osaka 2005, Toin Pelican [CROB05b].

1.1.3.8. RoboCup Junior

La liga RoboCup Junior es una iniciativa educativa a escala mundial orientada a la realización de proyectos que fomenta eventos de robótica a nivel local, regional e internacional para jóvenes estudiantes. Su objetivo es presentar RoboCup a alumnos de educación primaria y secundaria, incluyendo también a estudiantes de ciclos superiores que carezcan aún de medios para participar en la liga de adultos. La liga junior se centra especialmente en la educación [CROB05c].

Esta liga se compone de tres competencias:

- Soccer
- Rescue
- Dance

Capítulo 2

Aspectos Generales de Diseño

El objetivo de este capítulo es presentar un marco te**ó**rico acerca de los principales problemas a enfrentar en el desarrollo de un equipo de fútbol de robots. La mayoría de los temas que se exponen se encuentran enmarcados en el entorno de simulación, por ser este el entorno específico en el cual se desarrollará el proyecto que contiene a este trabajo.

2.1. Tecnología de Agentes y Sistemas Multiagente[1]

Los roles en los sistemas multiagenes son fundamentales, dado que disminuyen el espacio de estados posible que debe manejar cada agente del sistema, reduciendo así la complejidad. Un agente con determinado rol tendrá menos comportamientos entre los cuales elegir. Por otro lado, los roles en el fútbol existen por definición del propio deporte. Por lo que, aparece un motivo directo para el uso de los avances en sistemas de multiagenes para el dominio del fútbol de robots. Lo que se presenta al respecto en esta sección son los principales conceptos teóricos, sin pretender ser un estado del arte en el tema.

La inteligencia artificial distribuida (IAD) se ha definido como un subcampo de la Inteligencia Artificial (IA) que se centra en los comportamientos inteligentes colectivos que son producto de la cooperación de diversas entidades denominadas *agentes*. A pesar de ser una área de investigación relativamente joven, existen tres enfoques que abordan este tema:

- la IAD clásica la cual se centra en el estudio de la conducta colectiva de los agentes, en oposición a la IA que estudia la conducta individual de los mismos.
- la IAD autónoma la cual se centra en el estudio de los agentes individuales situados en un mundo social. Esta enfoque modifica la visión inicial de la IAD en que la perspectiva social significa que la sociedad prima sobre los individuos, que se explican a partir de su función en la sociedad, y se centra en estudiar agentes autónomos en un mundo multiagente.
- IAD comercial la cual se centra en la aplicación de la IAD clásica y autónoma, desarrollando agentes (denominados genéricamente *agentes de software*) con características muy diferenciadas (agentes móviles, personales,etc.) que están siendo explotados de forma comercial.

A continuación, presentamos un resumen de las principales características de los dos primeros enfoques por ser los mas relevantes para el marco de este trabajo. El tercer enfoque, si bien es interesante, refiere mas bien a la aplicación de la tecnología de agentes a Sistemas de Información, los cuales se encuentran fuera del alcance de interés de la investigación realizada.

2.1.1. IAD Clásica: Perspectiva de grupo de agentes

La perspectiva de grupo estudia los elementos que caracterizan a un grupo de agentes, su comunicación e interacciones. Los elementos característicos pueden descomponerse a su vez en tres aspectos: organización del grupo, comunicación y coordinación.

[1] Partes de esta sección fueron tomadas directamente de [IGL]

La inteligencia artificial distribuida clásica es un subcampo de la inteligencia artificial dedicado al estudio de las técnicas y el conocimiento necesario para la coordinación y distribución del conocimiento y las acciones en un entorno con múltiples agentes. Podemos distinguir dos áreas principales de investigación [WEI]:

- **Resolución (cooperativa) de problemas distribuidos** (DPS, Distributed Problem Solving) estudia cómo un conjunto de módulos (o agentes) cooperan para dividir y compartir el conocimiento de un problema y en el desarrollo de la solución.
- **Sistemas Multiagente** (MAS, Multiagent Systems) estudia la coordinación de la conducta inteligente entre un conjunto de (posiblemente pre-existentes) agentes inteligentes autónomos.

La principal diferencia entre ambas áreas se encuentra en la flexibilidad de la coordinación entre los agentes. En la DPS, las interacciones y tareas que cada agente realiza están prefijadas de antemano: hay un plan centralizado de resolución del problema. Suele haber un miembro que ejerce un control global que centraliza los resultados parciales y datos entre el resto de los componentes del sistema. En contraposición, en los MAS, los agentes tienen un grado de autonomía mayor y pueden decidir dinámicamente qué interacciones son adecuadas, qué tareas deben realizar, quién realiza cada tarea y, además, es posible mantener conocimiento que no es globalmente consistente, incluso los agentes pueden mantener objetivos globales diferentes. Esta distinción permite diferenciar entre sistemas que se centran en el comportamiento global, con una conducta fija de los agentes (DPS) y sistemas que se centran en la conducta de los individuos que como resultado, obtienen una conducta del sistema (MAS). Si realizamos una analogía con las sociedades humanas, las dos posturas podrían compararse con la resolución de un problema por el estado central (el diseñador) que planifica y regula las conductas de los individuos (que serán predecibles) o dejar que el problema se resuelva por la libre iniciativa de los individuos.

Los problemas básicos que estudia IAD clásica, y que serán comunes a todos los sistemas, son:

- Cómo formular, describir, descomponer y asignar problemas y sintetizar los resultados entre un grupo de agentes inteligentes.
- Cómo capacitar a los agentes para que se comuniquen e interactúen: qué lenguajes de comunicación o protocolos deben utilizarse, qué y cuándo deben comunicarse, etc.
- Cómo asegurar que los agentes actúan coherentemente al tomar decisiones o realizar acciones, cómo manejar los efectos globales de las decisiones locales y prevenir interacciones no deseadas.
- Cómo capacitar a los agentes para representar y razonar sobre acciones, planes y conocimiento de otros agentes para coordinarse; cómo razonar sobre el estado de su proceso de coordinación (inicio o terminación).
- Cómo reconocer y resolver puntos de vista e intenciones conflictivas entre un conjunto de agentes para coordinar sus acciones; cómo sintetizar los puntos de vista y los resultados.
- Cómo utilizar técnicas de IA y desarrollar sistemas con IAD. Cómo diseñar plataformas de MAS y metodologías de desarrollo con técnicas de IAD.

Desarrollaremos a continuación y brevemente, los aspectos relacionados con la perspectiva de grupo. La misma recoge los aspectos de organización entre los agentes, las posibles formas de comunicación y los aspectos de control dinámico para conseguir una conducta global coherente.

A. Descripción, descomposición y asignación de tareas

Uno de los problemas de la IAD es cómo representar los problemas, ya que la descomposición de los mismos depende en gran medida de su formulación. La descripción debe incluir la información sobre las características y atributos del problema así como la información del dominio y el entorno del problema. Una vez dada la descripción de una tarea, la descomposición de la misma y asignación de las subtareas a múltiples agentes debe tener en cuenta que estos tengan capacidad para llevarlas a cabo así como disponibilidad de recursos.

Se han identificado varias dimensiones comúnmente utilizadas para realizar la descomposición de un problema:

- *Nivel de abstracción:* los agentes que actúan como solucionadores de problemas pueden ser asociados con un nivel de abstracción del mismo. La descomposición del problema en niveles de abstracción proporciona una buena base para la descomposición de tareas.
- *Dependencias de control:* las tareas pueden descomponerse tomando como principio la reducción de las dependencias de datos o control entre tareas, de forma que se restrinja la comunicación. Las dependencias de los datos pueden ser semánticas, temporales o procedentes de la distribución de los datos de entrada. Cuando una tarea se distribuye, suele ser necesario introducir tareas de control para coordinarlas.
- *División funcional/producto:* la descomposición funcional consiste en agrupar a los agentes que realizan funciones similares, mientras que la división por productos consiste en agrupar a los que trabajan en la producción de un mismo producto.
- *Necesidad de redundancia:* para garantizar la robustez del sistema o para incluir diferentes perspectivas puede ser necesario duplicar tareas entre diferentes solucionadores de problemas. En general, debe evitarse que una tarea sea sólo resoluble por un agente.
- *Minimización de recursos:* debe minimizarse la utilización de los recursos para evitar una sobrecarga de comunicación y coordinación para acceder a dichos recursos limitados.

Las técnicas más empleadas para realizar una descomposición automática de las tareas por parte de los agentes son:

- *Tareas inherentemente descomponibles:* la propia descripción de la tarea incluye su descomposición.
- *Descomposición* realizada por el programador.
- Planificación jerárquica: es uno de los sistemas más empleados para descomponer de forma automática. Las tareas se definen en forma de planes que satisfacen determinados objetivos.
- Agregación de subtareas: enfoque ascendente en vez de enfoque descendente en la descomposición.

B. Comunicación

Los agentes pueden mejorar su coordinación y coherencia gestionando qué, cómo y cuándo se comunican entre sí. La comunicación puede proporcionar a los agentes el conocimiento necesario para desarrollar sus acciones con una visión menos local y poder sincronizarse con el resto de agentes. Sin embargo, una excesiva comunicación puede dar lugar a una sociedad de agentes burocráticos, cuya sobrecarga de comunicación sea mayor que el trabajo efectivo realizado. Podemos distinguir un rango amplio de formas de comunicación, que van desde la falta de comunicación hasta la comunicación de alto nivel y la interacción hombre máquina.

Sin comunicación

Los agentes pueden interactuar sin comunicarse, infiriendo las intenciones de otros agentes. Esta situación puede darse debido a fallos de hardware, a la imposibilidad de comunicarse o al deseo de que los agentes tengan mayor autonomía. Normalmente, para que sea posible la cooperación se supone que los agentes disponen de la información sensorial suficiente para poder inferir los objetivos intencionales del resto de agentes. Para estudiar este tipo de interacción se ha recurrido a la teoría de juegos, empleando matrices de costes, en las que se representa la ganancia que obtiene cada agente dependiendo de la acción que realice otro agente. Este enfoque se ha extendido con técnicas probabilísticas para representar la incertidumbre sobre los posibles movimientos de los otros agentes[IGL].

Comunicación primitiva

La comunicación se restringe a un número de mensajes con interpretaciones fijas. Este enfoque ha sido aplicado a planificación multiagente para coordinar a dos agentes que tienen un conflicto (de escasez de recursos por ejemplo) mediante un mediador, pero la coordinación que puede lograrse es limitada (ceder el control a través de un mediador que sincroniza a los agentes)[IGL].

Pasaje de mensajes

Los sistemas de paso de mensajes permiten que un agente envíe un mensaje (tanto de peticiones de servicios e información como de respuesta a dichas peticiones) a uno o más agentes cuyos nombres debe conocer. Con este enfoque los agentes deben mantener conocimiento sobre su entorno para saber a qué agentes deben dirigir sus mensajes. Este modelo tiene su origen en la programación orientada a objetos concurrente.

Numerosos sistemas multiagente han adoptado el paso de mensajes definiendo un formato para dichos mensajes, tipos de mensaje y un protocolo para procesar dichos mensajes[AGU, IGL].

Comunicación de alto nivel

Se ha realizado un gran esfuerzo de investigación para estudiar las interacciones entre los agentes en el nivel de conocimiento en vez de en el nivel simbólico. Esto supone que los agentes puedan razonar sobre las intenciones, deseos y objetivos de otros agentes, y que puedan comunicar estos deseos, objetivos e intenciones.

Para analizar estas interacciones complejas, se han intentado aplicar a los sistemas multiagente algunas técnicas y teorías provenientes del campo del lenguaje natural. De acuerdo con las técnicas simbólicas de comprensión del diálogo, no basta con analizar el significado de cada frase para entender un texto, sino que es necesario comprender las intenciones de los interlocutores y cómo se desarrolla el diálogo para satisfacer sus objetivos, ya que no toda la comunicación suele ser explícita.

Interacción hombre-máquina

La comunicación entre un agente artificial y un agente humano ha tenido durante un mucho tiempo gran relevancia. Básicamente, se han tomado dos aproximaciones: encapsular al agente humano modelando sus interacciones en un lenguaje de comunicación de agentes o aprovechar la tecnología multiagente para simplificar las interfaces hombre-máquina.

C. Coherencia y coordinación

Se define coherencia como la propiedad de un sistema para comportarse como una unidad, de acuerdo con alguna dimensión de evaluación [IGL]. Podemos evaluar la coherencia de un sistema examinando varias dimensiones de su conducta:

- **Calidad de la solución:** habilidad del sistema para alcanzar soluciones satisfactorias, y la calidad de las soluciones que produce. Requiere que se alcancen tres condiciones:
  - *Cobertura:* cada tarea necesaria del problema debe ser realizada al menos por un agente.
  - *Conectividad:* los agentes deben interactuar de forma que las actividades cubiertas puedan ser desarrolladas e integradas en una solución global.
  - *Capacidad:* la comunicación debe ser suficiente para que puedan conseguirse la cobertura y la conectividad.
- **Eficiencia:** eficiencia global del sistema para alcanzar un fin.
- **Claridad:** claridad conceptual de las acciones del sistema y utilidad de su representación. Posibilidad de describir y representar la conducta del sistema de forma que un observador externo pueda entenderla. En un sistema describible y bien estructurado la auto-representación puede ser usada para comunicación interna, reorganización, diagnóstico de fallos, análisis de rendimiento, etc.
- **Robustez:** grado de degradación del sistema en presencia de fallos o incertidumbre.

Definimos *coordinación* como la propiedad de interacción entre un conjunto de agentes que realizan alguna actividad colectiva. El grado de coordinación exhibido por un conjunto de agentes es el área en que pueden evitar realizar un *trabajo de articulación* para coordinarse. La coordinación efectiva implica un cierto grado de predecibilidad mutua y falta de conflictos. Cuantos más conflictos inesperados se den, peor coordinados estarán los agentes. Definimos *cooperación* como una clase de coordinación entre agentes no antagonistas (los agentes antagónicos pueden ser coordinados, por ejemplo si son robots, para no chocar). La coherencia y la coordinación están relacionadas: una mejor coordinación debe guiar a una mayor coherencia por la reducción del trabajo de articulación, aunque la coordinación no garantiza la coherencia[IGL, BERb].

Podemos distinguir los siguientes mecanismos para facilitar la coordinación:

- Negociación: empleo de diálogo entre los agentes para resolver vistas inconsistentes y alcanzar un acuerdo sobre cómo trabajar conjuntamente.
- Cooperación funcionalmente precisa: la inconsistencia se supera intercambiando soluciones tentativas para resolver errores y converger en las soluciones del problema.
- Estructuración organizativa: utilización de conocimiento común sobre los roles generales de resolución del problema y patrones de comunicación para reducir la incertidumbre de los agentes y sobre cómo deben cooperar.
- Planificación multiagente: compartición de información para construir un plan de cómo los agentes deben trabajar juntos, distribuyendo y siguiendo este plan durante la resolución del problema.

D. Reconocimiento y resolución de discrepancias entre agentes

Según [IGL] pueden distinguirse cuatro tipos de diferencias de conocimiento: *incompletitud*, sucede cuando un agente tiene algún conocimiento que otro no tiene; *inconsistencia*, sucede cuando dos agentes tiene diferentes valores de verdad para la misma proposición lógica; *incompatibilidad*, sucede cuando el conocimiento es representado en formas incompatibles; e *inconmensurabilidad*, que sucede cuando el conocimiento está representado de la misma forma, pero las interpretaciones semánticas son diferentes.

Para que los agentes pueden reconocer las diferencias es necesario que tengan representaciones de la realidad compatibles. Para reconocer estas diferencias los agentes suelen representar sus propias creencias y las de otros agentes. Algunos de los métodos empleados para reconciliar las discrepancias o conflictos son [IGL]:

- **Obtención de un conocimiento común:** cuando la diferencia se debe a conocimiento incompleto puede solicitarse este conocimiento a otros agentes.
- **Revisión de premisas:** cuando se detectan proposiciones inconsistentes, pueden revisarse las premisas de dichas proposiciones para descubrir si son estas premisas las causas de la inconsistencia.
- **Autoridad y mediación:** en numerosos casos, es necesario el arbitrio, o un mediador, o un criterio jerárquico para resolver el conflicto.
- **Resolución basada en casos:** los conflictos pueden resolverse recurriendo a casos similares sucedidos en el pasado.
- **Resolución de restricciones:** en el caso de que el conflicto se dé por restricciones conflictivas, el conflicto puede resolverse relajando las restricciones no esenciales.
- **Negociación:** la negociación, otra vez, suele ser una técnica empleada para resolver conflictos.
- **Estandarización:** la experiencia acumulada en la resolución de conflictos puede conducir a la estandarización de las conductas para evitar o resolver los conflictos.

2.1.2. IAD autónoma: Perspectiva del agente

La IAD autónoma se centra en los micro-aspectos de la IAD, es decir, en los agentes inteligentes, más que en los macro-aspectos (tratados en la IAD clásica), si bien estos macroaspectos son relevantes para abordar los micro-aspectos. Dichos aspectos se dividen en tres áreas [IGL]: *teoría de agentes*, que trata de responder a la pregunta de qué es un agente y de la utilización de formalismos matemáticos para representar y razonar sobre las propiedades de agentes; *arquitecturas de agentes*, que trata de las arquitecturas software/hardware que permiten reflejar las propiedades enunciadas por los teóricos; y *lenguajes de agentes*, que son herramientas de software para programar y experimentar con agentes.

A. Teoría de Agentes

Las teorías de agentes son especificaciones para conceptualizar los agentes. Debido a que la definición de agente ha resultado ser un tanto controvertida como la definición de inteligencia artificial, se ha optado por una definición de un conjunto de propiedades que caracterizan a los agentes, aunque un agente no tiene por que poseerlas todas:

- *autonomía:* los agentes pueden operar sin la intervención de humanos o de otros agentes;
- *sociabilidad:* los agentes son capaces de interactuar con otros agentes (humanos o no) a través de un lenguaje de comunicación entre agentes;
- *reactividad:* los agentes son capaces de percibir estímulos de su entorno y reaccionar a dichos estímulos;
- *proactividad, iniciativa:* los agentes no son sólo entidades que reaccionan a un estímulo, sino que tienen un carácter emprendedor, y pueden actuar guiados por sus objetivos;
- *movilidad:* capacidad de un agente de trasladarse a través de una red;
- *veracidad:* se asume que un agente no comunica información falsa a propósito;
- *benevolencia:* se asume que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos;
- *racionalidad:* se asume que un agente actúa de forma racional, intentando cumplir sus objetivos si son viables.

Por otro lado, se pueden distinguir otras dos nociones de agentes [IGL]:

- Una *noción débil* de agente consiste en definir un agente como a una entidad que es capaz de intercambiar mensajes utilizando un lenguaje de comunicación de agentes. Esta definición es la más utilizada dentro de la ingeniería software basada en agentes, cuyo fin es conseguir la interoperabilidad entre aplicaciones a nivel semántico utilizando la tecnología de agentes.
- Una *noción más fuerte* o restrictiva de agente lo define como una entidad cuyo estado es visto como un conjunto de componentes mentales, tales como creencias, capacidades, elecciones y acuerdos.

Los agentes suelen ser considerados como sistemas intencionales, esto es, sistemas cuya conducta puede ser predecida atribuyendo creencias, deseos y una conducta racional. Para representar estas intenciones, se han empleado diversos formalismos lógicos, de entre los que cabe destacar la teoría de la intención de Cohen y Levesque[IGL], la lógica multi-modal BDI y la familia de lógicas para especificar sistemas multiagente.

B. Arquitecturas de agentes

Las arquitecturas de agentes describen la interconexión de los módulos software/hardware que permiten a un agente exhibir la conducta enunciada en las teorías de agentes.

Una primera clasificación de las arquitecturas puede ser realizada según todas las capas tengan acceso a sensores y actuadores (horizontales) o sólo la capa más baja tenga acceso a sensores y actuadores (verticales), tal como se muestra en 2.1. Las arquitecturas horizontales ofrecerán la ventaja del paralelismo entre capas a costa de un alto conocimiento de control para coordinar las capas, mientras que las verticales reducen este control a costa de una mayor complejidad en la capa que interactúa con los sensores.

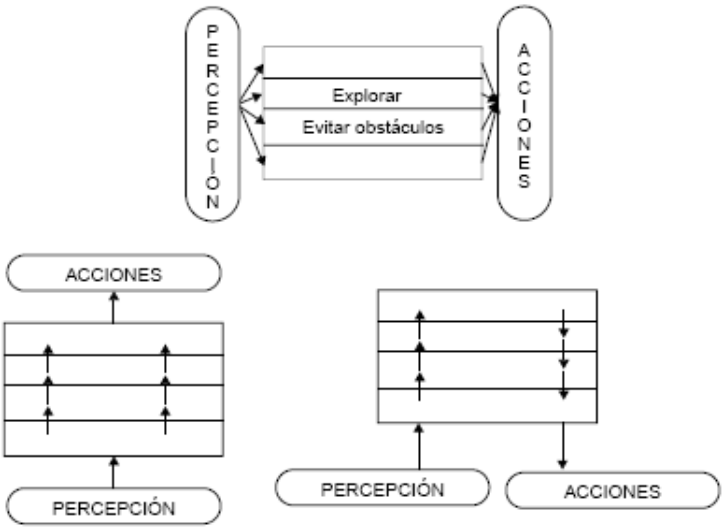

Figura 2.1: Modelo de arquitectura horizontal (fig. superior) y vertical (fig. inferior) [IGL]

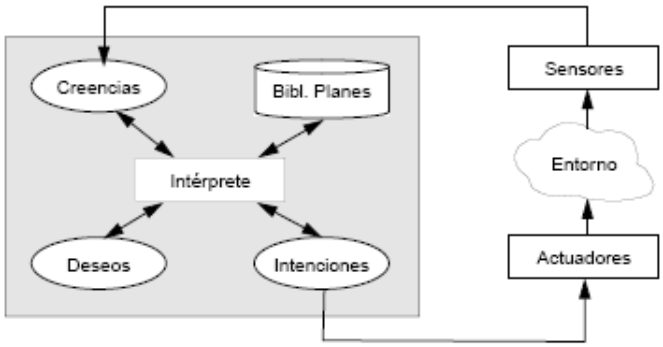

Figura 2.2: Arquitectura de un agente deliberativo[IGL]

También podemos clasificar las arquitecturas según el tipo de procesamiento empleado [BERb] en deliberativas, reactivas e híbridas.

Arquitecturas deliberativas o basadas en lógica

Las arquitecturas de agentes deliberativos suelen basarse en la teoría clásica de planificación de inteligencia artificial: dado un estado inicial, un conjunto de operadores/planes y un estado objetivo, la deliberación del agente consiste en determinar qué pasos debe encadenar para lograr su objetivo, siguiendo un enfoque descendente (top-down).

Podemos distinguir los siguientes tipos principales de arquitecturas deliberativas o simbólicas: arquitecturas intencionales y arquitecturas sociales.

Las agentes intencionales se distinguen por ser capaces de razonar sobre sus creencias e intenciones. Se pueden considerar como sistemas de planificación que incluyen creencias e intenciones en sus planes.

Los agentes sociales se pueden definir como agentes intencionales que mantienen además un modelo explícito de otros agentes y son capaces de razonar sobre estos modelos.

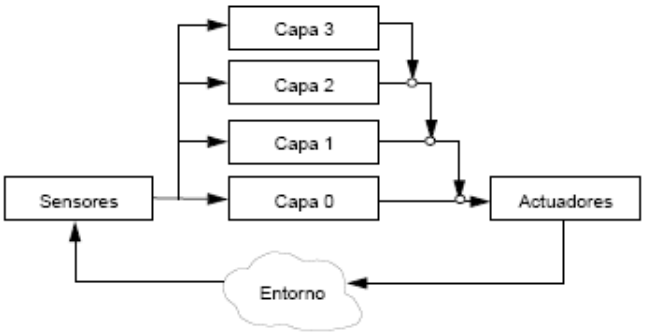

Figura 2.3: Ejemplo de arquitectura reactiva [IGL]

Dentro de las arquitecturas intencionales, cabe destacar aquellas que han tomado como punto de partida la teoría de agentes BDI (2.2) en su implementación, representando explícitamente las actitudes intencionales de los agentes. Estos sistemas también suelen utilizar planificación para determinar qué acciones deben llevar a cabo pero, a diferencia de los agentes planificadores, emplean planes en que se comprueban creencias, deseos e intenciones. Las creencias son el conocimiento que el agente tiene sobre sí mismo y su entorno. Los deseos son objetivos del agente a largo plazo. Como normalmente no puede cumplir todos los objetivos a la vez, ya que tiene unos recursos limitados, se introducen las intenciones, que son los objetivos que en cada momento intenta cumplir el agente. Normalmente también se introduce el concepto de planes, que permiten definir las intenciones como los planes que un agente está realizando en un momento dado.

Los agentes sociales pueden clasificarse en dos grandes grupos: agentes intencionales cuya arquitectura ha sido aumentada para abordar el razonamiento sobre otros agentes, y arquitecturas que siguiendo la IAD clásica han prestado más atención a los aspectos cooperativos (cuándo, cómo y con quién cooperar), sin modelar necesariamente las intenciones de los agentes.

Las arquitecturas deliberativas pueden clasificarse como horizontales porque los estímulos recibidos del exterior son procesados en varias capas de diferente nivel de abstracción y al final el nivel superior decide qué acciones hay que llevar a cabo (y las realiza directamente o se lo indica a las capas inferiores).

Arquitecturas reactivas

Las arquitecturas reactivas cuestionan la viabilidad del paradigma simbólico y proponen una arquitectura basada en un modelo estímulo-respuesta. Las arquitecturas reactivas no tienen un modelo del mundo simbólico como elemento central de razonamiento y no utilizan razonamiento simbólico complejo, sino que siguen un procesamiento ascendente (bottom up), para lo cual mantienen una serie de patrones que se activan bajo ciertas condiciones de los sensores y tienen un efecto directo en los actuadores. Esta discusión entre mantener una representación explícita del modelo o no, no es una discusión específica del campo de agente sino de la inteligencia artificial en general, de hecho las primeras arquitecturas de agentes reactivos se basan en los planificadores reactivos.

Las arquitecturas reactivas pueden clasificarse como verticales porque los estímulos recibidos del exterior son procesados por capas especializadas que directamente responden con acciones a dichos estímulos y pueden inhibir las capas inferiores, como se puede apreciar en 2.3.

Arquitecturas híbridas

Dado que algunos investigadores opinan que para la construcción de agentes no es del todo acertado utilizar una arquitectura totalmente deliberativa, o totalmente reactiva, se han propuesto sistemas híbridos que pretenden combinar aspectos de ambos modelos. Una primera propuesta puede ser construir un agente compuesto de dos subsistemas: uno deliberativo, que utilice un modelo simbólico y que genere planes de accion, y otro reactivo, centrado en reaccionar a los eventos que tengan lugar en el entorno y que no requiera un mecanismo de razonamiento

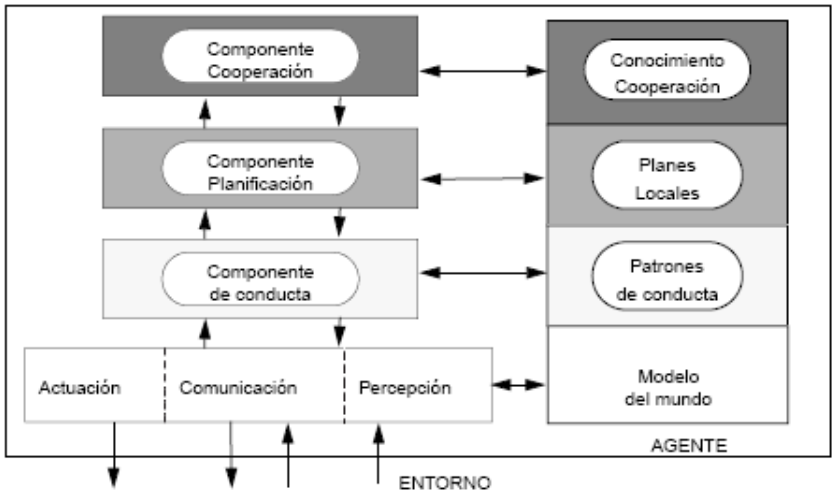

Figura 2.4: Ejemplo de arquitectura Híbrida (vertical): proyecto *Interrap* [IGL]

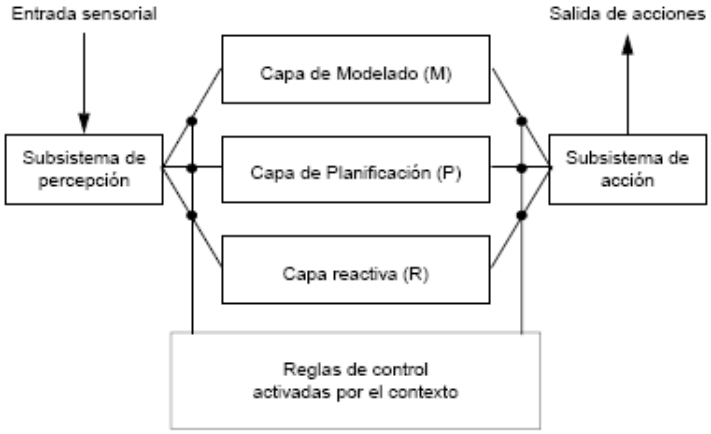

Figura 2.5: Ejemplo de arquitectura Híbrida (horizontal): proyecto *TouringMachine*[IGL]

complejo. Por su propia naturaleza, estas arquitecturas son propicias para una estructuración por capas, que puede ser o bien vertical, o bien horizontal [BERb]. En 2.4 y 2.5 se muestran ejemplos de estas arquitecturas.

C. Lenguajes de agentes

Los lenguajes de agentes se definen como lenguajes que permiten programar agentes con los términos desarrollados teoricamente. Podemos distinguir dos tipos principales de lenguajes de programación [IGL]:

- *Lenguajes de agentes de propósito general:* lenguajes destinados a programar agentes genéricos utilizables en cualquier aplicación.
- *Lenguajes de agentes específicos:* lenguajes para un tipo de agentes específicos, por ejemplo los lenguajes para agentes móviles (por ejemplo *Telescript* o *Agent-Tcl*).

2.2. Modelado de la realidad

Una característica común a todos los agentes es que perciben el estado de su entorno por medio de sensores y actúan sobre el mismo a través de efectores. Una forma sencilla de resolver esto es realizar un mapeo directo entre sus percepciones y sus acciones, llevando al agente a responder inmediatamente ante un cambio en el estado de la realidad. Cuando los agentes se desempeñan en entornos dinámicos y no deterministas, como lo es el fútbol de robots, esta solución es practicamente irrealizable. Otra forma más eficiente y sofisticada de resolver el problema es no consumir directamente los datos "crudos" de los sensores, sino realizar una interpretación de los mismos previamente. Para lograr esto se requiere almacenar y procesar los datos usando algún modelo interno de representación de la realidad. Este modelo le permite al agente guardar información acerca de como percibe el estado actual de su entorno el cual puede ser actualizado de dos formas: por el procesamiento de nueva información sensorial o por la predicción de acciones en base al último estado conocido. El principal cometido del modelado de la realidad es proveerle al agente métodos de alto nivel que le permitan razonar el estado de su entorno y así poder decidir sus acciones de la mejor forma posible. Por estas razones, el modelo de la realidad es considerado como uno de los componentes más importantes de la arquitectura de un agente.

Un agente de fútbol de robots requiere conocer la posición y velocidad de sí mismo como así también la de los restantes jugadores y de la pelota. En el caso de la liga de simulación este tipo de información es provista a los agentes por el simulador de juego a través del envío de mensajes con información sensorial acerca del estado de la realidad (física, auditiva y visual). Además de contener información sensorial, los mensajes contienen una "marca" de tiempo que indica el momento en que se originó la información. Guardando esta marca de tiempo junto con la información sensorial el agente puede determinar que tan confiable (actualizado) es el contenido del modelo de la realidad, dato por demás relevante al momento de elegir que acción realizar. Un aspecto importante a destacar acerca de la información visual provista por el simulador es que la misma viene expresada en coordenadas relativas a la posición del agente. Se requiere por parte del agente transformar estas coordenadas relativas a coordenadas globales al campo de juego con la ayuda de objetos fijos que se encuentran dentro del mismo (lineas, flags, arcos, etc) de las cuales el agente conoce sus coordenadas. Se debe resaltar que esta tarea es inevitable ya que si el agente guardara en el modelo la información con coordenadas relativas a su posición, las mismas quedarían obsoletas ante el primer movimiento del agente luego de haberlas almacenado.

En lo que resta de esta sección se explicarán algunos de los aspectos más relevantes a tener en cuenta en el modelado de la realidad para un equipo de fútbol de la liga simulada. Para esto se tomará como base lo realizado por el equipo UvA Trilearn [DEBa].

2.2.1. Actualización del modelo

Para cada objeto dinámico del campo de juego es guardada una estimación de su posición y velocidad (entre otras cosas) junto con un valor de confianza que indica la fiabilidad de esa estimación. Este valor de confianza es derivado directamente de la marca de tiempo recibida en cada mensaje sensorial proveniente del servidor. El modelo de la realidad es actualizado siempre que el agente recibe nueva información sensorial. Para los objetos que el agente ha visto durante el último ciclo el modelo es actualizado con información de alta confianza. Para el resto de los objetos, el agente realiza una estimación de cual sería su ubicación y velocidad actual en base a la última información recibida del mismo y actualiza el modelo junto con una disminución de su valor de confianza. De esta forma, el agente podrá distinguir entre información proveniente del servidor (actualizada) e información estimada por el propio agente. Además de estas dos formas de actualizar el modelo, los agentes generalmente se comunican entre sí para informarse de la ubicación de los objetos, principalmente la pelota.

2.2.2. Interacción con el modelo

Con el objetivo de ayudar al agente en la resolución del probema de elegir la mejor acción posible, el modelo de la realidad debe proveerle métodos de alto nivel que le permitan interactuar con el mismo. Existen cuatro tipos de métodos que idealmente un modelo debería ofrecer:

- *Métodos de Recuperación*. Le permiten al agente obtener la información de los objetos (coordenadas, velocidad, etc.).
- *Métodos de Actualización*. Le permiten al agente actualizar el modelo en base a nueva información sensorial.

- *Métodos de Predicción*. Le permiten al agente predecir el futuro estado del modelo en base a percepciones pasadas.
- *Métodos de Alto Nivel*. Le permiten al agente sacar conclusiones de alto nivel en base a la información almacenada en el modelo.

En [DEBa] se presenta una discución detallada acerca de como implementar estos métodos.

2.2.3. Atributos del modelo

Los atributos pueden ser considerados como bloques de información que en su conjunto conforman el modelo de la realidad y le permiten al agente razonar la mejor acción a realizar en cada momento. El valor de estos atributos representan el estado actual de la realidad y son actualizados cuando el agente recibe nueva información sensorial. El equipo UvA Trilearn propone una larga lista de atributos los cuales pueden ser agrupados en cuatro categorías: información del entorno, información del partido, información de los objetos e información de las acciones.

Información del entorno

El modelo de la realidad almacena una serie de atributos que contienen información especifica acerca del entorno de simulación que son provistos por el simulador de fútbol. Estos atributos representan los parametros del simulador y los parámetros de los tipos de jugadores. Una lista detallada de ellos puede encontrarse en **[CHE]**.

Información del partido

El modelo de la realidad almacena un conjunto de atributos que contienen información general acerca del estado actual del juego. Los atributos son:

- *Tiempo*. Representa el tiempo del servidor (por ejemplo, el ciclo actual del partido).
- *Número de Jugador*. Representa el número de camiseta del jugador. Su valor es fijo durante todo el partido. Este atributo sirve para identificar al jugador.
- *Lado de juego*. Representa el lado del campo de juego (derecha o izquierda) donde el equipo esta jugando.
- *Nombre del equipo*. Representa el nombre de equipo del jugador. Este atributo le permite al jugador saber si la información que recibe pertenece a un compañero o a un contrario.
- *Modo de juego*. Representa el modo de juego actual del partido (sección 3.5).
- *Tanteador del partido*. Representa la diferencia de goles en el partido. Si es 0 significa que estan empatados, si el valor es positivo significa que el equipo va ganando y si es negativo significa que va ganando el oponente.

Información de los objetos

El modelo de la realidad almacena información acerca de todos los objetos del campo de juego. Estos objetos pueden ser divididos en *estacionarios*, por tener una ubicación fija dentro del campo de juego (flags, lineas,etc) y *dinámicos*, que se mueven en el campo de juego (jugadores y pelota). Como se mencionó anteriormente la ubicación y velocidad de los objetos son almacenadas en coordenadas globales y se utilizan los objetos *estacionarios* para el cálculo de las mismas. En la figura 2.6 se muestra un diagrama UML de la jerarquía de objetos utilizado por el equipo Uva Trilearn para el modelado de la realidad.

El modelo utiliza el numero de camiseta de los jugadores para identificarlos. Uno de los problemas importantes a resolver es cuando se tiene nueva información sobre un jugador pero la distancia que existe a él es tal que no permite saber cual es su numero de camiseta. Esto ocasiona que al momento de actualizar el modelo no se sepa a cual jugador asignarle esta nueva información. En estas ocaciones, la última posición y velocidad conocidas de cada jugador son utilizadas para resolver el problema de ambigüedad. Utilizando los métodos de predicción que ofrece el modelo, se puede estimar la nueva ubicación de cada jugador para encontrar uno que concuerde con la nueva información. En caso de no haber ningún jugador que concuerde con esta información (supera cierto margen de error) se elimina del modelo al jugador que tiene menor valor de confianza en su ubicación y se da ingreso a un nuevo jugador sin identificación. Si bien esto ocasiona que el agente no pueda identificar a ese jugador, le permite saber que hay un objeto en esa posición.

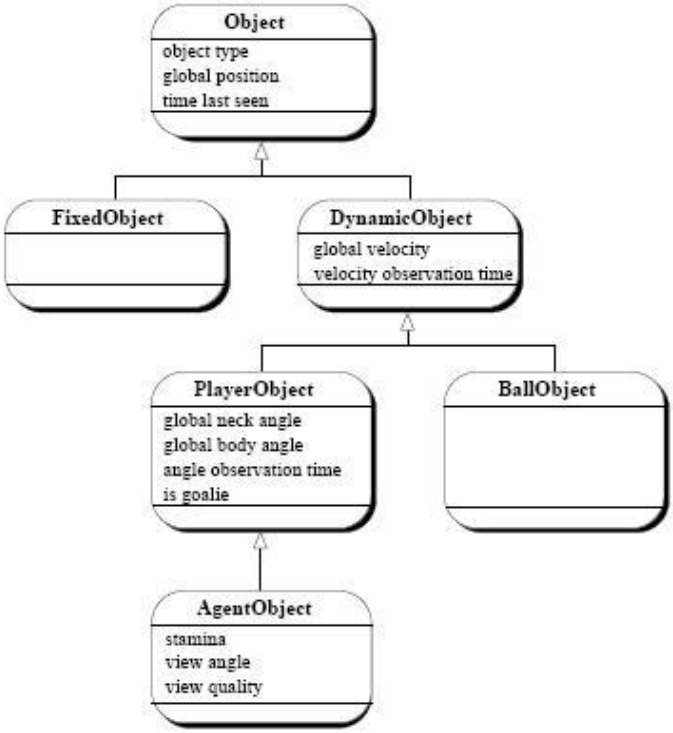

Figura 2.6: Jerarquía de objetos del modelo de realidad del equipo UvA Trilearn [DEBa]

Información de las acciones

El modelo de la realidad almacena atributos que contienen información acerca de las acciones que el agente a realizado previamente. Los atributos son:

- *Nro. de patadas, Nro. de corridas, Nro. de giros, Nro. de mensajes enviados, Nro. de giros de cuello, Nro. de atrapadas, Nro. de movimientos, Nro. cambio de vistas*. Estos atributos contienen el número total de cada acción que el agente ha realizado en lo que va del partido. Los valores de estos atributos son obtenidos directamente de los mensajes enviados por el servidor. Esta información sirve para conocer si una acción enviada al servidor ha sido ejecutada o descartada. Por ejemplo, si antes de enviar un comando kick el contador estaba en 10 y luego de pasar el ciclo el contador aumento a 11 significa que la acción fue ejecutada, si el contador sigue con el valor 10 la acción fue descartada.
- *Cola de acciones*. Representa una lista con las acciones que el agente envió al servidor en el ciclo previo.
- *Acciones realizadas*. Representa una lista de valores booleanos que indica para cada acción de la *Cola de acciones* si fue realizada o no. La lista de *Acciones realizadas* le sirve al agente para saber que acciones fueron ejecutadas y así poder actualizar adecuadamente el modelo de la realidad.

2.3. Habilidades de los jugadores

Las habilidades de los jugadores pueden llegar a ser muchas y con distintos niveles de complejidad. Por este motivo se trata de clasificarlas en grupos jerárquicos donde los niveles más altos utilicen los servicios de los más bajos. Una división general que plantea [CAS] las divide en: acciones, comportamientos y jugadas.

2.3.1. Acciones

Las acciones son las de nivel de complejidad más bajo. Se pueden considerar como las acciones básicas o atómicas en un dominio genérico. En ambientes de fútbol simulado, estas acciones son soportadas por el propio simulador y no es necesario desarrollarlas. Para el caso de robots físicos, las acciones pueden ser de gran complejidad dependiendo del tipo de robots. Mostraremos a continuación algunas habilidades que podrían clasificarse como acciones junto con sus características en los robots físicos:

Mover o Avanzar Refiere a desplazar al robot en una determinada dirección y sentido. Esta acción (que ya viene implementada dentro de las ligas simuladas) puede ser muy compleja de realizar en los robots físicos. El caso más complejo es el de los robots bípedos ó humanoides debido a los cálculos necesarios para mantener el equilibro, y la sincronización para modelar las articulaciones del mismo. Otros robots, como los cuadrúpedos o con ruedas, requieren también de sistemas complejos que coordinen un mismo desplazamiento para cada pata (o rueda) de forma de obtener el movimiento global esperado.

Impulsar Es la base para el pateo de la pelota. Dependiendo del tipo de robot y las restricciones de la liga donde se desempeñen, los robots pueden contar con dispositivos específicos para golpear la pelota ó la impulsan golpeando con su propio cuerpo. Para los casos donde se dispone de dispositivos para golpear la pelota, estos se construyen en base a resortes, impulsores que se activan nuematicamente con el contacto o mediante aire, etc. En algunos casos los robots poseen en el frente elementos estáticos con los que al golpear generan efectos sobre la pelota. Por otro lado, los robots cuadrúpedos deben utilizar las patas delanteras para impulsar la pelota.

Girar La acción de girar implica que el robots gire sobre su propio eje para cambiar su orientación. Este es otro caso donde el soporte del robot define la complejidad y tipo de problema a resolver. Para el caso de robots con ruedas multi-direccionales el problema puede ser sencillo. Por otro lado, la peor parte la llevan los humanoides y cuadrúpedos donde el giro requiere de una secuencia de movimientos.

Atrapar Esta acción es permitida solamente para los arqueros. Dependiendo de las restricciones de la liga, los robots poseen algún dispositvo para retener la pelota durante algún tiempo.

Rotar percepción Esta acción es relevante en las ligas con visión local. Consiste en cambiar el ángulo de percepción sin cambiar la dirección del cuerpo. En los robots físicos, se relaciona con el manejo de las cámaras móviles que llevan los robots.

2.3.2. Comportamientos

Según se presenta en [CAS], los comportamientos se pueden clasificar en elementales y complejos. Los elementales son los que representan el primer contacto con el entorno y sirven como base para realizar comportamientos más complejos. Los comportamientos complejos son jugadas básicas, las mismas son descriptas en la próxima sección.

Comportamientos Elementales

Para la implementación de comportamientos elementales se utilizan distintos tipos de técnicas. Algunas simplemente codifican y determinan las opciones, en otros casos se utilizan aproximaciones, algoritmos de optimización y también son comunes las técnicas de aprendizaje. Para desarrollar estos comportamientos, los robots simulados poseen el soporte brindado por el simulador y los físicos las acciones antes descriptas. El objetivo fundamental para la selección de comportamientos elementales, es que sean lo suficientemente genéricos como para ser utilizados por varios comportamientos complejos.

Navegar A diferencia de la acción mover, navegar es moverse de un lugar a otro teniendo en cuenta el entorno. El destino se puede considerar como estático o dinámico.

Para navegar se utiliza una secuencia de acciones de avance y giro. Una consideración importante son las colisiones, si bien dentro de un ambiente simulado no es tan costoso una colisión, en los robots físicos podría dejar al robot sin participación en el juego durante un tiempo. También es importante el ángulo de llegada al destino, ya que en algunas ligas el jugador solo puede patear la pelota en dirección a su cuerpo.

Para este comportamiento se cuenta en general con dos tipos de algoritmos. El primero, son los algoritmos de planificación de trayectorias, en los cuales antes de comenzar se realiza una planificación total de la trayectoria, y luego se pasa a su ejecución. Este tipo de algoritmos utilizados en ambientes dinámicos como el fútbol requieren de recálculos de la planificación inicial. Por problemas de performance en general se utilizan mecanismos de optimización donde la planificación de un paso es rehusada en recálculos posteriores ó simplemente se hacen planificaciones parciales.

El segundo grupo de algoritmos, son los llamados métodos heurísticos. Sobre la base de heurísticas particulares, estos algoritmos deciden en forma independiente el próximo paso a realizar en cada momento. Esta opción es mucho mas performante pero puede generar que las decisiones en cada paso lleven al robot a un camino sin salida. Dentro de la liga simulada es común la utilización de heurísticas para determinar los parámetros (ángulo y potencia) del comando *dash* a través del cual el agente se desplaza.

Patear Consiste en impulsar la pelota hacia un determinado lugar, generalmente el arco contrario o hacia un compañero. Para las ligas que consideran el pateo como colisionar la pelota, se puede dejar la responsabilidad del pateo a la navegación. Otra forma es que la navegación llegue hasta una distancia próxima a la pelota y desde ahí se encargue la técnica de pateo, buscando la velocidad y ángulo necesario para impulsar la pelota al punto deseado.

Es común la utilización de técnicas de aprendizaje (algoritmos genéticos, redes neuronales, etc.) para determinar los parámetros necesarios para patear la pelota (el ángulo y la potencia de pateo para enviar la pelota a un punto determinado). Esto se debe a la gran dificultad de calculo que requieren los parámetros, incluso en la liga simulada, el software de simulación incorpora ruido haciendo más dificíl la tarea.

Rastrear un objeto Consiste en seguir el movimiento de un objeto sin moverse del lugar de manera de no perderlo de vista. Este comportamiento es más común en arqueros, ya que busca conocer siempre la posición de la pelota de forma de estar preparado si se le efectúa un tiro al arco. Su implementación también se puede aplicar en comportamientos complejos donde se quiere marcar un jugador contrario o mantener referencia a algún objeto del campo.

Para las ligas con visión global la tarea no parece difícil, de todos modos hay que contemplar los casos que la información es confusa y por algún momento se pierde la posición de algún objeto del campo. En estos casos se recurre a información vieja y a la simulación del comportamiento del oponente para estimar su posición actual.

Para las ligas con visión local, su implementación es mas compleja dado que el jugador solo tiene información de una parte del campo de juego. Por este motivo, aquí entran en juego la cooperación entre agentes y el modelado del mundo. Este comportamiento incluye la acción de *Rotar Percepción*, la cual es utilizada para conocer el entorno del jugador.

Comportamientos Complejos

Los comportamientos complejos son la unidades básicas ó atómicas de las jugadas. Las jugadas se presentan en la siguiente sección.

Interceptar la pelota. Consiste en desplazarse de forma de colisionar la pelota en movimiento. Este comportamiento es fundamental en varias instancias del juego. Para su realización es necesario utilizar todos los comportamientos básicos descriptos anteriormente: navegar, rastrear un objeto y patear. Para interceptar la pelota debemos determinar la trayectoria del jugador para que pase por un punto en la trayectoria de la pelota en un mismo momento.

Luego de encontrado el punto de intersección, el jugador debe desplazarse hacia el mismo con cierta velocidad y un ángulo determinado (depende del tipo de pateo soportado). Las formas de cálculo del punto de intersección

son muy variadas, se utiliza simulación para determinar estados futuros del juego, optimización y mecanismos de aprendizaje como redes neuronales entre otros.

Bloquear la pelota Consiste en ubicarse en algún punto futuro de la trayectoria de la pelota. Este comportamiento lo podemos considerar como un caso particular del anterior. La diferencia básica es que no lleva pateo.

Posicionarse Consiste en elegir y posicionarse en un punto determinado de la cancha. Este comportamiento se lo puede ver como sencillo pero es uno de los más complejos y estratégico para el equipo. Se basa en que los jugadores tomen posiciones estratégicas del campo que puedan servir en momentos futuros. En la sección 4 veremos buenas implementaciones de este comportamiento. Algunos aspectos que se deben considerar para elegir una posición en el campo de juego son:

- Permanecer cerca de la posición origen del agente.
- Permanecer dentro del campo de juego.
- Evitar quedar en posición adelantada.
- Posicionarse donde sea posible un pase.
- Posicionarse bloqueando el pase a un oponente.
- Seguir la formación del equipo.

Llevar la pelota Consiste en desplazarse de un lugar a otro sin perder control sobre la pelota (durante el documento también se utiliza el término *dribliar* del inglés). Este comportamiento requiere de los tres comportamientos elementales (patear, navegar y rastrear) y su implementación depende sustancialmente del tipo de robot.

Para robots con caras cilíndricas la tarea les será más difícil, en cambio lo que tienen caras planas o cóncavas podrán valerse de ellas para su realización. Existen ligas donde los robots tienen dispositivos donde retiene la pelota para desplazarse junto a ella. Para el caso de liga simulada de RoboCup, se suelen utilizar pateos cortos y desplazamientos que acompañen el movimiento de la pelota.

Perseguir un objeto Consiste en mantenerse cerca de un objeto móvil conservando la distancia. Este comportamiento esta relacionado con la marca a jugadores oponentes. A través del rastreo del objeto se intenta mantener, buscar y navegar hacia una posición cercana del mismo evitando colisionarlo.

2.3.3. Jugadas

Las jugadas son comportamientos de alto nivel del robot o agente. Las mismas son ejecutadas por el robot a través de algún mecanismo de selección que determine cual es la indicada para ejecutar en cada momento. Si bien se destacan algunas, toda acción asociada al fútbol que desee realizar un jugador en determinado momento se puede considerar como una jugada. Describimos a continuación las que consideramos mas relevantes:

Atajar Es la acción del arquero tendiente a evitar que la pelota ingrese al arco. Para las ligas que lo permiten, el arquero puede retener la pelota por algunos segundos (en estos casos los robots físicos poseen un soporte físico para su implementación). En la liga simulada de RoboCup esto se permite siempre y cuando la pelota se encuentre dentro de una región próxima al arquero. Para los casos que la retención no se permite, la acción se implementa a través de interceptar y bloquear la pelota. En general los arqueros utilizan también el comportamiento de rastrear un objeto de forma de estar preparados para un eventual tiro al arco.

Despejar la pelota Esta jugada consiste en impulsar la pelota fuera del área defensiva reduciendo el peligro para el arco propio. Es utilizada cuando un defensa posee la pelota y es marcado por jugadores oponentes. Frente a la posibilidad de perder la pelota y que el contrario quede en posición de gol, es preferible patear la pelota lejos (donde no exista peligro). Se utilizan comportamientos de pateo de la pelota y de posicionamiento.

Para realizar esta jugada se deben considerar entre otras cosas:

- Las posiciones de los jugadores contrarios: la posibilidad (ó probabilidad) de los contrarios de interceptar el despeje de la pelota.
- La posición del jugador que posee la pelota: Reglas que determinen que lugares del campo pueden generar más peligro al momento de despejar la pelota.
- Las posiciones de los compañeros.

Patear al arco Esta jugada intentar convertir un gol y se basa en el comportamiento de pateo de la pelota. Primeramente es necesario determinar si la mejor opción es patear al acro o pasar la pelota a un compañero que pueda quedar con mayores posibilidades de convertir el gol. Si la opción elegida es la de patear al arco, se debe determinar el punto del arco donde sea mas probable la concreción del gol. Para realizar esta jugada se deben considerar entre otras cosas:

- La posición del arquero.
- Las posiciones de jugadores contrarios: la posibilidad que tengan los mismos de bloquear la pelota.
- Las posiciones de los compañeros: para determinar si es mejor hacer un pase.

Existen técnicas de aprendizaje on line que modelan el comportamiento del oponente enfocadas en el arquero. También se utilizan técnicas de aprendizaje off line si existen datos pasados del arquero que vamos a enfrentar. Y aunque no sea el mismo arquero, con las técnicas de aprendizaje off line podemos obtener heurísticas para decidir donde conviene patear para convertir el gol.

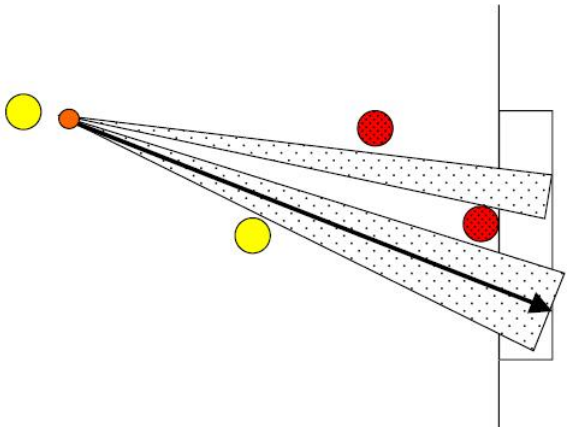

Figura 2.7: Técnica para seleccionar la dirección de tiro al momento de patear al arco [CAS].

Otra buena técnica muy sencilla de implementar, es la que vemos en la figura 2.7. Se elige el mayor ángulo libre entre la pelota y el arco y, si hay espacio suficiente para que pase la pelota, se establece su bisectriz como dirección del tiro.

Realizar y recibir un pase Estas dos jugadas las tratamos juntas debido que a tienen gran dependencia. Realizar un pase consiste en enviar la pelota a una posición donde, potencialmente, quede bajo el control de un robot compañero. La jugada de recibir un pase, consiste en posicionarse para recibir un pase de un compañero.

Para la realización de un pase es necesario un alto nivel de coordinación. El jugador que posee la pelota debe predecir el comportamiento futuro de su compañero; a su vez, el que espera el pase también debe predecir el lugar donde su compañero va a enviar la pelota para correr a su encuentro. De forma de hacerlo más sencillo, el jugador que recibe la pelota podría empezar a correr luego de conocer la dirección de la pelota, el problema es cuando el tiempo que demora el jugador en determinar la dirección hace que la acción de interceptarla sea tardía. Los mecanismos pueden diferir según si existe o no comunicación entre los agentes. También existen técnicas en las que un agente puede predecir el comportamiento de otro agente “poniéndose en su lugar”. Vemos un ejemplo de estas jugadas en la figura 2.8.

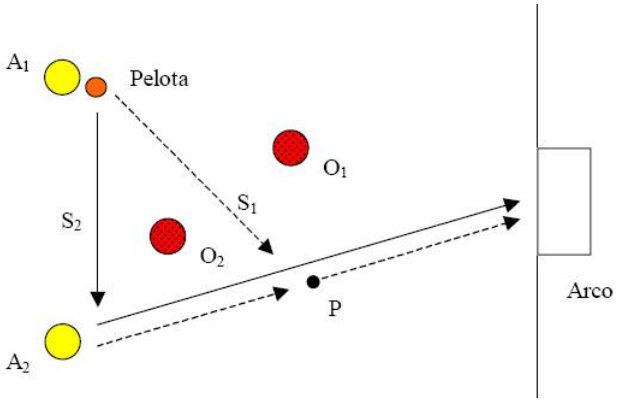

Figura 2.8: Jugada de pase de la pelota [CAS].

Marcar un jugador Esta jugada consiste en ubicarse cerca de otro jugador de forma de anticiparse a sus movimientos y evitar que tome control de la pelota, o que la patee al arco propio. Se puede implementar esta jugada con comportamientos de posicionamiento y persecución. Una técnica sencilla es ubicarse en la intersección de la pelota y el jugador marcado. En momentos donde la pelota se encuentra lejos, se pueden utilizar técnicas de modelado del oponente para determinar la formación del equipo contrario y ubicarse en función de estas.

Liberarse de una marca Esta jugada consiste en ubicarse de forma que los defensores del equipo contrario no interfieran en posibles recepciones de la pelota o de tiros al arco. Es muy utilizada por los atacantes, usa comportamientos de posicionamiento e involucra la jugada *Recibir un pase*. La idea es alejarse de los jugadores contrarios para ser un buen candidato para recibir un pase.

2.4. Estrategia de Equipo

La estrategia de un equipo de fútbol de robots puede definirse como el plan global y colectivo que siguen sus jugadores durante el partido con el objetivo de ganarle a su adversario. Algunos de los principales aspectos que deben ser tenidos en cuenta en la estrategia de un equipo son: su formación dentro del campo de juego, los roles de cada jugador, las formas de comunicación entre los jugadores, las jugadas preparadas, los comportamientos individuales de cada jugador, el dinamismo de la estrategia, etc. Además de las características de juego mencionadas anteriormente, en el fútbol de robots se deben tener en cuenta las restricciones que impone el ambiente en donde se desarrolla el juego. Para el caso de la liga Simulación de fútbol de RoboCup estas restricciones son dadas por el simulador de juego y por las propias reglas de la liga. En lo que resta de esta sección se dará una breve descripción de los principales problemas que deben ser tenidos en cuenta en la definición de la estrategia de un equipo del fútbol de robots, centrándonos principalmente en las características y restricciones que se tienen en la liga de Simulación de RoboCup, competencia 2D.

2.4.1. Características del juego

Formación del equipo

Una de las principales características de la estrategia de un equipo de fútbol es su actitud dentro del campo de juego. Se puede tener un juego *ofensivo*, en donde el equipo trata de que el partido se juegue más cerca del arco rival teniendo como principal objetivo anotar goles; o un juego *defensivo*, en donde el equipo tiene como principal objetivo lograr una defensa sólida para evitar recibir goles en contra. En este sentido, la formación de un equipo esta directamente relacionado con la actitud del mismo dentro del campo de juego.

La formación de un equipo se define como la distribución de sus jugadores dentro del campo de juego. Dependiendo de la cantidad de jugadores que el equipo tenga en cada campo (propio o rival) se define el estilo de juego del equipo (ofensivo, defensivo, conservador, etc.). Uno de los principales aspectos a tener en cuenta en la formación de un equipo es si ésta es *dinámica* o *estática*. Si el equipo cuenta con una formación dinámica, tendrá la posibilidad de adaptar su forma de juego según las circunstancias del partido, en caso contrario el equipo tendrá

el mismo estilo de juego durante todo el partido sin la posibilidad de adaptarse ante la evolución del partido. En algunas ligas de fútbol de robots los equipos tienen la posibilidad de parar el partido y hacer ajustes en la formación del equipo (en el caso de la liga de Simulación de RoboCup esto no es posible). Por esta razón, contar con este tipo de dinamismo en la estrategia es un aspecto por demás relevante.

Roles de los Jugadores

Otro aspecto relevante en la estrategia y que se encuentra relacionado directamente con la formación del equipo es la definición de roles para los jugadores. Esta idea es tomada del fútbol tradicional en donde típicamente existen 4 roles claramente diferenciados: *Arquero, Defensores, Medio campistas y Delanteros*. Dentro del fútbol de robots, esta división permite una disminución en la complejidad de desarrollo de los agentes. Cada rol tiene distintos objetivos y pueden utilizar distintas técnicas o métodos para lograrlos. Por ejemplo, los delanteros pueden tener un método avanzado para realizar tiros al arco o triangulaciones con sus compañeros con el objetivo de eludir a los contarios. Los defensas por el contrario, pueden estar interesados en despejar la pelota hacia una parte del campo de juego en donde se asegure que el rival no va a tener la posibilidad de recuperarla. Es importante que los jugadores tengan la posibilidad de intercambio o reasignación de roles durante un partido dependiendo del estado del juego. Por ejemplo, un equipo puede tener más o menos defensas dependiendo de si posee la pelota o no, o si el partido se esta jugando más en su campo que en el del rival.

Modelado del Oponente

Una buena estrategia no puede dejar de tener en cuenta la forma de juego del equipo contrario. El modelado del oponente es uno de los aspectos más importantes a tener en cuenta al momento de evaluar un posible cambio en la estrategia. Teniendo en cuenta la forma de juego del equipo contrario, la estrategia de nuestro equipo puede variar en busca de explotar los defectos del adversario o contrarrestar sus virtudes. Si el oponente ataca con muchos jugadores es posible que se requiera más defensas, si ataca siempre por uno de los laterales se debería reforzar dicho sector, si defiende con pocos jugadores se podría aumentar la cantidad de atacantes y así se podría seguir enumerando diversas situaciones en donde un ajuste en la estrategia del equipo sería de gran utilidad para maximizar las posibilidades de una victoria.

Coach

Hasta ahora se ha hablado de cuales son los aspectos a tener en cuenta para la estrategia de un equipo pero nada se ha hablado acerca de quien toma las decisiones de cambiarla o ajustarla según las circunstancias del partido. Dependiendo de la liga en la que se este jugando es como ser resuelve este problema. Para el caso de la liga de Simulación 2D de la RoboCup existe la figura del *Coach*. Este es un agente especial que tiene la posibilidad de visualizar en todo momento la totalidad del campo de juego y tiene como principal objetivo dar asistencia a sus jugadores en busca de mejorar la forma de juego y así ganar el partido. Todos los aspectos que se han mencionado acerca de la estrategia tienen como principal restricción que se necesita la visión global de partido para tomar decisiones al respecto. El coach es el único agente del equipo que tiene la posibilidad de esto por lo que es el responsable de analizar la forma de juego de su equipo (y del contrario) como para poder tomar decisiones que impliquen un cambio en la estrategia.

Para realizar una variante en la estrategia del equipo generalmente el coach se basa en algunos de los siguientes elementos[2]:

- **Resultado parcial del partido:** una diferencia importante, tanto positiva como negativa, puede inducir a cambiar el tipo de estrategia del equipo.
- **Tiempo de juego restante:** en muchas ocasiones este dato se complementa con el anterior. Por ejemplo, si el resultado es favorable y resta poco tiempo de juego es posible que se adopte un estilo de juego defensivo y lento. En cambio, si resta poco tiempo y el equipo está perdiendo el partido, es muy probable que el equipo se adelante para conseguir goles.
- **Estrategia del equipo oponente:** el equipo puede contar con información almacenada sobre el estilo de juego de sus oponente y variar su estrategia consultando estos datos. También puede contar con mecanismos para realizar un estudio *on line* de su rival y actuar en consecuencia.

[2]Estos elementos fueron tomados directamente de [CAS].

- **Información estadística:** también se puede optar por la alternativa de recopilar información propia y de los equipos rivales para generar una base de información estadística a ser considerada durante la toma de decisiones. Esta base puede ser creada previamente y actualizada en forma *on line* durante el trascurso del partido.

2.4.2. Características del ambiente

En esta sección hablaremos acerca de las principales características del entorno en donde se desarrollan los partidos de la liga de Simulación 2D de la RoboCup y como éstas influyen en la estrategia del equipo.

Arquitectura descentralizada

Los equipos de esta liga son sistemas multi-agente en donde cada jugador es un agente autónomo que decide y actúa en base a las percepciones que tiene sobre su entorno. Esta característica afecta directamente en el diseño de la estrategia del equipo. Para lograr que los 11 jugadores de una equipo actúen cooperativamente con un objetivo en común (ganar el partido) se requiere que cada agente tenga la capacidad de predecir el comportamiento de sus compañeros y en base a esto tomar decisiones que le permita interactuar "inteligentemente" con ellos. Una de las forma de lograr esto es la utilización de técnicas de IA y mecanismos de comunicación que permitan la coordinación de todo el equipo.

Visión limitada

La mayor parte del comportamiento de una agente está dirigido por las percepciones (visuales, auditivas y físicas) que éste tiene sobre el entorno donde actúa. Una de las restricciones que tiene la liga de Simulación de fútbol de RoboCup es que los agentes (jugadores) solo pueden visualizar parte de la realidad del estado del juego (sección 3.5). Esto ocasiona que el agente tenga una visibilidad reducida del campo de juego, dificultando la tarea de tomar decisiones. Generalmente, se trata de dotar a los agentes con mecanismos que le permitan construir la situación global del juego. Una forma de lograrlo es mediante el modelado probabilistico de la realidad en base a las percepciones recibidas en el pasado como se mencionó en la sección 2.2. Teniendo la ubicación anterior de un objeto (jugador o pelota) junto con su velocidad y dirección en ese momento, el agente puede predecir su nueva ubicación. Otra forma que se tiene para rearmar la situación del estado del juego es por medio de mensajes entre los jugadores, informando acerca de la ubicación de los objetos (generalmente se informa sobre la ubicación de la pelota).

Comunicación

Como se mencionó anteriormente, una forma de coordinar a los jugadores de un equipo es mediante la comunicación entre ellos. En esta liga, toda la comunicación entre los agentes debe pasar por el simulador del juego. De esta forma, el simulador recepciona los mensajes de un jugador y los distribuye al resto con las siguientes restricciones [CHE]:

- existe limite en la cantidad de mensajes que un agente puede enviar
- el alcance del mensaje es limitado, no todos los jugadores llegan a recibirlo (a excepción del coach)
- lo pueden escuchar jugadores de ambos equipos

Todas estas restricciones afectan a la estrategia del equipo ya que no se puede utilizar con total libertad este potente medio de coordinación. Otro problema que existe en relación a la comunicación es que abusar del mismo puede llegar a ocasionar graves problemas de performance sobre el equipo.

Jugadores Heterogéneos

Como en el fútbol real, el simulador de la liga permite tener jugadores con distintas caracterísitcas físicas. Para esto, existen en el simluador distintos tipos de jugadores llamados *jugadores heterogéneos* en donde cada tipo tiene distintas características de resistencia, velocidad, fuerza, etc. Además, el simulador modela el cansancio de los jugadores de forma que a medida que pasa el tiempo van perdiendo su energía. Toda esta información debe ser tenida en cuenta en la estrategia del equipo ya que de otra forma puede llegar a fracasar o a no obtener los resultados

esperados. Por ejemplo, si el coach decide pasar a un juego ofensivo y sumar varios jugadores al ataque pero no cuenta con jugadores rápidos o los que están no tienen suficiente energía, seguramente no vaya a ser efectivo el estilo de juego adoptado.

Capítulo 3

La liga de Simulación de Fútbol RoboCup

3.1. Historia de la liga

En julio de este año se ha realizado el 10º campeonato oficial de la liga Simulada de RoboCup en Osaka, Japón [CROB05a] en donde se reunieron los 12 mejores equipos de todo el mundo. El campeón de este año fue el equipo *Brainstormers* de la Universidad de Dortmund, Alemania.

La primera competición de la liga tuvo lugar en el evento llamado preRoboCup-96 el cual se realizó en el año 1996 en Osaka, Japón. Esta competición tuvo como principal objetivo ser un banco de prueba para el software de simulación de la liga en preparación de lo que sería la primera competición oficial a nivel mundial de la liga en el año siguiente. En esa ocación participaron 7 equipos, 5 de los cuales era japoneses. El ganador de la competición fue el equipo japones *Ogalets* de la Universidad de Tokio. En este torneo, los equipos presentaron estrategias de juego muy simples y poco flexibles.

En 1997 se llevo a cabo la primera competición oficial en Nagoya, Japón [CROB97]. En esta competencia participarón cerca de 29 equipos (desde la fase previa) de diferentes partes del mundo. El ganador de este año fue el equipo *At-Humboldt* de la Universidad de Humboldt, Alemania. Este equipo demostro una clara superioridad en las habilidades de bajo nivel, principalmente en el pateo en donde tenía la capacidad de darle a la pelota una velocidad superior de la que podía darle cualquier otro equipo. Esta ventaja se debió a que la versión del simulador utilizada para ese campeonato no tenía implementado restricciones sobre la velocidad maxima de la pelota. Esta restricción fue incorporada al simulador al año siguiente.

Las competencias desarrolladas en los años siguientes (1998 - 2005) [CROB98, CROB99, CROB00, CROB01, CROB02, CROB03, CROB04, CROB05a] han contribuido al desafio de la RoboCup gracias a la incorporación de nuevas técnicas de IA y sistemas multi-agente que permitieron una mejora sustancial respecto a las estrategias y habilidades de los equipos. El número equipos participantes de esta liga ha ido aumentando año tras año hasta el punto que al día de hoy se superan largamente los 100 equipos en todo el mundo.

3.2. Campeonatos

3.2.1. RoboCup World Championship

Es el principal campeonato de la RoboCup que se juega a nivel mundial. Se viene realizado año tras año desde 1997 y su última edición ha sido en Julio de 2005 en Osaka, Japón [CROB05a]. En él se congregan los mejores equipos del mundo de cada una de las ligas de la RoboCup. Además de la competición, este evento sirve para exponer los avances que se han logrado en el último año dentro del área de la inteligencia artificial, robótica y sistemas multi-agente. El próximo evento será realizado en Junio de 2006 en la ciudad de Bremen, Alemania[1].

3.2.2. Simulated Soccer Internet League (SSIL)

En este campeonato solo puede participar equipos de la liga de Simulación cuyo fin es permitir que los mismos evaluen continuamente sus avances en la liga. La competencia es abierta a cualquier equipo que quiera participar,

[1] Generalmente en los años que se juegan los campeonatos mundiales de fútbol de la FIFA, el evento es llevado a cabo en el mismo país.

solo se requiere registrarse con un usuario y subir los binarios a los servidores de la liga para competir con el resto de los equipos. Generalmente se realizan unas pocas competencias por año.

La participación en esta liga es de vital importancia para los equipos que aspiran a participar de los campeonatos mundiales de la RoboCup ya que para clasificar a los mismos se toma en cuenta los resultados obtenidos en esta competencia [LSIM].

3.2.3. Otros Campeonatos

Además de las competencias mencionadas anteriormente, existen otras de nivel regional que se realizan año tas año en distintas partes del mundo:

Abierto de Alemania Este campeonato se realiza desde el año 2001 y reúne a los principales equipos de la región de todas la ligas de la RoboCup [CRAL05, CRAL01]. Resulta una campeonato bastante competitivo para la liga de Simulación ya que en él participan varios de los equipos que se destacan a nivel mundial.

Abierto de Estados Unidos Este campeonato se realiza desde el año 2003. Participan equipos de todas las ligas de la RoboCup y reúne a las principales universidades de los Estados Unidos que se destacan en el área de inteligencia artificial, robótica y sistemas multi-agente. El próximo campeonato se dará lugar en Octubre de 2005 en el Instituto Tecnológico de Georgia, Atlanta [CREU05].

Abierto de Eslovaquia Este campeonato es organizado por la Facultad de Informática de la Universidad Tecnológica de Bratislava desde el año 2000 [CRAS05]. Los equipos participantes son desarrollados enteramente por estudiantes de ésta universidad. El campeonato tiene como principal objetivo motivar a los estudiantes en la investigación y desarrollo de nuevas técnicas de inteligencia artificial. Vale la pena destacar que ninguno de los equipos de este campeonato han logrado participar en competencias oficiales de la RoboCup.

Además de los campeonatos anuales mencionados anteriormente, existen competencias que se realizan esporádicamente en distintas partes del mundo: Japón, Australia, Irán, Holanda, Brasil y el Abierto Europeo.

3.3. Reglas de la liga Simulación 2D

Para continuar con el desarrollo de la definición del problema, deben ser explicadas las reglas básicas de la liga. Las reglas son muy importantes y permiten entender la complejidad de este juego. Las mismas, por otra parte, deben ser tenidas en cuenta en las estrategias de los equipos participantes.

Durante el juego, las reglas son aplicadas por 2 tipos diferentes de jueces: el juez automático, provisto por el simulador, y el juez humano que interactúa con el simulador, corrigiendo o cambiando el estado de juego manualmente. El juez automático es encargado de aplicar las siguientes reglas: *Kick-Off*, *Goal*, *Out of Field*, *Player Clearance*, *Play-Mode Control*, *Offside*, *Back passes*, *Free Kick Fault*, *Half-Time* y *Time-up*. El juez humano es el encargado de aplicar las reglas del juego que se cumplen según su criterio sobre la intencionalidad de la jugada o del jugador en ese momento.

Reglas Aplicadas por el juez automático

Kick-Off

Al momento de iniciar o reanudar el juego, comienzo del 2^o tiempo o luego de un gol, todos los jugadores deben estar dentro de su campo de juego. Para permitir esto luego de anotado un gol, el partido se suspende durante 5 segundos para que todos los jugadores vuelvan a su campo.Si pasado estos 5 segundos existen jugadores que aún están en falta, el juez los colocará en una posición aleatoria dentro de su campo de juego.

Goal

Cuando un equipo anota un gol, el juez realiza las siguientes tareas:

- Comunica el gol a todos los jugadores

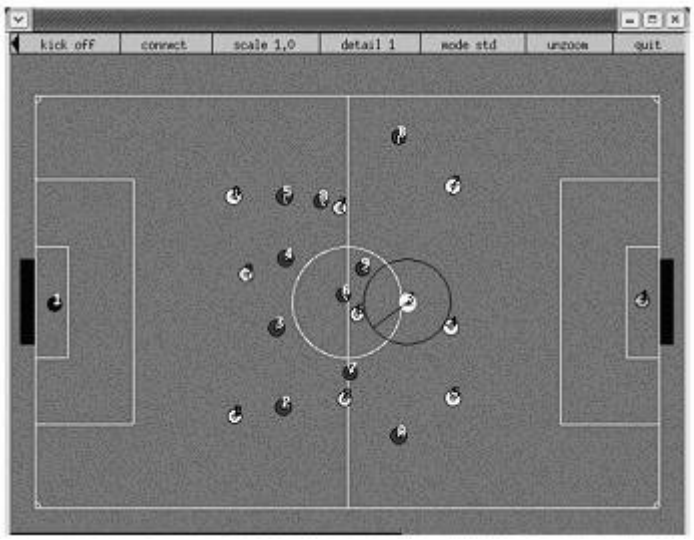

Figura 3.1: Círculo de distancia que deben mantener los jugadores ante un tiro libre (centro en la pelota y radio 9.15 mts)[WAR].

- Actualiza el tanteador del partido
- Coloca la pelota en el centro del campo
- Cambia el modo de juego a *kick_off_x*[2], donde x puede ser *r* (derecha) o *l* (izquierda) según el lado del campo donde este jugando el equipo al cual le anotaron el gol.
- Suspende el juego por 5 segundos antes de reanudarlo (*Kick-Off*)

Out of Field

Cuando la pelota sale fuera del campo de juego, el juez la coloca en la posición adecuada (líneas laterales, corner o saque de arco) según la parte del campo por la que haya salido. Inmediatamente después, cambia el modo de partido a *kick_in*[3], *corner_kick*[4] o *goal_kick*[5].

Player Clearance

Esta regla es utilizada para marcar una distancia entre los jugadores contrarios y la pelota cuando se va a ser efectivo un tiro con pelota parada.

Esta regla se aplica cuando el juego se encuentra en modo *kick_off*, *free_kick*[6], *kick_in*, o *corner_kick*. Cuando el juego se encuentra en alguno de estos modos, el juez reubica automáticamente a todos los jugadores del equipo contrario que se encuentren dentro del circulo centrado sobre la pelota y con radio de 9.15 metros (este parámetro puede ser modificado en el simulador).

Cuando el juego se encuentra en modo *offside*[7], todos los jugadores contrarios que se encuentren dentro del área del offside, o que estén dentro del circulo centrado en la pelota y con radio 9.15 metros, serán reubicados automáticamente por el juez.

Cuando el juego se encuentra en modo *goal_kick*, todos los jugadores del equipo contrario son movidos fuera del área penal. Estos jugadores no pueden volver a entrar al área mientras no se cambie el modo. El modo cambia a *play_on*[8] inmediatamente después que la pelota sale del área penal.

[2]Comienzo o reanudación del juego por parte del equipo *x*.
[3]Saque desde el lateral.
[4]Tiro desde la esquina (corner).
[5]Saque desde el arco.
[6]Tiro libre.
[7]fuera de juego.
[8]Pelota en juego.

Play-Mode Control

Cuando el modo de juego se encuentra en *kick_off*, *free kick*, *kick_in*, o *corner_kick*, el juez cambiará a modo *play_on* inmediatamente después de que la pelota es movida por el comando *kick*.

Offside

Un jugador esta en offside si se dan simultáneamente las siguientes condiciones:

- Esta en el campo contrario
- Esta más cerca de la meta contraria que los últimos dos defensas del equipo contrario
- Esta más cerca de la meta contraria que la pelota
- Esta a menos de 2.5 mts. de la pelota

Back passing

No se permite que el arquero atrape la pelota si le llego a él por un pase de un jugador de su propio equipo. En caso de suceder esto, el juez cobrará *back_pass_l* o *back_pass_r* (*l* - equipo izquierda, *r* - equipo derecha) y asignará *free_kick* para el equipo contrario. Esta permitido pasar la pelota al arquero sin que éste la atrape.

Free Kick faults

Si un jugador va a patear la pelota (comando *kick*) cuando el juego se encuentra en modo *kick_off, free_kick, kick_in, corner_kick*, no se permite que éste se haga un auto-pase. Si sucediera esto, el juez cobrará *free_kick_fault_l* o *free_kick_fault_l* y le asignará *free_kick* al equipo contrario.

Halft-Time y Time-Up

Cuando el primer o segundo tiempo termina, el juez para el partido. Cada tiempo dura 3000 ciclos (aproximadamente 5 minutos).

Reglas aplicadas por el juez humano

Todas las reglas mencionadas anteriormente pueden ser fácilmente detectadas por un juez automático (se cumplen siempre que se den ciertas condiciones), sin embargo, existen otras reglas que para ser aplicadas se debe medir la intencionalidad de los jugadores o de la jugada. Se hace extremadamente difícil lograr que un juez automático contemple estos criterios. Por esta razón, existe un juez humano que observa el partido y tiene la potestad de parar el partido y de interactuar con el simulador manualmente para hacer cambios sobre el estado de juego (pelota y jugadores).A continuación se ofrece una guía, conformada para la competición de la RoboCup 2000 (aún siguen vigentes), de las situaciones más comunes del partido en donde podría ser necesario la intervención del juez humano.

- Cuando un jugador esta continuamente rodeando la pelota
- Cuando el arquero es bloqueado por varios jugadores del equipo contrario
- Cuando la pelota no es puesta en juego luego de cierto cantidad de ciclos
- Cuando un jugador bloquea intencionalmente el movimiento de otros jugadores
- Cuando el arquero de un equipo abusa del comando catch. Por ejemplo cuando el arquero usa repetidamente el comando kick y catch.
- Cuando un jugador esta desbordando de mensajes el servidor (manda mas de 3 o 4 mensajes por ciclo).
- Cuando un jugador interfiere en el juego de una forma inapropiada.

En [CANb] se encuentran las reglas de la liga que son validas (además de las mencionadas anteriormente) para la competencia de Osaka 2005.

3.4. Software de Simulación para la liga

La liga de simulación 2D [LSIM2D] ofrece un software base que debe ser utilizado por quienes desean participar en la misma. El mismo es desarrollado por la propia comunidad RoboCup de estudiantes, docentes, investigadores y participantes en general, en un esfuerzo conjunto. Existe una instancia anual de llamados a presentación de proyectos en relación al software de simulación. Las propuestas realizadas por los investigadores son evaluadas por un comité el cual además determina la línea de trabajo e investigación a futuro. La propuesta de proyecto que se destaque a juicio de dicho comité se anuncia durante el ciclo de conferencias o simposios de cada campeonato oficial RoboCup. A esta actividad se la conoce dentro de dicho evento como *Development.* Hoy en día los esfuerzos de investigación se encuentran dirigidos al simulador 3D[9].

En paralelo al software oficial, la comunidad genera gran cantidad de herramientas, algunas de las cuales sirven a los equipos en su investigación. Muchos equipos han desarrollado sus propios debuggers, monitores (software que sirve para la visualización de los partidos) ó también los llamados *Asistentes* que nos brindan la posibilidad de acceder de manera más rápida y mejor a la información para la evaluación y construcción de los equipos.

Una de las características que resulta más atractiva para quienes observan el fútbol simulado, es su presentación. Aparecen continuamente nuevos monitores mejores a los anteriores que cuentas con mas cámaras y funcionalidades en general.

Además de este tipo de software, se pueden encontrar bibliotecas para construir agentes básicos (jugadores) en distintos lenguajes, las cuales ayudan a los principiantes a dar sus primeros pasos dentro de la liga simulada.

Software Oficial

Dentro del software oficial, algunos de los paquetes que vamos a encontrar son:

- **rcsssoccersim**: Este paquete contiene todos los “sub-paquetes” del que esta formado el software oficial. No es actualizado muy periódicamente, las versiones de los sub-paquetes en general están más actualizadas. Todas sus versiones soportan varias distribuciones de los sistemas linux y unix, y en alguna de ellas también se cuenta con versiones para windows.
- **rcssserver** (2D): Este paquete es el servidor de la liga simulada 2D.
- **rcssbase** (2D): Son librerías utilizadas por el servidor y otros sub-paquetes.
- **rcsslogplayer** (2D): Sirve para poder grabar y reproducir partidos.
- **rcssmonitor_classic** (2D): Es el monitor clasico, aunque es él más rudimentario, sigue estando vigente.
- **rcssmonitor** (2D): Este monitor dista muy poco del clasico. Cuenta con mas funcionalidades, la más importante es que tiene integrado el logplayer (figura 3.2).
- **manual** (2D): Al igual que el software, el manual es un esfuerzo conjunto de la comunidad.
- **rcssserver3d** (3D): Es el servidor 3D. A diferencia que el 2D, el mismo viene con todos los paquetes integrados.

Otros Monitores, Asistentes y herramientas en general.

Aparte del software oficial, existen múltiples herramientas que apoyan el trabajo en la liga simulada de la RoboCup. Uno de los ejemplos son las librerías que se pueden encontrar para la construcción de agentes. Las mismas se encuentran disponibles para lenguajes como C++, Java y otros. Aparte de estas librerías con agentes básicos, muchos equipos dejan accesible su código fuente para que puedan ser reutilizados.

También existen otras versiones del servidor (para plataformas windows o implementados en Java[10]). En general estas iniciativas van perdiendo vigencia frente al avance del simulador oficial debido al aporte que hace toda la comunidad RoboCup.

A continuación vamos a desarrollar dos de los puntos fuertes que tiene el software alternativo al oficial, en primer lugar hablaremos de los monitores y en segundo de los asistentes.

[9] Simulador presentado en el Robocup 2004.
[10] El servidor oficial de la liga simulada, esta desarrollado en C++.

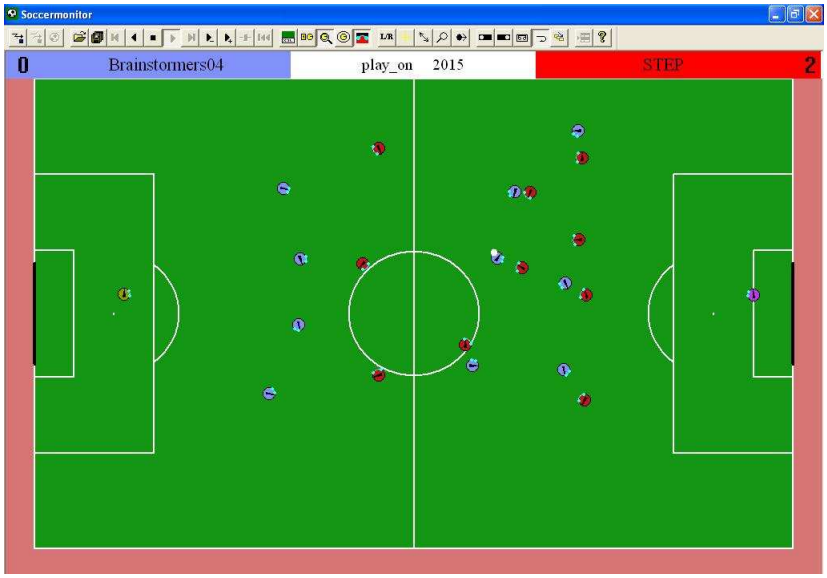

Figura 3.2: Captura de una reproducción de la final de Portugal 2004 - Monitor 2D.

Monitores

Figura 3.3: Captura monitor 3D del equipo FC-Portugal.

Los monitores son fundamentales en un juego donde la visual tiene un rol preponderante para el observador. En este sentido se generaron varios proyectos para el desarrollo de monitores 3D para la liga simulada 2D. Con estos monitores no vamos a ver que la pelota se levante del piso, tenemos visualización en tres dimensiones pero la información que recibe el monitor esta en dos dimensiones. Uno de los proyectos mas viejos es el *Virtual RoboCup* [VIR]. En los últimos tiempos han mejorado bastante y nos parece importante mostrar dichos avances.

El monitor que se muestra en la figura 3.3 fue desarrollado por el equipo FC-Portugal [POFCP]; el mismo funciona solo en ambientes Windows[11]. El monitor del la figura 3.4 del equipo Wright Eagle [POWRI] esta distribuido para ambientes linux y windows, el mismo solo sirve como reproductor de logs. El equipo RoboLog [POROB] también a desarrollado un monitor que fue utilizado oficialmente en el campeonato de Fukuoka 2002 [CROB02]. Este software es la base del monitor de la liga 3D. En la figura 3.5 se pueden observar imagenes de este monitor en donde los jugadores son representados como esferas.

[11] El monitor y el servidor puede ejecutarse en máquinas diferentes ya que se comunican a través de una arquitectura Cliente-Servidor.

Figura 3.4: Captura monitor 3D - Magic Box para linux y windows (solo reproduce logs).

Figura 3.5: Monitor del Equipo RoboLog, base del simulador de la liga 3D [POA4T].

Asistentes

Los asistentes son monitores en los que el investigador puede visualizar información interna de los agentes, debaguearla, analizarla y reproducirla. Por ejemplo, se puede ver el cono de visión o se pueden visualizar ver los comandos que envían al servidor los agentes e incluso realizar gráficos de distintos tipos en base a ellos.

El asistente del equipo SBCE [POSBC] declarado en RoboCup 2002 como ”Best Game Presentation and Analysis Tool” tiene las siguientes características:

- **Logplayer**: Se puede reproducir un partido y permite visualizar todas las características de cada jugador. La visión de los jugadores puede ser desplegada gráficamente y se puede observar el rastro de jugadores y de la pelota.
- **Debugger gráfico**: El Debugger permite representar graficamente las sentencias guardadas en el log. Estas sentencias tienen que ver con el modelo del mundo (jugador/pelota posiciones/velocidades) y la conducta de los jugadores (pases, pateos, dribles[12], cambios de posición, posesión e intercepción de la pelota, marcado de oponentes y escapes de marcas). En la figura 3.6 se puede ver: el cono de visión en cada ciclo para el jugador nro. 3, la energía (stamina) para el jugador nro. 4 y las coordenadas respecto al campo de juego del jugador nro. 8. Además, se muestran estadísticas del jugador nro. 10. en la figura 3.7 se puede apreciar un grafico de barras realizado por el analizador off line del asistente, que muestra el estudio de los pases realizados por los jugadores de ambos equipos.
- **Analizador**: Se propusieron los siguientes eventos a analizar basados en la información de los logs: Posesión de la pelota, Pases, Perdida de la pelota, Dribles y peteos. El analizador es capaz de reconocer estos eventos y mostrarlos gráficamente en el campo. También, el usuario puede generar los reportes asignando pesos a la ocurrencia de diferentes eventos de cada jugador de forma de evaluar la performance del equipo.

[12] Trasladarse con la pelota.

Figura 3.6: Captura del asistente desarrollado por el equipo SBCE.

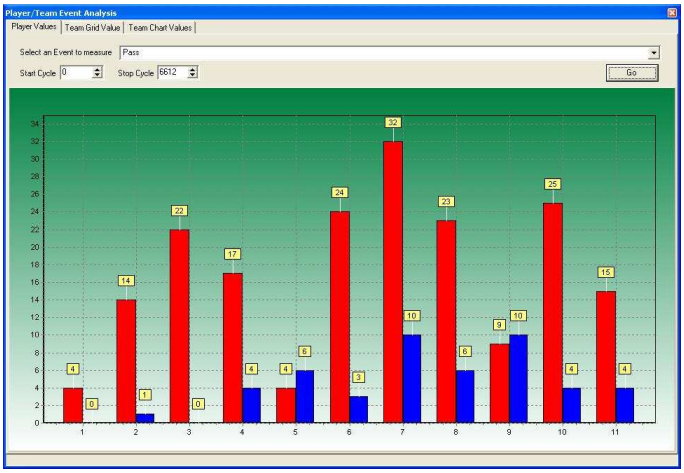

Figura 3.7: Captura del analizador off-line del asistente - Análisis de pases.

3.5. Servidor Oficial de la liga 2D

El simulador de la RoboCup (de aquí en más *SoccerServer* o *servidor*) contempla muchas de las complejidades del mundo real. Plantea un gran desafío a los investigadores y brinda una plataforma realista para el desarrollo y manejo de agentes.

Este simulador, SoccerServer de aquí en más, es casi un sistema de tiempo real que trabaja con intervalos de tiempo discretos usualmente de 100ms de duración. Estos intervalos de tiempo son denominados ciclos del simulador y son su unidad de tiempo.

Durante un ciclo, un cliente conectado al servidor, al cual se denomina agente, recibe información de distinto tipo obtenida de los sensores que lo rodean y del campo de juego. Por este motivo, si los agentes desean decidir en función de información vigente, deben responder inmediatamente a cada ciclo para completar una acción. Esto requiere tomar decisiones en tiempo real.

Cuando un agente decide lo que quiere hacer, la acción se ejecuta recién cuando se termina un ciclo del simulador. Por lo tanto, podemos decir que el servidor usa un modelo de acciones discretas, mientras que por otro lado, los agentes deben decidir en tiempo real.

Cada agente es un programa cliente separado e independiente, que puede ser desarrollado potencialmente en cualquier lenguaje. Estos agentes no pueden comunicarse directamente, la comunicación permitida en la liga es solo en forma indirecta a través del simulador. Todos los agentes del campo de juego usan el mismo canal de comunicación el cual tiene poco ancho de banda y también es poco confiable.

Para el simulador, hay dos grandes tipos de agentes, los jugadores y los entrenadores o coachs. Los agentes que son del tipo jugador, pueden desarrollar diferente tipo de acciones, las cuales están divididas en dos categorías: acciones primarias y acciones corrientes. Durante un ciclo de simulador solo puede ejecutarse una acción primaria, mientras que varias acciones del tipo corriente pueden ejecutarse simultáneamente con una acción primaria. Aunque un agente solicite la ejecución de mas de una acción primaria, el simulador solo ejecutara una de ellas.

Por otra parte, los agentes del tipo Coach son clientes privilegiados del SoccerServer que tienen como objetivo dar asistencia a sus jugadores. Existen dos tipos de coach, online y trainer (este último también llamado offline).

Los jugadores pueden ser divididos en diferentes tipos de jugadores, característica denominada jugadores heterogéneos. Si se desea jugar de este modo, cuando comienza un juego nuevo cada equipo puede seleccionar los tipos de jugadores que quiere tener, ya que cada tipo tiene características propias como velocidad de pique, energía, etc.

El SoccerServer también cuenta con un árbitro automático que controla el juego. Este puede cambiar el modo de juego y cuando lo hace informa de estos cambios a los agentes conectados al servidor. Estos mensajes son privilegiados en el sentido de que los jugadores pueden escucharlos en cualquier situación independientemente del numero de mensajes que hayan escuchado de otros jugadores (recordar que los mensajes a los jugadores están limitados).

En las siguientes secciones se presenta un resumen de sus principales características. Para obtener una descripción más detallada del funcionamiento del simulador dirigirse a [CANa, CHE].

3.5.1. Arquitectura

El simulador de la liga simulada 2D de la RoboCup cuenta con una arquitectura Cliente-Servidor. La comunicación entre clientes y el servidor se realiza a través del protocolo UDP/IP. La incorporación de este estilo de arquitectura ha permitido que los clientes puedan ser programados en cualquier lenguaje (debe soportar la comunicación vía sockets) y bajo cualquier plataforma. Cada cliente controla un único jugador o coach. Por ahora solo diremos que en un partido contamos con 24 procesos clientes que se dividen en 22 jugadores (uno por cada jugador de cada equipo) y 2 coach[13]. Los clientes pueden enviar y recibir comandos a través del servidor para controlar a los jugadores.

3.5.2. El Tiempo para el Simulador

El SoccerServer puede ser considerado como un sistema de tiempo real que trabaja con intervalos de tiempo discretos usualmente de 100ms de duración. Estos intervalos son denominados ciclos del simulador y son su unidad

[13]Los Monitores y LogPlayers también se comunican con el SoccerServer a través de sockets bajo la arquitectura Cliente-Servidor.

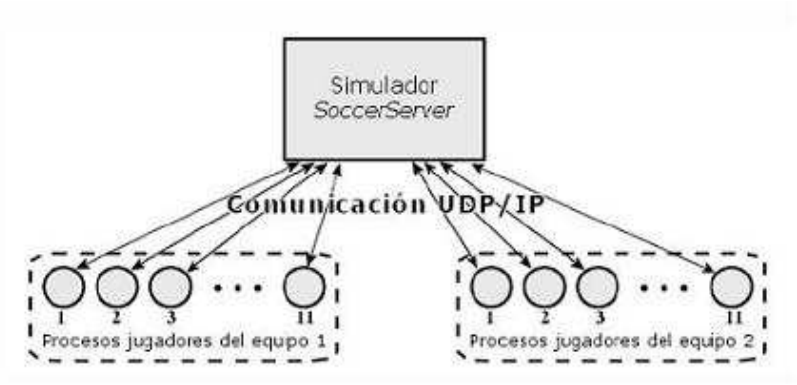

Figura 3.8: Comunicación entre el SoccerServer y los agentes jugadores [MAT].

de tiempo. Mientras el simulador utiliza un modelo de acciones discreto donde actualiza el estado del ambiente en cada ciclo, cada uno de los clientes toma sus decisiones en tiempo real y se las envía al servidor en cualquier momento. El servidor recepciona las acciones de los jugadores durante el curso de un ciclo pero las ejecuta sobre el final del mismo.

Si un cliente envía más de un comando durante el transcurso de un ciclo, el servidor elige aleatoriamente uno de ellos para su ejecución descartando a los restantes. Esto hace que cada jugador trate de no envíar más de un comando por ciclo. Por otro lado, si un jugador no envía ningún comando al servidor durante un ciclo, tendrá que esperar al siguiente para poder actuar en el juego. Esto puede ser una gran desventaja en un sistema de tiempo real en donde interviene un adversario.

Si bien el simulador actualiza el estado de su mundo cada 100ms, envía información visual a los clientes cada 150ms, información física cada 100ms e información auditiva en el momento que se genere.

Los partidos generalmente se juegan a 6000 ciclos (aproximadamente 5 minutos cada tiempo) aunque esto es configurable en los parámetros del servidor.

3.5.3. Modelado de la realidad

Los objetos del mundo que maneja el simulador se pueden ver en el diagrama UML de la figura 3.9. Todo objeto del juego posee una distancia y una dirección. Los objetos del campo de juego poseen una posición en el mismo y se pueden dividir en móviles y estáticos. Dentro de los objetos móviles se encuentran la pelota y los jugadores.

Propiedades de los Jugadores:

- Nombre del Equipo
- Lado (l | r) por izquierda o derecha. Cambia en el medio tiempo.
- Número de camiseta (1..11)
- Dirección del cuerpo (-180..+180)
- Dirección del cuello (minneckang..maxneckang[14]) relativo a la posición del cuerpo.

Los objetos estáticos (figura 3.10) que maneja el simulador son marcas en el campo de juego y resultan fundamentales para que el jugador pueda tener una noción de espacio dentro del mismo. Un aspecto importante a destacar es que las coordenadas percibidas por el jugador sobre los objetos siempre son relativas a su posición dentro del campo de juego. Al conocer las coordenadas globales de estas marcas, el jugador puede transformar las coordenadas relativas a globales del resto de los objetos que percibe.

Como se puede ver en la figura 3.10, Las marcas están divididas en izquierda(r) - derecha(l) y arriba(t) - abajo(b). Además se cuenta con la línea de gol (goal), la línea del medio campo (c) y la línea del área del penal (p).

[14] minnneckang y maxneckang son valores constantes y configurables desde los parámetros del servidor.

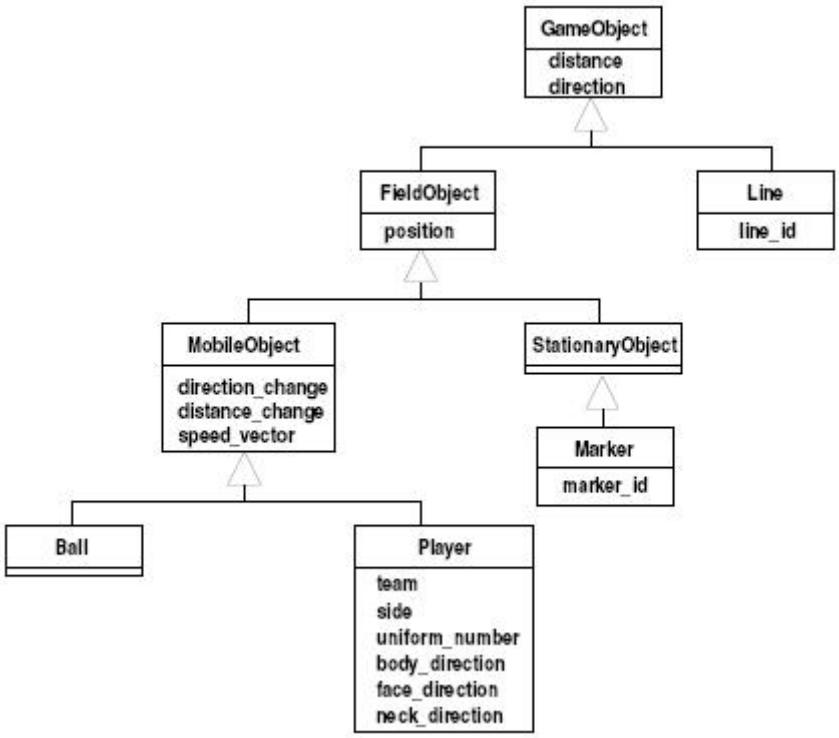

Figura 3.9: Diagrama UML de los objetos manejados por el SoccerServer [CHE].

3.5.4. Modelo Sensorial

Cada jugador posee la capacidad de percibir la realidad a través de tres sensores. El *Sensor Auditivo*, por donde recibe mensajes del referee, los coachs y del resto de los jugadores de ambos equipos. El *Sensor Visual*, que le permite recibir información de los objetos visibles para el jugador. Y por último, el *Sensor Corporal* que le permite obtener al jugador información sobre su estado físico (energía, velocidad, ángulo del cuello, etc.).

3.5.4.1. Sensor Auditivo

El *sensor auditivo* recibe información desde el servidor cuando un jugador, coach o el referee, envía un mensaje a través del comando *Say*. Estos mensajes son recibidos por el jugador inmediatamente sin necesidad de que pase un ciclo. El formato de los mensajes recibidos desde el servidor es:

(**hear** *Time Sender Message*)

Donde *Time* representa el tiempo transcurrido hasta el momento (número de ciclo). *Sender* indica quien originó el mensaje (tipo de mensaje). Si el mensaje proviene de un jugador, se brinda su ubicación relativa a quien recibe el mismo, de lo contrario, se indica con *referee, online_coach_left o online_coach_rigth* dependiendo de quien haya sido el emisor del mensaje. Por último, en *Message* se almacena el contenido del mensaje. El tamaño del mismo tiene un límite y es configurable desde los parámetros del servidor.

Capacidad Auditiva Un jugador puede escuchar un mensaje si su capacidad auditiva es mayor o igual a *hear_decay*[15], luego de escuchado el mensaje su capacidad auditiva disminuye en este valor. El jugador va recuperando en cada ciclo su capacidad auditiva *hear_inc*[16] unidades hasta llegar a su valor máximo (*hear_max*[17]). La configuración actual de los parámetros del servidor esta determinada para que se pueda escuchar un mensaje de cada equipo por ciclo. Esta restricción no aplica a los mensajes enviados por el propio jugador o por el referee.

[15] capacidad auditiva mínima para escuchar un mensaje (parámetro del servidor)
[16] recuperación de la capacidad auditiva en cada ciclo (parámetro del servidor)
[17] máxima capacidad auditiva de un jugador (parámetro del servidor)

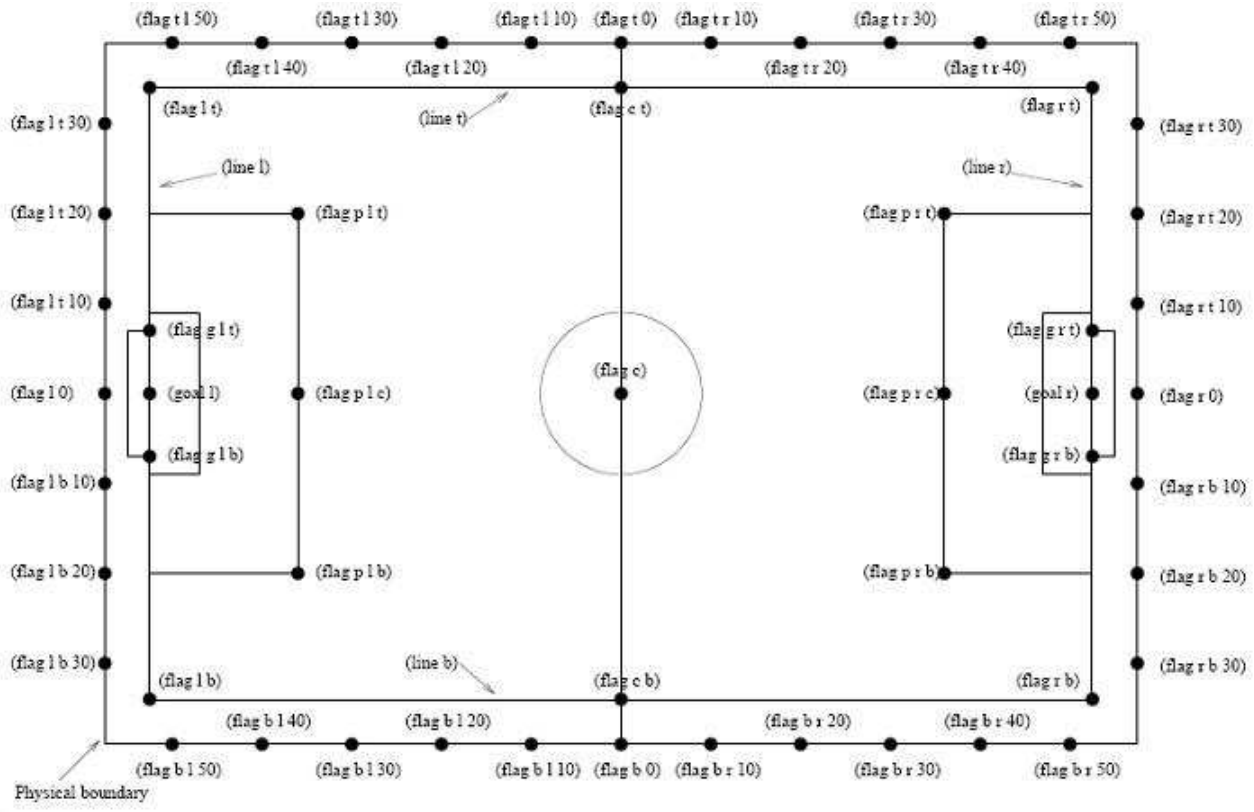

Figura 3.10: Marcas del campo de juego [CHE].

Rango de Comunicación Un mensaje enviado por un jugador puede ser escuchado por todo jugador que se encuentre dentro de un radio máximo de *audio_cut_dist*[18] metros del jugador emisor. Los mensajes enviados por el referee no cuentan con esta restricción.

3.5.4.2. Sensor Visual

El sensor de visión le permite al jugador obtener información del servidor acerca de la ubicación y velocidad[19] de todos los objetos del campo que se encuentren dentro de su cono de visión. Este tipo de información es enviada a cada cliente por medio del comando *see* cada *sense_step*[20] (actualmente el valor es 150 ms). El formato del comando *see* es:

(**see** *ObjName Distancia Direccion DistChng DirChng BodyDir HeadDir*)

Donde:

ObjName Es el tipo de objeto visualizado, el mismo puede ser un jugador (p), la pelota (b) o una marca del campo (Figura 3.10).

Distancia Es la distancia entre el objeto observado y el observador.

Dirección Representa el ángulo del objeto observado relativo a la dirección de la cabeza del observador.

DistChng Cambio en la distancia entre ambos objetos.

DirChng Cambio de la dirección entre ambos objetos.

[18] distancia máxima para escuchar un mensaje (parámetro del servidor)
[19] Solo aplica para los objetos móviles (jugadores y pelota).
[20] Frecuencia de actualización visual (parámetro del servidor)

BodyDir Es la dirección del cuerpo del objeto observdado.

HeadDir Es la dirección del cuello del objeto observado.

Rango de Visión La visión de un jugador depende de múltiples factores. Dos de ellos son los parámetros *sense_step* (150 ms) y *visible_angle* (90 grados) que corresponden a la frecuencia con la que se recibe información visual y los grados del cono de visión normal del jugador. El jugador puede influir en la frecuencia y calidad de la información que recibe cambiando la amplitud o profundidad de su visión a través del comando *change_view***.** La idea es que cuanto más angosto es el ángulo de visión, el jugador podrá ver mas lejos; además, si su cono de visión es disminuido, la información será actualizada con más frecuencia.

Ejemplo de Visión de un Jugador Veremos un ejemplo basándonos en la figura 3.11. Un jugador puede ver un objeto si se encuentra dentro del radio *visible_distance*, aunque se encuentre fuera de su cono de visión (*view_angle*). En este caso, el jugador solo podrá saber el tipo de objeto (ball, player, goal ó flag) pero no conocerá su nombre o número de camiseta.

En el ejemplo de la figura, solo los objetos *g* y *b* no son visibles. El cono de visión es de 90 grados, el jugador *f* se encuentra de frente con un ángulo relativo de 0 grados, el *e* a -40 grados y el *d* a +20 grados relativos a la cabeza del jugador observador. Este ejemplo también muestra como se afecta la calidad de la información debido a la distancia de los objetos; de los jugadores que se encuentren cerca se podrá visualizar el nombre del cuadro al que pertenecen y su número de camiseta, a medida que se incremente la distancia irá bajando la probabilidad de conocer esta información.

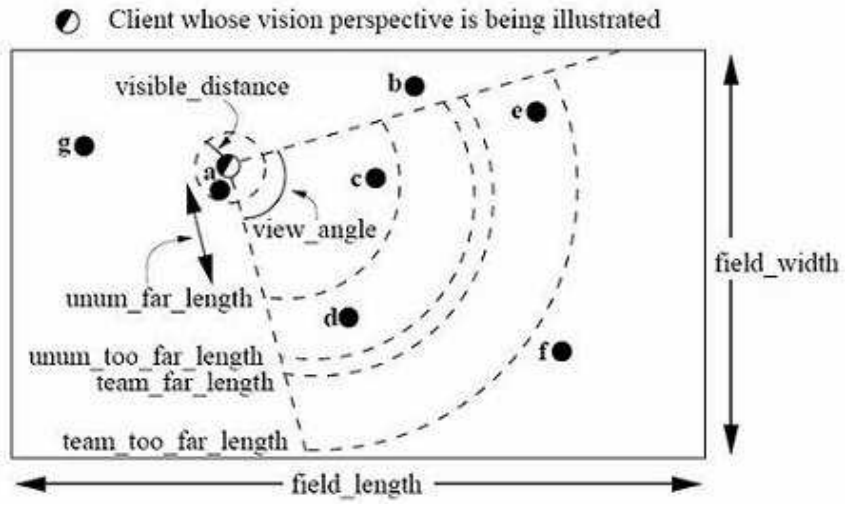

Figura 3.11: Ejemplo de la visión de un jugador [CHE].

El servidor maneja las siguientes distancias:

$$unum\_far\_length =< unum\_too\_far\_length =< team\_far\_length =< team\_too\_far\_length$$

Si tomamos *Dist* como la distancia del objeto observador al objeto observado, tenemos que:

- Si $Dist =< unum\_far\_length$ siempre se reconoce el equipo y el número de camiseta
- Si $unum\_far\_length < Dist < unum\_too\_far\_length$ siempre se reconoce el equipo pero la probabilidad de ver el número de camiseta decrece de 1 a 0 a medida que aumenta la distancia
- Si $Dist => unum\_too\_far\_length$ el número de camiseta no es visible
- Si $Dist =< team\_far\_length$ siempre se reconoce el equipo

- Si *team_far_length* <*Dist* <*team_too_far_length* la probabilidad de reconocer el equipo decrese linealmente de 1 a 0 a medida que aumenta la distancia
- Si *Dist* >*team_too_far_length* el equipo no es visible

Del ejemplo se desprende que:

- de *c* conocemos el equipo y el numero de camiseta
- de *d* conocemos el equipo y con 50 % de posibilidades, su numero de camiseta
- de *e* solo conocemos con 50 % de posibilidades, su equipo
- a *f* lo vemos solo como un jugador anónimo

Modelo de Visión con Ruido El servidor incorpora ruido a la información de visión, el cual es cuantificado en función de la distancia de los objetos observados. Por lo tanto, se puede decir que mientras exista distancia a un objeto no se va a conocer la posición exacta de éste.

3.5.4.3. Sensor Corporal

El *sensor corporal* brinda información del estado físico del jugador. Esta información es enviada automáticamente al jugador cada *sense_body_step*[21] (actualmente 100 ms). Parte de la información recibida es: calidad y amplitud de visión, energía, velocidad, dirección del cuerpo y cuello del jugador. También se recibe información estadística sobre cuantos comandos de cada tipo fueron ejecutados.

3.5.5. Modelo de Movimientos

En cada ciclo del simulador, los movimientos de cada jugador se simulan según las ecuaciones que se muestran en la figura 3.12. La velocidad es influenciada positivamente por la aceleración y negativamente por el parámetro *decay*. La aceleración aparece en los jugadores al ejecutar el comando *dash* y en la pelota cuando algún jugador la patea con el comando *kick*, ambos comandos llevan como parámetro un valor de potencia que se utiliza para determinar la aceleración aplicada al jugador o a la pelota. El parámetro *decay*[22] permanece constante y puede verse como la fuerza de rozamiento que deben superar los objetos para movilizarse. Existe un valor *decay* para los jugadores y otro para la pelota.

$(u_x^{t+1}, u_y^{t+1}) = (v_x^t, v_y^t) + (a_x^t, a_y^t)$: aceleración

$(p_x^{t+1}, p_y^{t+1}) = (p_x^t, p_y^t) + (u_x^{t+1}, u_y^{t+1})$:desplazamiento

$(v_x^{t+1}, v_y^{t+1}) = decay * (u_x^{t+1}, u_y^{t+1})$:decremento de velocidad

$(a_x^{t+1}, a_y^{t+1}) = (0,0)$: reinicio de aceleración

Donde:

(v_x^t, v_y^t) =velocidad en el ciclo t

(p_x^t, p_y^t) =posición en el ciclo t

decay = rozamiento (distinto para la pelota ó jugador)

(a_x^t, a_y^t) =formula que surge de la potencia y ángulo con el que se ejecutó el comando kick(patear) o dash(correr).

Figura 3.12: Ecuaciones que utiliza el SoccerServer para realizar el cálculo de los movimientos [CHE].

[21] Frecuencia de actualización de la información corporal (parámetro del sevidor)
[22] Fuerza de rozamiento del jugador (parámetro del servidor)

Modelo de Movimientos con Ruido

Para reflejar los movimientos inesperados de los objetos del mundo real, el simulador le ha incorporado ruido. El ruido es incorporado al vector de velocidad en cada ciclo y a los parámetros de los comandos *dash* y *kick*. También se incorpora un valor de ruido por el viento, el mismo está determinado por una fuerza y una dirección. Básicamente se le suma a dichos elementos un valor aleatorio de distribución uniforme dentro de un rango. En el caso en donde dos objetos colisionan, se multiplica al vector de velocidad de los objetos por un factor igual a -0,1.

3.5.6. Acciones

Los jugadores pueden realizar acciones a través de los comandos que les proporciona el simulador. Las acciones se dividen en primarias: *kick* (patear), *dash* (correr), *turn* (girar), *catch* (agarrar), *move* (moverse) y concurrentes: *say* (hablar), *turn_neck* (girar el cuello), *change_view* (modificar la visión), *sense_body* (consultar el estado) y *score* (resultado). Durante un ciclo, solo se puede ejecutar una de las acciones primarias y simultáneamente a ésta se pueden ejecutar varias de las acciones concurrentes. A continuación se da una breve explicación de cada uno de estos comandos.

Comando Catch

El arquero es el único jugador habilitado a tomar la pelota con el comando *catch*. Esta acción es realizada con éxito si la pelota se encuentra dentro del área penal y dentro de la región *catchable_area*.

Si el arquero realiza por ejemplo un *catch 45º* (figura 3.13), el rectángulo de lados *catchable_area_l* y *catchable_area_w* delimita la región donde la acción tendría éxito con probabilidad *catch_probability*. Si el arquero logra tomar la pelota, podrá utilizar el comando *move* para moverse con ella dentro del área penal antes de patearla.

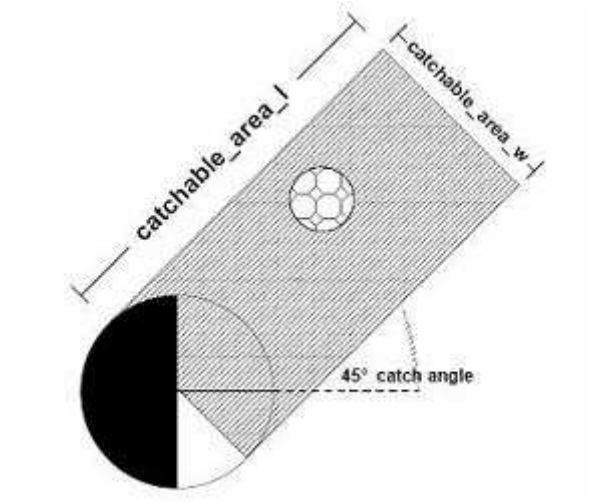

Figura 3.13: Región donde el arquero puede tomar la pelota con el comando *catch* [CHE].

Comando Dash

El comando *dash* se utiliza para acelerar al jugador en la dirección que se encuentre su cuerpo. Este comando requiere como parámetro la potencia de aceleración que se le va a aplicar al jugador. Para que se pueda realizar esta acción, el jugador en cuestión debe poseer la energía (*estamina*) requerida por este movimiento[23] . Cuando un jugador acelera con una cierta potencia, la misma se le quita de su energía. A través de la potencia, se calcula el vector de velocidad y su posición en el próximo ciclo (sección 3.5.5).

[23]La cantidad de energía necesaria por el comando depende de la aceleración deseada.

Modelo de Energía (Stamina)

La energía está compuesta por tres variables: la cantidad de energía (*stamina*), la recuperación (*recovery*) y el esfuerzo (*effort*) . La energía disminuye cuando un jugador corre pero en cada ciclo se recupera un porcentaje de la energía gastada. El valor *recovery* se encarga de determinar cuanta energía puede recuperar un jugador luego de un ciclo y *effort* da una noción de la efectividad al correr. Al comienzo del partido y del segundo tiempo la energía de cada jugador es establecida en *stamina_max*[24].

Comando Kick

El comando *kick* le permite al jugador patear la pelota con una dirección y potencia dada. El comando requiere como parámetros la potencia del tiro y el ángulo[25] con el que se le pega a la pelota. Una vez que llega el comando al servidor este verifica que se den las condiciones necesarias como para hacer efectivo el tiro. La pelota debe estar a una distancia del jugador menor a *kickable_margin*[26] metros y no debe haber ningun jugador del mismo equipo que genere *offside*. Antes de realizar el tiro, el simulador calcula la potencia efectiva de la patada (*ep*) sobre la cual influye, la distancia de la pelota al jugador (cuanto más lejos menor potencia) y el ángulo con el que se le pega a la pelota. El peor caso al patear la pelota se da cuando su *ep* se reduce en un 50 %.

Comando Move

El comando *move* permite mover los jugadores a cualquier posición dentro del campo de juego. Este comando tiene solo dos usos, uno de ellos es para posicionar al equipo dentro del campo de juego cada vez que se reanuda el partido (al comienzo de cada tiempo y luego de anotado un gol). El otro uso que se la da a este comando es por parte del arquero para moverse dentro del área cuando tiene la pelota en su poder (es limitada la cantidad de veces que puede usar este comando antes de patearla).

Comando Say

Con este comando los jugadores pueden comunicarse a través de mensajes. Los mensajes tienen un tamaño máximo de *say_msg_size* y pueden ser escuchados por cualquier jugador que se encuentre a una distancia menor de *audio_cut_dist* metros. Los mensajes son enviados inmediatamente al servidor (no requiere esperar al final de un ciclo). El uso de este comando esta limitado por la capacidad auditiva de los jugadores (sección 3.5.4.1).

Comando Turn

El comando *turn* le permite al jugador cambiar la dirección de su cuerpo. El comando recibe el parámetro *momento*. Si el jugador esta en reposo, *momento* representa el ángulo de giro efectivo que se le dará al jugador. En caso de que el jugador se encuentre en movimiento y considerando la dificultad de girar debido a la inercia existente por el mismo, el giro efectivo del jugador será disminuido.

Comando Turn neck

Este comando le permite al jugador girar su cuello sin afectar la dirección de su cuerpo. La dirección de su cuello permite determinar el ángulo de visión del jugador. Mientras que *turn* gira el cuerpo del jugador, *turn_neck* gira su cuello en relación a su cuerpo. Esto significa que el ángulo de visión del jugador esta condicionado al giro del cuerpo y del cuello. Este comando puede ser ejecutado junto con los comandos *dash*, *turn* y *kick* en el mismo ciclo.

Comando Score

El comando *score* es utilizado por los jugadores para saber el resultado parcial del partido. Al ejecutarlo el servidor retorna un mensaje con el siguiente formato:

score *Time OurScore TheirScore*

[24] Energía máxima de un jugador (parámetro del servidor)
[25] Con este ángulo y a la posición del jugador es que se calcula la dirección con la que sale la pelota luego del tiro.
[26] Distancia máxima de la pelota al jugador para patearla (parámetro del servidor)

Donde *Time* es el ciclo actual del partido, *OurScore* los goles convertidos por su equipo y *TheirScore* los goles convertidos por el contrario.

3.5.7. Jugadores Heterogéneos

Para simular las diferencias de energía, potencia y recuperación que existen entre los jugadores humanos, el SoccerServer introdujo el concepto de tipo de jugador, *player_type*. Los distintos tipos de jugadores junto con sus características son configurables desde los parámetros del servidor. Ambos equipos tienen disponibles la misma cantidad de tipos de jugadores. A medida que estos se van conectando, el servidor les informa los tipos que se encuentran disponibles para poder ser elegidos.

3.5.8. El Referee

El juez automático del SoccerServer envía mensajes a los jugadores indicándoles el modo de juego actual del partido (fuera de juego, tiro libre, fuera del campo, etc.). Adicionalmente, les envía información de otros eventos. En el cuadro 3.1 se muestran los distintos modos de juego que tiene un partido, los cuales son comunicados a los jugadores por el referee. En el cuadro 3.2 se muestran los mensajes que puede enviar el referee a los jugadores.

Play Mode	**Descripción**	$\mathbf{T_C}$	Sig. Play Mode
before_kick_off	antes de comenzar cada tiempo	0	kick_off_*Side*
play_on	Jugando		
time_over	juego terminado		
kick_off_*Side*	comienza el juego *Side*		
kick_in_*Side*	saca *Side*		play_on
free_kick_*Side*	tiro libre para *Side*		play_on
corner_kick_*Side*	corner a favor para *Side*		play_on
goal_kick_*Side*	saque de arco para *Side*		play_on
goal_*Side*	gol para *Side*		
drop_ball	pelota afuera	0	play_on
offside_*Side*	offside de *Side*	30	free_kick_*Side*

Cuadro 3.1: Modos de juego en un partido de la liga de Simulación 2D [CHE].

Mensaje	**Descripción**	T_C	**Sig. Play Mode**
goal_*Side*_n	anuncia el gol nro. n para *Side*	50	kick_off_*OSide*
foul_*Side*	foul de *Side*	0	free_kick_*OSide*
goalie_catch_ball_*Side*	el arquero *Side* atrapo la pelota	0	free_kick_*OSide*
time_up_without_a_team	si no hubo oponente hasta el fin del segundo tiempo	0	timer_over
time_up	termino el partido	0	timer_over
half_time	termino el primer tiempo	0	before_kick_off
time_extended	tiempo extra	0	before_kick_off

Cuadro 3.2: Mensajes enviados por el referee [CHE].

Donde *Side* puede ser *l* o *r* y representa al equipo que juega a la izquierda o derecha del campo de juego, *Oside* es el equipo contrario y $T_C$es el tiempo (en cantidad de ciclos) que va a pasar hasta que el siguiente *Play Mode* sea anunciado por el referee.

3.5.9. Clientes Coach

Los coach son clientes privilegiados del SoccerServer que tienen como objetivo dar asistencia a sus jugadores. Existen dos tipos de coach, *online* y *trainer* (también llamado *offline*). En [CANa] se encuentra una lista con los comandos dle servidor que son utilizados por este tipo de agente.

Trainer

Este coach solo puede ser utilizado en la etapa de desarrollo o aprendizaje del equipo. Su principal cometido es interactuar con el server, generando sesiones de entrenamiento para ayudar y asistir a los jugadores en el aprendizaje de sus habilidades. El trainer tiene las siguientes habilidades:

- Tiene control sobre el modo del partido
- Puede mandar mensajes a sus jugadores (comandos o información de estado). El formato de los mensajes es libre
- Puede cambiar la ubicación, direcciones y velocidades de cualquier objeto móvil dentro del campo de juego
- Obtiene información libre de ruido de todos los objetos móviles del campo de juego

Conexión al SoccerServer con o sin juez automático Por defecto, el SoccerServer tiene activado el juez automático. En caso de que se quiera que el trainer tenga el control total del partido, se puede desactivar el juez. Se debe tener en cuenta que todas la tareas de las que el juez es responsable (cambiar el modo del partido, mover a los jugadores, etc.) ya no se realizan más automáticamente y deben ser realizadas por el trainer. La activación o desactivación del juez se debe indicar al levantar el SoccerServer, ya sea por comando o en la configuración del server.

Online Coach

A diferencia del trainer, el coach online puede ser conectado en los partidos oficiales. Su principal cometido es observar el partido y brindarle información a sus jugadores en pos de mejorar el juego del equipo. Las capacidades del coach online son:

- Comunicarse con sus jugadores (con restricciones)
- Obtener información libre de ruido de todos los jugadores y de la pelota

Para que se puedan desarrollar los coach online, independientemente del equipo que lo utilice, se creó el *estándar coach language* (CLang) para ser usado en la comunicación con el server y los jugadores. Por más información referirse a [CHE].

Comunicación con los jugadores La comunicación entre el coach y los jugadores se realiza por medio de mensajes alfanuméricos de largo configurable (*say_coach_msg_size*[27]).

Cuando el juego se encuentra en modo *play_on* existen restricciones en el tiempo de envío y en la cantidad de mensajes. Cada 600 ciclos (especificado en *freeform_wait_period*[28]) de modo *play_on*, se puede mandar mensajes en los próximos 20 ciclos (especificado en *freeform_send_period*[29]). Estas restricciones no son aplicadas cuando el partido se encuentra en un modo distinto de *play_on*.

El coach puede enviar *say_coach_cnt_max*[30] mensajes por partido. En caso de que el partido tenga tiempo adicional se le acreditan *say_coach_cnt_max*[31] mensajes cada 6000 ciclos (o lo que dure el tiempo adicional).

Cambio de Tipo de Jugador (palyer types) Usando el comando *change_player_type* el coach online puede cambiar el tipo de los jugadores. Cuando el partido se encuentra en modo *before_kick_off*, se pueden cambiar cuantas veces quiera (según las configuración del server). Luego de comenzado el partido, se podrán realizar solamente 3 variantes (especificado en *subs_max*). A los jugadores de ambos equipos se les informa las sustituciones.

[27] Tamaño máximo de los mensaje (parámetro del servidor)
[28] Período de tiempo que debe esperar el jugador mandar mensajes (parámetro del servidor)
[29] Período de tiempo durante en donde el jugador puede mandar mensajes (parámetro del servidor)
[30] Cantidad máxima de mensaje que puede enviar el referee (parámetro del servidor)
[31] Cantidad de mensajes extra que puede enviar el referee (parámetro del servidor)

3.6. El Problema de la Sincronización

Como se mencionó en la sección 3.5, el SoccerServer es el responsabe, entre otras cosas, de ejecutar los pedidos de acciones de cada agente (jugador) y de actualizar adecuadamente el estado juego. El simulador cada cierto intervalo de tiempo envía información sensorial (visual, auditiva y física) a cada agente acerca del estado de la realidad[32]. Durante cada ciclo, los agentes pueden enviar pedidos al servidor para que éste los ejecute al final del mismo. En caso de tener dos pedidos de un mismo agente para ejecutar en un ciclo, el servidor elige aleatoriamente uno de ellos descartando el restante. Por otro lado, si un agente no envía pedidos durante un ciclo, éste se esta perdiendo la oportunidad de intervenir en el juego durante dicho ciclo, dándole ventajas al equipo contrario.

Por lo dicho en el parrafo anterior, un aspecto importante a tener en cuenta en el desarrollo de los agentes es la sincronización con el simulador. Para que una acción sea ejecutada en un determinado ciclo *t*, la misma debe llegar al simulador durante el transcurso del ciclo *(t - 1)*. En el caso de que una acción llegue al servidor en un ciclo distinto del que originalmente fue pensada, la acción puede llegar a ser descartada (se elige la otra acción disponible para ese ciclo) o puede ser ejecutada pero sobre un estado del juego distinto al pensado originalmente por lo que el resutado de aplicar dicha acción será impredecible. Es por esto que un buen método de sincronización en el envío de acciones al servidor tiene un alto impacto en la performance de los agentes y del equipo.

La llegada a tiempo de los mensajes (tanto desde como hacia el servidor) se ven influenciados por varios factores como ser: la capacidad de los equipos (CPU, memoria, etc.) y la velocidad de la red donde operan éstos. Dichos retardos deben ser tenidos en cuenta para que una acción que estaba destinada a ser ejecutada en cierto ciclo, no llegue y se ejecute en otro. Otro problema importante que dificulta la sincronización es que el jugador desconoce cuando se da el comienzo de un ciclo, solo sabe su duración.

En el simulador, la información sensorial y la acciones son asincrónicas. Actualmente el agente puede: ejecutar solo una accción cada 100 ms (duración de un ciclo), recibir información visual cada 150 ms, recibir información física cada 100 ms y recibe aleatoriamente información auditiva. El desafio del agente es tratar de basar la elección de su acción en un determinado ciclo sobre información sensorial del entorno lo más actual posible. Esto no siempre se puede realizar ya que hay ciclos en donde no recibe información visual o esta llega demasiado tarde como para que sea incluida en la elección de la acción de ese ciclo.

En Resumen, el problema de la sincronización trata de determinar, para cada ciclo, el momento optimo para enviar una acción al servidor. En general, el envío de los mensajes tiene que ser lo más cerca posible del término del ciclo para maximizar la posibilidad de que le llegue al agente nueva información visual (si es que la hay en ese ciclo). Sin embargo, no se debe dejar de tener en cuenta que cuanto más cerca del fin del ciclo se envíe el mensaje se corre el riesgo de que éste llegue al servidor en el ciclo siguiente. Un mal método de sincronización puede llevar a perder oportunidades de acción o hacerlas con información desactualizada.
[DEBa].

3.6.1. Tiempo de llegada de los Mensajes

Como se mencionó anteriormente, cada ciertos intervalos el SoccerServer envía a cada agente información sensorial acerca del estado del entorno. Para el problema de la sincronización son importantes dos tipos de mensajes: *información física* de los jugadores (el servidor las envía cada 100 ms) y la *información visual* del campo de juego (el servidor las envía cada 150 ms).

Para lograr una mejor comprensión del problema de la sincronización y de los métodos que serán presentados en la sección 3.6.2 para su resolución, se debe encontrar y entender la relación existente entre estos dos tipos de mensajes. En [DEBa] se presenta un análisis detallado de varias pruebas realizadas acerca del envio de estos mensajes y la relación que existe entre ellos. Las pruebas consistieron en la realización y ejecución de dos programas, uno permite detectar la diferencia de tiempo existente entre la llegada de mensajes del mismo tipo y el orden en que éstos llegan al agente y el segundo programa permite observar el tiempo maximo que puede esperar un agente desde el arribo de un mensaje de información física para enviar al servidor una acción y que éste la reciba en el mismo ciclo[33].

A continuación se ofrecen las conclusiones obtenidas por [DEBa] apartir de estas pruebas:

[32] Cada agente recibe una porción del estado de la realidad deacuerdo a su ubicación dentro del campo de juego.

[33] Esta prueba se realizo con varias configuraciones del cliente y servidor: cliente local, cliente remoto, servidor ocupado, cliente o cupado, etc.

- La diferencias entre sucesivos mensajes del mismo tipo se corresponden con los intervalos especificados por el servidor (100 ms para la información física y 150 ms para la información visual).
- Los mensajes de información física llegan siempre primeros que los de información visual
- El porcentaje de mensajes que llegan al servidor en el mismo ciclo que fueron enviados por el agente luego de haber esperado entre 91 y 100 ms esta en el entorno del 97 % en su configuración más típica (cuadro 3.3).

Configuración				Tiempo (ms)			
nro	**cliente**	**server**	**cliente**	**81-90**	**91-100**	**101-110**	**111-120**
1	local	libre	libre	100.00 %	99.30 %	0.03 %	0.01 %
2	remoto	libre	libre	100.00 %	96.80 %	0.00 %	0.00 %
3	remoto	ocupado	libre	100.00 %	96.78 %	74.54 %	31.32 %
4	remoto	libre	ocupado	99.70 %	79.94 %	0.00 %	0.00 %

Cuadro 3.3: Procentajes de mensajes que llegan en el mismo ciclo para diferentes configuraciones del sistema y tiempos de espera [DEBa].

La configuración nro. 4 es la que se da típicamente en un partido de fútbol de esta liga. Existen tres máquinas conectadas en la misma red, en una se ejecuta el servidor y en cada una de las restantes se ejecuta un equipo con 11 jugadores (agentes) conectados al servidor enviando y recibiendo mensajes.

De todo el análisis anterior y recordando el problema que el agente no conoce el comienzo de un ciclo, se concluye que dicho momento puede ser considerado en el instante donde le llega un mensaje de información física. Esta conclusión es por demás relevante ya que todos los métodos que serán presentados en la sección 3.6.2 se basan en esto.

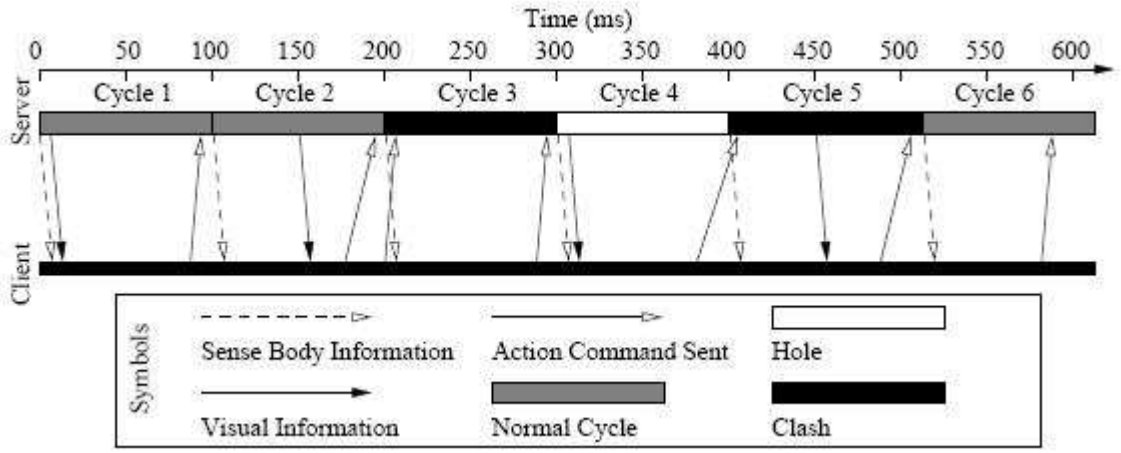

Figura 3.14: Problema de la Sincronización [BUL].

3.6.2. Métodos de Sincronización

En esta sección se presentarán cuatro métodos distintos presentados por [DEBa] y que pueden ser tomados como posibles soluciones al problema de la sincronización.

3.6.2.1. External Basic

En este método el agente solo realiza una acción cuando existe un cambio en el entorno. Esto significa que solo enviará un acción al servidor cuando el agente reciba un mensaje de información visual. Una ventaja de

este método es que el agente puede estar conformado por un único hilo de ejecución. De esta forma el agente, cada vez que ejecute una acción lo hará basándose en información actualizada del estado del juego. Una de las principales desventajas de este método es que el agente solo ejecuta acciones en los ciclos donde existan mensajes de información (se dan 2 veces cada 3 ciclos), esto lleva a que el agente solo utilice un 67 % de sus posiblidades de acción, dándole una clara ventaja al oponente en caso de que éste utilice un método con mayor porcentaje de utilización. En el algortimo 1 se da el seudo-código para este método y en la figura 3.15 se muestra un ejemplo de su aplicación.

Algorithm 1 Seudo-código para el método de sincronización External Basic [DEBa].

while server is alive **do**
  wait for see message
  determine next action
  send action command to server
end while

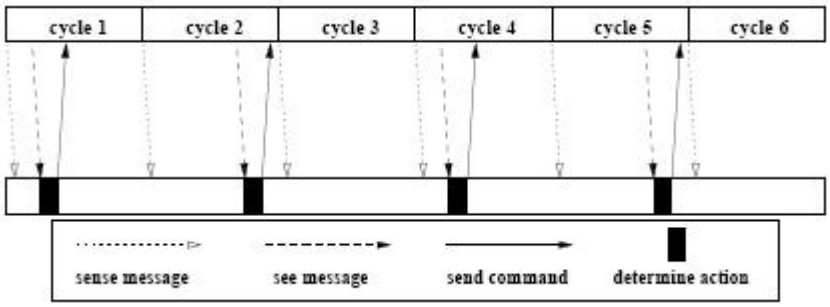

Figura 3.15: Ejemplo usando el método de sincronización External Basic [DEBa].

3.6.2.2. Internal Basic

Otra posible solución al problema de la sincronización es la de enviar una acción al servidor cada 100 ms usando algún tipo de reloj interno (señal) que permita contar estos intervalos. En el algoritmo 2 se muestra el seudo-código de este método. Como la duración de estos intervalos es igual a la de los ciclos, este método tiene la ventaja de que el agente envía una acción en cada ciclo sin desperdiciar oportunidades de acción. Como desventaja se tiene que si la señal de los 100 ms interrumpe cerca del comienzo del ciclo, no dando tiempo a que llegue el mensaje con información visual, el agente puede tomar una acción basándose en un estado del juego desactualizado. Lo deseable es que la señal interrumpa sobre la mitad del ciclo como para que el agente tenga tiempo de recibir la información actualizada y decidir que acción tomar antes de que termine el ciclo. En la figura 3.16 se muestra un ejemplo de aplicación de este método.

Algorithm 2 Seudo-código para el método de sincronización Internal Basic [DEBa].

{Main thread}
install signal that comes every 100ms

{Signal handler - called when signal arrives}
determine next action
send action command to server

3.6.2.3. Fixed External Windowing

El principal problema con el método anterior es que el envío de una acción al servidor no esta relacionada con la llegada de mensajes de información (física y visual) desde el servidor. Esto puede llevar a que el agente base sus decisiones sobre un estado del juego desactualizado. Sin embargo, si se utilizara un mecanismo externo de

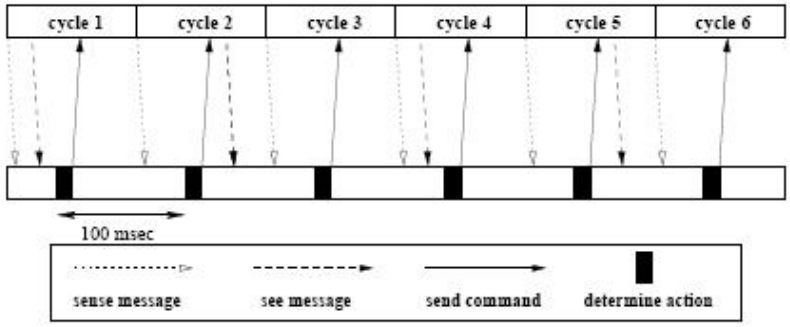

Figura 3.16: Ejemplo usando el método de sincronización Internal Basic [DEBa].

temporalización que permita relacionar el momento de envío de acciones al servidor con el comienzo del ciclo, este problema podría minimizarse. Esta idea es incorporada por el método *Fixed External Windowing* y se basa en el supuesto de que la llegada de un mensaje de información física indica el comienzo del ciclo (sección 3.6.1). De esta forma, cada vez que llega un mensaje de este tipo, una señal es seteada para que interrumpa 90 ms[34] más tarde. Durante este tiempo, el agente primero determina una acción a ejecutar en base a la información que tenga en ese momento. Si antes de que interrumpa la señal llega nueva información visual, el agente vuelve a determinar la mejor acción a realizar en base a esta información. Una vez que la señal interrumpe, el agente envía al servidor la última acción calculada. En el algortimo 3 se da el seudo-código para este método y en la figura 3.17 se muestra un ejemplo de su aplicación.

Como principal ventaja de este método se tiene que en los ciclos donde hay mensajes de información visual, el agente generalmente tiene la posibilidad de poder tomar sus decisiones con información actualizada del estado del juego. Una desventaja es que las acciones son enviadas al servidor muy cerca del fin del ciclo. En caso de que ocurra una mayor ocupación de la CPU o una sobrecarga en el trafico de la red, la acción podría llegar al servidor en el siguiente ciclo ocacionando los problemas planteados en la sección 3.6.

Algorithm 3 Seudo-código para el método de sincronización Fixed External Windowing [DEBa].

```
{Main thread}
while server is alive do
  block until server message arrives
  if type(message) == sense then
    set signal to go off after 90ms
  end if
  current action = determine next action
end while
{Signal handler (Act thread) - called when signal arrives}
send current action to server
```

3.6.2.4. Flexible External Windowing

Las dos principales características de un buen método de sincronización son: enviar una acción al servidor en cada ciclo y basar esas acciones en la información del estado del juego lo más actualizado posible. El método presentado anteriormente cumple con estas dos características pero con el riesgo de que algunas acciones lleguen al servidor en un ciclo incorrecto. Este problema se da porque el tiempo de espera para enviar una acción es fijo (90 ms). El método presentado en esta sección ofrece una política flexible en el tiempo de espera para el envío de acciones dependiendo de la llegada de mensajes de información visual. Esta política se basa en que la llegada de los mensajes de información visual cumplen el siguiente patrón cada tres ciclos (cuadro 3.4).

Utilizando este patrón es posible saber en cada ciclo si va a existir o no un mensaje de información visual y en caso de que lo haya, en que parte de ciclo va a llegar. Esto permite establecer un tiempo de espera para el envío

[34] La elección de 90ms se hace en base a los resultados obtenidos de la configuración nro. 4 del cuadro 3.3.

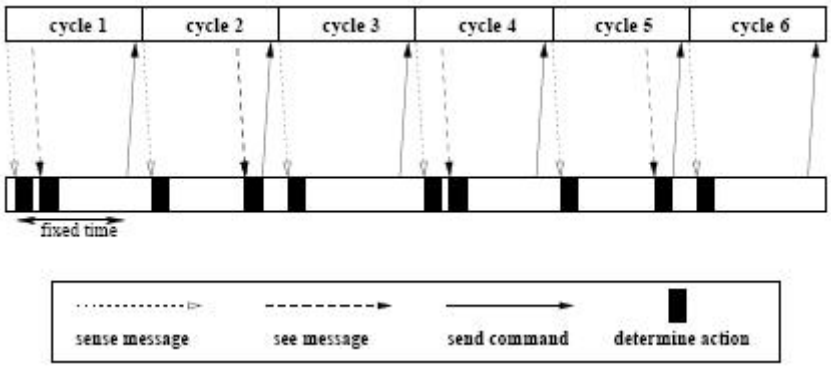

Figura 3.17: Ejemplo usando el método de sincronización Fixed External Windowing [DEBa].

	ciclo *x*	**ciclo *(x+1)***	**ciclo *(x+2)***
tiempo de llegada	1ª mitad del ciclo	2ª mitad del ciclo	no hay mensaje

Cuadro 3.4: Patrón que cumplen los mensajes del servidor con información visual [DEBa].

de acciones adecuado para cada situación. Claramente, no es necesario esperar 90 ms para el envío de la acción si se sabe que no va a llegar nueva información visual en lo que resta de este período. De esta forma, se puede tomar intervalos de espera distintos para cada uno de los tres casos. Un posible valor para el caso de que no haya mensajes visuales o que se den en la primera mitad de ciclo puede ser 70 ms, este valor nos asegura de que la acción va a llegar al servidor en el ciclo correcto. Para el caso en donde los mensajes llegan en la segunda mitad de ciclo se puede seguir manejando el valor de 90 ms, corriendo con los riesgos mencionados anteriormente. En el algortimo 4 se da el seudo-código para este método y en la figura 3.18 se muestra un ejemplo de su aplicación en donde se muestra en funcionamiento la política flexible del tiempo de espera.

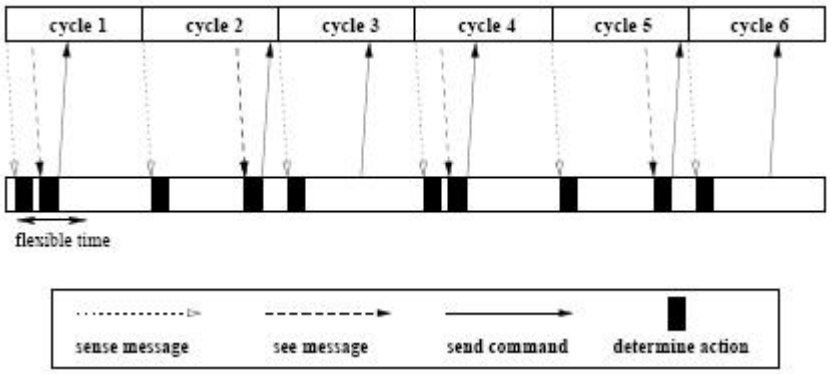

Figura 3.18: Ejemplo usando el método de sincronización Flexible External Windowing [DEBa].

Algorithm 4 Seudo-código para el método de sincronización Flexible External Windowing [DEBa].

```
{Main thread}
pattern index = 0 // 0 = see in 1st half, 1 = see in 2nd half, 2 = no see
while server is alive do
  block until server message arrives
  if type(message) == sense then
    sent message = false // new cycle starts
    if pattern index == 0 or pattern index == 2 then
      set signal to go off after 70ms
    else if pattern index == 1 then
      set signal to go off after 90ms
    end if
    pattern index = (pattern index + 1) mod 3
  else if type(message) == see then
    if no see message arrived in the previous cycle then
      pattern index = 1 // reset pattern: see in 2nd half in next cycle
    end if
  end if
  current action = determine next action
  if type(message) == see and sent message == false then
    set signal to go off immediately
  end if
end while
{Signal handler (Act thread) - called when signal arrives}
if sent message == false then
  send current action to server sent message = true
end if
```

3.7. La liga de Simulación 3D

La liga de simulación 3D [LSIM] comenzó en la competición oficial de RoboCup 2004, el objetivo principal fue poder realizar la competición y dar el puntapié inicial. Aunque el servidor cuenta con carencias muy grandes, la iniciativa entre la comunidad RoboCup fue muy bien recibida con la participación de muchos equipos. Entre los factores que han promovido con mas fuerza esta iniciativa, está el avanzar hacia un simulador físico real en tres dimensiones. Con esto se logra disminuir la brecha entre los robots físicos y los simulados e ir en dirección del objetivo planteado por RoboCup para el 2050. Otro aspecto importante pero no menos significativo, es realizar una competición más atractiva para los espectadores.

3.7.1. Características de un simulador físico

En esta sección describiremos los elementos más significativos del problema general de la simulación en ambientes físicos.

Modelado del Mundo

El modelado del mundo en este contexto es un gran desafío. La representación del mundo debe ser capaz de ser accedida por varios módulos de la simulación en tiempo real [KOG].

Por definición, *Un ambiente es un espacio tri-dimensional que contiene un conjunto de objetos. Cada objeto tiene una localización dentro del ambiente y puede ser percibido y manipulado por agentes. Un ambiente posee un conjunto de agentes, incluidos entre el conjunto de objetos.*

Existen conceptos importantes que surgen de esta definición:

- los agentes son siempre objetos del ambiente que pueden ser manipulados y percibidos entre ellos.
- todos los objetos tienen una localización en el ambiente (sistema de coordenadas en 3D).

- el ambiente esta definido por los objetos que están contenidos en él.

Representación de los Objetos

Los objetos en un ambiente físico tienen ciertas necesidades que deben ser cubiertas. Por ejemplo, para su visualización es necesaria cierta información de *como* el objeto debe ser visualizado. Llamamos a estas características, *aspectos.*

Aspectos de los Objetos

- *Aspecto Visual*: Este aspecto refiere a como se ve el objeto. Contiene toda la información necesaria para su visualización en una pantalla.
- *Aspecto Físico*: Considera el peso de los objetos y la forma en que actúan todas las fuerzas sobre los mismos (atracción gravitatoria, etc.).
- *Geometría*: Los objetos reales son irregulares, este aspecto considera la forma y el tamaño de los mismos. En general dentro de un sistema simulado esto es resulto a través de formas simples como esferas, cápsulas ó cajas. De la geometría del objeto surge como se lo puede identificar y como él colisiona con lo demás (modelo de colisiones).

Aspectos de los agentes En la definición de agentes aparece que los mismos son objetos. Los agentes agregan las capacidades de Sentir, Pensar y Actuar a diferencia del resto de los objetos, que no las poseen. De forma de representar estas capacidades adicionales, se define el término, *aspectos del agente.*

Estos aspectos incluyen: capacidad de *percibir* el ambiente (perceptors), *pensar* acerca de las próximas acciones y **actuar** (effectors).

3.7.2. SPADES

El Simulador de la liga 3D esta basado en SPADES como se muestra en la figura 3.19. Por ese motivo veremos brevemente algunas de sus características [RILb].

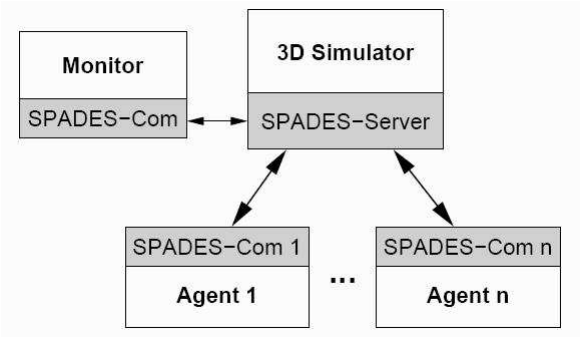

Figura 3.19: Arquitectura del Simulador 3D [MAR].

SPADES (System for parallel Agent Discrete Event Simulation) es un framework para construir simulaciones de agentes distribuidos.

- Es un híbrido entre sistemas de simulación basados en eventos y sistemas de simulación continuos.
- Implementa un mecanismo de ejecución basada en el agente. El agente se ejecuta dentro de un ciclo infinito: siento, pienso y actúo (figura)
- Procesamiento distribuido. Permite ejecutar los agentes en varias máquinas.
- Los agentes no están afectados por los retardos de la red o la carga del equipo donde se ejecuta. SPADES se encarga de que los eventos sean procesados en el orden adecuado.

- Permite la reproducción de las simulaciones.
- Resuelve la comunicación de bajo nivel. Los desarrolladores de agentes no se tienen que preocupar por sockets y direccionamiento. Además, es independiente del lenguaje de desarrollo.

Ciclo de ejecución de los agentes: Sentir-Pensar-Actuar

SPADES implementa el llamado ciclo sentir-pensar-actuar donde cada agente recibe sensaciones y responde con acciones. Esto significa que un agente solo es capaz de actuar luego de haber recibido un mensaje sensorial. Un agente puede requerir información sensorial, pero siempre una acción es precedida por información sensorial[35].

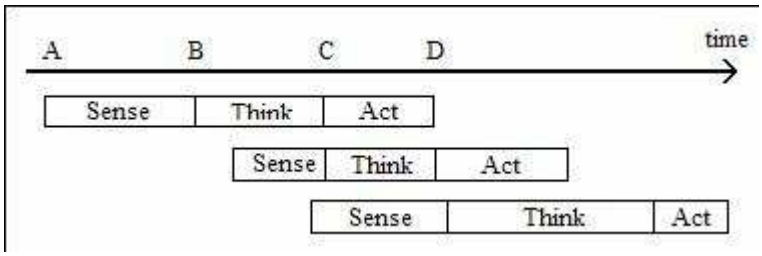

Figura 3.20: Ciclo Sentir-Pensar-Actuar [MAR].

La figura muestra el ciclo de ejecución y cuando cada uno de los componentes se ejecutan. Desde *A* hasta *B*, la información sensorial es enviada al agente, el agente decide que acción va a ser ejecutada. Desde *C* hasta *D* se envía al servidor la acción a ser ejecutada. En muchos agentes las sensaciones, pensamiento y acciones se solapan, solo existe una única restricción: los ciclos de pensamiento (think) no se pueden solapar.

3.7.3. Simulador 3D

A continuación se va a realizar una breve descripción de algunas de las características del servidor de la liga simulada 3D [S3D].

El servidor provee una plataforma para el desarrollo de robots simulados en un ambiente físico. Para entender su funcionamiento, debemos tener en cuenta el middleware de simulación SPADES [RILb] sobre el cual se desarrolla.

Un equipo consiste en un grupo de once robots con idénticas capacidades para la percepción de información y la realización de acciones.

Algunos datos técnicos

El simulador 3D posee un ciclo de simulacion al igual que el 2D. En este caso, la duración del ciclo es de 0.01s, 10 veces más que la frecuencia del simulador 2D.

Para el diseño de la cancha y otros elementos del juego,se usaron los estándares de FIFA[FIFA]:

- La cancha: largo entre 100 y 110 metros y ancho entre 64 y 75 metros. Arcos: 7.32 metros de ancho. Debido a que aún los agentes son pequeños y no pueden saltar, los arcos son de solo 0.5 metros de alto, mientras que el tamaño oficial en la FIFA es de 2.44 metros
- La pelota: tiene un diámetro de 0,222 metros y un peso entre 0.41 y 0.45kg

Los robots son representados por esferas (para el próximo año se espera tener una representación más sofisticada). El diámetro de todos los robots es de 0.44m y pesan 75kg. Los agentes en este momento no tienen una representación física y solo existe un único tipo de agentes el cual esta fijo. Otro punto importante y que tuvo que ser tenido en cuenta, es el valor asignado a la gravedad, el cual se fijo en $9.81 m/s^2$. El simulador 3D (al igual que el 2D) agrega ruido a los parámetros de los comandos y a la información de visón recibida por los agentes.

[35] SPADES permite explícitamente, el pedido de información sensorial vacía.

Figura 3.21: Monitor actual del simulador 3D [MAR].

Movilidad de los agentes

Los robots tienen un mecanismo especial para ayudar a dirigirlos, se les puede aplicar fuerza sobre los cuerpos y acelerarlos hacia alguna dirección. Dado que es posible saltar (muy poco por ahora), la aceleración solo es aplicada solo si el agente esta tocando el piso. Si se deja de aplicar fuerza al agente, el mismo se seguirá moviendo hasta detenerse. El vector de fuerza que recibe el comando que realiza esta acción (drive) esta en tres dimensiones y puede ser de hasta 100 unidades de potencia (la potencia máxima aplicable también depende de la batería ó energía asociada al agente).

La posibilidad de tener un comando *move* que permita que un robot se movilice desde un lugar a otro es rechazada. Por otro lado, existe el problema de que en los medios tiempos o inicio del juego, los jugadores deber ir a lado del campo de juego antes de un lapso de tiempo. Para esto se desea implementar jueces que puedan sacar tarjetas amarillas y rojas en caso que los jugadores no se comporten como debieran. Hasta el momento se permite la acción *move* solo antes del comienzo de cada tiempo.

Mover la pelota

Para mover la pelota, los robots simplemente pueden empujarla usando su cuerpo. Otra forma es utilizar la acción kick (este mecanismo artificial de pateo, rompe con el modelo físico real. En un futuro cuando se mejore la representación física de los agentes se desea eliminar). El comando kick, acelera la pelota a través de fuerza y torque (ó torsión) a la misma. El comando *kick* recibe el ángulo en la dimensión *z*. No es posible cambiar el ángulo aplicado en el plano xy (se debe girar el robot si se desea patear en otra dirección).

Dentro de las limitaciones actuales del simulador 3D, esta la carencia de la regla de juego offside y que los robots no se ven afectados por colisiones con la pelota.

Visión

La información recibida desde el servidor discrimina entre los tipos de objetos observados: otros robots, la pelota ó marcas del campo (8 en total, 4 de los arcos y 4 de los límites de la cancha). A diferencia del simulador

2D, el sistema de visión no brinda información a los agentes acerca de velocidades de los objetos e incluso estos pueden ocultarse detrás de otros.

Capítulo 4

Análisis de Equipos

Este capítulo presenta un resumen de las principales investigaciones y desarrollo de trabajos para la liga simulada de fútbol de RoboCup, competencia 2D. Concretamente se analizaron los equipos mejor rankeados en las últimas competencias de "RoboCup World Championship" condicionados a la disponibilidad de información. Además se incluyó el estudio de aquellas investigaciones que han sido pioneras en el área, pero aún siguen estando vigentes.

En el cuadro 4.5 se presentan los equipos que fueron estudiados para este trabajo junto con sus actuaciones más destacadas en competencias oficiales de la liga.

Equipo	Resultados en campeonatos
AT-Humboldt	1º en RoboCup-97 2º en RoboCup-98 7º en RoboCup-99
Brainstormers	1º en RoboCup-05 2º en RoboCup-00, 01, 04 3º en RoboCup-02, 03
CMUnited	1º en RoboCup-98, 99 4º en RoboCup-97, 00
FC-Portugal	1º en RoboCup-00, EuRoboCup-00 y Abierto de Alemania 2001 3º en RoboCup-01 5º en RoboCup-02,03
Mainz Rolling Brains	4º en el Abierto de Alemania 2003 5º en RoboCup-98,99
Uva Trilearn	4º en RoboCup-01,02 1º en RoboCup-03 1º en el Abierto de Alemania 2002, 2003, 2004 y 1º en el Abierto de Estados Unidos 2003

Cuadro 4.1: Equipos analizados en este trabajo y sus actuaciones más destacadas en competencias oficiales de la liga.

4.1. Mainz Rolling Brains

El equipo Mainz Rolling Brains [POMRB] fue creado en 1998 por integrantes del Departamento de Ciencias de la Computación de la Universidad de Johannes Gutenberg, Mainz - Alemania. Compite en la liga de Simulación 2D desde ese fecha y ha participado en todas las competencias oficiales hasta el año 2004. Sus mejores actuaciones se dieron en RoboCup-98 (5º), RoboCup-99 (5º) y en el Abierto de Alemania 2003 (4º). Desde el año 2004 se encuentran compitiendo en la liga de Simulación 3D.

Técnicas a Destacar

Este equipo se destaca por utilizar árboles de decisión basado en reglas para el control de las habilidades del jugador. La principal técnica de IA utilizada por el equipo es aprendizaje por refuerzo (Q-Learning), la cual es utilizada para la construcción de estos árboles.

Resumen de las Técnicas

Aprendizaje por Refuerzo

La idea básica de esta mecanismo es dar al agente un refuerzo ante la realización de una determinada acción. El refuerzo puede ser una recompensa (refuerzo positivo) o un castigo (refuerzo negativo) según que tan beneficioso haya sido el resultado de la acción. Dentro de esta área, existen diferentes técnicas. Q-Learning propone aprender la bondad de ejecutar una determinada acción en un determinado estado. Esta bondad está representada por un valor numérico denominado valor *Q* (refuerzo) [CER]. Las principales ventajas de este mecanismo son la capacidad de poder tomar decisiones secuénciales y el no requerir conocimientos previos. Como principal desventaja tiene que requiere una importante interacción con el ambiente para lograr un buen aprendizaje [CER].

Desarrollo de la Aplicación

Arquitectura

El principal aporte de este equipo ha sido su arquitectura de tres capas (Figura 4.1). La misma fue presentada en el año 2000 [MEY] y ha ido evolucionando con el trascurso de cada competición. La última versión de esta arquitectura es del año 2004 [ARN].

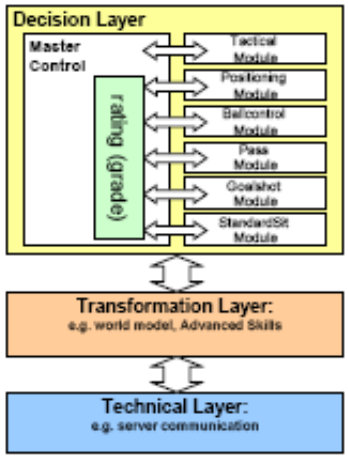

Figura 4.1: Arquitectura de los agentes del equipo Mainz Rolling Brains[ARN].

La capa *Technical* ofrece una interfaz para abstraer la comunicación con el SoccerServer. La capa *Trasformation* contiene la representación del modelo de la realidad y las distintas habilidades de los jugadores. Las habilidades se dividen según su nivel de complejidad: habilidades básicas (turn, kick, etc.) y habilidades complejas (intercept, dribble, etc.).
La capa *Decision* es considerada como el cerebro del jugador y tiene como cometido decidir cual es la mejor opción de juego que tiene el jugador en cada momento. Esta capa ha sido la de mayor evolución de la arquitectura y actualmente cuenta con seis módulos de comportamiento y uno de control (*Master Control*). Los módulos *Positioning*, *BallControl*, *Pass*, *Goalshot*, *StandarSit* se encargan de determinar los distintos comportamientos que el jugador puede tener. El módulo *Tactical* administra la información táctica (tipos de jugador, roles, etc.) de sus compañeros y de los oponentes. Por último, el *Master Control* es el que evalúa y decide que módulo tiene mayor prioridad de ejecución. A continuación se da una breve descripción de los pasos que debe seguir el *Master Control* en cada ciclo para la toma de una decisión:

- **feedback**

  Le solicita a cada módulo que evalúe los efectos de las decisiones y acciones realizadas anteriormente para actualizar sus esquemas y mecanismos de decisión.

- **evaluate**

  Le solicita a cada módulo (menos el Tactical) que evalúe el grado de conveniencia o urgencia de que este modulo realice una acción.

- **act**

  Le indica al módulo con mayor prioridad que ejecute su acción.

- **turnHeadEvaluate**

  Le solicita a cada modulo (menos el Tactical) que le indiquen que parte del campo o objeto desean tener información actualizada y cual es su urgencia.

- **tactical**

  Le solicita al módulo Tactical que evalúe la conveniencia de cambiar algún aspecto táctico del equipo (formación, tipo de jugadores, roles, etc.).

La principal ventaja de esta arquitectura se encuentra en el diseño de la capa Decision ya que ésta permite que fácilmente se incorporen nuevos módulos de comportamientos para ser tenidos en cuenta por el Master Control a la hora de evaluar la mejor acción a tomar por parte del jugador.

Roles del jugador

Desde hace ya unos años, el equipo cuenta con distintos roles para los jugadores: defensas, medio campistas y delanteros. Cada uno de estos roles tienen distintos objetivos y prioridades durante el trascurso de un partido. Por esta razón, cada módulo de comportamiento de la capa Decision tiene un árbol de reglas por cada rol que tenga definido el jugador.

Análisis del oponente

La principal característica en este sentido es el reconocimiento de la formación (figura táctica) del equipo contrario. Para lograr esto, desarrollaron un algoritmo que toma la ubicación de los jugadores del equipo contrario y las compara con las figuras tácticas más conocidas (preestablecidas de antemano en el algoritmo). Estas comparaciones dan como resultado un valor E, que representa el margen de error de la comparación según las desviaciones de cada jugador con la figura táctica modelo. La que tenga el menor valor E será tomada como la figura táctica del equipo contrario en ese momento. Esta tarea es llevada a cabo por el Coach del equipo y es una de las entradas para la evaluación de un posible cambio de formación de su equipo.

4.2. Brainstormers

El equipo Brainstormers [POBRA] fue creado en 1998 por integrantes del Instituto de Lógica y Sistemas Deductivos de la Universidad de Karlsruhe - Alemania y actualmente es desarrollado por integrantes del Departamento de Ciencias de la Computación de la Universidad de Dortmund - Alemania. Desde su creación, ha participado activamente de todas las competencias oficiales de RoboCup, teniendo las siguientes actuaciones destacadas: 2^o puesto en RoboCup-2000, 2001 y 2004, 3^o puesto en RoboCup-2002 y 2003, y 1^o puesto en RoboCup-2005.

Técnicas a Destacar

El principal interés del equipo es investigar y aplicar técnicas de *machine learning* en dominios complejos. Principalmente se centran en la utilización de la técnica de aprendizaje por refuerzo para los movimientos básicos de los jugadores (interceptar la pelota, posicionarse en la campo de juego, trasladarse con la pelota, etc.) y para la estrategia del equipo. Para aproximar el valor de la función que calcula el costo de ejecutar cierto movimiento o estrategia (que tan beneficioso resultaría realizarlo) se utiliza *redes neuronales*.

Resumen de las Técnicas

Markov Decision Process (MDP)

El proceso de aprendizaje puede ser formulado como un **P**roceso de **D**ecisión de **M**arkov (MDP) [MER]. MDP es una tupla *M:= [S, A, r, p]* donde *S* representa el conjunto de estados del entorno, *A* el conjunto de acciones disponibles en el sistema, *p(s´ | s,a)* describe el comportamiento del entorno, de forma que *p* es la probabilidad de que el entorno tome el estado *s´* si en el estado *s* se ejecuta la acción *a*. Finalmente, *r* representa el costo de tomar una determinada acción *a*.

Multi-Agent Markov Decision Process (MMDP)

Un MMDP [MER] puede ser definido como una tupla

$$M_n := [S, A, r, p],$$

donde *S* es el espacio de todos los estados, *A* representa al producto cartesiano del conjunto de acciones $A = A_1 \times A_2 \times \ldots \times A_n$ y *p* representa la probabilidad de transferencia de estado. Esto no difiere mucho de lo que es un MDP. La diferencia viene dada por la función de refuerzo *r.* En un problema MDP, esta función es de la forma: $S \times A \rightarrow R$ ya que es considerada para un solo agente que realiza una sola función. En un problema MMDP esto cambia debido a que la función *r* tiene en cuenta a todos los agentes del sistema. Por esta razón la nueva función *r* es de la forma:

$$r : S \times A \rightarrow R^n$$

Desarrollo de la Aplicación

Arquitectura

Hasta el año 2002 el equipo utilizo una arquitectura de tres capas: estrategia, políticas y movimientos [EHR] (Figura 4.2). La capa de estrategia se ocupa del comportamiento del equipo en el nivel más alto de abstracción. Ejemplos de estrategias pueden ser: juego ofensivo o juego defensivo. La capa de políticas es un nivel intermedio del comportamiento y se ocupa de cómo llevar a cabo la estrategia del equipo según las condiciones del juego. Ejemplos de políticas pueden ser: cuando el equipo tiene la pelota, cuando el equipo no tiene la pelota, cuando hay un tiro libre, etc. Por último, la capa de movimientos maneja las habilidades del jugador y como este las realiza. Ejemplos de habilidades pueden ser: interceptar la pelota, posicionarse en el campo de juego, patear la pelota, etc.

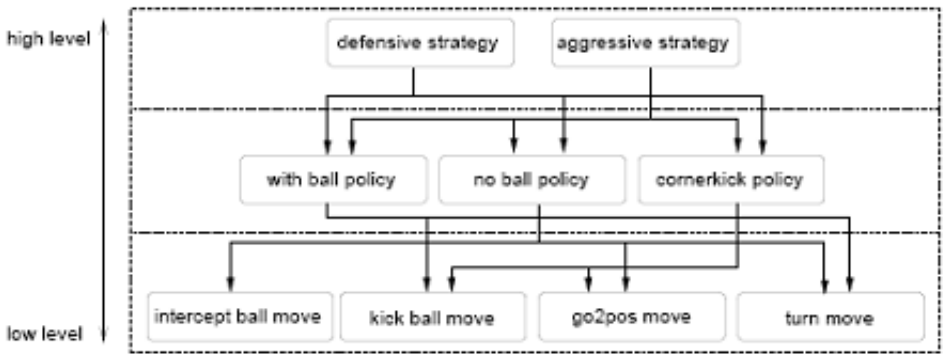

Figura 4.2: Arquitectura de los agentes del equipo Brainstormers hasta el año 2002 [EHR].

Esta arquitectura fue utilizada de forma exitosa durante varios años pero a lo largo de ese tiempo surgieron distintos problemas que llevaron al cuestionamiento de la misma. Algunos de estos problemas fueron:

- dificultad de decidir la ubicación de nuevas características (módulos) dentro de las capas definidas.
- las características del entorno (tiempo real, entorno dinámico, etc) han demostrado ser cada vez menos compatibles con este estilo de arquitectura.

Por estas razones, entre otras, decidieron evolucionar su arquitectura y basarse en estilos de arquitectura utilizados en los robots físicos (Figura 4.3). Esta arquitectura mantiene cierto grado de jerarquía pero elimina las restricciones de división por capas y el flujo de control top-down.

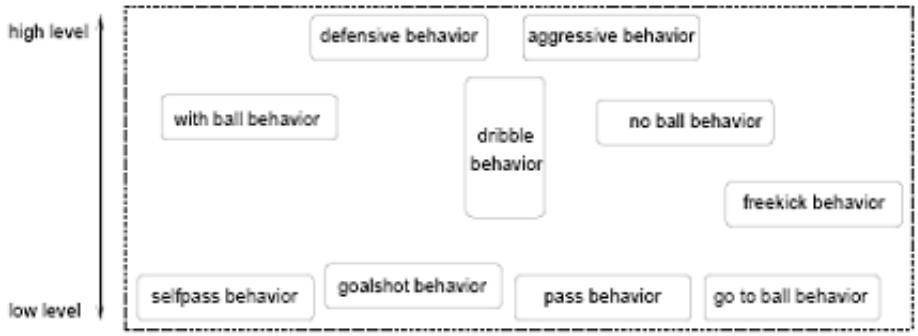

Figura 4.3: Arquitectura de los agentes del equipo Brainstormers a partir del año 2003 [EHR].

Movimientos

Los movimientos son las habilidades básicas de los jugadores y están compuestos por una determinada secuencia de comandos (turn, dash, kick, etc). Esta abstracción en el comportamiento del jugador permite disminuir la complejidad del problema de optimización en dos dimensiones: la cantidad de opciones a elegir en cada ciclo (de más 1000 opciones a unos 20 movimientos distintos) y la cantidad de ciclos que consume el llevar a cabo cierto objetivo (cada movimiento esta compuesto por unos pocos comandos) [BUC].

Actualmente, todos los movimientos del equipo están implementados utilizando la técnica de aprendizaje por refuerzo. Los principales movimientos del equipo son: patear, interceptar la pelota, trasladarse con la pelota, posicionarse en el campo de juego y parar la pelota [HOF].

En [MER] se plantea que el aprendizaje de los movimientos puede ser formulado como un problema de optimización dinámico (MDP). La idea es encontrar una serie de comandos básicos que minimicen el costo de ejecutar satisfactoriamente un movimiento. Con aprendizaje por refuerzo esto se realiza como sigue: en cada ciclo el agente elige una acción (comando) a. Esto causa un costo $r > 0$. Cuando el agente llega a un estado deseado S^+, la secuencia de acciones es terminada y se le asigna costo 0. En caso de que luego de ejecutar una acción a en un estado s se llegue a un estado que no permita cumplir con el objetivo del movimiento, se termina el movimiento y se le asigna un costo alto a la ejecución de la acción a en el estado s.

Como se mencionó anteriormente, se utiliza una red neuronal para el cálculo de la función de costo [MER]. Esto es hecho por dos razones: el espacio de estados es continuo y en principio tiene infinitos estados, y segundo, esta técnica les permite acelerar el proceso de aprendizaje de las habilidades. En [MER] se puede encontrar un ejemplo de cómo el jugador aprende a patear la pelota.

Estrategia del equipo

El aprendizaje de una estrategia de equipo puede ser formulado como un MMDP con aprendizaje individual por parte de los agentes participantes [EHR]. La idea de este enfoque es tener una función central de evaluación $V(s)$ para todos los jugadores donde se indique que tan favorable es el estado actual en pos de la estrategia del equipo. En otras palabras, $V(s)$ es una mapeo de un estado s a un valor entre [-1,1]. Cuanto más cerca este del valor 1 significa que el estado esta próximo a satisfacer el objetivo, un valor cerca del -1 significa que es probable que se fracase.

El estado final $s_n$de la secuencia de movimientos va a tener recompensa 1 ($V(s_n) = 1$) si se cumple con el objetivo y -1 ($V(s_n) = -1$) en caso contrario. El valor del resto de los estados de la trayectoria depende de la distancia que tengan del estado s_n. El valor de esos estados puede ser computados por:

$$V(s_i) = dacay(n-i) * V(s_n) \text{ con } i = 1..(n-1)$$

Los estados, junto con sus valores asociados, son usados para el entrenamiento de la función de aproximación. En su implementación utilizan un red neuronal que fue entrenada con una variación del algoritmo de retroalimentación llamado RPROP [BRA].

Este enfoque utilizado para la estrategia tiene como principal ventaja su flexibilidad ante la incorporación de nuevas habilidades a los jugadores. Lo único que se requiere es implementar en el modelo de la realidad la lógica que permita predecir el estado futuro ante la ejecución de la nueva habilidad incorporada[EHR].

Una de las restricciones de este enfoque es que el modelo de las acciones asume que el oponente no se mueve. Otra restricción es que los jugadores solo consideran sus propios conjuntos de acciones para la búsqueda de la acción óptima. Teóricamente un jugador debería tener en cuenta el conjunto de acciones $a = (a_1,...,a_n)$ de todos los jugadores participantes de la jugada (los atacantes o defensas, dependiendo de la situación del partido) [EHR].

4.3. CMUnited

El CMUnited [POCMU] fue creado por Peter Stone [STOa] y forma parte del desarrollo de su Tésis de Maestría [STOb] llevada a cabo en la Universidad de Carnegie Mellon. El equipo participó activamente desde el comienzo de la liga hasta 1999. Luego de esto, Peter Stone formó el equipo ATTUnited [POATT] (2001-2002) y actualmente se encuentra formando parte del equipo UT Austin Villa [POUTA]. Ambos equipos tiene como base al CMUnited. Las posiciones más destacadas del CMUnited fueron: 4º en la preRoboCup - 1996, RoboCup - 1997 y 2000[1], y 1º en la RoboCup - 1998 y 1999.

Este equipo, si bien no se actualiza desde 1999, ha sido considerado como el mejor equipo de la historia de la RoboCup y ha servido como base de varios de los equipos que han participado con gran suceso en la RoboCup desde ese entonces.

Técnicas a destacar

Este equipo tiene como principal contribución la generación de un nuevo paradigma dentro de las técnicas de machine learning llamado Layered Learning [STOb], este paradigma es de propósito general y permite su aplicación en dominios complejos como lo es el fútbol de robots.

Resumen de las Técnicas

Layered Learning

Es un paradigma de machine learning definido por un conjunto de principios que le permiten la construcción de una solución a una tarea compleja basada en una jerarquía de aprendizaje [STOb]. Esta definida por cuatro principios:

Principio 1: Motivada por el fútbol de robots, esta técnica fue diseñada para dominios que son muy complejos como para aprender directamente del mapeo entre los sensores de entrada del agente y sus efectores de salida. Por esta razón, el paradigma básicamente consiste en dividir el problema en varias capas de comportamiento, usando técnicas de machine learning en cada una de ellas. Utiliza una descomposición de tareas de forma jerárquica (bottom-up), donde las capas inferiores resuelven el comportamiento de más bajo nivel. El proceso de ir creando nuevas subtareas (nuevas capas) continúa hasta que se resuelve el comportamiento de más alto nivel que cubra toda la complejidad del dominio.

Principio 2: La granularidad en la división de los comportamientos y los aspectos que se quieren aprender de éstos, están determinados en función del dominio en el que se quiera aplicar este paradigma. La tarea de descomponer en capas de aprendizaje no es automática.

[1] El equipo fue el mismo que el del año 1999 por haber sido el campeón en ese año. Esto se hace para medir el avance de los equipos de una competición a otra.

Principio 3: Machine learning es usado como la parte central de este paradigma en lo que respecta al análisis de los datos para el entrenamiento o adaptación de todo el sistema. Machine learning es muy usado para el entrenamiento de comportamientos que son difíciles de realizar manualmente. También es usado para la adaptación cuando el entorno no es completamente conocido y cambia dinámicamente.

Principio 4: La característica principal de esta técnica o paradigma es que el aprendizaje de cada capa afecta al aprendizaje de la capa superior.

Desarrollo de la aplicación

Layered Learing aplicado en el CMUnited

Como parte de sus tésis, Peter Stone aplicó este nuevo paradigma en el CMUnited. En [STOb] se plantea un posible conjunto de niveles de aprendizaje de comportamiento para el fútbol de robots (Cuadro 4.2). La capa de más abajo contiene las habilidades de bajo nivel de los agentes como pueden ser: interceptar la pelota. La segunda capa contiene comportamiento a nivel de multi-agente (requiere interactuar con otros agentes). Un ejemplo de comportamiento de esta capa puede ser: evaluación de pase. La tercera capa maneja el comportamiento cooperativo del equipo como puede ser: selección del pase. La cuarta capa se ocupa de la formación del equipo dentro del campo de juego. Por último, se tiene una capa que analiza el comportamiento del equipo contrario para evaluar una posible adaptación de la estrategia o formación del equipo en el partido.

Capa	Nivel de estrategia	Tipo de comportamiento	Ejemplo
1	habilidades básicas	individual	intercepción de la pelota
2	jugador a jugador	multi-agente	evaluación de pase
3	uno a muchos jugadores	equipo	selección de pase
4	formación del equipo	equipo	estrategia de posicionamiento
5	equipo contrario	adversario	estrategia de adaptación

Cuadro 4.2: Ejemplos de distintos niveles de comportamiento en el fútbol de robots utilizando Layered Learning [STOb].

Analizando la división de capas realizada anteriormente, se puede observar que cada uno de los comportamientos de una capa utiliza los comportamientos o habilidades de la capa inferior. Para la evaluación de un pase se utiliza el conocimiento de interceptar la pelota para evaluar el éxito o fracaso de dicho pase. Para la selección de un pase se evalúan todos los posibles pases que tiene el jugador, la estrategia de posicionamiento se entrena con la selección de pase, y estas dos últimas sirven para determinar el comportamiento del equipo contrario.

Actualmente el CMUnited solo contiene tres subtareas de aprendizaje correspondientes a las tres primeras capas del Cuadro **4.2.** En el Cuadro 4.3 se muestran junto con las técnicas de machine learning utilizadas en cada una de ellas. El comportamiento de selección de pase de la capa 3 ha sido entrenada utilizando un nuevo método de aprendizaje por refuerzo multi-agente llamado TPOT-RL[2] junto con la habilidad de evaluación de pase de la capa inferior.

Capa	Comportamiento	Método de aprendizaje
1	intercepción de la pelota	redes neuronales
2	evaluación de pase	árboles de decisión
3	selección de pase	TPOT-RL

Cuadro 4.3: Métodos de aprendizaje utilizados por el CMUnited en cada una de las capas [STOb].

Otras características importantes del CMUnited son:

- Los agentes utilizan memoria predictiva que les permite tener un modelo preciso y certero del estado actual del campo de juego en cada momento, permitiéndole modelar de forma probabilística las partes del campo que no puede visualizar.

[2] **T**eam-**P**artitioned **O**paque-**T**ransition **R**einforcement **L**earning: este método es usado para maximizar la recompensa en un ambiente muti-agente donde los agentes tienen información limitada del estado del ambiente.

- Implementaron un protocolo de comunicación que les permite de forma eficiente y confiable la comunicación entre los jugadores de su equipo. Este protocolo es utilizado para asegurar la coordinación del equipo.
- Puede cambiar dinámicamente la formación del equipo y los roles de los jugadores.
- Los agentes utilizan un método llamado SPAR[3] [BOW] para determinar el posicionamiento estratégico dentro del campo de juego. Cuando se utiliza este método para el posicionamiento, el agente usa una función multi-objetivo con puntos de atracción y repulsión. De esta forma, la función busca maximizar la distancia a otros jugadores y minimizar la distancia a la pelota y al arco contrario.
- Los agentes modelan el comportamiento del oponente para que la toma de decisiones se base según las características del oponente.
- Los agentes cuentan con "jugadas predefinidas" (set-plays) para ser utilizadas en situaciones que ocurren repetidamente durante un juego (corners, tiros libres, etc.).

4.4. FC-Portugal

El equipo FC-Portugal [POFCP] se ha destacado desde que se desarrolló en el año 2000, año en el cual obtuvo el primer lugar en el campeonato europeo EuRoboCup2000 [CRAE00] y en el campeonato oficial RoboCup-2000 [4]. También logró el primer puesto en el abierto de Alemania del año 2001, y un tercer lugar en el campeonato RoboCup-2001. En los campeonatos oficiales de 2002 y 2003 obtuvo el 5º puesto, en 2004 el 6º y en 2005 terminó en el 12º lugar.

Este equipo fue desarrollado por tres estudiantes de la Universidad de Aveiro [DETUA] y Porto [LIACCUP]. Las habilidades básicas del equipo estan basadas casi enteramente en el equipo campeón de RoboCup-1999, CMUnited, perimitiéndoles así concentrarse en perfeccionar las habilidades de alto alto nivel. Desde el año 2000 el equipo FC-Portugal se ha convertido en un equipo referente de la liga simulada.

A continuación se presenta un breve resumen de las técnicas utilizadas por este equipo, las mismas se desarrollan con mayor profundidad en [LAUa, LAUb, LAUc, COS].

Técnicas a destacar

Entre las técnicas utilizadas por FC-Portugal se destaca *Situation Based Strategic Positioning* ó Posicionamiento Estratégico Basado en Situación (de aquí en mas, SBSP). Innovaron en el año 2000 con un mecanismo de posicionamiento que distingue entre situaciones activas y situaciones estratégicas. En situaciones estratégicas los jugadores utilizan SBSP para calcular la posición estratégica la cual se basa en el tipo de jugador, el estado del juego, la táctica que se esta empleando y la formación del equipo. Para situaciones activas, la posición del jugador es calculada utilizando un mecanismo que difiere según se este en posesión, recuperación ó disputa de la pelota.

Otra técnica utilizada es la descripta como Posicionamiento Dinámico e Intercambio de Roles (DPRE) la cual permite intercambiar posiciones y roles en la formación del equipo durante el partido.

Por último, este equipo se caracteriza por la utilización de un mecanismo inteligente de comunicación (ADVCOM) y otro de percepción estratégica (SLM). Por otra parte, sus agentes utilizan la técnica de árboles de decisión para seleccionar la acción apropiada en una situación dada.

Resumen de las técnicas

Situation Based Strategic Positioning - SBSP

SBSP es un sistema de posicionamiento desarrollado para situaciones no críticas (cuando no es inminente realizar una acción). La posición estratégica es calculada dinámicamente analizando la situación del juego: la táctica, la formación del equipo y posiciones de los jugadores. En base a estos datos se calcula la posición estratégica del jugador para esa formación. Esta posición calculada es ajustada de acuerdo a la pelota (posición y velocidad), la situación e información estratégica asociada al rol del jugador (atacante, defensivo, oportunidad de gol, etc.). De

[3] **S**trategic **P**ositioning by **A**ttraction and **R**epulsion

[4] Entre el EuRoboCup-00 y el RoboCup-00 convirtió 180 goles y no recibió ningún gol en contra.

esta forma, sin información sensorial (visual o sonora) un jugador a través de un análisis de situación puede saber el posicionamiento de todos sus compañeros en el campo de juego (es lo que podemos ver en el fútbol de humanos). Esta técnica asegura que durante todo el partido los jugadores van a estar correctamente distribuídos en el campo de juego, tendrán una buena cobertura de las posiciones defensivas y contarán con múltiples opciones de ataque.

Mecanismo de Posesión, Recuperación y Disputa de la Pelota.

Casi todas las decisiones individuales que debe realizar un agente surgen del conocimiento adquirido bajo el dominio del fútbol. Dentro de estas decisiones están: qué hacer cuando se está en poseción de la pelota (pasarla, patear al arco, pasar la pelota hacia adelante para que algún compañero corra y pueda alcanzarla, retener, dribliar[5], etc.), cuando no (marcar un oponente, marcar un pase, ir hacia la pelota, cubrir el arco o desmarcarse, etc.), o cuando la pelota esta en disputa. Básicamente el mecanismo evalúa entre las mejores opciones de todas las alternativas. Por ejemplo, si se considera el caso de posesión, se buscará la mejor opción de pase. Las opciones se califican según un sistema de bonos y penalizaciones. Luego se evalúa nuevamente entre las mejores opciones de cada alternativa y se elige la mejor.

Por ejemplo, cuando se debe determinar que pase realizar, se utiliza una matriz con algunas de las características de los tipos de pase. Los tipos de pase pueden ser: directos a un jugador destino, pase por líneas de bajo congestionamiento, pases a puntos oportunos con bajo congestionamiento cerca de un jugador y pases a la posición esperada de un jugador dada la situación del juego. La evaluación y selección del mejor pase se realiza en base a el valor de la posición, congestión de las líneas, confianza, congestión final, etc.

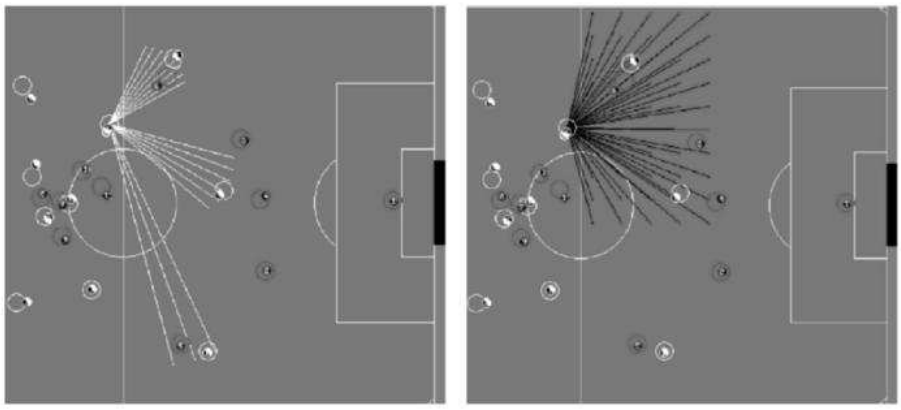

Figura 4.4: Ejemplo de evaluación de pase a jugador y pase hacia delante (forwards)[LAUc].

Dentro del mecanismo de recuperación se consideran alternativas del tipo: marcar la pelota, marcar a un oponente, cubrir el arco (o un gol), cubrir una línea de pase, etc.

Posicionamiento Dinámico e intercambio de roles - DPRE

DPRE permite el posicionamiento e intercambio de roles entre los agentes. Esta técnica esta basada en un trabajo previo de Peter Stone [STOb, STOc] donde se sugiere el uso flexible de los roles en los agentes y plantea protocolos para su intercambio. Con DPRE se realizan los intercambios de roles o posiciones entre los jugadores cuando es beneficioso para el equipo. La herramienta de intercambio de posiciones utiliza la distancia desde la posición actual del jugador a su posición estratégica y la importancia de esta posición dentro de la formación del equipo. A diferencia con la técnica implementada por Peter Stone [STOa] para el equipo CMUnited-99, la cual no fue de gran ayuda (se descarto para el RoboCup-99), los resultados experimentales de DPRE incrementaron significativamente la performance del equipo[6] (los jugadores retuvieron mayor nivel de energía y hubo menos posiciones estratégicas sin cubrir). El resultado de la técnica se destaca en los casos en los cuales se debe jugar con menor cantidad de jugadores.

[5] El término driblin se refiere a patear la pelota manteniendo su control

[6] FC-Portugal con DPRE gano el 85 % de la veces contra FC-Portugal sin DPRE.

Mecanismo Inteligente de Comunicación - ADVCOM

La comunicación sobre canales con ruido, con muy poco ancho de banda y alcance limitado debería usarse solo en casos que sea beneficioso desde un punto de vista global del equipo. El principal desafío de esta investigación es saber *qué y cuándo comunicar*. FC-Portugal utiliza la comunicación con dos cometidos:

1. Actualizar el estado del mundo de sus agentes compartiendo el estado del mundo individual.
2. Aumentar la coordinación del equipo comunicando eventos relevantes (por ejemplo el cambio de posición entre jugadores).

El principal aporte de la técnica es la utilización de un mecanismo de modelado de equipo y de información de estado (como velocidad de la pelota) para evaluar la utilidad de la comunicación y de esta forma decidir si comunicar o no. Básicamente los agentes comunican cuando creen que la utilidad de la información es mayor para alguno de sus compañeros. Por ejemplo porque no hay mejores observadores de un evento dado.

Percepción Estratégica - SLM

En un dominio tan complejo y dinámico como el del fútbol, la información sensorial debe ser utilizada y coordinada de forma inteligente para poder mantener un preciso estado del mundo. Se debe considerar que el interés acerca del estado del mundo para cada agente puede variar según su situación. Para un jugador que se acerca al arco contrario con la pelota, la información de donde se encuentra el arquero, hacia donde se mueve y con que velocidad es una parte muy importante del estado del mundo, mientras que la posición de los jugadores que se encuentran en el medio del campo tendrá menos interés para él. El mecanismo de percepción inteligente de FC-Portugal decide la dirección de la mirada del jugador de acuerdo a la situación en que se encuentra y a la confianza de los datos que posee del resto de los objetos del campo de juego[7]. Para cada una de las direcciones posibles, se realiza una medida de utilidad y se mueve el cuello en la dirección mas útil.

Otras Características

Modelado del Oponente

Este módulo analiza el juego y utiliza máquinas de aprendizaje asociadas a simples heurísticas para aprender algunos parámetros y características del oponente. Estos parámetros pueden incluir: el número del arquero, la posición del arquero en los saques de arco y que velocidad y dirección le aplica a la pelota, la dirección y velocidad de los tiros al arco, formación del oponente y posición predecible de los jugadores oponentes en cada situación. Esta técnica está distribuida entre los agentes jugadores y el agente coach [LAUa].

Estrategia Defensiva del Arquero

El comportamiento del arquero es uno de los más afectados por el uso de un simulador de dos dimensiones dado que no existe la posibilidad de levantar la pelota por encima del mismo. FC-Portugal mantiene casi siempre el arquero dentro de los límites del área chica, su comportamiento considera los siguientes casos [LAUc]:

- **Marcar posición**: El arquero se ubica cerca del límite del área del penal y utilizando una visión estrecha mantiene la vista en la pelota. Busca ubicarse en la línea conformada entre la pelota y el punto medio del arco.
- **Intercepción activa y pasiva**: Pasiva es cuando no tienen oponentes cerca (adaptada de CMUnited) y activa es usada en una situación más disputada.
- **Intercepción de un gol**: Es cuando el arquero alerta que la pelota va rápidamente hacia el arco y debe interceptarla para evitar el gol.
- **Atajar**: Es cuando la pelota esta tan cerca que puede agarrarla.

[7] CMUnited propuso un método en el que cada objeto visualizado se le da un valor de (1.0), este valor decae con el tiempo cuando el objeto deja de ser visualizado. FC-Portugal agregar un factor de precisión relacionado con la distancia del objeto observado, dado que aumenta el ruido con la distancia.

Habilidades Básicas

Las habilidades básicas del equipo fueron casi enteramente tomadas del equipo CMUnited-99 [POCMU] [8]. Solo se hicieron las siguientes modificaciones: FC-Portugal implementó una poderosa habilidad de pateo basado en técnicas de optimización online (la habilidad de pateo es estratégica en el logro del gol, además de que garantiza la movilidad de la pelota con los pases), también mejoró la habilidad de driblin y movimientos del arquero (observando la pelota) [LAUc].

Configuración del equipo y Coach

Las estrategia de alto nivel de FC-Portugal es fácilmente configurable y por tanto flexible para distintos tipos de oponentes. Se pueden cambiar antes o durante el partido, si es antes se realizan a través de archivos de configuración. Antes del juego un coach humano define la estrategia a utilizar, así mismo, a través del modelado del oponente el coach puede cambiar la estrategia durante el partido[LAUd].

Técnica de Evaluación de Tiros al Arco basado en los Movimientos del Arquero

Para el 2003, FC-Portugal incorporó un mecanismo para que los jugadores puedan seleccionar el ángulo de tiro que le proporcione mayores posibilidades de anotar, este trabajo se puede ver en [LAUe].

4.5. UvA Trilearn

Historia del Equipo

El equipo UvA Trilearn [POUVA] de simulación fue desarrollado por dos estudiantes de la Universidad de Amsterdam [UAM] para su proyecto de maestría en 2001. En el marco de su proyecto, decidieron no reutilizar código de otros equipos y se enfocaron en el desarrollo de las habilidades de bajo nivel de los agentes[9], logrando generar un equipo con bases sólidas para el futuro. Luego, en años posteriores continuaron en el desarrollo de las habilidades de alto nivel del equipo. Su inicio en la RoboCup fue importante, logrando un 4º puesto en los campeonatos oficiales de 2001 y 2002. Posteriormente lograron ganar los abiertos de Alemania 2002 y 2003, el campeonato oficial 2003 y el abierto de Estados Unidos 2003.

Técnicas a Destacar

UvA comenzó en el 2001 y hasta la fecha ha incorporado distintas características. En sus comienzos en 2001 se destacan: habilidades de bajo nivel muy robustas, una arquitectura de tres capas multi-hilada, un avanzado mecanismo de sincronización descrito buena parte en 3.6, un mecanismo efectivo para determinar el punto óptimo del arco donde patear la pelota para convertir el gol (el mismo se describe como "política óptima de score"). En 2002 mejoran su sistema de localización utilizando filtrado de partículas y atacan el problema de la selección de acciones [DEBd]. Para el año 2003 (año que lograron ganar varios campeonatos) realizaron mejoras considerables en las habilidades de interceptar la pelota y en la evaluación de opciones de pase [GROa] e incorporan un método de coordinación para los agentes llamado *"Grafo de Coordinación"* (CG) el cual continúan mejorando hasta el día hoy. Otra de las técnicas de alto nivel destacada del equipo UvA, es la llamada "Mutual Modeling of Teammate Behavior" (MMTM) ó "Modelado del Comportamiento Mutuo de los Jugadores" [KOKa] que esta muy ligada a SBSP del equipo FC-Portugal descripta en 4.4.

A continuación desarrollamos dos de las técnicas, que consideramos más relevantes.

Resumen de las Técnicas

Política óptima de score

El objetivo de esta técnica es determinar el punto óptimo del arco al momento de patear al mismo. La idea es poder obtener una probabilidad de conversión del gol asociada a:

- La posición desde donde se patea la pelota.

[8] También se tomó parte del código del modelado del mundo y del action effect predicting.

[9] La estrategia de alto nivel utilizada en el 2001, fue basada en el equipo FC-Portugal, ganador del RoboCup-2000.

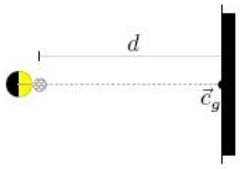

Figura 4.5: Posición utilizada en el experimento para calcular el ruido acumulado de la pelota[DEBa].

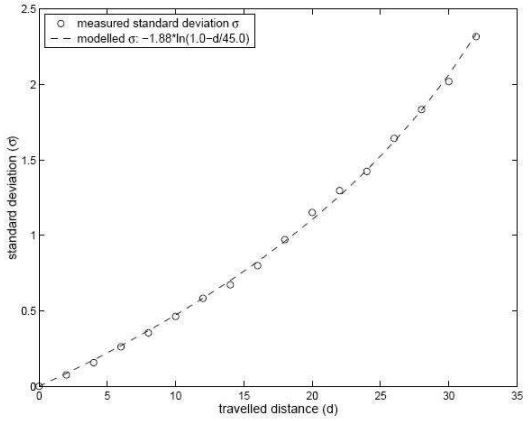

Figura 4.6: Desviación estándar de la pelota en función de la distancia recorrida.

- La posición del arquero.

De esta forma, se puede tener un mecanismo para determinar cual es el punto óptimo que nos conviene elegir para patear al arco ó también decidir si la mejor opción es patear o pasar la pelota a un compañero que tenga una posición más ventajosa para convertir el gol.

Para resolver el problema óptimo de score, se lo dividió en dos sub-problemas independientes:

1. La probabilidad de que la pelota entre al arco por un punto determinado pateando desde una posición dada.
2. La probabilidad de que el arquero pueda interceptar la pelota en una situación dada.

Dado que los problemas con considerados independientes, la probabilidad total es el producto de ambas probabilidades.

Subproblema 1 - Probabilidad de que la pelota entre el arco. Este problema se atacó, calculando la desviación obtenida del punto original elegido donde patear la pelota. En principio este problema puede parecer sencillo en un ambiente como el simulado donde poseemos un comando *kick* que recibe el ángulo con el que se quiere patear, el problema surge por el ruido incorporado por el simulador a los parámetros del comando *kick* como también al vector de velocidad aplicado la pelota (en cada ciclo del simulador). Existen dos formas de resolver el problema: Una de ellas es analítica, se puede tomar la formula utilizada por el simulador para incorporar el ruido[10] y luego de cálculos muy complejos llegar al resultado. La complejidad de realizar estos cálculos en cada situación, lleva a la alternativa de estimar el ruido acumulado a través de la experimentación.

Se colocó a un jugador de frente al arco, con dirección perpendicular a la línea de gol como se muestra en la figura 4.5. Se experimento con distancias *d* de entre 0 y 32 metros y se realizaron 1000 pateos de cada una de las distancias siempre pateando al medio del arco.

Se pudo comprobar empíricamente que la desviación σ para cada una de las distancias es diferente. Se comprobó que la misma es una función de la distancia, monótona creciente. Se realizó una aproximación a través de:

[10] El vector de velocidad (v_x^{t+1}, v_y^{t+1}) en el ciclo *t+1* es igual a: 0.94 * (v_x^t, v_y^t) + $(\widetilde{\nabla}_1, \widetilde{\nabla}_2)$ donde $\widetilde{\nabla}_1 y \widetilde{\nabla}_2$ son números aleatorios con distribución uniforme en [-rmax, rmax], con rmax = 0.05 * $\|(v_x^t, v_y^t)\|$.

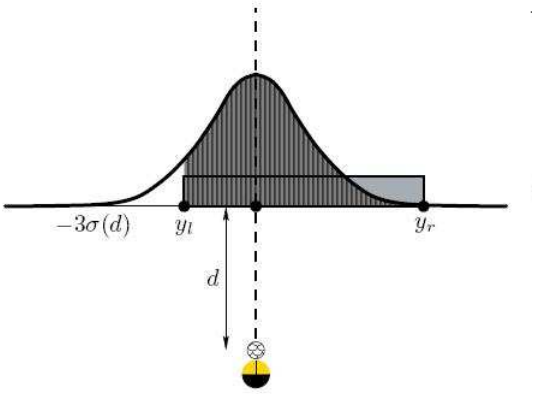

Figura 4.7: Distribución para tiros perpendiculares a la línea de gol [DEBa, DEBc].

$$\sigma(d) = -1.88 * \ln(1 - \frac{d}{45}).$$

En siguientes pasos de los estudios expuestos en [DEBa, DEBc] se calculo la distribución de la pelota, de alcanzar la línea de gol. Los resultados obtenidos se lograron aproximar por una distribución Gaussiana como se muestra en la figura 4.7 para tiros perpendiculares a la línea de gol. Luego, estos cálculos se llevaron al caso general donde el ángulo de pateo puede variar.

Subproblema 2 - Probabilidad de sobrepasar al arquero. La posibilidad de interceptar la pelota, es mucho más fácil para el arquero, debido a que tiene la posibilidad de agarrar la pelota, por lo tanto para que la pelota entre al arco, debe pasar a una distancia mayor que la distancia en la que el arquero puede agarrar la pelota (max_catchable_distance). El problema a resolver depende fuertemente del comportamiento implementado para el arquero, a no ser que pueda determinarse el comportamiento óptimo (no es el caso de este estudio), es necesario optar; para este estudio te tomo el arquero de FC-Portugal, uno de los mejores disponibles (equipo campeón del último campeonato al momento de la realización de este estudio).

Este problema se atacó, como un problema de clasificación de dos clases. Dado un punto de pateo, la posición del arquero y la pelota, queremos predecir en cual de las dos clases vamos a estar:

1. Clase 1 - La pelota fue interceptada
2. Clase 2 - La pelota NO fue interceptada

Mas aún, lo que interesa es la probabilidad asociada a la futura pertenencia a alguna de ambas clases. Para realizar el entrenamiento, se utilizó una posición fija para el jugador (frente al arco) y se vario la posición relativa del arquero con la pelota en forma aleatoria. Se utilizaron con conjunto de 10.000 resultados, los mismos revelaron que las características importantes para la clasificación son: el ángulo absoluto entre el arquero y el punto de pateo (a), la distancia entre el arquero y el jugador (d). En la figura 4.8 podemos observar que la las clases pueden dividirse aproximando a través de una función discriminante lineal.

Como se puede ver en [DEBa, DEBc], la función discriminante se puede expresar como:

$$u = (a - 26.) * 0.043 + (d - 9.0) * 0.09 - 0.2 = 0.043 * a + 0.09 * d - 2.1323$$

El siguiente paso fue profundizar el estudio para llegar a la función de probabilidad (como se desarrolla en [DEBa, DEBc]), el resultado fue el siguiente:

$$\text{P(pasar al arquero / } u \text{)} = \frac{1}{1+\exp(-9{,}5)u}$$

Como se puede observar, ambos problemas se pueden calcular con funciones de baja complejidad. Como ya se menciono, la probabilidad total, es el producto de la probabilidad para cada uno de los problemas resueltos. Veamos los resultados para los casos: que el arquero cubre bien el arco (figura 4.9) y cuando el jugador tiene buenas posibilidades de convertir el gol (figura4.10).

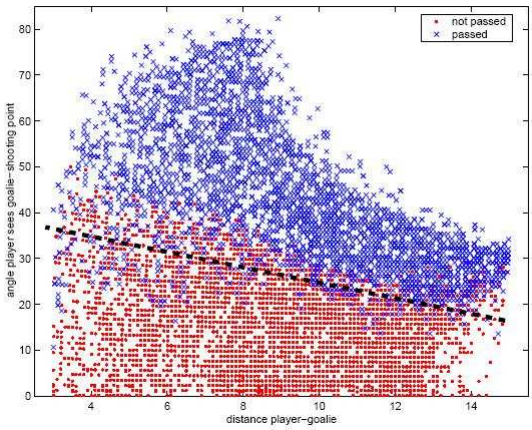

Figura 4.8: Conjunto de datos y función discriminante [DEBa, DEBc].

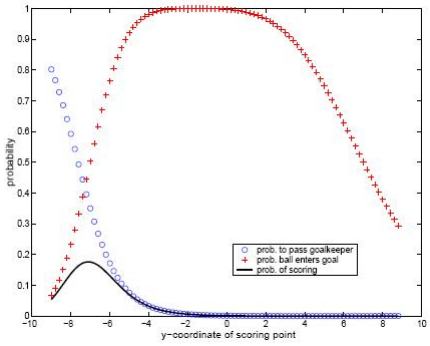

Figura 4.9: Caso 1 - El arquero cubre correctamente el arco [DEBa].

Grafo de Coordinación (CG)

En 2003 el equipo UvA Trilearn incorporó un mecanismo de coordinación para los agentes denominado "*Grafo de Coordinación"* (CG) [KOKc], posteriormente lo perfeccionó en 2005 [KOKe, KOKd].

Dependiendo de la situación, los agentes deben coordinar sus acciones dentro del campo de juego. Por ejemplo: un agente con la pelota debe decidir a cual de sus vecinos pasarla, los atacantes deben prepararse para recibir pases futuros y los defensas deben coordinar sus acciones para poder bloquear al oponente con mayor efectividad.

Desde un punto de vista teórico, la estrategia de juego se puede ver como una tupla: (n, $A_{1..n}$, $R_{1..n}$) donde n es el número de agentes, A_i es el conjunto de acciones que puede realizar el agente i y R_i es el valor que retorna la función de utilidad del agente i. Esta función de utilidad aplicada sobre el conjunto de acciones A=A_1x...xA_n nos retorna un valor real. Cada agente selecciona una acción y la misma tiene valor de utilidad. El objetivo de los agentes es seleccionar la acción (a través de sus propias decisiones) que sea más rentable para el conjunto.

Sigamos con un ejemplo práctico. Para llevar a cabo la coordinación, inicialmente a todos los agentes se les asigna un rol, en base a lo que perciben de su contexto (los roles nos permiten reducir el espacio de acciones posibles de un agente). Supongamos la coordinación que se debe realizar para decidir un pase a un compañero, los roles participantes son:

- **el que intercepta**: el jugador que se encuentra mas cerca de la pelota pero que no puede patearla
- **el que pasa**: el que puede patear la pelota
- **el que recibe**: los jugadores que se encuentran dentro de un radio próximo a la pelota
- **el pasivo**: los jugadores que se encuentran lejos de la pelota

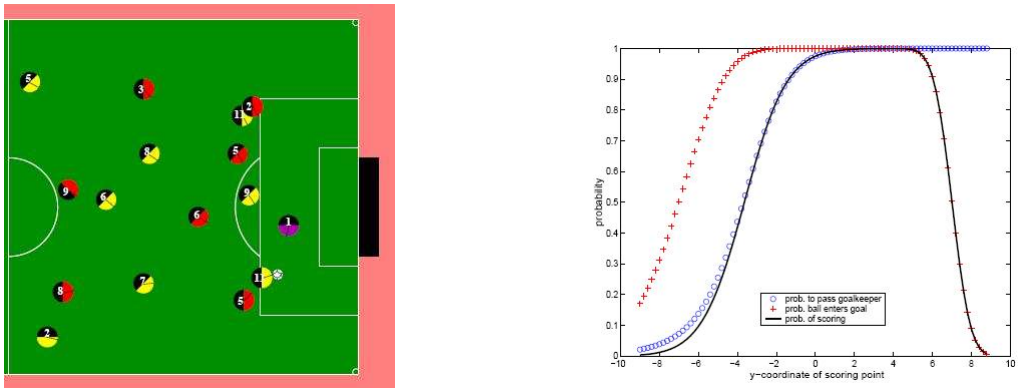

Figura 4.10: Caso 2 - El jugador posee una buena opción de gol [DEBa].

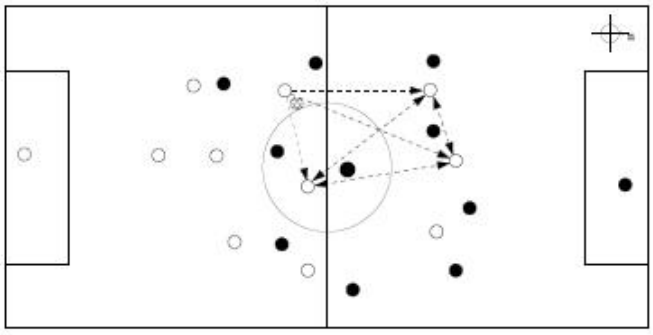

Figura 4.11: Ejemplo de una situación donde se aplica CG[KOKc].

En la figura4.11 podemos ver el grafo de coordinación donde: él jugador que posee la pelota tiene el rol de *pasador* (el que pasa), los tres conectados al anterior son *recibidores* (que pueden recibir) y el resto son *pasivos* (no pertenecen al grafo). Ahora bien, todos los agentes conectados deben coordinar sus acciones y para esto, cada agente puede seleccionar una de las siguientes acciones:

- *passTo(i, dir)*: pasar la pelota en dirección *dir* relativa al agente *i*.
- *moveTo(dir)*: moverse en dirección *dir.*
- *dribble(dir)*: moverse con la pelota en dirección *dir*.
- *score*: tratar de convertir un gol.
- *clearBall*: patear hacia la cancha contraria entre los oponentes.
- *intercept*: ir en dirección de la pelota.

Además se definen variables de estado acerca de información relevante del contexto:

- *paseBloqueado(i, j, dir)*: indica si el pase entre el agente *i*, y el agente *j* esta bloqueado con dirección *dir*.
- *frenteAlArco(j)*: indica si el jugador *j* esta frente al arco contrario.
- *espacioLibre(i, dir)*: indica que en dirección *dir* relativa al agente *i*, no hay jugadores contrarios.

Finalmente, para definir completamente la estrategia del equipo debemos especificar reglas de valores que nos especifiquen la utilidad global para un contexto dado. Veamos algunos ejemplos:

- $<p_1^{pasador}$; *esRecibidor(j)* $\wedge$ *Not paseBloqueado(i,j,dir)* $\wedge$ a_i= *passTo(j,dir)* $\wedge$ a_j= *moveTo(dir)*: *u(j,dir)*>$\forall j \neq i$

- $<p_2^{pasador}$; *espacioLibre(i,n)* $\wedge$ a_i= *dribble(n)*: *30*>
- $<p_3^{pasador}$; *espacioLibre(i,n)* $\wedge$ a_i= *clearBall*: *10*>
- $<p_4^{pasador}$; *frenteAlArco(i)* $\wedge$ a_i= *score*: *100*>
- $<p_5^{recibidor}$; *esInterceptador(j)* $\wedge$ *Not paseBloqueado(j,i,dir)* $\wedge$ a_i= *moveTo(dir)*: *u(i,dir)*>$\forall j \neq i$
- $<p_6^{interceptador}$; *intercept*: *100*>

Como podemos ver en las reglas, cada una retorna un valor de utilidad, el cual es utilizado para definir que acción tomar, para el caso que la comunicación entre los agentes no es posible (por mas información, ver [KOKf]), un agente se tiene que hacer pasar por el otro para evaluar sus funciones de utilidad, en caso de ser posible, lo ideal es que un agente se alimente por la evaluación de los demás agentes. En los caso de p_1y $p_5$la utilidad depende de la función *u*.

	Con CG	Sin CG
Ganados	5	2
Empatados	3	3
Perdidos	2	5
Gol Average	0.9	0.2

Cuadro 4.4: Resultado de la utilización de CG en el equipo UvA Trilearn [KOKc].

En el cuadro 4.4 se pueden ver los resultados obtenidos con CG. Para poder conocer con mas detalle este método utilizado por el equipo UvA Trilearn, referirse a [KOKc, KOKb]. En 2005 se modifico una parte del método CG, inicialmente se utilizaba un algoritmo de eliminación variable y en 2005 se paso a utilizar max-plus[KOKe, KOKd].

Desarrollo de la Aplicación

Durante los inicios del proyecto para la realización del equipo UvA Trilearn en 2001 [DEBa], se analizo la posibilidad de rehuso de las habilidades básicas de otros equipos. La falta de documentación asociada al código disponible y la falta de prácticas relacionadas con la Ingeniería de Software llevaron a que decidieran desarrollar el equipo desde cero. Se enfocaron inicialmente en desarrollar las habilidades de bajo nivel del equipo bajo criterios de calidad que les permitieran en un futuro proseguir su implementación. Las habilidades de alto nivel fueron sí, desarrolladas en base al equipo FC-Portugal, campeón en el año 2000. Los equipos en los que se basaron fueron: CMUnited (el parceo de mensajes y algunas habilidades de bajo nivel), Cyberoos [POCYB] (sincronización), Essex Wizards [POESS](arquitectura multi hilada) y FC-Portugal (las habilidades de alto nivel).

Arquitectura

La arquitectura presentada en UvA consta de tres capas jerárquicas como se muestra en la figura 4.12, donde cada una tiene un nivel de abstracción diferente.

- **Capa de Interacción**: Se encarga de la interacción con el simulador (SoccerServer). De esta forma encapsula los detalles del simulador para las capas superiores.
- **Capa de habilidades**: Consume los servicios que le ofrece la capa de interacción, y se encarga del modelado del mundo y de varias de las habilidades que puede realizar el agente (por ejemplo, interceptar la pelota)
- **Capa de Control**: Esta capa contiene el módulo de razonamiento del agente. Se encarga de seleccionar la acción adecuada que pertenece a la capa de habilidades, para su cometido utiliza el estado del mundo y la estrategia elegida en un momento dado.

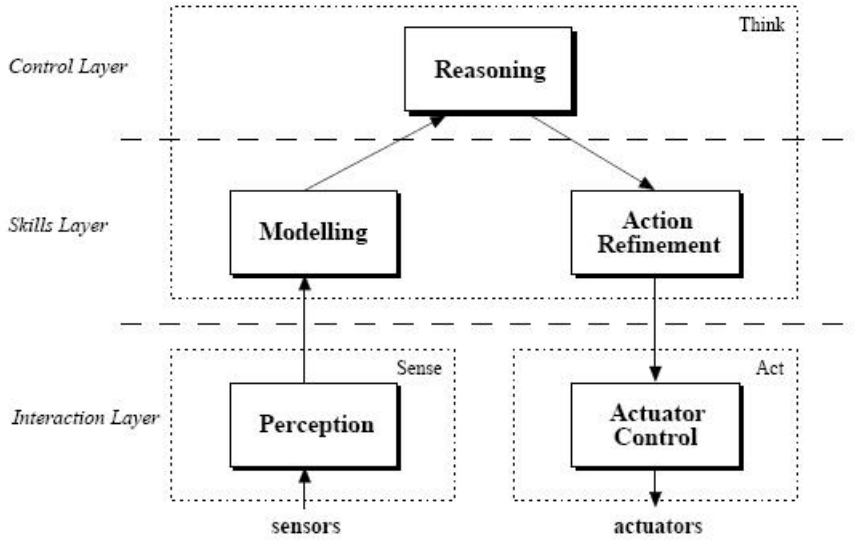

Figura 4.12: Arquitectura del equipo UvA Trilearn [DEBa].

Las tres capacidades de los agentes: percepción, rezonamiento y acción pueden ser ejecutadas en paralelo. Debido a que los agentes, actúan un ambiente dinámico de tiempo real, es sumamente necesario que estas tres tareas puedan ser ejecutadas en paralelo. UvA realiza el paralelismo a través de hilos, los mismos son:

- **Sense**: representa el módulo de percepción (Figura 4.12)
- **Act**: representa el módulo de acción (Actuator Control)
- **Think**: representa los módulos de las capas de habilidades y control.

De esta forma, el agente puede razonar sin los retardos que genera la entrada/salida.

4.6. AT-Humboldt

Introducción

En esta sección se describen las principales características del equipo AT-Humboldt [POATH] (de aquí en mas ATH). Luego de mencionar aspectos básicos del equipo, la sección, se divide en dos partes bien diferenciadas: la primera corresponde a las características principales del equipo incorporadas desde sus inicios [BURa] (se mantienen hasta hoy) y en la segunda, se describen las técnicas incluídas en el equipo de 2003 [BERa].

Identificación del equipo

El equipo ATH (AT por "Agent Team") de simulación fue desarrollado por estudiantes de la Universidad de Humboldt [UHU]. Participó en el primer campeonato oficial de la RoboCup en el año 1997 y obtuvo el primer premio; su fortaleza fueron las habilidades básicas de los jugadores[11]. En el campeonato de 1998 en Paris, mucho mas parejo que el anterior, ATH se volvió a destacar obteniendo el segundo lugar por detrás de CMUnited. En el campeonato de 1999 logro el 7^{o} lugar, para de a poco ir quedando relegado a los últimos lugares en el campeonato de 2001. Desde 2002 comenzó un proceso de renovación del equipo que presento en 2003. En el campeonato de 2004 fue 10mo en la categoría 2D, año que comenzó también a participar en 3D, destacándose con un segundo lugar en esa categoría en Portugal-2004.

[11] Durante el campeonato del 97, el equipo ATH obtuvo una ventaja comparativa al resto, debido a que lograba patear la pelota con la máxima potencia, pateando varias veces durante un mismo ciclo de reloj. Luego de este campeonato se modifico el simador para que no siga ocurriendo.

Técnicas a destacar

AT-Humboldt incorpora una Arquitectura BDI belief-desire-intention ó creencia-deseo-intención basada en una estructura mental deliberativa. También utiliza razonamiento basado en casos (CBR). Entendemos por CBR, el aprendizaje a través de experiencias (casos); especialmente lo utiliza en situaciones donde la información no es suficiente para la aplicación de reglas [BURa].

Desde 2003, ATH incorporó al enfoque BDI una arquitectura de doble pasada llamada DPA (double pass architecture). El principal aporte de la misma es manejar paralelamente dos hilos donde uno se encarga de procesar una estrategia de largo plazo y el restante se activa cuando son necesarias acciones inmediatas. De esta forma, en la misma arquitectura convive el enfoque reactivo y el deliberativo, necesarios ambos en el dominio del fútbol [BERa].

Resumen de las técnicas

Arquitectura BDI

Dentro de una arquitectura BDI, cada agente maneja:

- Un modelo del mundo llamado **creencia** (se diferencia entre conocimiento y creencia considerando que: el conocimiento es verdadero y la creencia no se sabe).
- **Deseos**, son los objetivo del agente
- **Intenciones** son los deseos realizables en un momento dado. Las mismas son consideradas como los planes (parciales) para la concreción del objetivo o deseo.

A continuación se describen brevemente los pasos del proceso de decisión de un agente, siguiendo el modelo BDI [BURc, CAS]:

1. El agente recolecta tanta información del ambiente como le es posible y la almacena en su modelo del mundo (**fase de creencia**).
2. Luego, busca entre sus comportamientos realizables y determina cual de ellos tiene la mayor prioridad (**fase de deseo**).
3. Finalmente, el agente ejecuta el comportamiento deseado (**fase de intención**).

La elección del deseo se lleva a cabo mediante lo que se llama **deliberación**. Es importante el compromiso con un determinado plan (intención) para satisfacer el deseo. La estabilidad de las intenciones tiene impacto sobre el uso racional de los recursos, una (relativa) estabilidad previene la sobrecarga en la deliberación, cambios innecesarios en los planes y ayuda a una cooperación confiable entre los agentes.

Proceso Deliberativo

Primero, se busca un plan existente y se evalúan las condiciones para su continuación. Si no se decide continuar con el plan, se evalúan todas las opciones en base a sus beneficios. Las mejores opciones son elegidas como deseos, es decir como candidatos para una nueva intención. Se chequea la factibilidad del plan del deseo mejor rankeado. Si es factible, se elige como intención; sino el siguiente deseo es chequeado. Luego de adherir a una nueva intención, un nuevo plan o comportamiento debe ser calculado y ejecutado.

Razonamiento basado en casos (CBR)

El éxito de CBR esta dado en la medida de poseer una memoria eficiente de los "casos" que permita su recuperación en corto tiempo. RoboCup ofrece infinidad de escenarios para aprender de la experiencia. Podemos distinguir entre **aprendizaje off-line** en el cual podemos experimentar y recabar muchos casos y el **aprendizaje on-line** durante los partidos en el cual solo podemos recabar unos pocos casos para aprender las habilidades y tácticas del oponente.

Desarrollo de la aplicación

Características del Desarrollo

El desarrollo para el año 1997 lo realizaron alrededor de 9 personas y luego siguió en los años posteriores con 5 miembros aproximadamente. El equipo de 1998 contaba con 25.000 líneas de código. Por tener una solución que se inclinaba a la P.O.O, evaluaron C++ y Java, la performance de la JVM para 1998 los inclino a utilizar C++.

Componentes

Los agente del equipo ATH poseen diferentes habilidades como: Ir a una posición, patear al arco ó dribliar (correr con la pelota). Cada habilidad puede considerarse como un plan parcial.

El componente **creencia** modela lo que cree el agente acerca del ambiente y de él mismo, de acuerdo a la información sensorial que percibe y de una simulación que realiza del movimiento de los objetos a su alrededor. Le provee al agente una forma de simular, evaluar y especular acerca del estado futuro de forma de tomar decisiones.

El componente **deseo** le permite al agente evaluar opciones (deseos potenciales) de acuerdo a su creencia. Las opciones típicas pueden ser: interceptar la pelota, patear al arco, ó dribliar.

El componente **intención** le permite al agente encontrar el mejor plan para realizar el deseo elegido. En base a habilidades predefinidas busca el plan que le haga maximizar el éxito esperado. Cada plan es una secuencia de acciones atómicas realizables por el jugador.

Luego que el agente deliberó, un conjunto de acciones atómicas son enviadas el componente de **ejecución**, que además, es responsable de la sincronización con el SoccerServer.

El proceso de decisión depende del rol del jugador, según el rol se definen preferencias como por ejemplo: la dirección de pateo o si usar mucho dribling, etc. [12].

Deliberación(deseo, intención, plan)

Se intenta elegir la opción correcta de entre las posibles para luego convertirla en deseo y posteriormente en acciones concretas.

En ambientes de tiempo real el proceso deliberativo posee **restricciones de tiempo**. Muchas veces el ambiente fuerza a una decisión rápida y en otros casos caemos en comportamientos inadecuados por realizarlos tarde.

Debemos también considerar como afecta el agregar, quitar o alternar opciones durante el proceso deliberativo. Como ya se ha mencionado, la estabilidad de los planes es fundamental y uno de los problemas a resolver es cuando realizar **re-planeación**. Por un lado nos podemos ver forzados re-planear luego de determinado tiempo y por el contrario no podremos re-planear dentro de ciertos intervalos de tiempo mínimos para no generar sobrecargas.

Aprendizaje

Dentro del aprendizaje **off-line**, este equipo utilizó la herramienta del coach para entrenar al equipo en distintas habilidades. Para habilidades como la de patear, se recolectó información para determinar los parámetros a utilizar.

Otra habilidad que se entreno fue la de interceptar la pelota, la misma es fundamental en el proceso deliberativo por su complejidad. Dependiendo de la distancia del resto de los jugadores a la pelota, hay que determinar sí: correr hacia la pelota, bloquear un pase, desmarcarse para recibir, además de los cálculos por la desaceleración de la pelota, etc.. En una primer implementación, el equipo utilizó simulación para determinar los estados futuros de la pelota y los jugadores, y de esta manera determinar que acción tomar. El tiempo consumido para el cálculo era inaceptable. En una segunda etapa se paso a utilizar una tabla de posibles puntos obtenida mediante ejemplos, los que luego se reducen aplicando técnicas de aprendizaje.

Dentro del aprendizaje **on-line**, se realizo un experimento que consistía en grabar posiciones del oponente en determinados momentos del juego, y en base a estos datos determinar, que lugares ocupar dentro del campo de juego. Se aplico en algunos partidos del campeonato de Paris-98 pero no dio los resultados esperados.

Arquitectura DPA incorporada en el 2003

El principal aporte, fue la incorporación de una arquitectura descripta como de doble pasadas (DPA), consiste en dos módulos, uno deliberativo y otro de ejecución que corren en hilos diferentes y cada uno realiza una pasada. La arquitectura sigue estando basada en BDI (belief, desire and intention).

[12] En zona defensiva no se recomienda despejar la pelota hacia el medio del campo o dribliar jugadores.

Motivación

Bajo la premisa que los acercamientos de arquitecturas reactivas en el dominio de la RoboCup son las mas exitosas (o arquitecturas híbridas utilizadas para vehículos no tripulados). ATH considera que dentro del fútbol de humanos se va mas allá de comportamientos reactivos a comportamientos mas complejos de estrategias a largo plazo. Incluso si se utilizará un agente reactivo para contemplar los cambios bruscos e inesperados, las estrategias complejas se hacen difícil de introducir y controlar intencionalmente. En las clásicas arquitecturas híbridas organizadas en capas donde las capas mas bajas poseen los comportamientos reflejos y las capas mas altas la planificación de alto nivel, las mismas poseen restricciones: Los requerimientos de tiempo real se aplican solo a los comportamientos básicos y no a nivel mas altos por lo que, el horizonte del tiempo y la cantidad de comportamientos no son escalables.

Por tanto ATH se inclina por la necesidad de arquitecturas deliberativas para la RoboCup, se introduce esta arquitectura de doble pasada (DPA) con las siguientes características:

- tiempo de la deliberación a largo plazo independiente del reactivo.
- menos compromiso con los datos necesarios al momento de la ejecución
- escalabilidad en el número y la complejidad de comportamientos.
- mantener la capacidad de tiempo real en todas las capas de control, incluso para la re-planeación ó comportamientos de alto nivel
- la persistencia natural de los objetivos
- control del comportamiento coordinado que implica a más de un jugador

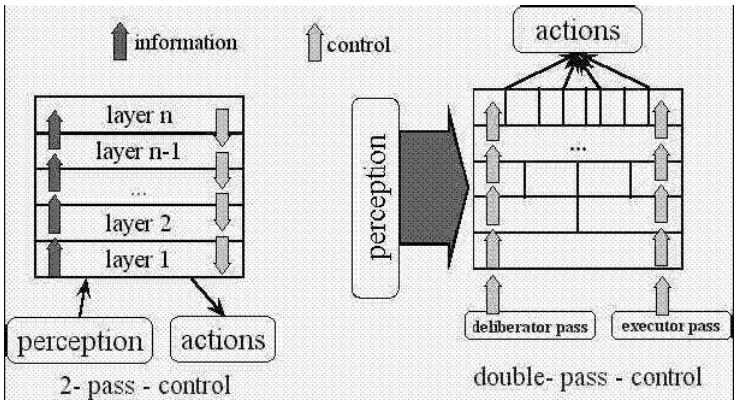

Figura 4.13: Comparación de una clásica arquitectura en capas de dos pasadas con la arquitectura DPA [BERa].

Arquitectura DPA

El elemento principal de la arquitectura DPA es una estructura de árbol donde los nodos son opciones. Estas opciones representan comportamientos específicos a diferentes niveles de abstracción. Los comportamientos mas complejos se componen de comportamientos mas simples, los de largo plaza se encuentran mas cerca de la raíz. Se distingue en tres tipos de opciones.

- **Opciones básicas**: Que se mapean con los comandos soportados por el SoccerServer.
- **Opciones de elección**: son comportamientos complejos realizables a través de una de sus sub-opciones. Se analizan alternativas y se determinan parámetros estratégicos.
- **Secuencia de opciones**: son comportamientos complejos realizables a través de la ejecución en secuencia de sus sub-opciones.

Solo las opciones de elección y de secuencia pueden estar anidadas.

El módulo **Deliberador** consume todo su tiempo en la planeación de largo plazo. La deliberación comienza en la raíz evaluando las sub-opciones y el estado del juego. El resultado es un plan parcial que corresponde a los deseos e intenciones de la metodología BDI. El plan es continuamente actualizado y completado mientras el tiempo pasa, también puede considerar finalizar un plan o comenzar uno nuevo. El deliberador no realiza cálculos que están demasiado ligados al estado actual del juego como ser parámetros de la acciones básicas. El tiempo del deliberador es independiente y fue desarrollado asincrónicamente para ser interrumpido por componentes con necesidades de tiempo real.

El módulo **Ejecutador** es invocado siempre que el manejador del tiempo entiende que es necesario ejecutar una acción. En este momento se recorre el árbol de decisión hasta encontrar una opción básica para ser ejecutada. Se evalúan condiciones y se resuelven parámetros para dicha acción. Esto esta basado en el trabajo previo realizado por el deliberador sobre el árbol de decisión, solo se le deja al ejecutador un trabajo mínimo que queda determinado por el último estado del juego. El promedio de ejecución del ejecutador durante RoboCup-2003 fue de 1 milisegundo.

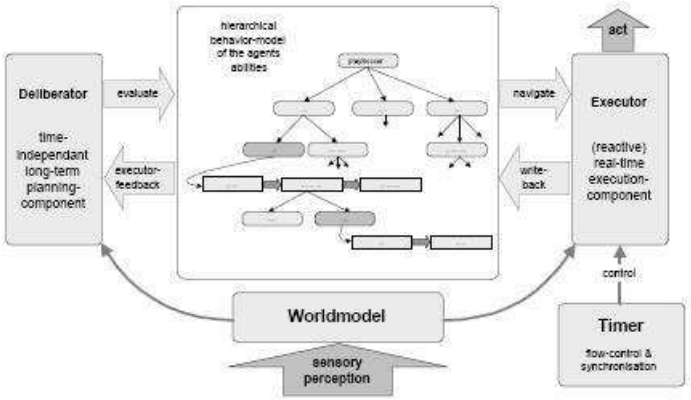

Figura 4.14: Visión general de la arquitectura DPA [BERa]

Otras Características de ATH 2003

ATH utiliza comportamiento cooperativo basado en roles dinámicos junto a los planes. Los roles se pueden asignar como parte de la ejecución del plan y son independiente de roles estáticos como defensor o atacante.

En el 2003 se continuo con la utilización de CBR y se profundizo en muchos aspectos. Se está investigando la utilización de CBR para el reconocimiento y predicción de la conducta del oponente.

Para la descripción del comportamientos del agente utiliza XML.

ATH ha desarrollado herramientas para el desarrollo y debuger de los clientes que son ofrecidas open source. Las mismas se llaman ClientControl y ADT(Agent Debug Tool).

Y por último el coach, desde 2001 ATH ha incorporado un coach con pocas responsabilidades. Las mismas son las siguientes:

- Jugadores Heterogéneos son rankeados a través de heurísticas y son sustituidos de acuerdo a este ranking. Al 2003 solo se sustituye antes del partido aunque se espera incorporar la sustitución durante el partido en base al manejador de energía y la adaptación al oponente.
- Se detecta rápidamente durante el partido el tipo de jugadores utilizados por el oponente.
- Se comunica la posición de la pelota en los tiros libres.
- Maneja la estructura de los jugadores en el campo al inicio del juego buscando las fortalezas y debilidades del oponente.

4.7. Guía de referencia

Para concluir este capítulo presentamos un resumen de todas las referencias utilizadas en el análisis de los equipos presentado en las secciones anteriores. En el cuadro se muestra para cada uno de estos equipos las referencias utilizadas.

Equipo	Referencias
AT-Humboldt	[BURb, BURa, BURc, BERa]
Brainstormers	[MER, EHR, BUC, HOF]
CMUnited	[STOa, STOb, BOW]
FC-Portugal	[LAUb, LAUc, LAUe, COS]
Mainz Rolling Brains	[CER, MEY, ARN]
Uva Trilearn	[DEBa, DEBc, KOKc, KOKe]

Cuadro 4.5: Referencias de los equipos analizados en esta capítulo.

Bibliografía

[AGU] Aguirre, J.L.; Brena Pinero, R.; Conant, S.; Garrido, L. *"Conocimiento Distribuido y Agentes Inteligentes"*. Reporte de Investigación - CSI-RI-03-002, Junio de 2003. Proyecto: Tecnologías de conocimientos distribuídos y agentes inteligentes - Tecnológico de Monterrey, México. Disponible en versión pdf y MWord: <http://lizt.mty.itesm.mx/catedra/3.htm>[Consulta: Julio 2005].

[AKI] Akin, L.,Custodio, L., Jacoff, A., Kiat Ng, B., Kraetzscmar, G., Lima, P., Obst, O., Rofer, T., Takahashi, Y., Zhou, C.*"RoboCup 2004 Competitions and Symposium: A Small Kick for Robots, a Giant Score for Science"*. 2004.

[AND] Anderson, J.; Baltes, J. *"Doraemin user manual"*. <http://sourceforge.net/projects/RoboCup-video>[Consulta: Agosto 2005].

[ARN] Arnold, A.;Flentge, F. *"Descripción del equipo Mainz Rolling Brains 2D, año 2004"*. Departamento de Ciencias de la Computación, Universidad de Johannes Gutenberg, Mainz - Alemania, 2004. (D-55099).

[ASA] Asada, M., Kitano, H., Kuniyoshi Y., Noda I., Osawa E. *"RoboCup: The Robot World Cup Initiative"*. En: Proceedings, IJCAI-95 Workshop on Entertainment and AI/Alife, 1995.

[BAR] Barman, R., Kingdon, S., Mackworth, A., Pai, D., Sahota, M., Wilkinson H., Zhang, Y.*"Dynamite: A Testbed for Multiple Mobile Robots"*. En Proceedings, IJCAI-93 Workshop on Dynamically Interacting Robots, Chambery, France, 1993.

[BERa] Berger, R.; Burkhard, H.D.; Gollin, M. *"AT-Humboldt 2003 Team Descriptions"*. En: RoboCup2003: Robot Soccer Word Cup VII, LNAI. Springer, 2004. Departamento de Ciencias de la Computación - Universidad de Humboldt, Berlín.

[BERb] Bernardos Barbolla, A.; Garcia Herrero, J.; Molina Lopez, J. *"Agentes y Sistemas Multiagente"*. Centro de Difusión de Tecnologías, Universidad Politécnica de Madrid. CEDITEC 2004. Disponible en versión pdf: <http://www.ceditec.etsit.upm.es/InfTecnologia/ceditec_agentes.pdf>[Consulta: Junio 2005]. Disponible en Web: <http://giaa.inf.uc3m.es>[Consulta: Junio 2005].

[BUC] Buckland, M. *"Programming game AI by example"*. 1ra edición, EEUU: Worldware Publishing Inc., 2005. 495p. ISBN: 1-55622-078-2.

[BOW] Bowling, M.; Stone, P.; Veloso, M. *"Anticipation: A key for Collaboration in a Team of Agents"*. En: Tercer Conferencia Internacional de Agentes Autonomos, 1999.

[BRA] Braun, H.; Riedmiller, M. *"A direct adaptive method for faster backpropagation learning: The RPROP algorithm"*. En: IEEE International Conference on Neural Networks (ICNN), page 586 - 591, San Francisco 1993.

[BUC] Buck, S.; Dilger, S.; Ehrmann, R.; Frommberger, L.; Hofmann, A.; Merke, A.; Riedmiller, M.; Sinner, A.; Thate, O. *"Karlsruhe Brainstromers: Design Principles"*. RoboCup-99: Robot Soccer World Cup III, LNCS, Springer. p. 59 - 63.

[BUL] Bulter, M.; Howard, T.; Prokopenko, M. *"Flexible Synxhronisation within RoboCup Enviroment: a Comparative Analysis"*. . Artificial Intelligence in e-Business Group, CSIRO Mathematical and Information Sciences, Australia.

[BURa] Burkhard, H.D. *"AT-Humboldt in RoboCup 99"*. En: Team Descriptions RoboCup 1999.

[BURb] Burkhard, H.; Gugenberger, P.; Schroter, K.; Wendler, J. *"AT Humboldt in RoboCup-98 (Team Description)"*. En: Asada, M.; Kitano, H.; (Eds.), "RoboCup-98: Robot Soccer World Cup II", p. 358-363, Springer, 1999.

[BURc] Burkhard, H.; Duffert, U.; Lotzsch, M.; Myritz, H.; Werner, M. *"Humboldt Heroes"*. En: Stone, P.; Balch, T.; Kraetzschmar, G.; (Eds.), "RoboCup 2000: Robot Soccer World Cup IV", p. 651-654, Springer, 2001.

[CANa] Canales, R.; Casella, S.; Rodríguez, P. *"Fútbol de Robots: Liga de Simulación 2D RoboCup, Características del Simulador"*. Tejera, G. (tutor). Trabajo de Grado. Facultad de Ingeniería. Instituto de Computación - Montevideo, Junio 2005.

[CANb] Canales, R.; Casella, S.; Rodríguez, P. *"Fútbol de Robots: Liga de Simulación 2D RoboCup, Reglas de la Liga - Reglas de la Competencia Osaka 2005"*. Tejera, G. (tutor). Trabajo de Grado. Facultad de Ingeniería. Instituto de Computación - Montevideo, Junio 2005.

[CAS] Castelo, C.; Fassi, H.; Scarpettini, F. *"Fútbol de Robots: Revisión del Estado del Arte y Desarrollo del Equipo UBASot de Simulación"*. Tesis de Grado, Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires, Diciembre de 2002.

[CER] Cerchiari, C.; Frank, J.; Varela, M. *"Agentes Inteligentes: Informe Final"*. Garbusi P.; Rodríguez F. (tutores). Proyecto de Grado. Facultad de Ingeniería. Instituto de Computación, Montevideo, 2003.

[CHE] Chen, M.; Dorer, K.; Foroughi, E.; Heintz, F.; Huangy, Z.; Kapetanakis. S; Kostiadis, K; Kummeneje, J.; Murray, J.; Noda, I.; Obst, O.; Riley, P.; Steens, T., Wang, Y.; Yin, X. *"RoboCup SoccerServer: User Manual"*. Comunidad de RoboCup, Febrero 2003.

[COS] Costa, E.; Lau, N.J.; Reais,L.P. *"Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents"*, 2001. En: Markus Hannebauer, Jan Wendler and Enrico Pagello Editors, Balancing Reactivity and Social Deliberation in Multi-Agent System From RoboCup to Real-World Applications, Springer's Lecture Notes in Artificial Intelligence, Vol. 2103, pp. 175-197, Berlin, 2001.

[CRAE00] Campeonato RoboCup - Abierto Europeo, EuRoboCup 2000. <http://www.RoboCup.org/games/euro2000/3241.html>[Consulta: Agosto 2005].

[CRAL01] Campeonato RoboCup - Abierto de Alemania 2001. <http://www.ais.fraunhofer.de/GO/2001/>[Consulta: Agosto 2005].

[CRAL02] Campeonato RoboCup - Abierto de Alemania 2002. <http://www.ais.fraunhofer.de/GO/2002/>[Consulta: Agosto 2005].

[CRAL03] Campeonato RoboCup - Abierto de Alemania 2003. <http://www.ais.fraunhofer.de/GO/2003/>[Consulta: Agosto 2005].

[CRAL04] Campeonato RoboCup - Abierto de Alemania 2004. <http://www.ais.fraunhofer.de/GO/2004/>[Consulta: Agosto 2005].

[CRAL05] Campeonato RoboCup - Abierto de Alemania 2005. <http://www.ais.fraunhofer.de/GO/2005/>[Consulta: Agosto 2005].

[CRAS05] Campeonato RoboCup - Abierto Eslovaco 2005. <http://www.fiit.stuba.sk/RoboCup/2005/eng/turnaj_menu.htm>[Consulta: Agosto 2005].

[CREU03] Campeonato RoboCup - Abierto de Estados Unidos 2003. <http://www.cs.cmu.edu/~AmericanOpen03/>[Consulta: Agosto 2005].

[CREU05] Campeonato RoboCup - Abierto de Estados Unidos 2005. <http://www.RoboCup-us.org/>[Consulta: Agosto 2005].

[CROB97] Campeonato Oficial RoboCup 1997 - Nagoya, Japón. <http://www.RoboCup.org/games/97nagoya/311.html>[Consulta: Agosto 2005].

[CROB98] Campeonato Oficial RoboCup 1998 - París, Francia. <http://www.RoboCup.org/games/98paris/312.html>[Consulta: Agosto 2005].

[CROB99] Campeonato Oficial RoboCup 1999 - Estocolmo, Suecia. <http://www.ida.liu.se/ext/RoboCup-99/>[Consulta: Agosto 2005].

[CROB00] Campeonato Oficial RoboCup 2000 - Melburne, Australia. <http://www.RoboCup.org/games/2000melbourne/314.html>[Consulta: Agosto 2005].

[CROB01] Campeonato Oficial RoboCup 2001 - Seattle, USA. <http://www.cs.cmu.edu/~RoboCup2001/>[Consulta: Agosto 2005].

[CROB02] Campeonato Oficial RoboCup 2002 - Fukuoka, Japón. <http://www.RoboCup.org/games/02Fukuoka/316.html>[Consulta: Agosto 2005].

[CROB03] Campeonato Oficial RoboCup 2003 - Padova, Italia. <http://www.RoboCup.org/games/03Padova/317.html>[Consulta: Agosto 2005].

[CROB04] Campeonato Oficial RoboCup 2004 - Lisboa, Portugal. <http://www.RoboCup2004.pt/>[Consulta: Agosto 2005].

[CROB05a] Campeonato Oficial RoboCup 2005 - Osaka, Japón . <http://www.RoboCup2005.org/>[Consulta: Junio 2005].

[CROB05b] Campeonato Oficial RoboCup 2005 - Osaka, Japón. <http://RoboCup.org/games/05Osaka/images/0717/index.html>[Consulta: Agosto 2005].

[CROB05c] Campeonato Oficial RoboCup 2005 - Osaka, Japón. <http://www.gizmodo.com/gadgets/japan/gizmodo-japan-RoboCup-2005-osaka-114466.php>[Consulta: Agosto 2005].

[DEBa] De Boer, R; Kok, J.R. "*The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team*" PhD tésis. Universidad de Amsterdam, Holanda 2002.

[DEBb] De Boer, R; Groen, F.; Kok, J.R. "*UvA Trilearn 2001 Team Description*". En: RoboCup-2001: Robot Soccer World Cup V. Springer Verlag, Berlin, 2002.

[DEBc] De Boer, R; Kok, J.R; Vlassis, N. "*Towards an optimal scoring policy for simulated soccer agent*" En: Proc. 7th Int. Conf. on Intelligent Autonomous Systems, p. 195-198, IOS Press, Marina del Rey, California, Marzo 2001.

[DEBd] De Boer, R; Groen, F.; Kok, J.R.; Vlassis, N. "*UvA Trilearn 2002 Team Description*". En: G. Kaminka, P. Lima, and R. Rojas, editors, RoboCup 2002: Robot Soccer World Cup VI, page 549, Springer-Verlag, Fukuoka, Japón, 2002.

[DEL] Delladio, T.; García, A.; García, D.; Gottifredi, S.; Martin F.; Rostsein, N.; Simari, G.; Tucat, M. *"Cognitive robotics in a soccer game domain: A proposal for the e-league competition"*. En: 6to workshop de Investigadores en Ciencias de la Computación (WICC), Universidad Nacional del Comahue, 2004. p. 289-293,.

[DETUA] Departamento de Electrónica y Telecomunicaciones - Universidad de Aveiro, Portugal. <http://www.det.ua.pt/>[Consulta: Agosto 2005].

[EHR] Ehrmann, R.; Hofmann, A.; Meier, D.; Merke, A.; Riedmiller, M.; Sinner, A.; Thate, O. *"Karlsruhe Brainstromers: A Reinforcement Learning approach to robotic soccer"*. Instituto de Lógica y Sistemas Deductivos de la Universidad de Karlsruhe - Alemania, 2000. (D-76128 Karlsruhe, FRG).

[FIFA] Federación Internacional de Fútbol Asociado. <http://www.fifa.com>[Consulta: Agosto 2005].

[GROa] Groen, F.; Kok, J.R.; Vlassis, N. "*UvA Trilearn 2003 Team Description*". En: Proceedings of the 11th International Conference on Advanced Robotics, ICAR'03, p. 1124-1129, Coimbra, Portugal, Junio 2003.

[GROb] Groen, F.; Kok, J.R.; Vlassis, N. "*UvA Trilearn 2004 Team Description*". En: Proceedings CD RoboCup 2004, Springer-Verlag, Lisboa, Portugal. Julio 2004.

[GSI] *"Agentes Inteligentes"*. Material para curso de grado del Grupo de Sistemas Inteligentes - Dpto. Ingeniería de Sistemas Telemáticos (GSI-DIT), España. Disponible en versión pdf: <http://www.gsi.dit.upm.es/~gfer/ssii/agentes-04.pdf>[Consulta: Junio 2005].

[HOF] Hofmann, A.; Merke, A.; Nickschas, M.; Riedmiller, M.; Withopf, D.; Zacharias, F. *"Brainstromers 2002: Tem Description"*. Instituto de Informática de la Universidad de Dortmund - Alemania, 2002.

[IGL] Iglesias Fernandez, C.A. *"Capítulo 2 Fundamentos de Inteligencia Artificial Distribuída"*. En: "Definición de una metodología para el desarrollo de sistemas multiagente". Tésis de Doctorado. Departamento de Ingeniería de Sistemas Telemáticos - Universidad Politécnica de Madrid, España. Enero 1998.

[KOG] Kogler, M.; Obst, O. *"Simulation League:The Next Generation"*. En: RoboCup2003: Robot Soccer Word Cup VII, LNAI. Springer, Marzo 2003.

[KOKa] Kok, J.R.; Vlassis, N. *"Mutual Modeling of Teammate Behavior"*. En: Technical Report IAS-UVA-02-04, Instituto de Ciencias de la Computación, Universidad de Amsterdam, 2002.

[KOKb] Kok, J.R.; Spaan, M.T.J.; Vlassis, N. *"An approach to noncommunicative multiagent coordination in continuous domains"*. En: Benelearn 2002: Proceedings of the Twelfth Belgian-Dutch Conference on Machine Learning, pp. 4652, Utrecht, Holanda. Diciembre 2002.

[KOKc] Kok, J.R.; Spaan, M.T.J.; Vlassis, N. *"Multi-robot decision making using coordination graphs"*. En: Proceedings of the 11th International Conference on Advanced Robotics, ICAR'03, p. 1124-1129, Coimbra, Portugal. Junio 2003.

[KOKd] Kok, J.R.; Vlassis, N. "*UvA Trilearn 2005 Team Description*". En: Proceedings CD RoboCup 2005, Springer-Verlag, Osaka, Japón, Julio 2005.

[KOKe] Kok, J.R.; Vlassis, N. "*Using the Max-Plus Algorithm for Multiagent Decision Making in Coordination Graphs*". En: RoboCup 2005: Robot Soccer World Cup IX, Osaka, Japón, Julio 2005.

[KOKf] Kok, J.R.; Spaan, M.T.J.; Vlassis, N. *"Non-communicative multi-robot coordination in dynamic environments"*. En: Robotics and Autonomous Systems, 50(2-3):99-114, Elsevier Science. Febrero 2005.

[LAUa] Lau, N.J.; Reais,L.P.; Seabra Lopez, L. *"F.C.Portugal Team Description"*, 2001. En: RoboCup-2000: Robot Soccer World Cup IV, Eds. Peter Stone, Tucker Balch and Gerhard Kraetzschmar, LNAI 2019, 29-40, Springer Verlag, Berlin.

[LAUb] Lau, N.J.; Reais,L.P. *"FC Portugal Most Interesting Research Overview"*, 2000. En: <http://www.ieeta.pt/RoboCup/documents/FCPortugalInteresting.ps.zip>, 2000.

[LAUc] Lau, N.J.; Reais,L.P. *"FC Portugal Team Description: RoboCup 2000 Simulation League Champion"*, 2001. En: Peter Stone, Tucker Balch and Gerhard Kraetzschmar, editors, RoboCup-2000: Robot Soccer World Cup IV, Springer Verlag Lecture Notes in Artificial Intelligence, Vol. 2019, pp.29-40, Berlin. 2001.

[LAUd] Lau, N.J.; Reais,L.P. *"FC Portugal 2001 Team Description: Flexible Teamwork and Configurable Strategy"*. En: Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro, editors, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Berlin. 2002.

[LAUe] Lau, N.J.; Reais,L.P.; Teixeira, C. *"FC Portugal 2003 shoot evaluation based on goalie movement prediction"*, 2004. En: Proc. Scientific Meeting of the Portuguese Robotics Open 2004, FEUP Edições, Colecção Colectâneas, Vol. 14, pp.149-155, ISBN 972-752-066-9, 23-24 Abril, 2004.

[LIACCUP] Laboratorio Inteligencia Artificial y Ciencias de la Computación – Universidad de Porto, Portugal. <http://www.ncc.up.pt/liacc/index.html>[Consulta: Agosto 2005].

[LSIM] Liga simulada de RoboCup. <http://sserver.sourceforge.net/index.html>[Consulta: Agosto 2005].

[LSIM2D] Liga Simulación RoboCup, Competencias 2D. <http://sserver.sourceforge.net/index.html>[Consulta: Abril 2005].

[MAR] Marques, H; Lau, N.; Reis, L.P. *"FC-Portugal 3D simulation team: Architecture, low-level skills and team behaviour Optimizad for the new RoboCup 3D simulator"*. En: RoboCup 2005 Osaka, Japón.

[MAT] Matellán, V. *"El simulador de la RoboCup"*. En: V Workshop Agentes Físicos, Universidad de Girona, 2004.

[MER] Merke, A.; Riedmiller, M. *"Using machine learning techniques in complex multi-agent domains"*. Instituto de Lógica y Sistemas Deductivos de la Universidad de Karlsruhe - Alemania, 2001. (D-76131 Karlsruhe, FRG).

[MEY] Meyer, C.; Schappel, B.; Schulz, F.; Uthmann, T. *"Descripción del equipo Mainz Rolling Brains 2D, año 2000"*. Departamento de Ciencias de la Computación, Universidad de Johannes Gutenberg, Mainz - Alemania, 2000. (D-55099).

[POA4T] Equipo a4ty <http://dssg.cs.rtu.lv/en/research/RoboCup/Default.htm>[Consulta: Agosto 2005].

[POATH] Equipo AT-Humboldt. <http://www.RoboCup.de/AT-Humboldt/index.shtml>[Consulta: Agosto 2005].

[POATT] Equipo ATTUnited. <http://www.cs.utexas.edu/~pstone/RoboCup/United2002-sim.html>[Consulta: Agosto 2005].

[POBRA] Equipo Brainstormers. <http://amy.informtik.uos.de/asg/projects/brainstormers>[Consulta: Agosto 2005].

[POCYB] Equipo Cyberoos, <http://www.cmis.csiro.au/IIT/Projects/RoboCup/RoboCup.htm>[Consulta: Agosto 2005].

[POCMU] Equipo CMUnited, Carnegie Mellon University - Estados Unidos. <http://www.cs.cmu.edu/~robosoccer/simulator/>[Consulta: Agosto 2005].

[POESS] Equipo Essex Wizards <http://cswww.essex.ac.uk/staff/hhu/Essex_Wizards/ew.html>[Consulta: Agosto 2005].

[POFCP] Equipo FC-Portugal. <http://www.ieeta.pt/RoboCup>[Consulta: Agosto 2005].

[POMAT] Equipo MateBots, RoboCup Liga E-League. <http://cs.uns.edu.ar/~ajg/matebots/>[Consulta: Agosto 2005].

[POMRB] Equipo Mainz Rolling Brains. <http://amy.informatik.uos.de/asg/projects/brainstormers/>[Consulta: Agosto 2005].

[PONIM] Equipo NimBro, RoboCup Liga Humanoide <http://www.nimbro.net/>[Consulta: Agosto 2005].

[POROB] Equipo RoboLog <http://www.robolog.org/>[Consulta: Agosto 2005].

[POSBC] Equipo SBCEE <http://www.sbcee.net/>[Consulta: Agosto 2005].

[POUTA] Equipo UT Austin Villa. <http://www.cs.utexas.edu/~AustinVilla/>[Consulta: Agosto 2005].

[POUVA] Equipo UvA Trilearn. <http://staff.science.uva.nl/~jellekok/RoboCup/>[Consulta: Agosto 2005].

[POWRI] Equipo Wright Eagle. <http://wrighteagle.org/>[Consulta: Agosto 2005].

[RILa] Riley, P.; Stone, P.; Veloso, M. CMUnited-99 source code, 1999. <http://www.cs.cmu.edu/~pstone/RoboCup/CMUnited99-sim.html>[Consulta: Agosto 2005].

[RILb] Riley, P. *"SPADES System for Parallel Agent Discrete Event Simulation - Users Guide and Reference Manual"*, Diciembre 2003. <http://spades-sim.sourceforge.net>[Consulta: Agosto 2003].

[RILc] Riley, F.G; Riley, F.P. *"SPADES - A distributed agent simulation environment with software-in-the-loop execution"*. En: Proceedings of the 2003 Winter Simulation Conference. p. 817-825. Eds Chick, S.; Ferrin, D.; Morrice, D. J.; Sánchez, P. J., 2003.

[ROBa] Federación de Fútbol de Robots - RoboCup. <www.RoboCup.org>[Consulta: Agosto 2005].

[ROBb] Reportes del campeonato oficial RoboCup Osaka 2005. <www.RoboCup2005.org/news/RoboCup_reports.aspx>[Consulta: Agosto 2005].

[S3D] Manual del Servidor de la liga Simulada 3D de RoboCup. "Text_instead_of_a_manual.txt" En: rcssserver3D, Software de base de la liga 3D de RoboCup [LSIM].

[SSIL] Liga RoboCup SSIL. <http://sserver.sourceforge.net/league/index.html>[Consulta: Agosto 2005].

[STOa] Stone, Peter <http://www.cs.utexas.edu/~pstone>[Consulta: Agosto 2005].

[STOb] Stone, P. *"Layered Learning in Multi-Agent Systems"*, PhD Thesis, School of Computer Science, Carnegie Mellon University, 1998.

[STOc] Stone, P.; Veloso, M.; *"Task Decomposition, Dynamic Role Assignment, and LowBandwidth Communication for RealTime Strategic Teamwork"*. En: Artificial Intelligence, 110 (2), p. 241-273, Junio 1999.

[UAM] Universidad de Amsterdam. <http://www.english.uva.nl/>[Consulta: Agosto 2005].

[UHU] Universidad de Humboldt - Berlín, Alemania. <http://www.hu-berlin.de/>[Consulta: Agosto 2005].

[VIR] Proyecto Virtual RoboCup <http://www.techfak.uni-bielefeld.de/ags/wbski/3DRoboCup/3DRoboCup.html>[Consulta: Agosto 2005].

[WAR] Wardzynski D. *"Efficient Offensive Strategy for Simulated Robotic Soccer"*. PhD tésis. School of Electrical Engineering, Departament of Numerical Analysis and Computer Science, Royal Institute of Tecnology. Estocolmo, Suecia 2005. (TRITA-NA-E05051).

[WEI] Editado por Weiss, G. *"Multiagent Systems A modern approach to Distributed Artificial Intelligence"*. 2da edicion, EEUU: M.I.T. Press, 2000. 619p. ISBN 0-262-23203-0.