Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Tesis de Licenciatura

Fútbol de Robots:
Revisión del Estado del Arte y Desarrollo del Equipo UBASot de Simulación

Diciembre de 2002

Claudia C. Castelo
Héctor R. Fassi
Flavio E. Scarpettini

Director de Tesis
Dr. Juan M. Santos

A Phoebe, la cuarta integrante honoraria de esta tesis.

A nuestras familias.

Indice

Agradecimientos

A Juan, por tu apoyo y seguimiento en este proyecto.

A Cristina, por guiarnos en nuestras intentos anteriores de tesis.

A Phoebe, por tu traducción y aliento continuo.

Claudia, Héctor y Flavio

A Olga, por darme tanto amor, por tu amistad y tu apoyo.

A Andrés, por tu amor, tus enseñanzas y ser mi guía en la vida.

A mi familia, por su cariño y su compresión.

A mis mejores amigos, por compartir momentos tan lindos y permitirme alcanzar este sueño.

Claudia

A mis papis, Ida y Victor, por sus sacrificios para que pudiera llegar hasta aquí.

A mi familia, por su apoyo y aliento a pesar de tantos fines de semana de ausencia.

A Phoebe, Claudia y Flavio, por haber hecho tan placenteros estos largos años de estudio y amistad.

Héctor

A María Laura, por tu empuje, apoyo, amor y tiempo.

A Ornella y Piero, por acompañarme y enseñarme lo importante de aprender.

A Anita, Renata y Pino, por confiar en que llegaría.

A mi familia y amigos, por comprender mis ausencias.

A Héctor, Claudia y Phoebe, por soportarme en este largo camino.

Flavio

Resumen

El fútbol de robots provee un amplio campo de experimentación para el desarrollo de conocimiento e integración de tecnología en el área de sistemas de múltiples robots que cooperan entre sí para resolver un problema común. El problema resulta sumamente interesante dado que provee un entorno dinámico donde se mezcla el aspecto cooperativo con situaciones adversas.

El presente trabajo de tesis propone realizar un estudio del problema desde el análisis de cada uno de los aspectos que entran en juego en el desarrollo de un equipo de fútbol de robots, así como también de las distintas técnicas utilizadas en referencia a sistemas multiagentes y al desarrollo de habilidades especiales en robots móviles. El objetivo del trabajo es proveer un marco teórico que sirva para futuros trabajos y desarrollos en el área.

Asimismo se presenta en esta tesis la implementación realizada de un equipo de fútbol simulado que fue presentada en el Campeonato Mundial de Fútbol de Robots FIRA 2002 en el mes de mayo de ese año en Corea del Sur, obteniendo el tercer puesto en su categoría entre un total de cincuenta y nueve equipos participantes provenientes de doce países.

Abstract

Robot soccer provides a terrain rich for experimentation in developing knowledge and integrating technology in the area of systems where multiple robots cooperate with each other in order to solve a common problem. This problem proves to be highly interesting in the sense that it offers a dynamic environment where robot cooperation and situational adversity meet.

This thesis presents a study of the aforementioned problem which consists of a detailed analysis of each aspect in the development of a robot soccer team, as well as the various techniques employed in reference of multi-agent systems and the development of special abilities for mobile robots. The objective of the study is to provide an adequate theoretical framework for further research in the field.

As part of the thesis it is also included the implementation of the simulated team which participated in 2002 FIRA Robot Soccer World Championship celebrated in May of the same year in South Korea, and has ranked as high as the third place among a total of fifty nine participants from twelve countries in the category of simulation.

Capítulo 1
Introducción

El fútbol de robots provee un campo de testeo para sistemas de múltiples robots autónomos que, a través de una tarea común como el juego de fútbol, estimula la investigación de problemas que involucran robots moviéndose rápidamente y cooperando entre sí para cumplir objetivos específicos en entornos dinámicos y bajo situaciones adversas.

En la actualidad existen diversas competencias de fútbol con robots. El interés teórico detrás de estas experiencias es el uso del fútbol como un prototipo de sistema complejo y adaptativo que permita el desarrollo de técnicas para la construcción de robots capaces de llevar a cabo tareas más útiles.

La formación de un equipo capaz de competir en alguno de los certámenes de fútbol de robots presupone integrar desarrollos sobre distintos comportamientos tales como rastreo de agentes y de la pelota, movilización hacia un objetivo determinado, habilidad para llevar la pelota a través de un recorrido, evitar obstáculos móviles, patear la pelota al arco, realizar un pase a otro compañero, ubicarse dentro del campo de juego, evaluar su posición con respecto a sus compañeros y oponentes, interceptar la pelota en movimiento, compartir una estrategia de equipo y reaccionar ante estímulos externos. También supone trabajar sobre áreas como robótica para diseñar robots capaces de aprender y realizar estos comportamientos, procesamiento de imágenes para poder brindar a los robots un estado actual del escenario, procesamiento en tiempo real para poder responder a los estímulos recibidos, y lenguajes de comunicación entre los robots.

La propuesta del presente trabajo de tesis consistía originalmente en hacer un relevamiento del estado del arte respecto del fútbol de robots y de las distintas

técnicas utilizadas en referencia a sistemas multiagentes y desarrollo de habilidades especiales sobre robots, para que sirva de marco teórico para futuros trabajos y desarrollos. Luego, seleccionar e implementar algunas técnicas aplicables a robots móviles en una plataforma de simulación como paso inicial hacia la formación futura de un equipo que pueda participar en los proyectos internacionales como los organizados por FIRA y RoboCup.

Una vez realizado el relevamiento mencionado en el párrafo anterior, se comenzó con la implementación de comportamientos elementales sobre la plataforma de simulación de RoboCup. Avanzada esta etapa se presentó la posibilidad de que la Universidad de Buenos Aires participara en el Campeonato Mundial de Fútbol de Robots 2002 organizado por FIRA, en Corea del Sur, a través del Grupo de Inteligencia Computacional Aplicada a Robótica Cooperativa en Sistemas Multirobots del Departamento de Computación de la Facultad de Ciencias Exactas y Naturales. Dada la excelente posibilidad que representa la participación en un certamen de estas características para comprobar los resultados de una implementación, se encaró el desafío de desarrollar un equipo completo y competitivo en el marco de la categoría de simulación 5 vs 5 (*Middle League SimuroSot*) de FIRA. El equipo desarrollado obtuvo el tercer puesto dentro de su categoría entre un total de cincuenta y nueve equipos participantes de doce países.

El presente trabajo se divide en tres partes: en la primera parte se describen las generalidades del fútbol de robots como ambiente de desarrollo e investigación para sistemas multiagentes, las características de los robots que participan en el juego y los sistemas de simulación que se han desarrollado; en la segunda parte se abordan los temas relacionados con el diseño de un equipo de fútbol de robots en forma general, independientemente del ambiente donde se jueguen los partidos, las particularidades de los agentes y las ligas específicas donde pueden participar los equipos; por último, en la tercera parte, se presenta la implementación que se ha llevado a cabo para lograr el equipo de simulación UBASot, la estrategia, las

jugadas y los comportamientos desarrollados, los experimentos realizados y los resultados obtenidos.

La Parte I abarca los capítulos 2, 3, 4 y 5. Estos capítulos comprenden la introducción al entorno del fútbol de robots, los problemas que este entorno presenta, las distintas arquitecturas y características físicas que poseen los robots y las características de los sistemas de simulación, respectivamente.

La Parte II comprende desde el capítulo 6 al capítulo 12. Para una mejor compresión de los problemas que se deben resolver durante la implementación de un equipo, se propone una división en niveles y cada uno de ellos es abordado en uno de los capítulos. En el capítulo 6 se describe el modelado del ambiente. Entre los capítulos 7 y 10 se detallan, respectivamente, las acciones, comportamientos elementales, comportamientos complejos y jugadas que deben ejecutar los robots durante el juego. El capítulo 11 comprende el diseño de las tácticas, planes a corto plazo ejecutados por los agentes, y el capítulo 12 corresponde a las estrategias, planes a largo plazo perseguidos por el equipo en su conjunto.

La Parte III está conformada por los capítulos 13 y 14. En el primero de estos capítulos se describe la implementación del equipo UBASot, la estrategia, las jugadas y los comportamientos desarrollados. En el capítulo 14 se muestran los problemas que se presentaron durante la etapa de desarrollo y la resolución dada a los mismos. En este mismo capítulo también se incluyen los experimentos realizados con el equipo y los resultados obtenidos.

Por último, en el capítulo 15 se presentan las conclusiones de este trabajo y los lineamientos para el desarrollo de trabajos futuros en el tema.

Parte I
Fútbol de Robots

Los primeros capítulos del presente trabajo, los cuales conforman la Parte I del mismo, tienen por objetivo exponer y plantear las generalidades del fútbol de robots como ambiente de desarrollo e investigación de sistemas multiagentes.

En el capítulo 2 se brinda una introducción al entorno del fútbol de robots y luego, en el capítulo 3, se plantean los problemas que dicho entorno involucra.

A continuación se describen, en el capítulo 4, las distintas arquitecturas y características físicas que poseen los robos destinados a jugar fútbol. Por último, en el capítulo 5, se detallan las características de los software de simulación de fútbol de robots que brindan el marco necesario para implementaciones de estrategias de equipo sin necesidad de contar con el hardware de los robots.

Capítulo 2
Introducción al Fútbol de Robots

2.1 ¿Qué es Fútbol de Robots?

El fútbol de robots provee un campo de testeo para sistemas de múltiples robots autónomos que, a través de una tarea común como el juego de fútbol, estimula la investigación de problemas que involucran robots moviéndose rápidamente y cooperando entre sí para cumplir objetivos específicos en entornos dinámicos y bajo situaciones adversas.

Una manera efectiva de promover el desarrollo y la investigación es fijar un objetivo significativo a largo plazo. La iniciativa RoboCup [KIT1] ha fijado como objetivo para mediados del siglo XXI, el desarrollo de un equipo de robots humanoides totalmente autónomos que sean capaces de ganarle a la selección de fútbol humana campeona del mundo. Un objetivo, un poco más modesto, pero no menos ambicioso, sería lograr un robot que sea capaz de jugar al fútbol en forma similar a un humano.

Si bien el desafío parece muy grande y lejano en base al estado actual del conocimiento y la tecnología, medio siglo aproximadamente fue el tiempo que trascurrió desde el primer avión de los hermanos Write hasta la llegada de la misión Apollo a la Luna; o desde la invención de la computadora digital hasta el triunfo de Deep Blue sobre el campeón humano de ajedrez.

Este tipo de grandes proyectos a largo plazo, son conocidos como *landmark projects*. Un proyecto de esta magnitud proclama un objetivo atractivo e interesante, el cual es suficientemente complejo para requerir una serie de importantes mejoras o invenciones tecnológicas para alcanzarlo. Adicionalmente,

la tecnología desarrollada para el proyecto debe ser lo suficientemente innovadora como para, posiblemente, fundar nueva generación industrial. Esta última característica resulta mucho más importante que el objetivo del proyecto propiamente dicho. El *landmark project* más importante y exitoso de la historia ha sido,quizás, el proyecto Apollo, el cual tenía un claro objetivo "*Llevar un hombre a la Luna y traerlo sano y salvo de vuelta a la Tierra*", según fue proclamado por John F. Kennedy en 1961. Si bien el cumplimiento de este objetivo marcó el curso de la humanidad, no existieron significativas ventajas directas, ni siquiera desde el punto de vista militar, por el hecho en sí de haber llegado a la Luna. Sin embargo, el traslado de la tecnología desarrollada a la aviación, la electrónica, la producción de materiales y la informática fundaron una nueva era en la industria de los Estados Unidos [ASA] [KIT2].

El fútbol de robots también puede ser considerado un *Problema Estándar* ya que en él varias teorías, algoritmos y arquitecturas pueden ser evaluadas. Un típico ejemplo de problema estándar es el ajedrez. Varios algoritmos fueron evaluados y desarrollados en este dominio. Con el reciente logro del equipo Deep Blue al vencer a Kasparov, el gran maestro de ajedrez, usando las reglas oficiales del juego, el desafío está cercano a finalizar. Muchos investigadores piensan que el fútbol de robots puede ser el nuevo desafío que fomente un conjunto de tecnologías para la próxima generación industrial.

Si bien muchos problemas de la robótica han sido tratados y resueltos por la comunidad científica, varios temas específicos, cuyos resultados serán altamente beneficiosos para la robótica, son impulsados por el fútbol de robots. La tarea de múltiples robots móviles que cooperativamente llevan a cabo una misión en un ambiente adverso, no ha sido ampliamente investigado en el pasado. Sin embargo, cuando los robots son usados en la vida diaria, deben ser capaces de aprender y entender como colaborar con otros robots y humanos. La colaboración de múltiples agentes cooperativos es un tema crítico en el fútbol de robots.

El fútbol es también un ámbito propicio para el desarrollo de robots con amplia movilidad y comportamientos flexibles ya que un jugador debe ser capaz de caminar, correr, saltar y patear la pelota muy rápidamente. También debe ser capaz de hacer frente al contacto con otros robots o seres humanos. Tradicionalmente, los desarrolladores de robots asumen que no hay contacto con los robots o, a lo sumo, que el contacto es muy limitado y no hostil. Esta suposición no es válida si se desea desarrollar robots para el mundo real: la gente puede golpear accidentalmente a un robot que trasporta materiales frágiles. El robot debería ser capaz de, rápidamente, evitar la colisión o de balancear su cuerpo de manera que los objetos trasportados no se dañen. El fútbol de robot ofrece un campo ideal de testeo ya que en el fútbol el contacto es inevitable [KIT2].

2.2 Motivación

La formación de un equipo capaz de competir en alguno de los certámenes de fútbol de robots presupone integrar desarrollos sobre distintos comportamientos tales como rastreo de agentes y de la pelota, movilización hacia un objetivo determinado, habilidad para llevar la pelota a través de un recorrido, evitar obstáculos móviles, patear la pelota al arco, realizar un pase a otro compañero, ubicarse dentro del campo de juego, evaluar su posición con respecto a sus compañeros y oponentes, interceptar la pelota en movimiento, compartir una estrategia de equipo y reaccionar ante estímulos externos. También supone trabajar sobre áreas como *Robótica* para i) diseñar robots capaces de aprender y realizar estos comportamientos, ii) procesamiento de imágenes para poder brindar a los robots un estado actual del escenario, iii) procesamiento en tiempo real para poder responder a los estímulos recibidos y iv) lenguajes de comunicación entre los robots.

Cada robot o agente debe poder decidir si espera por un pase, corre por la pelota, cubre un ataque del equipo oponente, o ayuda a un compañero; siguiendo al mismo tiempo las reglas del juego. La característica más importante de un equipo de fútbol de robots es que cada agente es autónomo y soporta en sí mismo todas las capacidades esenciales. Hay tres tipos de posiciones de juego: arquero, defensor y delantero. La posición que un robot tiene en el campo de juego influye en la percepción que tiene de su entorno y sus estrategias de juego en el campo. Mientras cada robot tiene distintas motivaciones y objetivos, todos ellos comparten la misma arquitectura general y hardware básico [SHE1].

El proyecto de fútbol de robots brinda un marco de referencia concreto para el desarrollo sobre las distintas áreas descriptas arriba. Abordar este proyecto requiere la integración de un amplio espectro de temáticas tales como diseño de agentes autónomos, fusión de sensores en tiempo real, comportamiento reactivo, procesamiento de imágenes, reconocimiento de entorno, robótica, control de motores, desarrollo de tareas colectivas, aprendizaje de comportamientos, estrategias de decisión y adquisición de estrategias [KIT1].

Los problemas estándar han sido la fuerza direccional de la investigación en Inteligencia Artificial (IA). Un claro ejemplo de esto, como ya se mencionó, es el ajedrez computacional que ha dado pie al desarrollo de algoritmos de búsqueda potentes. Otros problemas, incluyendo *Yale Shooting* y *Monkey Banana* han contribuido a la investigación en IA. mostrando las dificultades presentes en el razonamiento diario. Las críticas a estos problemas se han enfocado en que ellos se basan en tareas abstractas y no consideran las dificultades del mundo real. Los propulsores de estas criticas argumentan que los problemas del mundo real deben ser objeto de serios estudios. Si bien estas criticas tienen cierto fundamento, tratar este tipo de problemas involucran dificultades dependientes del dominio, además de las sociales y económicas. Por otro lado, la investigación sobre sistemas reales utilizables está por encima de los recursos y fondos de muchos grupos de

investigación. Es por todo esto que se requiere un problema estándar real pero a la vez alcanzable para la mayoría de los grupos de investigación [KIT1].

Las competencias de fútbol de robots están diseñadas para cubrir la necesidad de manejar las complejidades del mundo real a través de un entorno reducido, con un tamaño y costo alcanzables y que además abarca e integra varias ramas de la inteligencia artificial y la robótica. Sobre el tema ya se han reportado numerosos informes. Los esfuerzos más importantes sobre el tema son la iniciativa RoboCup (*Robot Soccer World Cup*) y FIRA (*Federation of Robot-soccer Asociations*).

2.3 Evolución

La idea de "Robots que jugaran Fútbol" fue mencionada por primera vez por el Profesor Alan Mackworth (University of British Columbia, Canada) en el año 1992. Su proyecto se denominó *Dynamo* y su objetivo fundamental consistía en el desarrollo de un ambiente flexible para realizar experimentos con múltiples robots controlados por radio. La aplicación de prueba para este ambiente fue el fútbol de robots. En el marco de este proyecto, los robots en forma individual fueron capacitados para armar su plan de acción y llevarlo a cabo, sólo con el simple objetivo de hacer goles en el arco contrario y evitar que la pelota ingrese en el propio. Como experimento se realizaron competencias entre dos robots (uno contra uno) guardando la esperanza de formar en trabajos futuros equipos de robots competitivos [BAR].

En Octubre de ese mismo año, en forma independiente, un grupo de investigadores japoneses organizó en Tokio un *workshop* sobre "Grandes Desafíos en Inteligencia Artificial". Durante estos talleres se plantearon serias discusiones sobre el uso del juego del fútbol para promover la ciencia y la tecnología. Una serie de investigaciones fueron realizadas, estudios de impacto

social y de factibilidad tecnológica y financiera. También fueron diseñadas las reglas de juego, el prototipo de robot y de sistema de simulación. Los resultados de estos estudios, llevaron a concluir que el proyecto era realizable. En Junio de 1993, un grupo de investigadores, incluyendo a Minoru Asada, Yasuo Kuniyoshi y Hiroaki Kitano, decidieron lanzar una competencia robótica, tentativamente denominada *Robot J-League* (J-League era el nombre dado a la Liga Japonesa de Fútbol Profesional). El interés que despertó el proyecto fuera de Japón, hizo que la iniciativa se presentara como un proyecto internacional con el nombre de RoboCup (*Robot Soccer World Cup*).

En Septiembre de 1993, se realizó el primer anuncio público de esta iniciativa y se diseñaron sus reglamentos. También fue desarrollado y distribuido el primer sistema de simulación para el dominio de fútbol posibilitando la investigación sobre sistemas multiagentes.

Durante la "Conferencia Internacional de Inteligencia Artificial" (ICAI-95) realizada en Agosto de 1995 en Montreal, se anunció la organización del primer Campeonato Mundial de Fútbol de Robots (RoboCup) para el año 1997. Al mismo tiempo, se decidió la organización de una competencia durante el año 1996 (Pre RoboCup'96) con la finalidad de detectar potenciales problemas que pudieran suscitarse en la primera competencia a gran escala, RoboCup'97.

Pre RoboCup'96 se llevó a cabo en Osaka, RoboCup'97 en Nagoya y RoboCup'98 en París. Año tras año, las competencias se van superando en complejidad y cantidad de equipos participantes. En los años siguientes, 1999, 2000, 2001 y 2002 las sedes de estos campeonatos fueron Estocolmo (Suecia), Melbourne (Australia), Seattle (EE.UU) y Fukuoka/Busan (Japón), respectivamente [ROB].

También en el transcurso de 1995, durante el mes de Septiembre, el Profesor Jong-Hwan Kim de KAIST, Corea, fundó un comité para la organización

de un Campeonato Mundial de Fútbol de Micro Robots (MiroSot) en ese país. Durante los años 1996 y 1997, KAIST se encargó de la organización de estos eventos [FIR].

En Junio de 1997, también por iniciativa de J. H. Kim se creó la *Federación Internacional de Fútbol de Robots* (FIRA). Desde su fundación, FIRA ha sido responsable de los campeonatos subsiguientes: Francia en 1998, Brasil en 1999, Australia en 2000, China en 2001 y Corea en 2002. Habiendo comenzado con sólo una categoría orientada a la competencia de robots pequeños, anualmente FIRA incorpora nuevas categorías donde participan robots con diversas características y habilidades [KIM1].

Capítulo 3
El Problema

3.1 Introducción

Cuando se encara la tarea de conformar un equipo de fútbol de robots, se presenta una amplia gama de problemas a resolver. El gran desafío que asume el proyecto se ve como una serie de vallas que hay que ir sorteando una a una, paso a paso. Es importante tener una visión clara del problema general para poder particionarlo y clasificarlo en problemas de menor complejidad que, una vez resueltos, son integrados posteriormente.

Cada uno de estos subproblemas tiene sus propias características. Las mismas serán abordadas en este capítulo. El problema del fútbol de robots puede ser subdividido en tres grandes temas:

- ***Compresión del entorno:*** Tomando como base a lo percibido por el robot sobre su entorno, se conforma un modelo que le permite comprender el ambiente en el cual se desenvuelve el juego. Mediante la implementación de distintos sentidos se debe intentar captar el estado del ambiente de la manera más fiel posible.
- ***Decisiones tácticas y estratégicas:*** Combinando el conocimiento y la información con que se obtiene del entorno, se debe decidir cuales son las acciones a llevar a cabo considerando las reglas del juego y de las tácticas del equipo.
- ***Ejecución de Acciones:*** Este punto está relacionado con las habilidades físicas del robot. Al momento de llevar a cabo una acción, se ponen en juego distintos actores, los cuales deben ser comandados y coordinados para que el resultado de la acción sea lo más cercano posible al óptimo.

3.2 Comprensión del Entorno

Para recolectar la información que se necesita para jugar al fútbol, se debe conocer en tiempo real la dinámica del campo de juego. Se debe, particularmente, observar el comportamiento de aquellos objetos móviles que no se pueden controlar o predecir completamente. También se deben percibir objetos y marcas estáticas de la cancha a los efectos de que el robot pueda orientarse y autolocalizarse. Los mensajes sonoros que puedan ser emitidos por el árbitro o bien por algún miembro del equipo propio o rival, deben ser interpretados y el contacto que el jugador mantenga con un objeto también debe ser advertido. Para lograr todo esto, se deberían implementar, idealmente, tres sentidos humanos: la visión, la audición y el tacto. Eventualmente, se podrían considerar capacidades de percepción adicionales.

3.2.1 Sentido de la Visión

La visión es el principal y más importante sentido que un agente puede implementar para obtener datos sobre el ambiente que lo rodea.

En base a la información captada, el jugador conoce su posición en la cancha e identifica los distintos sectores de la misma. También se advierte la posición, orientación y velocidad de los demás jugadores y de la pelota. En base a la información que se puede obtener es fácil advertir que la eficiencia del sistema de visión es fundamental para alcanzar un juego exitoso. Esto redunda en el requerimiento de un alto grado de precisión en la información que será entregada al módulo de decisión.

Además de enfrentar los problemas del requerimiento de procesamiento en tiempo real y la precisión, la implementación del sistema de visión debe también

manejar en forma robusta las distintas condiciones del ambiente como pueden ser sombras, reflejos, diversas condiciones de iluminación y oclusiones parciales o totales de los objetos.

3.2.2 Sentido de la Audición

Un jugador puede utilizar la audición para interpretar indicaciones de su entrenador, mensajes de sus compañeros y también del entrenador y jugadores rivales para sacar provecho de ello. Interpreta indicaciones sonoras del árbitro en base a las reglas de juego y también mediante el oído puede advertir movimientos o acciones que quizás no perciba en forma visual (un jugador corriendo, la pelota picando, etc). Si se tiene en cuenta que el escenario donde se juega el partido es por naturaleza ruidoso, se suma una importante tarea que consiste en identificar la información sonara relevante dentro del ruido ambiente.

La capacidad auditiva de un agente puede ser dividida en dos categorías:

- ***Comprensión de habla:*** El agente debe entender sentencias o instrucciones verbales dadas por su entrenador u otro agente y poder de esta manera actuar en consecuencia.
- ***Comprensión del ambiente sonoro:*** El análisis de este ambiente puede ser muy útil para completar información sobre el ambiente de juego. Por ejemplo, interpretando el sonido ambiente se pude detectar un jugador corriendo detrás del agente o que la pelota fue pateada. También es necesario para comprender mensajes no verbales, como ser, los distintos sonidos del silbato del árbitro [KIT2].

3.2.3 Sentido del Tacto

El sentido del tacto es utilizado por los agentes, fundamentalmente, para

detectar y controlar colisiones de manera de resguardar su integridad física, la de los restantes jugadores y respetar las reglas de juego. También es utilizado para detectar y controlar la pelota, por parte del arquero y para el seguimiento de la misma, cuando está en poder de un jugador de campo: Cuando el jugador está llevando la pelota, no necesita ir mirándola constantemente para saber donde está, sino que se basa en el contacto de su pie con la pelota.

Los sistemas que implementan el sentido del tacto, deberían ser capaces de identificar el área de contacto, la fuerza y la presión ejercida sobre el objeto detectado considerando contactos muy suaves, variaciones de textura y de calor.

3.2.4 Otras Formas de Percepción

Si bien no existen otros sentidos humanos aplicables, los agentes podrían reforzar su sistema de percepción con otros sentidos implementando la capacidad de algunas especies animales como, por ejemplo, el sonar mediante el cual pueden determinar la presencia y distancia de objetos sin necesidad de la visión, o el sistema de orientación absoluta que poseen otras especies.

3.3 Decisiones Tácticas y Estratégicas

Los agentes deben tomar decisiones en tiempo real basándose en la información de su entorno con la que cuentan y actuar en consecuencia, en forma rápida, precisa y coordinada. A la vez, deben llevar adelante un plan coordinado para lograr el objetivo común, que considere la posibilidad de realizar ajustes a medida que cambia el entorno del juego.

Existen tres áreas de suma importancia para la toma de decisiones: el Aprendizaje, la Estrategia de Equipo y Modelado del Oponente. El aprendizaje supone generar en el agente los conocimientos para llevar a cabo los

comportamientos necesarios para jugar al fútbol, dado que sería imposible contemplar en un algoritmo todos los estados de juego posibles. La estrategia de equipo hace que los agentes se enfrenten a su adversario en forma organizada y siguiendo un plan que los ayude a lograr su objetivo. Finalmente, el modelado del oponente intenta descifrar el plan del adversario para poder predecir las acciones de sus agentes y tomar ventaja de ello.

3.3.1 Aprendizaje

En esta área se pueden presentar dos tipos de aprendizaje, cada uno asociado con momentos bien diferenciados. El primero corresponde al aprendizaje *off line*, que se realiza en forma previa al juego y por única vez. El segundo tipo de aprendizaje es el *on line*, llevado a cabo durante los encuentros.

El aprendizaje *off line* se realiza generalmente en ambientes controlados, mediante lotes de prueba a partir de los cuales se deducen características comunes. Las características que se quieren aprender en este caso son estáticas y no varían de partido en partido. Una vez que el agente aprende a patear la pelota, no necesita aprenderlo nuevamente antes de cada partido.

El aprendizaje *on line* se realiza básicamente durante los partidos. En este caso, el entorno está cambiando constantemente. Los agentes deben monitorear en cada instante las acciones de sus compañeros y adversarios para considerarlas en sus decisiones y alcanzar una mejor performance en el juego.

3.3.2 Estrategia de Equipo

El trabajo en equipo en un ambiente multiagente y dinámico como el fútbol, requiere de coordinación altamente flexible para superar la incertidumbre producida por los cambios que el accionar de otros jugadores, ya sean rivales o

compañeros, introducen al medio. Esta coordinación enfrenta la dificultad de representar y contemplar todos los estados posibles del juego.

Si bien se pueden lograr comportamientos coordinados y flexibles a través de acuerdos o planes conjuntos, el trabajo en equipo es más que un conjunto de comportamientos individuales, aunque éstos estén coordinados. Un claro ejemplo de esto es el tráfico que, si bien es coordinado y simultáneo, no presenta trabajo de equipo. El trabajo en equipo implica tener metas comunes, planes y compromisos adquiridos. El equipo debe llevar adelante un plan bien definido.

Finalmente se deben considerar las jugadas pre-establecidas, comúnmente llamadas *"Jugadas de Pizarrón"*. Estas se disparan en momentos particulares bajo una precondición fijada con anterioridad e implementan estrategias específicas para un dominio acotado del juego.

3.3.3 Modelado del Oponente

El modelado del adversario es un aspecto importante e interesante en el juego de fútbol. Deducir y modelar los objetivos, planes, conocimientos y capacidades de los adversarios puede permitir optimizar los propios planes. Este modelado se puede encarar mediante tres enfoques:

- Rastreo, haciendo el seguimiento dinámico y en tiempo real de los jugadores oponentes.
- Reconocimiento de estrategias, si se tiene la posibilidad de contar con un agente fuera de la cancha que observe y analice el juego.
- Análisis posterior del juego, observando los partidos jugados para apreciar fortalezas y debilidades del equipo. Este análisis se realiza en forma *off line.*

3.4 Ejecución de Acciones

En el juego de fútbol, cada jugador debe moverse veloz y ágilmente dentro de la cancha y debe poder impulsar la pelota en forma precisa principalmente, con sus pies; secundariamente, con su cabeza y en forma complementaria, debe ser capaz de detener la pelota con su pecho. Habilidades adicionales se esperan de un arquero: debe ser capaz de atrapar en forma segura la pelota y requiere de destreza para saltar y arrojarse para interceptar pelotas a alta velocidad para lo cual el desplazamiento pedestre no es suficiente.

En cuanto a su desempeño, los agentes deben tener un comportamiento robusto. Durante el partido, pueden sufrir accidentes inesperados que los hagan caer o incluso dañar alguna de sus partes. El jugador debe rápidamente recuperar la postura y volver al modo operacional. Si hubiese sido dañada alguna de sus partes, debe ser capaz de ajustar el movimiento del resto del cuerpo para llevar a cabo su tarea lo mejor posible. Como no es posible previamente considerar todos los posibles casos, es fundamental que se cuente con un esquema de aprendizaje y adaptación para ajustarse a la situación [KIT2].

La ejecución de las acciones responde a las decisiones tomadas en base a situaciones externas y capacidades propias de los agentes. El sistema que ejecuta las acciones debe ser capaz de controlar tanto movimientos simples como complejos. Durante el juego intervienen múltiples movimientos que deben ser coordinados y sincronizados para obtener el efecto deseado.

Otra cuestión que en muchos casos el agente debe controlar es su energía, ya que debe reservarla para jugadas en las que deberá realizar un alto esfuerzo físico. Por otro lado debe tener en cuenta el tiempo restante de juego de manera de administrar su capacidad física para actuar eficientemente hasta el final del partido.

Capítulo 4
Arquitectura de Robots Reales

4.1 Introducción

En el presente capítulo se describen las características físicas de los robots móviles que, en general, se utilizan en fútbol de robots. Dentro de las federaciones FIRA [FIR] y RoboCup [ROB], cada categoría introduce ciertas limitaciones en cuanto a las dimensiones, dispositivos de percepción y tipo de procesamiento que los robots participantes pueden tener. Básicamente, los tipos de robots se pueden clasificar según los siguientes criterios:

- ***Tipos de Movilidad***: Como el robot se desplaza dentro del ambiente.
- ***Tipos de Percepción***: Como obtiene el robot información sobre el ambiente que lo rodea.
- ***Tipos de Procesamiento***: Como el robot elabora la información necesaria para actuar.

Algunos diseños de robots móviles destinados a jugar fútbol incluyen dispositivos especiales, generalmente destinados a patear la pelota, los cuales se describen al final de este capítulo.

4.2 Tipos de Movilidad

Existe una amplia variedad de mecanismos de desplazamiento entre los robots destinados al fútbol. Los más frecuentes son aquellos que utilizan ruedas, en distintas disposiciones y cantidades. Mayor complejidad se asocia a a los robots cuadrúpedos, muy difundidos gracias a la categoría Sony Legged de

RoboCup. El tipo de movilidad más complejo para un robot es mediante dos patas sobre las cuales debe hacer equilibrio para moverse.

Dentro de los robots con desplazamiento mediante ruedas los más comunes son aquellos que poseen dos ruedas independientes. En la Fig.4.1 se puede observar un robot del equipo UBASot [SAN] participante de la categoría MiroSot de FIRA [FIR]. Luego, están aquellos robots con cuatro ruedas, donde no necesariamente todas tienen tracción. Por ejemplo, los robots de CMU Hammerhead [EME] tiene sólo tracción en las ruedas delanteras (Fig.4.2).

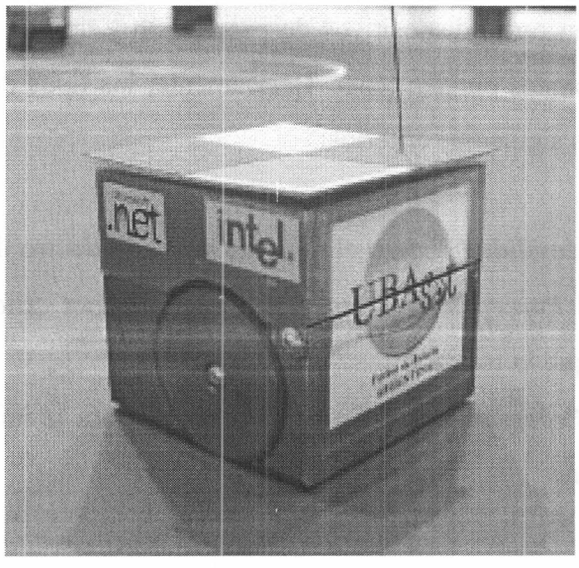


Figura 4.1: Robot UBASot.
Posee dos ruedas independientes que le permiten girar sobre si mismo y avanzar o retroceder describiendo arcos.

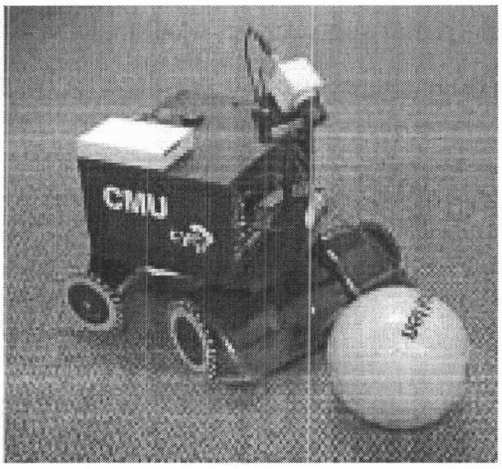


Figura 4.2: Robot CMU hammerhead.
Las ruedas delanteras poseen tracción independiente. Las traseras están montadas sobre un trailer que carga la unidad de proceso, las baterías y el sistema de visión.

Un método también muy utilizado para dar movilidad a los robots es el mecanismo omnidireccional donde, mediante ruedas dispuestas circularmente [DAN], o bien mediante esferas [MEN], se consigue que el robots pueda desplazarse en cualquier dirección. Esto no sucede con dos o cuatro ruedas paralelas, donde el robot no puede desplazarse lateralmente.

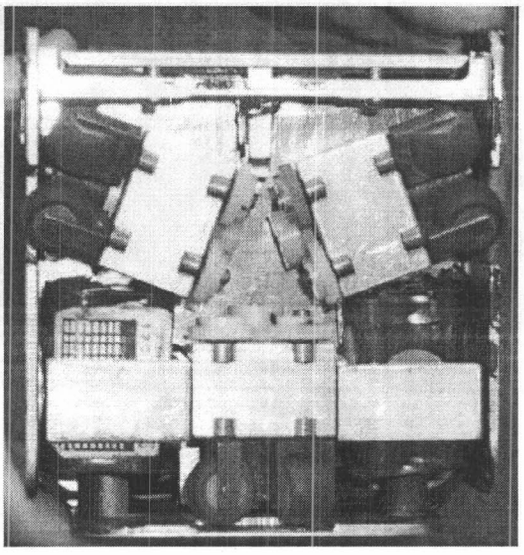
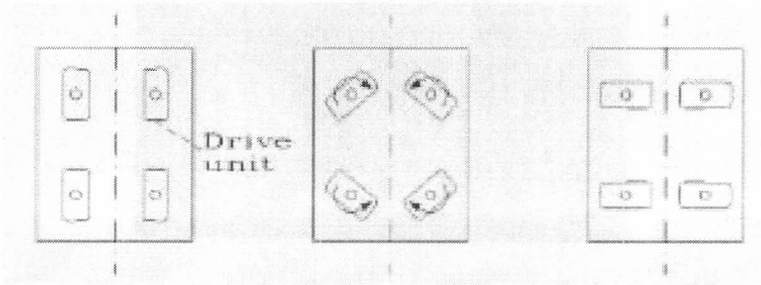


Figura 4.3: Mecanismo omnidireccional del equipo Cornell.

En [JAM] se expone un mecanismo de desplazamiento que consisten en cuatro ruedas que pueden rotar sobre su eje, con lo cual el robot puede desplazarse hacia delante y atrás, rotar sobre su eje o desplazarse hacia izquierda y derecha (Fig.4.4).



(a) (b) (c)

Figura 4.4: Mecanismo omnidireccional con cuatro ruedas.
(a) Desplazamiento hacia delante y atrás (b) rotación sobre centro geométrico (c) desplazamiento hacia izquierda y derecha.

Los robots cuadrúpedos requieren mecanismos más complejos para desplazarse ya que se deben coordinar las cuatro patas, cada una de las cuales posee 3 grados de libertad (Fig.4.5).

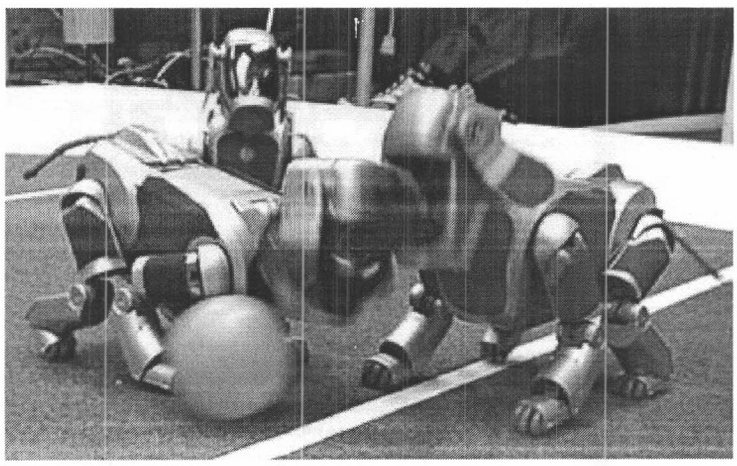


Figura 4.5: Robots cuadrúpedos Sony.

Los robots bípedos intentan reproducir el desplazamiento humano mediante dos piernas sobre las cuales caminan. En la Fig.4.6 se observa el aspecto exterior del robot PINO [YAM] el cual se asemeja a un ser humano.

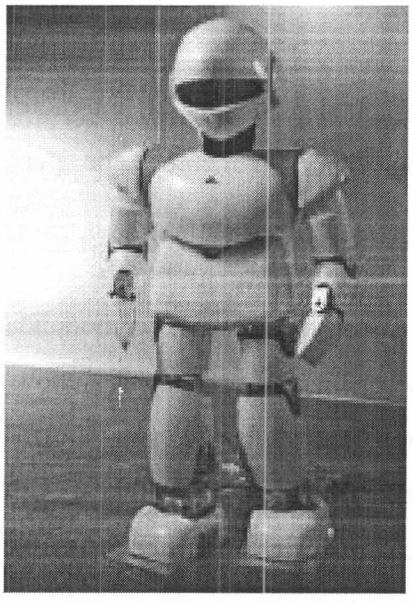


Figura 4.6: Robot humanoide PINO.

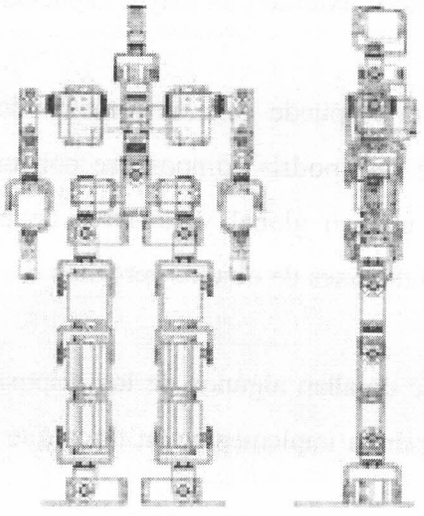


Figura 4.7: Diseño mecánico de PINO.
El robot posee 26 servomotores los cuales brindan cada uno de los grados de libertad.

El robot PINO posee 26 grados de libertad: 6 en cada pierna, 5 en cada brazo, 2 en el cuello y 2 en el tronco.

4.3 Tipos de Percepción

Para que un robot pueda, en general, interactuar con su ambiente y, en particular, jugar al fútbol, necesariamente debe percibir información de este ambiente.

Básicamente, se pueden mencionar dos modalidades de percepción: global y local. Una involucra mecanismos de percepción externos al robot, generalmente comunes a todo el equipo, los cuales relevan información de todo el entorno. Luego de un procesamiento, también externo, entregan a cada agente la información sobre ubicación y orientación de cada objeto en el campo de juego. En el otro caso, el propio robot cuenta con dispositivos de percepción, lo que le permiten recavar información sobre su propio entorno. En general, el propio robot procesa la información percibida aunque en algunos casos, puede que se trasmita la percepción a un procesador externo con mayor capacidad de cómputo.

Una variante mixta se puede generar combinando perceptores globales y locales. Un caso de este tipo podría componerse con una cámara externa a los robots que provea una imagen global y sensores de proximidad para obtener mayor precisión en las mediciones de objetos cercanos.

A continuación se detallan algunos de los dispositivos de percepción más utilizados así como también la implementación física que algunos equipos llevan a cabo.

4.3.1 Dispositivos de Visión

La visión es el mecanismo de percepción por excelencia. Para su implementación, es necesario el uso de una o más cámaras que capturen imágenes para luego ser procesadas por los módulos de software correspondientes.

En los equipos con percepción global, generalmente se cuenta con una cámara - o varias - ubicada en forma perpendicular al campo de juego y enfocando hacia abajo a una altura tal que toda la cancha pueda ser observada.

Cuando el dispositivo de visión está montado sobre el cuerpo del robot, en general se percibe una pequeña porción del ambiente que rodea al robot, complicación que se suma al hecho de que las formas, distancias y movimientos de los objetos son relativos al punto de vista del agente, situación que no se presenta con visión global.

Existen varios mecanismos comúnmente implementados para visión local. El más simple es el montaje de una cámara fija en la parte superior o en el frente del robot, de manera de captar la imagen del entorno delante del robot. Para cambiar el ángulo de visión, se debe desplazar el robot completo (Fig.4.8). Algunos diseños incluyen dos o más cámaras fijas, con lo cual se amplia el campo visual del robot, como por ejemplo en [SHE2] donde se sitúa una cámara al frente y otra hacia atrás, dándole cada una de ellas un ángulo de 165°. También existen implementaciones de visión binocular como en el arquero del equipo ART-00 [ADO] que le provee una visión de 220°.

Otra opción es el montaje de una cámara móvil con uno o dos grados de libertad, con lo cual se puede variar el ángulo de visión sin necesidad de desplazar al robot, siendo posible así ampliar o completar el modelo del ambiente de forma más fácil. Esto ha sido implementado en el jugador TotTino del equipo Art-00

[ADO] y en el robot humanoide PINO [YAM] cuyo cuello tiene dos grados de libertad.

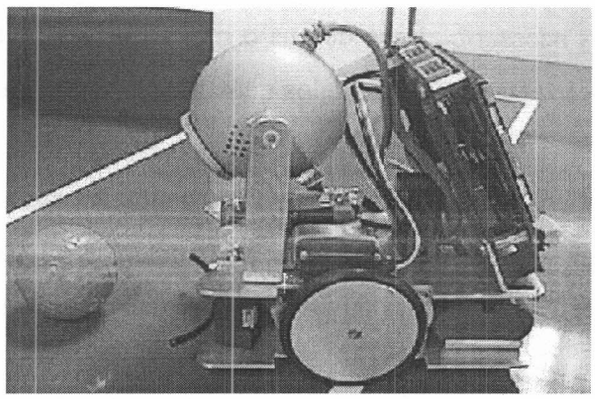


Figura 4.8: Robot con cámara fija.
La cámara está fija y orientada en un ángulo de, aproximadamente, 45° hacia el piso de manera de detectar la pelota.

Dado que resulta altamente beneficioso para un robot contar con visión de 360°, se podría pensar que emulando la visión global, colocando una cámara en forma perpendicular al suelo por encima del robot, se logra el objetivo. El problema reside en que dicha cámara debería estar muy alto comprometiendo la estabilidad y violando las restricciones de la mayoría de las categorías de fútbol de robots. Un método muy utilizado para obtener visión omnidireccional es la utilización de un espejo cónico o semiesférico montado sobre el robot hacia donde está dirigida una cámara (Fig.4.9). El espejo refleja todo el entorno circundante al robot y esa imagen distorsionada (Fig.4.10) es captada por la cámara, y provista a un proceso que deberá interpretarla [MEN].

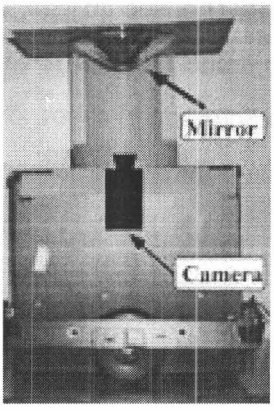
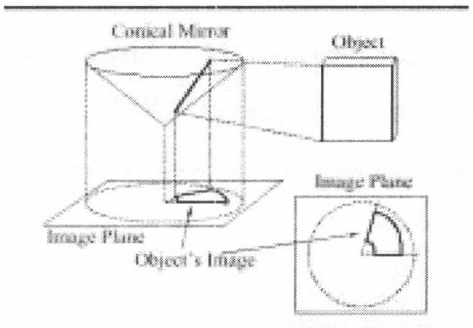
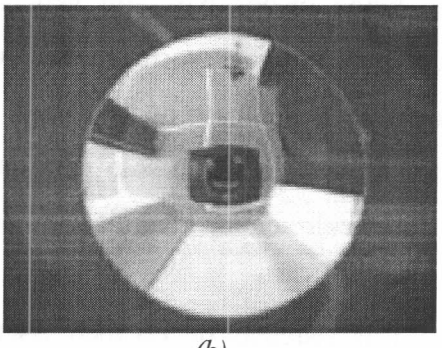


Figura 4.9: Robot con visión omnidireccional.



(a)



(b)

Figura 4.10: Visión Omnidireccional.
(a) Esquema de la imagen obtenida con visión omnidireccional (b) Imagen real reflejada en el espejo cónico.

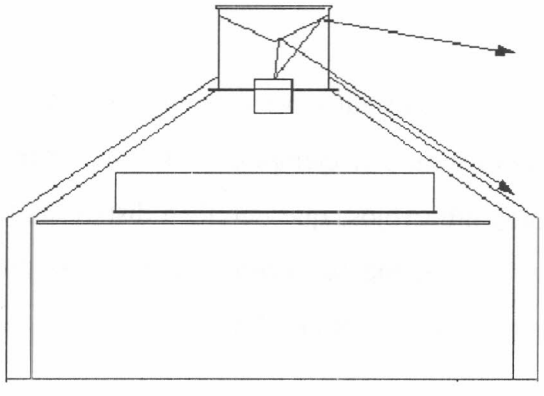


Figura 4.11: Sistema de visión omnidireccional.
La forma cónica del robot permite visualizar los objetos que hacen contacto con el cuerpo del robot en toda su circunferencia.

El equipo descripto en [PRI] posee un espejo cónico sostenido por un tubo de plástico transparente que contiene también la cámara. Para obtener una visión de la pelota cuando está en contacto con el robot, o cuando se está en presencia de obstáculos, se adoptó una forma cónica para la parte superior del robot, según el esquema que se puede observar en la Fig.4.11. La visión obtenida se extiende a un radio de 1 m desde el robot.

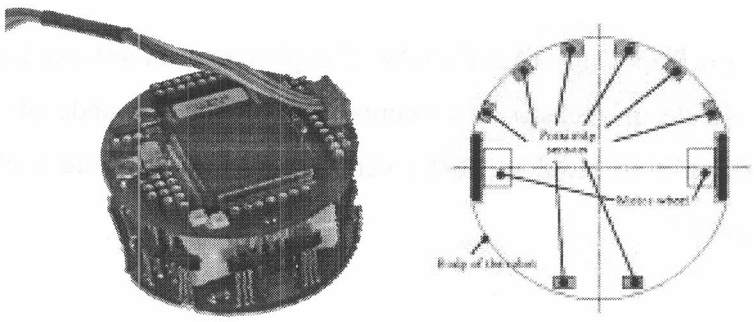
4.3.2 Escaners Láser

Estos dispositivos son generalmente utilizados para autolocalización y detección de objetos. Normalmente, son el complemento de un sistema de visión limitado. Este es el caso de los robots del equipo CS Freiburg [WEI], que cuentan con un sistema de visión sólo para detectar la pelota, utilizando el escaneo láser para autolocalización e identificación de objetos.

Los sistemas de escaneo láser como el utilizado por el equipo T-Team Tuebinger [PLA] tiene un rango de 180°, con una resolución angular de 0.5° y pueden medir distancias de hasta 15 metros con una precisión de 10 mm.

4.3.3 Sensores de Proximidad

Estos sensores son utilizados para detectar obstáculos u objetos que se encuentran próximos al robot. Por ejemplo, los robots Khepera [MON] miden su entorno con 8 sensores infrarrojos, que perciben distancias hasta 5 centímetros y la intensidad de luz del ambiente, gracias a lo cual pueden navegar entre obstáculos o perseguir una fuente de luz (Fig.4.12).



(a) (b)

Figura 4.12: Robot Khepera.
(a) Aspecto del robot (b) Diagrama donde se muestra la disposición de los 8 sensores de proximidad.

4.3.4 Sensores Táctiles

Un sistema de sensores táctiles es utilizado para detectar la colisión del robot con otros objetos, ya sea la pelota, compañeros, rivales o paredes.

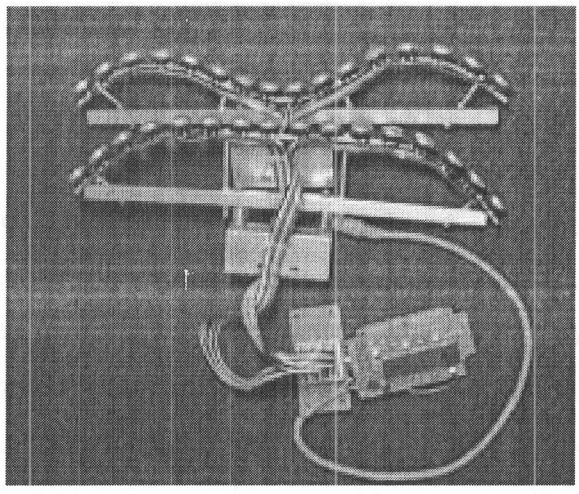


Fig.4.13: Sensores táctiles del Robot NAIST.

Un sistema táctil es principalmente útil para compensar posibles limitaciones del sistema de visión, por ejemplo, cuando la colisión se produce fuera del campo visual, o cuando el sistema de visión no posee la precisión necesaria para distinguir si un objeto está en contacto con el robot.

El equipo RoboCup-NAIST [NAK1] implementa un sistema táctil a partir de la adaptación de un teclado de computadora, el cual se puede observar en la Fig.4.13. El dispositivo táctil se puede ver acoplado en el frente y el fondo del robot en la Fig.4.14.

4.3.5 Otros Sensores

El robot PINO [YAM] posee 8 sensores de fuerza montados en sus pies lo que permite medir su fuerza de pateo. También posee un sensor de balance en su pecho, necesario para mantener el equilibrio.

Para tareas de autolocalización y evitado de obstáculos, algunos equipos como T-Team [PLA] utilizan sensores ultrasónicos.

4.4 Tipos de Procesamiento

Básicamente, se pueden dividir los robots según tengan o no capacidad de procesamiento local. Cuando un robot no tiene capacidad de procesamiento, o ésta es muy limitada, deberá comunicarse con un procesador externo que será quien le provea información y comandos, como ocurre en la categoría MiroSot de FIRA. Estos robots son llamados *brainless* (descerebrados) y poseen menor autonomía que aquellos con poder de procesamiento. Generalmente, son utilizados en categorías con visión global. Al tener el procesamiento fuera del robot, los comandos son transmitidos generalmente por medio de transmisores de radio-frecuencia.

Los robots con procesamiento local tienen diversas capacidades y su procesador puede estar ensamblado de distintas maneras, siendo común encontrar diseños en los cuales directamente los robots tienen acoplada una computadora

portátil, como por ejemplo los miembros del equipo RocoCup NAIST [NAK1]. Su uso se puede observar en la Fig.4.14.

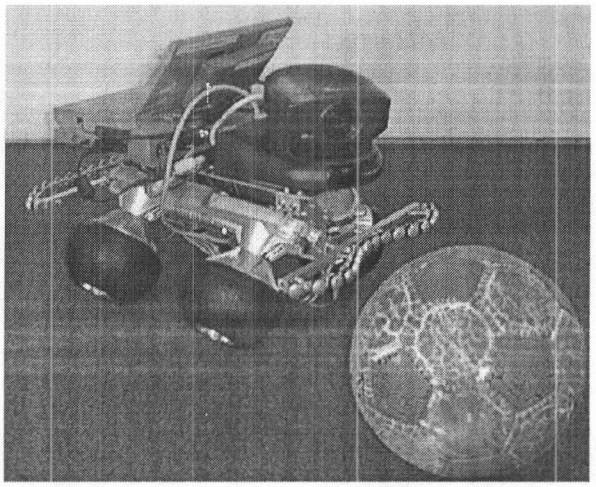


Figura 4.14: Robot NAIST.
El robot posee acoplada una PC portátil. Al frente y atrás del robot, se pueden observar los sensores táctiles.

4.5 Dispositivos Adicionales

Los dispositivos adicionales que algunos diseños de robots incluyen, tienen por objeto controlar o manejar la pelota. Muchos robots incluyen dispositivos para impulsar la pelota, llamados *Dispositivos Impulsores* o *Dispositivos de Pateo*. Estos brindan la posibilidad de dar mayor impulso a la pelota de lo que se puede obtener colisionándola.

El dispositivo agregado en los robots del equipo RMIT Raiders [BRU3] consiste en un resorte liberado por un solenoide el cual es capaz de enviar la pelota a 20 metros de distancia (Fig.4.15).

El equipo CS-Freiburg que participó en RoboCup 98 [GUT1] implementó el dispositivo de pateo que se puede apreciar en la Fig.4.16, que permite impulsar la pelota hasta 3 metros de distancia.

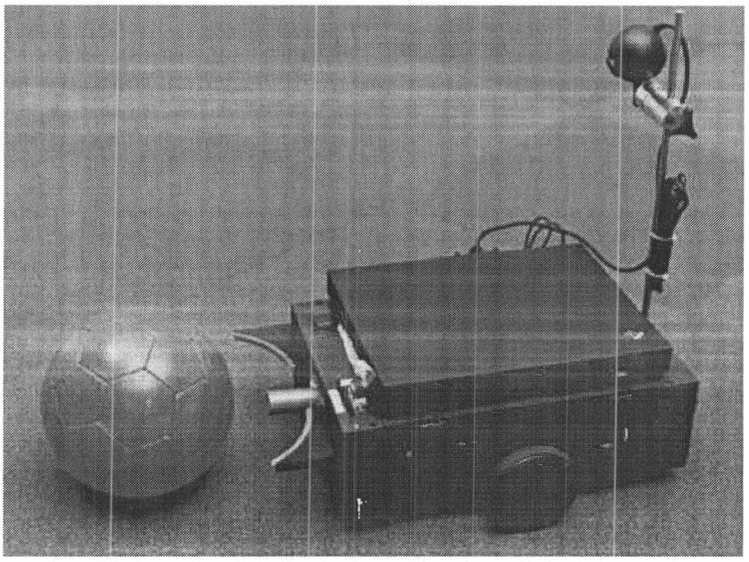


Figura 4.15: Dispositivo de pateo de equipo RMIT Riders.

El mismo equipo CS-Freiburg, en la competencia RoboCup 2000 introdujo cambios en el dispositivo de pateo, el cual se puede observar en la Fig.4.17 y que consiste de cuatro resortes que son comprimidos por un motor de limpiaparabrisas. Cuando los resortes son liberados, se produce un potente pateo de la pelota [WEI].

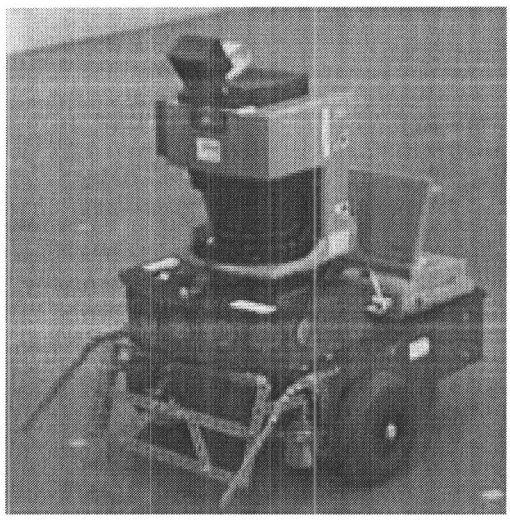
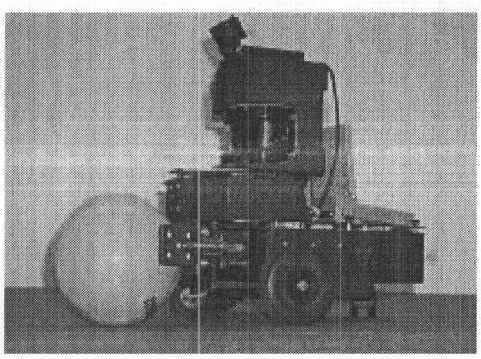


Figura 4.16: Dispositivo de pateo de equipo CS-Freiburg 98.



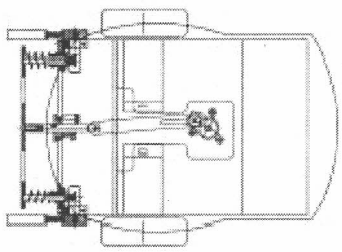


Fig.4.17: Dispositivo de pateo de equipo CS Freiburg 2000.

Existen también dispositivos de pateo de propulsión neumática, como el utilizado por el equipo T-Team Tuebingen [PLA]. El cilindro neumático está controlado por una válvula solenoide y posee un tanque de aire comprimido de 500 cm^3 ubicado en la parte posterior del robot y cuya capacidad es suficiente para aproximadamente 30 pateos.

Capítulo 5
Software de Simulación

5.1 Introducción

Además del fútbol con robots físicos también han sido desarrollados entornos o ambientes virtuales. Para crear el ambiente que permite el desarrollo de un partido de fútbol en forma virtual se han desarrollado herramientas especiales denominadas *Simuladores*.

El simulador provee el entorno que soporta las competencias entre equipos de fútbol virtuales, considerando un ambiente con múltiples agentes, cambiante y con respuestas en tiempo real. Estas herramientas también proveen una interfase gráfica que permite visualizar el desarrollo de los partidos en una pantalla.

El fútbol en ambiente simulado presenta una ventaja importante con respecto al fútbol implementado mediante robots físicos: permite focalizar la investigación en técnicas de cooperación y aprendizaje, abstrayéndose de los inconvenientes propios que pueden presentar los robots y el entorno físico. En este ambiente no es necesario considerar ni resolver problemas en el hardware de los robots, en los mecanismos de comunicación o en el reconocimiento de objetos.

Los simuladores desarrollados y disponibles en la actualidad son los utilizados en las competencias de las categorías de simulación de RoboCup y de FIRA. Aunque ambos simuladores brindan el entorno necesario para que se desarrolle el juego, a la hora de definir las características de los jugadores, cada uno ha aplicado distintos conceptos. En el caso de FIRA, los jugadores simulan a los robots de la categoría MiroSot, conservando la disposición de las ruedas, el tamaño y el peso. Esto hace que tanto los movimientos como las estrategias de

juego deban planificarse teniendo en cuenta estas características especiales. En cambio para RoboCup, los jugadores son agentes con características propias, que no responden en forma directa a ningún modelo o formato de robot en particular. A continuación se detallan las características de ambos simuladores, permitiendo apreciar sus diferencias conceptuales.

5.2 Simulador RoboCup Versión 7.07

El simulador provisto por RoboCup permite ejecutar un partido de fútbol entre agentes autónomos que consisten en programas desarrollados, potencialmente, en diferentes lenguajes. Como se mencionó anteriormente, estos agentes no simulan robots físicos, sino que poseen sus propios atributos. Este sistema presenta un esquema cliente/servidor. El servidor provee un campo de juego virtual, simula todos los movimientos de la pelota y los jugadores, y se encarga de controlar el juego de acuerdo con las reglas. Cada cliente controla los movimientos de un jugador.

La comunicación entre el servidor y cada cliente es a través de UDP/IP. Cada cliente se comunica con el servidor para transmitirle los comandos para controlar a su jugador y para recibir información de lo percibido por el jugador sobre si mismo y sobre el ambiente que lo rodea. En otros términos, el programa cliente se puede ver como el cerebro del jugador. La información recibida por el cliente puede ser de carácter visual, auditivo o relacionado con su estado físico.

Cada cliente puede controlar solamente un jugador, por lo tanto un equipo está compuesto por tantos clientes como jugadores posee. La comunicación entre clientes no está permitida, la misma se establece a través del servidor y mediante comandos especiales.

También existe un cliente especial que corresponde al *coach on line*. Este cliente también se comunica a través del servidor pero, a diferencia de los clientes

descriptos previamente, tiene la capacidad de recibir información global y libre de ruido sobre todos los objetivos del campo.

El servidor trabaja con intervalos de tiempo o ciclos. Cada ciclo tiene una duración específica, por lo que las acciones que necesitan ser ejecutadas en un intervalo dado, deben recibirse desde los clientes en el intervalo correcto. La duración de los ciclos de simulación es de 100 ms. y la frecuencia de comunicación entre los clientes y el servidor es un dato configurable. La coordinación entre el envío de mensajes y los ciclos de simulación, es de suma importancia para alcanzar una buena performance.

Las acciones o comandos que los clientes pueden enviar al servidor para su ejecución se describen a continuación:

- ***Turn(Moment)***: Girar el cuerpo en la dirección indicada. *Moment* se expresa en grados y los valores posibles son -180° y 180°.
- ***Dash(Power)***: Acelerar en la dirección actual. *Power* corresponde a un valor entre –100 y 100.
- ***Kick(Power,Direction)***: Patear la pelota con la potencia y dirección dada. Esta dirección es relativa a la dirección del jugador.
- ***Catch(Direction)***: Retener la pelota. Es un comando especial para el arquero. La dirección también es relativa a la dirección del jugador.
- ***Turn_neck(Angle)***: Girar solamente la cabeza del jugador con el ángulo indicado. *Angle* se expresa en grados y los valores posibles son -90° y 90°.
- ***Say(Message)***: Emitir un mensaje que cualquier jugador cercano puede escuchar.
- ***Sense_Body***: Obtener información física del jugador desde el servidor.
- ***Score***: Obtener información sobre el resultado del partido desde el servidor.
- ***Change_view(Width,Quality)***: Cambiar el campo visual y la frecuencia con que los sensores visuales toman información de su entorno.

Durante su ejecución el servidor accede a una tabla de parámetros o archivo de configuración que cada participante define con anterioridad a la competencia. Estos parámetros son descriptos en forma detallada en el Manual de Usuarios del Servidor de RoboCup [CHE].

5.2.1 Modelo Cinemático de los Jugadores

A través del comando *Dash*, se da al jugador un impulso instantáneo según la magnitud *Power*, lo que se traduce como una aceleración en dirección X e Y. Esta aceleración dura sólo un ciclo, volviendo a 0 en el instante t +1 y está dada por:

$$a_x = Power \times power\_rate \times cos(\theta) \tag{5.1}$$

$$a_y = Power \times power\_rate \times sen(\theta) \tag{5.2}$$

donde a_x y a_y son los componentes de la aceleración sobre el eje X e Y, respectivamente y *power_rate* un coeficiente parametrizable del servidor.

Esta aceleración se aplica en t_0 para calcular la velocidad inicial, a partir del cual se aplica un coeficiente constante de reducción de la velocidad (*decay*). Para simplificar, se muestran los cálculos sólo sobre una de las coordenadas. Un cálculo análogo es aplicable a la otra coordenada.

$$\begin{aligned} V_0 &= Min(V_{anterior} + a_x \times \Delta t, Velocidad\_max) \\ V_1 &= V_0 \times decay \\ &\vdots \\ V_n &= V_{n-1} \times decay = V_0 \times decay^n \end{aligned} \tag{5.3}$$

donde Δt = 1 y *Velocidad_max* es la máxima velocidad que puede alcanzar un robot y es parametrizable en el servidor.

5.2.2 Modelo Cinemático de la Pelota

El movimiento de la pelota se obtiene a través del comando *Kick(Power, Direction)*, cuyo efecto es similar al comando *Dash*. Las diferencias con éste último son las siguientes:

- La dirección efectiva de la pelota será la suma del ángulo que tenga el agente y la dirección pasada como parámetro.
- El comando sólo tiene efecto cuando la pelota está dentro de un *Kickable_area o Área de pateo*, la cual es la suma del radio del jugador (*player size*), el radio de la pelota (*ball size)* y el margen de pateo (*kickable margin)*:

$$Kickable\_area = player\_size + ball\_size + kickable\_margin \tag{5.4}$$

- La *Potencia* del tiro depende del ángulo y la distancia entre el jugador y la pelota. Esto se traduce en un coeficiente que se aplica para calcular el *kick_power*, como se muestra a continuación:

$$Kick\_power = Kick\_power\_rate\left(1 - 0.25 \times \frac{dir\_diff}{180} - 0.25 \times \left(\frac{dist\_ball - player\_size - ball\_size}{kickable\_margin}\right)\right) \tag{5.5}$$

donde *dir_diff* es el valor absoluto del ángulo entre la dirección hacia donde mira el jugador y la posición de la pelota; y *dist_ball* la distancia entre el jugador y la pelota.

5.2.3 Datos para Simulación

El servidor brinda datos que son esenciales para el desarrollo y ejecución de las estrategias de juego de los equipos de fútbol. El servidor *informa* a todos sus clientes en cada ciclo de simulación (cada 100 ms.). Cada cliente recibe su visión sobre el estado del juego debido a que la capacidad visual y auditiva de los jugadores es parcial. Esta información se asemeja a la brindada por un sistema de visión local en un partido de robots físicos. Tomando esta información, la estrategia *decide* la acción a aplicar en el próximo ciclo de simulación.

Los agentes de RoboCup reciben la información sobre el ambiente a través de mensajes enviados por el servidor. Existen tres tipos de mensajes y están asociados con las capacidades de los jugadores: auditiva, visual y física.

El servidor dispara un mensaje auditivo *Hear* cuando el árbitro, el *coach* u otro jugador emiten un comando *Say*. De esta manera el agente toma conocimiento de lo sucedido. La capacidad auditiva de los agentes está limitada por el grado de sordera que pueden tener y por la distancia que los separa del agente emisor. También está controlada la cantidad de mensajes que los agentes pueden procesar: a lo sumo un mensaje de cada equipo por cada segundo de simulación, los restantes son descartados. De este control se excluyen los mensajes provenientes del árbitro y los emitidos por si mismos. Tanto el grado de sordera como la distancia son datos parametrizables.

La información visual del entorno también es proporcionada por el servidor a través del mensaje *See*. En este caso los agentes reciben información de los jugadores y de otros elementos que se encuentren dentro de su campo visual: la pelota, los arcos y las marcas de referencia que tiene el campo de juego. La ubicación y dirección del elemento visualizado son relativas a la posición y dirección del jugador.

El campo visual de los agentes es limitado y corresponde a un cono. El ángulo de este cono y la frecuencia de actualización de los objetivos visualizados son datos configurables. Por defecto, el ángulo es de 90° y la actualización cada 150 ms.

El grado de precisión con que se visualizan los elementos depende de la distancia a la que se encuentren. En el gráfico que se muestra a continuación, el agente tiene información completa sobre el agente *c*; sobre el agente *d* conoce a que equipo pertenece y tiene un 50% de probabilidades de visualizar el número de jugador; para el agente *e* solo tiene un 25% de probabilidades de conocer el equipo y desconoce su numeración; el agente *f* es reconocido como agente pero sin identificación; por último, los agentes señalados como *a*, *b* y *g* están fuera de su campo visual.

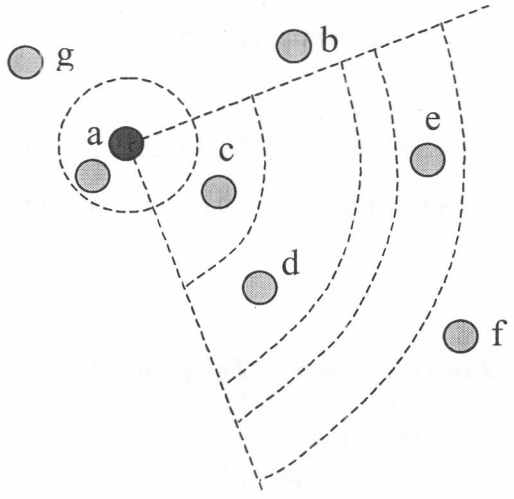


Figura 5.1: Campo de visión de los agentes
Los círculos corresponden a los agentes. Las líneas punteadas delimitan el alcance visual.

Por último el mensaje *Sense_body* provee información sobre el estado físico de los agentes: su grado de stamina, su velocidad y datos complementarios sobre la cantidad de comandos de cada tipo recibidos por el servidor. Estos datos se pueden utilizar para evaluar si hubieron comandos perdidos o demorados.

5.3 Simulador FIRA Versión 1.4

El simulador presentado por FIRA provee un ambiente virtual para llevar adelante un partido de fútbol entre robots no físicos. Los agentes simulados poseen las características físicas de los robots que participan en la categoría Middle League MiroSot. Independientemente de este ambiente, cada equipo desarrolla sus propias estrategias. En el transcurso de un partido, el simulador invoca las estrategias desarrolladas por los equipos participantes. En el lapso de un segundo el simulador invoca 60 veces cada estrategia.

Estas estrategias pueden ser desarrolladas en lenguaje Lingo o en C++. En el primer caso, las estrategias se almacenan en archivos de texto (TXT). En el segundo, los desarrollos C++ deben ser compilados como DLL. Antes de comenzar un partido, mediante un menú que presenta el simulador, cada equipo indica el nombre y el lenguaje de la estrategia a aplicar.

Para la ejecución del Simulador FIRA Middle League SimuroSot versión 1.4, las especificaciones mínimas requeridas para el equipamiento se muestran en la Tabla 5.1.

Especificaciones de Equipamiento	
Procesador	Pentium III 60 Mhz
Memoria	256 Mbytes
Placa de Video	TNT2 3D de 32 Mbytes
Lectora de CD	24x
Resolución de Pantalla	1024 x 768
Placa de Sonido	16 Bits
Entorno	Microsoft Windows 98 o superior
Espacio en Disco	10 Mbytes

Tabla 5.1: Especificaciones de equipamiento.
Características mínimas con que debe contar el equipamiento para la ejecución del simulador.

5.3.1 Modelo Cinemático de la Pelota

El movimiento de la pelota en el proceso de simulación sigue un movimiento rectilíneo uniforme retardado. Se puede representar como:

$$\begin{aligned} V_t &= V_{t-1} + \alpha\,\Delta t \\ X_t &= X_{t-1} + V_t\,\Delta t\,\cos\,\theta_{t-1} \\ Y_t &= Y_{t-1} + V_t\,\Delta t\,\sin\,\theta_{t-1} \end{aligned} \tag{5.6}$$

donde V_t, X_t e Y_t corresponden al estado de la pelota en el tiempo t; V_{t-1}, X_{t-1} e Y_{t-1} a su estado en el tiempo *t-1*. *V* representa la velocidad de la pelota, *X* la coordenada x, *Y* la coordenada y, α es la aceleración causada por la fricción, Δt es el lapso de tiempo y θ la dirección de la pelota.

5.3.2 Modelo Cinemático de los Jugadores

La velocidad de las dos ruedas determinan la velocidad y dirección de movimiento de los robots. Esto se puede expresar de la siguiente manera:

$$\begin{aligned} R &= \frac{L\,(V_M + V_m)}{2\,(V_M - V_m)} \\ \omega &= \left| \frac{V_M}{R + \frac{L}{2}} \right| \\ v &= \omega\,R \\ \beta &= \omega\,T \end{aligned} \tag{5.7}$$

$$\begin{aligned} X_t &= R\,\sin\,\beta\,\cos(\alpha_{t-1} + \beta/2) + X_{t-1} \\ Y_t &= R\,\sin\,\beta\,\sin(\alpha_{t-1} + \beta/2) + Y_{t-1} \\ \alpha_t &= \alpha_{t-1} + \beta \end{aligned} \tag{5.8}$$

donde L es la longitud lateral del robot; V_M es la mayor velocidad en valor absoluto de las velocidades de ambas ruedas; V_m es la menor velocidad en valor absoluto de las velocidades de ambas ruedas; R es un resultado intermedio; ω es la velocidad angular del robot; v es la velocidad lineal del robot; T es el intervalo de tiempo; β es el ángulo rotado en el intervalo de tiempo; X_t, Y_t y α_t corresponden al estado del robot en el tiempo t, X_{t-1}, Y_{t-1} y α_{t-1} corresponden a su estado en el tiempo $t-1$; X la coordenada x del robot; Y la coordenada y; α es la dirección.

5.3.3 Datos para Simulación

El simulador brinda datos que son esenciales para el desarrollo y ejecución de las estrategias. El simulador *informa* en cada ciclo de simulación (60 veces por segundo) el estado del juego, es decir la ubicación de los robots, de la pelota, la instancia de juego, etc. Esta información se asemeja a la brindada por un sistema de visión, en este caso global, en un partido de robots físicos. Tomando esta información, la estrategia *decide* la acción a desarrollar en el próximo ciclo de simulación.

Equipo Azul				***Equipo Amarillo***			
Robot	Nro	X	Y	Robot	Nro	X	Y
	1	90.47	42.13		1	10.66	42.25
	2	82.10	22.94		2	19.91	60.40
	3	81.30	60.44		3	19.74	23.03
	4	61.85	23.13		4	48.36	47.29
	5	61.45	60.38		5	39.89	23.09

Tabla 5.2: Posiciones Iniciales de Jugadores
Distribución de robots dentro del campo de juego con la cual se inicia la simulación.

Para cada robot propio y para cada oponente se conoce su posición en la cancha (X,Y,Z) y su orientación. La posición (X,Y) corresponde al punto central de

la superficie que ocupan los robots sobre la cancha y Z es la mitad de su altura. Este último dato es idéntico y estático para todos los robots. Los tres valores conforman el centro del cubo que representa a cada uno de ellos. Las posiciones (X, Y) iniciales de los robots se muestran en el cuadro anterior.

También están disponibles los datos de la pelota. En este caso solamente se conoce su posición (X, Y).

Para la ubicación de los elementos anteriores dentro de la cancha el simulador brinda la siguiente información de la dimensión del campo:

- Coordenada Y de la pared superior del campo de juego (77.2392).
- Coordenada Y de la pared inferior del campo de juego (6.3730).
- Coordenada X de la pared derecha del campo de juego (93.4259).
- Coordenada X de la pared izquierda del campo de juego (6.8118).
- Coordenada Y del extremo superior de los arcos (49.6801).
- Coordenada Y del extremo inferior de los arcos (33.9320).
- Coordenada X de la línea de fondo del arco derecho (97.3632).
- Coordenada X de la línea de fondo del arco izquierdo (2.8748).

Tanto estas dimensiones como las posiciones anteriores están expresadas en pulgadas (1 pulgada equivale a 2.54 centímetros) y son datos relativos con respecto a la pantalla del simulador. La esquina inferior izquierda de la pantalla corresponde a las coordenadas (0,0).

Cuando se hace referencia a la orientación o los ángulos de rotación de los robots, estos datos por defecto se expresan en grados, con valores entre 0° y 180° y 0° y –180°. Gráficamente estos datos varían según se muestra en la Fig.5.2.

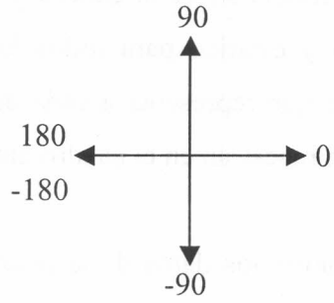


Figura 5.2: Orientación de los Robots
Rango de valores que puede informar el simulador para expresar la orientación de los robots.

Otro dato de suma utilidad en el desarrollo de un partido es su estado (*GameState*). Los posibles estados son:

0	En juego (*Play*)
1	Pique (*Free ball*)
2	Saque desde el círculo central (*Kick off*)
3	Tiro penal (*Penalty kick*)
4	Tiro libre (*Free kick*)
5	Saque desde el arco (*Goal kick*)

Por último, también se cuenta con información sobre el equipo que está reiniciando el juego después de una interrupción (*WhosBall*). Los valores posibles son:

0	Ninguno
1	Azul
2	Amarillo

Analizando la información descripta en este punto, y según la estrategia definida, se determina en forma individual para cada robot cual es su acción para el próximo ciclo de simulación. Las acciones se hacen efectivas con la asignación

de potencia a cada una de las ruedas. Las potencias son valores entre -125 y 125 (valores positivos para avanzar y negativos para retroceder). Las potencias son los únicos datos que requiere el simulador para realizar las acciones [FIR].

Parte II
Aspectos Generales de Diseño

En esta segunda parte se abordan todos los temas que involucra el diseño de un equipo de fútbol de robots en general, más allá del ambiente – real o simulado- , la arquitectura física de los robots –tipo de visión, actuadores, locomoción, etc – o categorías particulares –cantidad de jugadores, tipos de simuladores, reglas de juego, etc -. Algunos de los tópicos mencionados ya han sido detallados en la parte I del presente trabajo.

El abordaje de esta segunda parte se presenta en niveles, los cuales no tienen por qué asociarse con capas de diseño o arquitecturas particulares de los agentes, sin embargo la división de estos niveles puede contribuir a una mejor comprensión de los distintos problemas a ser resueltos al abocarse a la implementación de un equipo de fútbol de robots. Los niveles propuestos son los siguientes:

- Modelo del entorno
- Acciones
- Comportamientos elementales
- Comportamientos complejos
- Jugadas
- Tácticas
- Estrategias

Para poder actuar en un entorno lo primero que se necesita es entender y conocer este ambiente mediante el modelado del mismo. En el capítulo 6 se tocan los temas relacionados a este problema.

En el capítulo 7 se exponen las acciones que involucran la ejecución de los comportamientos. Estas acciones componen el conjunto de primitivas que un agente puede ejecutar .

Se denominan comportamientos a las tareas necesarias para llevar a cabo las jugadas necesarias para el entorno de fútbol. A su vez, cada comportamiento se debe implementar a través de un conjunto de comportamientos elementales. Una posible manera de caracterizar un comportamiento es a través de un mapeo $S \rightarrow A$, es decir $C(S)=a$ donde S es un estado y a una acción, pudiendo ser en ambos casos vectores. En el capítulo 8 y 9 se describen los comportamientos elementales y complejos, respectivamente, que un equipo de fútbol de robots debe implementar.

Cuando los comportamientos deben ser aplicados a un ambiente particular, las ejecuciones de estos comportamientos se agrupan y combinan según las distintos subproblemas que este ambiente contenga. En el capítulo 10 se detallan las jugadas básicamente necesarias para atacar el problema de fútbol.

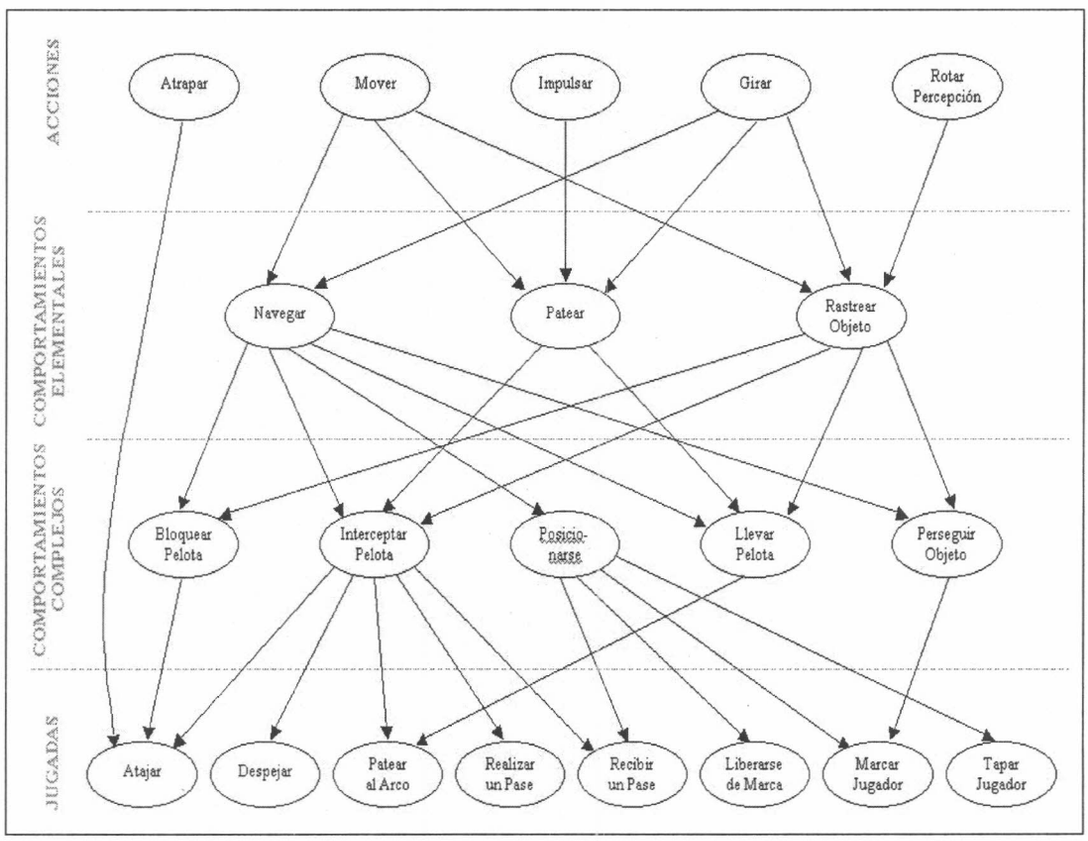
Cuando se habla de estrategias y tácticas, se trata de darles el significado clásico. Es decir, estrategia es el plan global el cual tiene un alcance amplio; mientras que la táctica es una actividad específica dentro de la estrategia y es de alcance limitado. La estrategia es la dirección y la táctica es la ejecución. En ajedrez, táctica son los movimientos a corto plazo para obtener una posición. Estrategia, la forma en que se gana el juego.

En el ambiente del fútbol, la estrategia está generalmente planteada por el entrenador quien decide las subestrategias de defensa y ataque y, en base a esta

estrategia, da las directivas tácticas a los jugadores. En el capítulo 11 se describen la táctica y, en el capítulo 12, la estrategia que involucra la implementación de un equipo de fútbol de robots.

Nótese que, mientras se desciende en esta jerarquía de niveles, menos influye la arquitectura física de los agentes. El nivel de acciones, por ejemplo, depende pura y exclusivamente de las capacidades que el robot .-real o simulado- tenga; mientras que el nivel de estrategias es menos dependiente del aspecto físico, aunque sí lo es de las reglas del juego o categoría.

A modo ilustrativo, el siguiente gráfico muestra la relación entre los niveles de acciones, comportamientos elementales, comportamientos complejos y jugadas.



Capítulo 6
Modelo del Entorno

6.1 Introducción

El entorno se refiere al ambiente que rodea al robot y a la interacción con el mundo real en donde se desenvuelve. El primer contacto con el entorno se da a través de la percepción del robot. Esta percepción puede estar dada por los sensores del robot o por la interfase con el servidor en el fútbol simulado.

Dentro de los elementos a identificar dentro del ambiente, el principal objeto a detectar es el propio robot. Este proceso es lo que se llama autolocalización.

La tarea de armado del modelo difiere significativamente si se cuenta con información global o sólo una parte de ésta como es el caso de los sistemas de visión local. Aquí, puede que se logre un modelo incompleto y local o bien, se puede intentar reconstruir el ambiente completo en base a información brindada por otros agentes.

Una vez obtenido el modelo general del entorno, se cuenta con la información estática del ambiente para el instante actual. Conociendo el modelo físico de los objetos que conforman el ambiente, mediante técnicas de predicción, se pueden anticipar estados futuros y actuar en consecuencia, o bien, en base a estados pasados, completar datos actuales incompletos.

En el presente capítulo se abordan los temas que deben ser tenidos en cuenta a la hora de construir el modelo del entorno, los cuales son enumerados a continuación:

- ***Sistemas de percepción:*** obtener información del ambiente a partir de la percepción.
- ***Autolocalización:*** ubicarse geográficamente dentro del campo de juego.
- ***Reconstrucción del estado global:*** obtener un estado completo del entorno.
- ***Modelado de objetos:*** definir un modelo físico de los objetos dentro del ambiente en el que se desarrolla el juego.
- ***Predicción:*** obtener un modelo del entorno en un tiempo futuro, o actual en base al pasado.

6.2 Sistemas de Percepción

Si bien se pueden citar distintos métodos de percepción del ambiente, al igual que en los seres humanos, la visión es el principal y más importante sentido que un equipo de fútbol de robots debe implementar para obtener información sobre el ambiente y, luego, actuar en consecuencia. El módulo de visión debe ser capaz de identificar los objetos dentro de la cancha (jugadores y pelota), determinar su orientación y, dependiendo de la arquitectura de la implementación, también determinar la velocidad con que estos objetos se desplazan. Evidentemente, toda esta información es imprescindible para la planificación del juego del equipo, por lo tanto la eficiencia el sistema de visión es fundamental para el éxito de las jugadas.

Básicamente un sistema de visión a ser utilizado en el entorno de fútbol de robots debe cumplir con los siguientes requisitos de performance [DIC]:

- ***Precisión*:** Los datos sobre posiciones, orientación y velocidad de los objetos deben poseer un alto grado de precisión debido a que en base a ellos se determinan las acciones de juego.

- ***Velocidad*:** Debido a lo dinámico del juego, es fundamental que el ciclo de procesamiento sea mínimo. Siendo el módulo de visión sólo uno de los componentes de dicho ciclo, debe procesar las imágenes e identificar los objetos en el menor tiempo posible.
- ***Robustez*:** El sistema de visión debe ser capaz de realizar su tarea a pesar de ruido en la imagen, distintas condiciones de iluminación, reflejos, sombras, etc.

Los sistemas de visión que se utilizan en fútbol de robots pueden dividirse en dos grandes grupos: *Visión Global*, donde una o varias cámaras toman la imagen del campo de juego completo desde una determinada altura por encima del mismo; y *Visión Local*, donde cada robot porta una cámara sobre si. El uso de uno u otro sistema depende, además de decisiones de diseño, del reglamento de las distintas ligas de los campeonato de fútbol de robots [FIR] [ROB].

Ya sea en visión global o en visión local existen dos tareas fundamentales que el sistema debe realizar: la detección de objetos y la identificación de los mismos. Otras tareas secundarias pueden depender o no del sistema usado. Por ejemplo, la normalización de la imagen debido al efecto de lentes en las cámaras. Los distintos equipos participantes en campeonatos mundiales han implementado estas tareas con distintas técnicas de procesamiento de imágenes o mediante el uso de software y hardware de visión comerciales.

6.2.1 Sistemas de Visión Globales

Típicamente un equipo con visión global consta de una cámara de video montada por encima del campo de juego y una computadora que recibe las imágenes y que realiza el procesamiento de las mismas. Ésta actúa como servidor de los módulos cliente que representan el cerebro de cada jugador [VEL1] [KIM2].

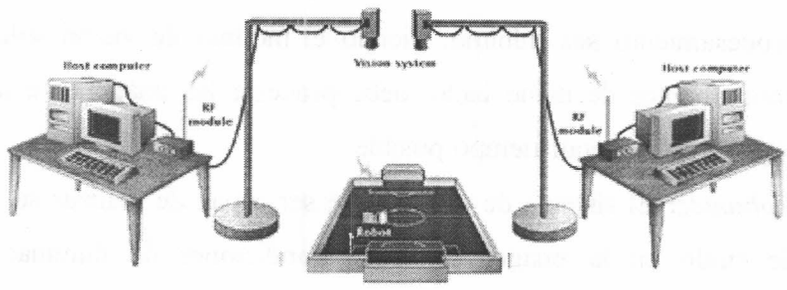


Figura 6.1. Esquema de sistema de visión global.

El sistema de visión global está permitido, entre otras, en la liga de robots de pequeño tamaño o F180 de RoboCup y en la categoría MIROSOT de FIRA [FIR][ROB]. Si bien el objetivo de estas competencias es el desarrollo de agentes autónomos, concepto al cual se adecua mejor la visión local, la visión global posibilita un mejor desempeño a nivel equipo por superar varias limitaciones obvias de la visión distribuida.

Una de las ventajas más destacables de la visión global reside en el hecho de que se cuenta con la imagen completa del campo de juego desde una altura fija durante todo el juego, lo que hace que los objetos sean siempre visibles y su tamaño, relativamente constante. Otra ventaja importante reside en el procesamiento centralizado de las imágenes [DIC]. Esto implica:

- Posibilidad de realizar procesos de alta complejidad en el procesamiento de las imágenes ya que se realizan en una unidad de proceso externa en vez de llevarse a cabo en las limitadas CPU de los robots.
- Menor requerimiento de poder de proceso en los robots.
- Eliminación de funciones repetidas como el procesamiento de imágenes.
- Los jugadores pueden focalizarse en tareas más específicas.
- La planificación de estrategias de equipo resulta más simple y mejor.

6.2.2 Sistemas de Visión Locales

Los equipos con sistema de visión local montan sobre cada jugador una o más cámaras que le permita adquirir en forma autónoma información sobre el entorno.

El procesamiento de esta información es realizado por una computadora central o por el mismo jugador. Cuando el módulo de visión reside en la computadora central, las imágenes percibidas son transmitidas y es la computadora la encargada de detectar los objetos, identificarlos y determinar su posición, devolviendo información y/o instrucciones a los robots. En otros casos, es el mismo jugador quien tiene la capacidad de procesar de las imágenes [KIM2].

Cuando el módulo de visión reside en una computadora central, se debe evaluar la cantidad de robots por equipo y la cantidad de cámaras asociadas. Estos datos determinan el número de imágenes a procesar en forma paralela. Su incremento, degrada el tiempo de respuesta afectando directamente el desempeño de los robots en el juego [KIM2]. En casos donde cada robot se encarga del procesamiento, esta situación no genera conflictos.

En relación a la utilización de estos sistemas en competencias de fútbol de robots, la visión local se aplica tanto en RoboCup como FIRA, en ligas donde participan robots medianos y grandes.

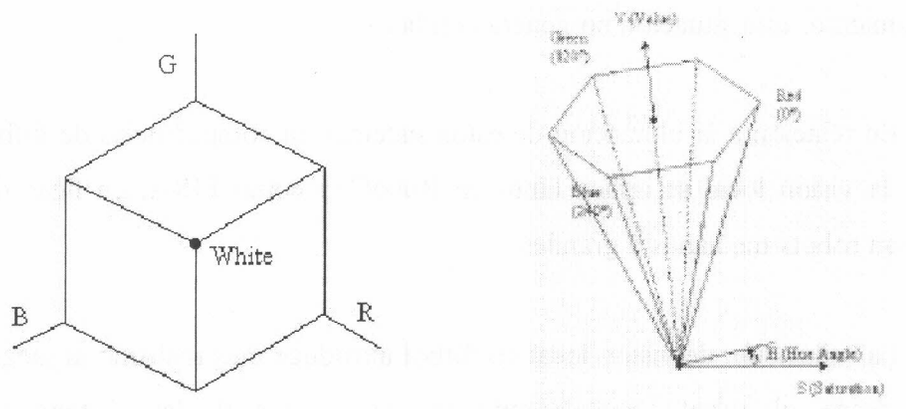
La utilización de visión local en fútbol introduce más realismo al juego. La visión desde el robot, considerando su campo visual, las distancias, las perspectivas, la superposición de los objetos, se asemeja en mayor medida a la visión de un jugador real.

Este realismo también dificulta la identificación de los objetos, de su posicionamiento y la ubicación del robot en si mismo. Los objetos no mantienen

el mismo tamaño en imágenes sucesivas, depende de la distancia y el ángulo de visión. La obtención de las coordenadas donde se encuentra el robot tampoco es tarea sencilla, pues la misma depende de su posición anterior y de los objetos fijos que puedan ser identificados en la imagen.

6.2.3 Espacios de Colores

RGB (*Red Green Blue*) es un espacio de colores muy utilizado en el procesamiento de imágenes, pero para muchas aplicaciones de visión en robótica presenta un inconveniente. En particular, en fútbol de robots, los elementos son marcados con distintos colores para facilitar su identificación (por ejemplo, una pelota de color naranja). Para ello es necesario que el software de clasificación sea robusto para resolver las variaciones de brillo por iluminación, por lo tanto es útil definir "naranja" en términos de porcentajes de intensidad de rojo (*Red*), verde (*Green*) y azul (*Blue*) en el píxel. Esto puede realizarse en el espacio RGB, pero el volumen definido por esta relación es un cuerpo rectangular y no puede ser representado con umbrales simples.



(a) (b)

Figura 6.2. Espacios de Color
a) El cubo del espacio RGB. b) El cono hexagonal de HSI (HSV)

En cambio, HSI (*Hue Saturation Intensity*) e YUV tienen codificada la coloración en dos de las dimensiones (H y S para HSI o U y V para YUV) y la intensidad en la tercera. Por lo tanto, un color en particular se puede describir como una columna variando las intensidades [BRC1].

6.2.4 Detección de Objetos

La primera tarea a ser realizada por el sistema de visión consiste en la detección de objetos dentro de la imagen. Para facilitar esta tarea, tanto en RoboCup como en FIRA, los reglamentos de las competencias fijan el color de la pelota, la cancha y los colores de identificación de cada equipo.

Dentro del proceso de detección, una tarea fundamental es la clasificación de colores. Esta tarea consiste en asociar a cada píxel una categoría dentro de una partición arbitraria de colores. Por ejemplo, la partición puede construirse en base a los objetos a identificar: pelota, arcos, robots, campo de juego, fondo, etc. Del método de clasificación se espera que sea capaz de generalizar a pesar de distintas condiciones de iluminación o saturación de color

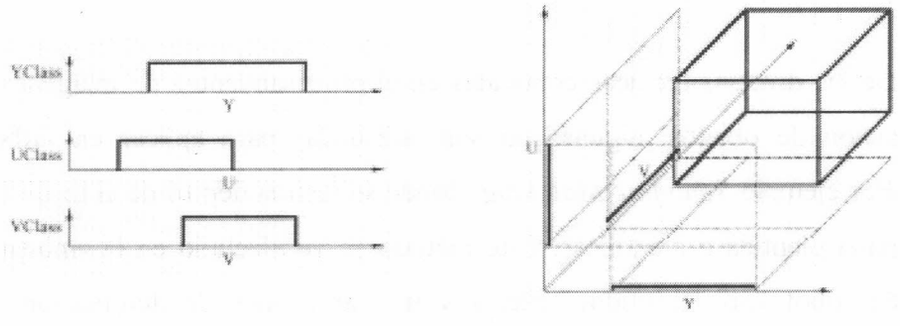
De las diversas técnicas conocidas en el procesamientos de imágenes para identificación de objetos, algunas no son adecuadas para aplicar en fútbol de robots. Por ejemplo *Template Matching,* donde se rastrea dentro de la imagen, una determinada plantilla a identificar. Este método se ve afectado en el ambiente de fútbol de robots por el ruido, reflejos y las variaciones de iluminación de la escena. Si bien algunos equipos utilizan este método, para superar estos inconvenientes deben combinar el método con otros de seguimiento de trayectoria. Otro método que no resulta adecuado es *Detección de Bordes*. Su principal desventaja es su alto costo computacional, por lo cual no se encuentran evidencias de su utilización en fútbol de robots [DIC].

6.2.4.1 Clasificación de Colores

A continuación se describen algunas técnicas de clasificación de colores y métodos de detección de objetos implementados por equipos participantes en competencias de fútbol de robots tanto en sistemas de visión local como visión global.

6.2.4.1.1 Clasificación por Umbrales

Un método simple y rápido para clasificar los píxels de una imagen consiste en comparar sus características (valores del espacio de colores utilizado: RGB, HSI o cualquier otro), con un conjunto de umbrales mínimos y máximos. Estos umbrales pueden ser fijados manualmente o bien a partir del análisis de una muestra de los píxeles de cada tipo de objetos. Para este último método se puede trabajar con la media y la varianza de cada componente de los píxeles. Por ejemplo, los umbrales se pueden fijar en la media más o menos N veces el desvío standard para fijar el umbral superior e inferior respectivamente.



(a) (b)

Figura 6.3: Región tridimensional del espacio de colores.
(a) Descomposición de la señal binaria del umbral. (b) Visualización del umbral en el espacio de colores.

Una vez determinados los umbrales, se forman "cajas" en el espacio de color. Cada píxel que cae dentro de la misma se considera que pertenece al objeto

considerado. Este método posee dos problemas fundamentales. El primero es que se asume que la distribución de colores se ajusta, prolijamente, en una caja. El segundo, las cajas a veces pueden estar superpuestas [BRU1].

6.2.4.1.2 Árboles de Decisión

Cualquiera sea el espacio de colores utilizado, existe una tendencia, especialmente con conjuntos grandes de datos, al solapamiento de los distintos objetos. Esto genera problemas a la hora de clasificar los distintos objetos, como se mencionó en el método de umbrales. Una alternativa es el uso de árboles de decisión los cuales permiten una más detallada partición del espacio de colores en comparación con lo permitido por el método de umbrales.

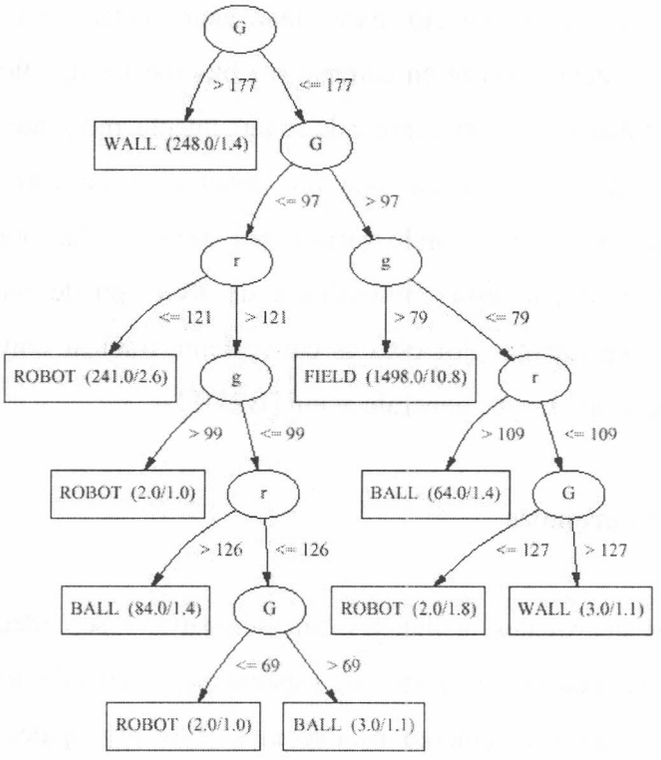


Figura 6.4: Ejemplo de un árbol de decisión.
El modelo de color utilizado en el ejemplo es rgG, donde la componente "G" corresponde al brillo. Los componentes están escalados en el rango [0,255]

Si bien los árboles de decisión pueden ser diseñados manualmente, lo más común es crearlos a partir de un conjunto de entrenamiento utilizando un algoritmo de aprendizaje.

Los árboles de decisión, cuando se basan en valores continuos, tienen la ventaja que pueden clasificar siempre cualquier dato, a diferencia del método de umbrales que puede dejar algún dato sin clasificar. Otra ventaja es que ningún dato tiene clasificación múltiple [BRU1].

6.2.4.1.3 Clasificación por Tablas de Búsqueda

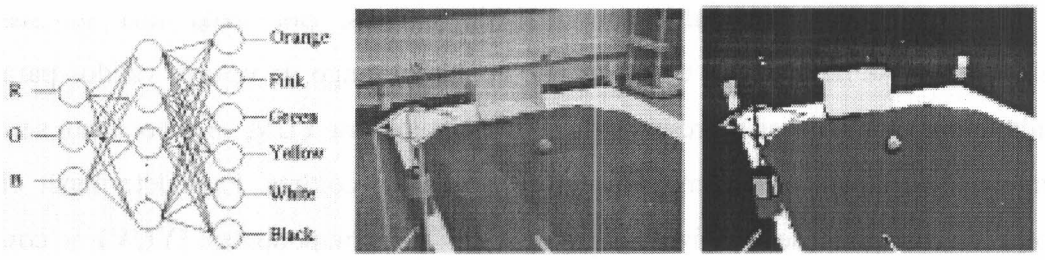
La forma de reconocer colores determinados que lleva a cabo este método es tener una tabla con cada posible valor de los píxeles de la imagen. Esto lleva a que se necesite un gran espacio para almacenar dicha tabla. Una posible optimización del espacio consiste en eliminar los bits menos significativos de cada componente. La tabla puede ser entrenada manualmente mediante la recolección de muestras de píxels de cada objeto. Desafortunadamente, esto genera típicamente un vector esparso donde variaciones sutiles de luz pueden afectar la clasificación debido a que esta combinación de RGB puede no haber estado presente en el entrenamiento. Por esto es conveniente trabajar con un espacio de colores que permita una mayor generalización [BRU1].

6.2.4.1.4 Redes Neuronales

Las redes neuronales artificiales han demostrado ser adecuadas para su aplicación a la clasificación de *patterns*, especialmente cuando los limites de la clasificación no están exactamente establecidos. Sus principales características son [DAS]:

- Capacidad para aproximar cualquier función

- Capacidad de generalización bajo situaciones donde la red no fue entrenada
- Capacidad de aprender a partir de ejemplos
- Robustez sobre errores en el entrenamiento



(a) (b) (c)

Figura 6.5: Ejemplo de uso de redes neuronales para la liga de perros de RoboCup
(a) Arquitectura de la red neuronal para clasificación. (b) Imagen original del campo de juego. (c) Salida de la red neuronal para la imagen b.

Un importante tema a tener en cuenta es elegir adecuadamente los datos de entrada que se le proporcionarán a la red. Para el problema de clasificación de colores, estas entradas corresponderán a determinadas características cromáticas de los píxels de la imagen. Utilizando las características de RGB de los píxels de una imagen del entorno de fútbol de robots y un perceptrón multicapa, se puede obtener un buen grado de generalización con un costo relativamente bajo de entrenamiento [DAS]. Otras implementaciones de clasificación con redes neuronales utilizan como datos de entrenamiento características del espacio de color HSI. Por ejemplo, a partir del histograma de saturación, se pueden generar N niveles a ser utilizados como entradas a la red, donde cada píxel se representa con un valor 1 en la neurona correspondiente al nivel de saturación al que pertenece y 0 en todas las demás [AMO].

6.2.4.2. Métodos de Detección

A continuación se explican algunos métodos de detección implementados

por equipos participantes en competencias de fútbol de robots tanto en sistemas de visión local como visión global.

6.2.4.2.1 Algoritmo de Regiones Conectadas

En un espacio multidimensional de colores, por cada una de sus dimensiones, se definen dos umbrales que acotan el rango de valores válidos para cada uno de los colores representados. Por ejemplo para YUV, se especifican seis umbrales: dos para cada dimensión en el espacio de colores. Para determinar el color de un pixel se evalúan los valores de sus componentes (YUV) y con respecto a los umbrales definidos para cada clase de color. Si los valores se encuentra entre los umbrales establecidos para un determinado color, el pixel pertenece a esa clase.

Después de clasificar los pixels en forma individual, se analiza la imagen por filas con la finalidad de conformar regiones considerando la semejanza de colores. Recorriendo cada fila, los pixels adyacentes y similares se agrupan formando *Series*.

Habiendo conectado horizontalmente los pixels, se intenta conectar estas series en forma vertical generando una estructura de árbol entre ellas. Las series agrupadas conforman regiones, tienen en común la tonalidad y en su origen pertenecen a filas adyacentes. De estas regiones se pueden obtener sus límites, su centroide y su tamaño.

Para finalizar, se analiza la proximidad de regiones de la misma tonalidad y se evalúa la integración de ambas regiones en una solamente. En esta integración se verifica que la densidad de color en la nueva región se mantenga sobre un umbral preestablecido. De no ser así, las regiones se mantienen independientes [BRC1].

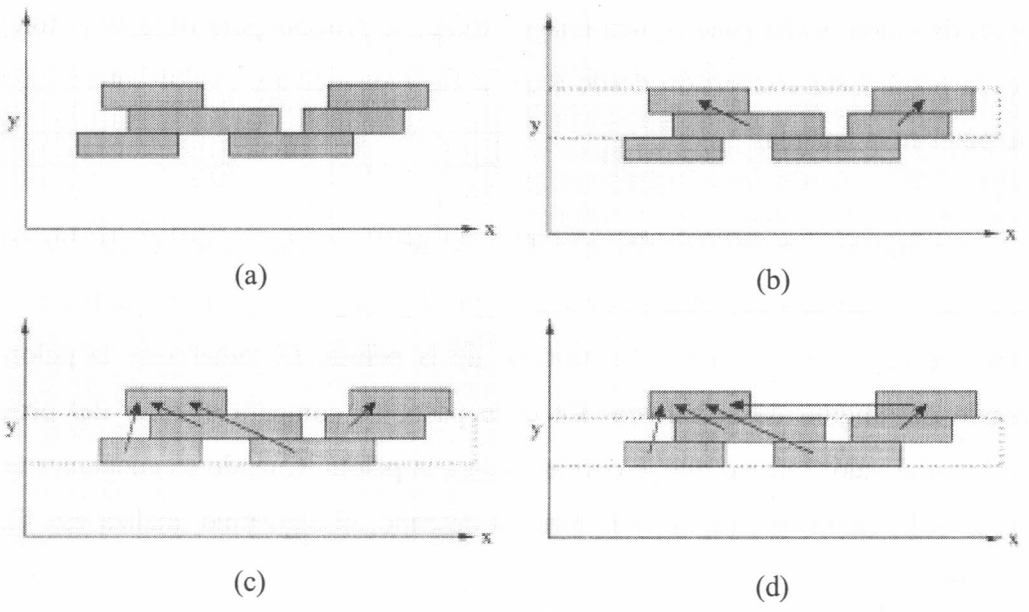


Figura 6.6: Cuatro etapas en la conformación del árbol de regiones.
(a) Estado inicial. (b) Combinación de regiones entre filas adyacentes. (c) Nueva asignación de raíces en forma transitiva. (d) Estado final

Esta técnica ha sido utilizada por los equipos Carnegie Mellon en RoboCup 99 en las ligas Sony y F180 [BRC1].

6.2.4.2.2 Cognachrome Vision System

Este sistema de visión, fabricado por Newton Research Labs, es un sistema comercial de visión basado en hardware utilizado por varios equipos de fútbol de robots. Dado que el sistema utiliza hardware dedicado, el procesamiento de imágenes es más veloz [BER]. Implementa clasificación de colores mediante umbrales fijos. Esta herramienta presenta dos limitaciones importantes: costo elevado [BRC2] y la capacidad de identificar solamente tres colores [YOU].

6.2.4.2.3 Algoritmo de Graf

El algoritmo de Graf comienza determinando el tamaño estimado de la pelota. Este dato es fundamental en el análisis de los cuadros tomados por la

cámara de video. Cada cuadro, una imagen RGB, se procesa para obtener el tono de cada píxel. Esta conversión desde RGB a HSV devuelve un valor entero para cada píxel en la imagen.

La imagen se recorre fila por fila, de arriba hacia abajo. Cada fila se analiza de izquierda a derecha, se evalúa un píxel y luego se ignora la cantidad de pixels que equivale a la mitad del tamaño de la pelota. El tamaño de la pelota depende del ángulo de la cámara. En cada paso, se compara el tono del píxel actual con el valor teórico que le corresponde a la pelota. Cuando se encuentra un píxel lo suficientemente cercano en tono al deseado, el algoritmo analiza esa fila en detalle.

Detectado el primer píxel con el color buscado, se localizan en la misma fila el extremo izquierdo y derecho con el mismo color. Luego se obtiene la distancia entre ambos extremos para determinar si corresponde al tamaño de la pelota. De ser así, se calcula el tono promedio de los pixels entre ambos extremos y se compara con el valor deseado. Siendo un valor cercano se considera el píxel central entre los extremos como el centro de la pelota. En caso contrario, se repite el procedimiento restringiendo el área, es decir acercando los extremos.

El mayor problema de esta técnica es la velocidad con la que se procesa la imagen. Cuando el promedio de procesamiento es de 3 cuadros por segundo, es posible que la pelota pase sin ser detectada. Este método ha sido implementado por el equipo CIIPS Glory [GAL].

6.2.4.2.4 Transformada de Hough Fill

La transformada de Hough se utiliza para generar información de características parametrizadas (círculos, líneas, etc.) a partir de información sobre bordes en la imagen. La aplicación de esta técnica, y en particular la detección de círculos, es adecuada en fútbol de robots ya que para algunas competencias los

demarcadores de jugadores son pelotas de ping-pong y la propia pelota en la imagen son vistos como círculos.

Cada píxel detectado como borde es candidato a ser parte del borde del círculo buscado. El algoritmo parte de la premisa que si se dibuja un círculo alrededor de ese píxel con el mismo radio, entonces el centro del círculo buscado formará parte del borde de este nuevo círculo. Si se trazan círculos para cada píxel detectado como borde, el punto más "dibujado" será aquel que forme parte de todos los círculos y por ende el centro del círculo buscado.

La desventaja de este método es la carga computacional que introduce y la necesidad de realizar previamente una detección de bordes en la imagen.

La técnica de Hough Fill se basa en el uso de la transformada de Hough para detectar figuras circulares en una imagen donde los objetos están bien diferenciados por color. De esta forma elimina su mayor desventaja de requerir información sobre bordes. Hough Fill combina las técnicas de clasificación por color y la transformada de Hough. El requisito de que los objetos en la imagen estén bien diferenciados por color se cumple en la aplicación de fútbol de robots. El método utiliza el prototipo de color HSV para clasificar la imagen.

El funcionamiento es básicamente similar al anterior. El algoritmo busca píxels del mismo color que la figura buscada. Cuando un píxel candidato es encontrado, la premisa de que el centro de la imagen buscada reside en el borde del círculo alrededor del píxel encontrado ya no es cierta. Esto se soluciona pues el algoritmo incrementa en la matriz acumulador todas las posiciones de la imagen que pertenecen al circulo con centro en el píxel candidato y no solo los bordes. Una vez procesados todos los candidatos, el máximo valor de la matriz corresponde al centro de la figura buscada.

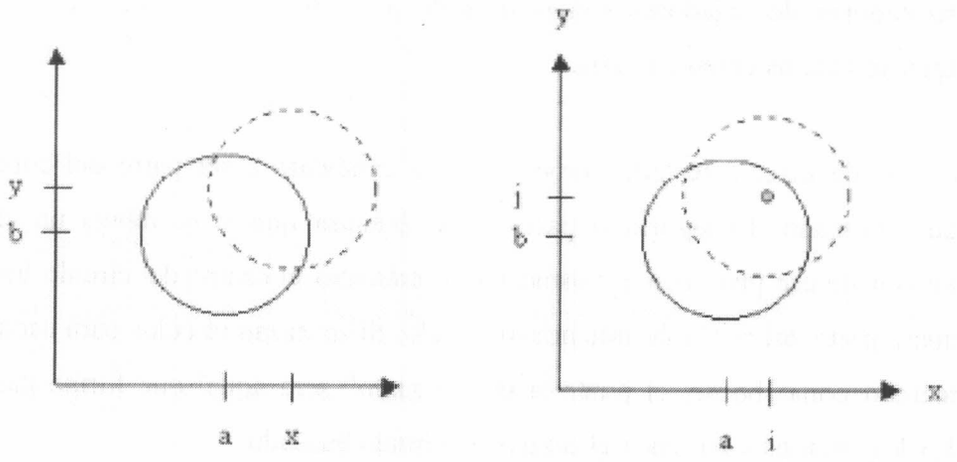


Figura 6.7 Transformada de Hough y de Hough-Fill
(a) El centro (a,b) del círculo actual se encuentra en un círculo del mismo radio R, centrado en un píxel del borde (x,y). (b) En la transformada de Hough-Fill, el centro del círculo (a,b) no necesita estar en el círculo centrado en un píxel candidato (i,j).

La técnica de Hough Fill hereda todas las ventajas de la transformada de Hough: es una técnica precisa para detección de figuras circulares, localiza el centro de la figura, es robusta pues puede trabajar sobre figuras parcialmente ocultas y agrega el hecho de ser relativamente rápida al eliminar el costoso proceso de detección de bordes [DIC].

6.2.5 Identificación de Objetos

Una vez que los objetos de la escena han sido detectados, es preciso determinar de que objeto se trata: pelota, arcos, robots rivales o compañeros. En sistemas de visión global, se presenta una tarea adicional: la individualización de cada uno de los robots que conforman el equipo propio. Este dato tiene fundamental importancia a la hora de transmitir información y/o instrucciones a cada uno de ellos. En sistemas de visión local, esta tarea no es necesaria ya que el mismo robot es quien procesa la imagen. De procesarse en una computadora central, ésta tiene individualizado el robot que percibe la imagen pues es quien le transmite la misma.

Una técnica para distinguir entre los distintos jugadores parece ser la asignación de un color adicional único a cada jugador. Esto hace que se deban utilizar tantos colores adicionales a los reservados para el campo de juego, marcas de la cancha, arcos, pelota e identificación de equipos, como jugadores compongan el equipo. Para esto se debe contar con colores lo suficientemente diferenciables para su correcta detección. La variación de luz sobre la cancha, los reflejos, sombras y ruido de la imagen son problemas que dificultan la aplicación de esta técnica en el entorno de fútbol de robots [DIC][VEL2].

Algunas de las técnicas utilizadas en fútbol de robots para la identificación de objetos se describen a continuación:

6.2.5.1 Asociación por Distancias Mínimas

La asociación de dos objetos entre cuadros consecutivos responde a distancias mínimas. Se considera el mismo objeto, al elemento más cercano a la última posición registrada para ese objeto. Es decir, al elemento que se encuentra a menor distancia con respecto a la última lectura de ese objeto.

Este algoritmo examina solamente un área rectangular de la nueva imagen centrada en la posición anterior del objeto. Está técnica reduce el tiempo que demanda la detección debido a que la búsqueda no se debe realiza en la imagen entera.

Mediante distancias y siguiendo los pasos anteriores, se obtienen en primera instancia las posiciones de los jugadores. Detectados los jugadores, es posible determinar su orientación. Para las competencias donde la cantidad de colores asignados a cada jugador no es una restricción, se pueden usar dos colores distintivos, uno correspondiente a su equipo y otro solamente para determinar su orientación. En estos casos, se calcula la distancia entre el color de equipo y los

elementos que se detecten con el segundo color. El elemento del color buscado con distancia mínima, determina la orientación del jugador [DIC].

6.2.5.2 Patterns Binarios

Mediante este método se asigna una identificación unívoca a cada robot sin necesidad de utilizar colores. Para esto se ubican en todos los robots n marcas de color blanco o negro formando una codificación binaria (no se utilizan los códigos con todas las marcas de igual color). Identificando estas marcas se elimina el riesgo de intercambiar robots cuando más de un jugador es perdido por el sistema de seguimiento. Este método ha sido utilizado por el equipo FU-Fighters en RoboCup 2000 [SIM].

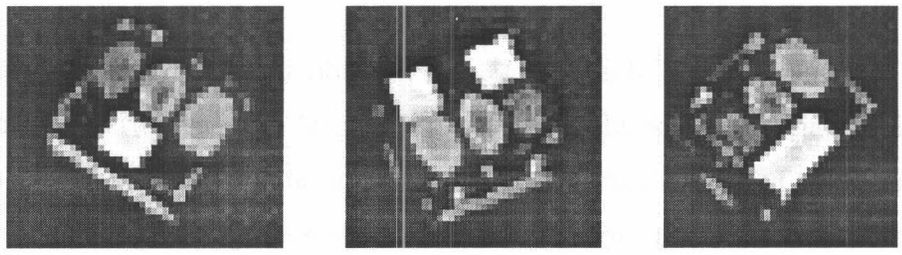


Figura 6.8: Identificación con patterns binarios.
Aquí se muestran tres códigos binarios, representados por las marcas cuadradas blancas y negras ubicadas al costado de las marcas de colores.

6.2.5.3 Ventanas de Búsqueda Variables

Este algoritmo tiene por objeto reducir la cantidad de imagen procesada en cada cuadro en base a la predicción de la posición de los objetos, procesando una pequeña ventana alrededor del mismo.

Básicamente, lo que hace el algoritmo es reducir lentamente el tamaño de las próximas ventanas cuando el objeto fue encontrado en su ventana y aumentar

el tamaño de la misma por un factor de dos si no fue detectado el objeto. Durante el juego regular, el tamaño promedio de las ventanas de búsqueda está cerca del mínimo. Las ventanas grandes sólo son necesarias cuando los objetos se mueven muy rápido o son cambiados manualmente de lugar. Para la pelota no hay problema ya que, al incrementarse su ventana, siempre se podrá volver a encontrar. El problema aparece en los robots, de los cuales se puede perder el rastro. Para ello se implementa un mecanismo adicional de búsqueda global por color. Si más de un robot se pierde, se asigna por cercanía a la posición anterior .Este método ha sido utilizado por el equipo FU-Fighters en RoboCup 2000 [SIM].

6.2.6 Normalización de la imagen

El sistema de visión debe proveer una exacta localización para cada objeto en el campo de juego. Esto resulta sencillo pues la forma rectangular del campo permite utilizar aproximación lineal. La Fig.6.9 muestra como la lente de la cámara puede obtener una imagen distorsionada del campo debido a que, por la poca distancia entre la cámara y la cancha, generalmente se utiliza un lente gran angular. En estos casos, a menos que la imagen sea corregida, no es posible obtener la información en forma precisa.

Un método heurístico presentado para sistemas de visión global [DIC] convierte la imagen a una imagen rectangular moviendo los pixels para que entren en un rectángulo, como se observa en la Fig.6.10. Cada píxel es movido una distancia proporcional. Para un punto P del campo de juego, su nueva ubicación (x,y) es calculada mediante la ecuación:

$$x_{new} = x_{old} + s\frac{B}{A}\Delta x \qquad (6.1)$$

$$y_{new} = y_{old} + s\frac{C}{D}\Delta y \tag{6.2}$$

donde s $\in \{-1,1\}$ indica el sentido del corrimiento y las variables *A, B, C* y *D* representan los segmentos que se muestran en la Fig. 6.10.

Para realizar la transformación se deben determinar los bordes. El algoritmo trabaja con los 4 puntos de las esquinas interiores del campo. Para cada punto en estas líneas, se procede a buscar, desde el centro hacia fuera, el primer píxel blanco. Este píxel es marcado como borde de la cancha. Los puntos se aproximan con una curva cuadrática para evitar outliers y suavizar los bordes. La curva se obtiene aplicando el método de cuadrados mínimos.

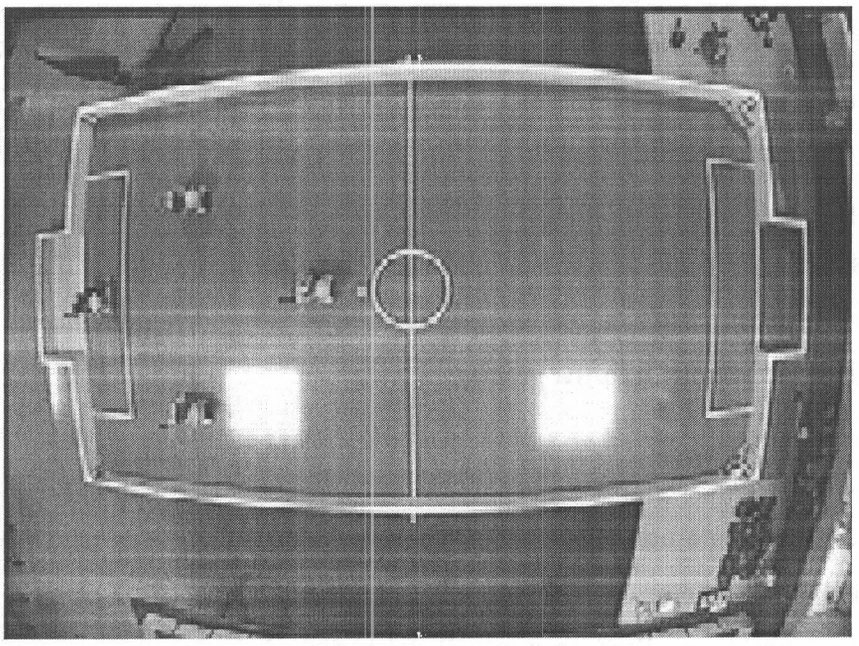


Figura 6.9: Imagen inicial distorsionada

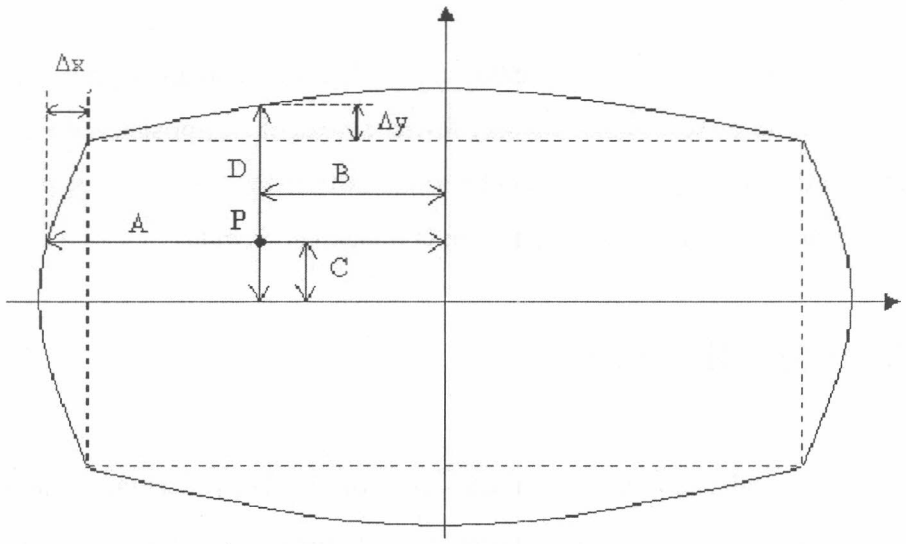


Figura 6.10: Corrimiento de los pixels para entrar en un rectángulo.

Solamente es necesario calcular las nueva posiciones en la imagen corregida una vez pues la cámara y el campo de juego permanecen estáticos. Las nuevas posiciones *(x,y)* son almacenadas en una tabla de búsqueda para un rápido acceso. Otra ventaja de utilizar una tabla de búsqueda es que los pixels que no forman parte de la cancha son marcados para no ser tenidos en cuenta.

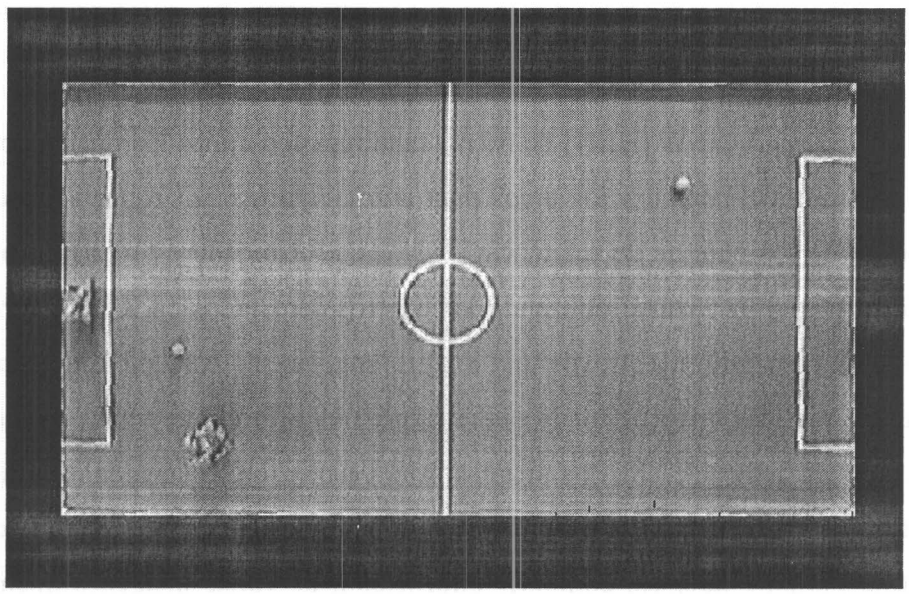


Figura 6.11 Imagen normalizada.
En la parte media derecha sobre el borde inferior pueden verse huecos donde no se asignó ningún píxel.

Si bien el método es sencillo, tiene dos desventajas significativas: la transformación no es reversible. No hay forma de calcular la posición de un punto en la imagen distorsionada. Esto se debe a que la transformación no es biunívoca. La transformación puede dejar agujeros en la imagen corregida.

6.3 Autolocalización

La autolocalización supone ubicar las coordenadas y orientación actual del robot en base a la información incompleta del medio. Nuevamente esta tarea resulta trivial en los sistemas de visión global a menos que el ruido en la percepción genere la ausencia de la posición actual del robot. En estos casos se debe recurrir al modelo del entorno y a la predicción. Ambos temas son ampliados más adelante.

La autolocalización cobra sentido en los sistemas de visión local, donde cada robot cuenta con información de los objetos que percibe relativa a su posición, aún desconocida. De esta forma, para poder ubicar estos objetos, primero deberá saber donde se encuentra él mismo.

El equipo UNSW [HEN] de robots cuadrúpedos utiliza los postes ubicados en lo costados del campo y los arcos para autolocalizarse. El robot almacena las posiciones de los objetos de referencia mientras se mantiene estático. En primera instancia, intenta localizarse utilizando los objetos detectados en la última imagen. Como la cámara utilizada provee una imagen angosta, no es posible detectar tres marcas en una sola imagen, por lo que cualquier algoritmo que utilice más de dos marcas no es aplicable. Si dos marcas son visibles en la imagen se utiliza el algoritmo de triangulación, a partir de la distancia y orientación de las marcas. El algoritmo permite, además, combinar información de varias imágenes mientras el robot no cambie de posición, para poder utilizar el algoritmo de triangulación.

Esto es útil cuando el robot se para y gira la cabeza para buscar la pelota. Para esto utilizan un vector donde se almacena la información de las marcas. Este vector es limpiado cada vez que el robot se mueve.

En este equipo, el módulo de modelado recibe información del módulo de locomoción para ajustar la posición del robot, en distancia en cm del desplazamiento sobre los ejes X e Y. Esta información no es muy exacta lo que produce que el error acumulado en cada paso lleve a un error más grande. Además, si el robot es obstruido, el módulo de locomoción envía información incorrecta sobre el desplazamiento. Para esto se desarrolló un método para ajustar la posición en base a la observación de una sola marca, como se muestra en la Fig.6.12. La base de este método es que, al percibir a lo largo del tiempo varias marcas, ajustar la posición paso a paso lleva a una aproximación a la triangulación.

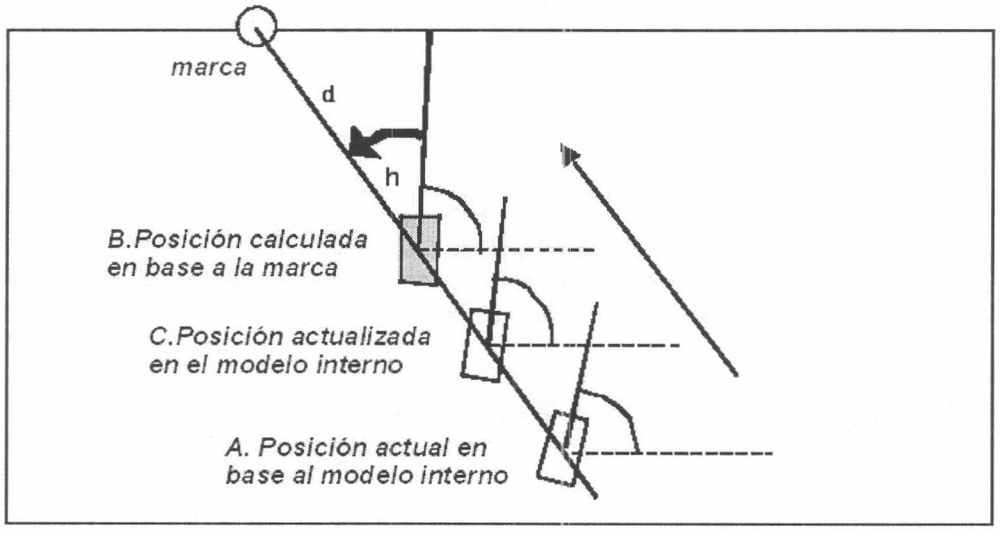


Figura 6.12:. Ajuste de la posición con una marca.
Para estimar la posición del robot, se dibuja una línea entre la marca y la posición actual estimada (A) del robot en el modelo interno. La posición calculada (B), relativa a la marca, es el punto en esa línea a d cm. de la marca. El algoritmo de localización promedia ambas posiciones obteniendo la nueva en (C)

El equipo CS Freiburg [GUT1] utilizó un método de autolocalización basado en escáners laser. Si bien generalmente estos métodos se utilizan para

ajustar una posición previamente conocida, el equipo utilizó los escáners laser para calcular su posición sin información adicional.

El algoritmo extrae segmentos de líneas de escáner que mapea con un modelo a priori del campo de juego. Para asegurarse de que las líneas correspondan a los bordes de la cancha, se tienen en cuenta solamente líneas bastante más extensas que el ancho de un robot. El escáner cubre 180 grados y es utilizado para autolocalizarse y para detectar robots. La Fig.6.13 muestra un ejemplo de este proceso.

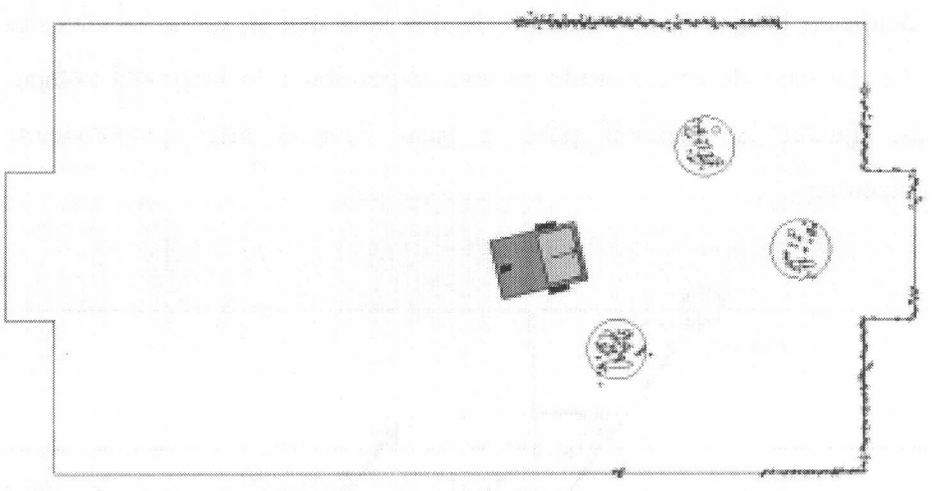


Figura 6.13:. Escáner laser de CS Freiburg.
Los segmentos de línea son extraídos del escáner y mapeados contra los bordes de la cancha. Tres robots son encontrados en el proceso.

6.4 Reconstrucción del Estado Global

Obtener un estado completo del entorno puede resultar una tarea sencilla o compleja dependiendo de cómo se perciba el mundo que rodea al robot. En los sistemas de visión global este problema está resuelto pues el sistema de percepción brinda un mapa completo. En la categoría de simulación de FIRA [FIR] por ejemplo, el servidor provee las coordenadas y orientación de cada robot y la posición de la pelota a cada instante. Lo mismo sucede con las categorías

MiroSot de FIRA y F180 de RoboCup [ROB] donde cada equipo puede colocar una cámara sobre el campo de juego obteniendo la imagen total del campo de juego. En este caso el problema consiste en procesar dicha imagen pero no es el tema que se abarca en esta sección.

En los sistemas de visión local, cada agente percibe una porción del entorno. En la categoría de simulación de RoboCup [ROB] cada agente recibe del servidor la información que puede visualizar en un ángulo de 90°, 45° hacia cada lado desde la orientación actual. En este caso el agente no cuenta con la posición de todos los robots y hasta puede no saber donde se encuentra la pelota. En estos casos completar el modelo implica percibir información del resto del espacio o compartir información con los otros agentes del equipo. En ambos casos, a causa del ruido en la percepción hay que lidiar con información incompatible y, a veces, incompleta para reconstruir el entorno.

El equipo de simulación AndHill [AND], en base a la información parcial y confusa que recibe cada agente del servidor arma el estado global. Los jugadores utilizan el comando *say* para compartir la información propia con el resto de sus compañeros.

6.5 Modelado de Objetos

Modelar el ambiente implica armar un modelo mental de cada objeto incluyendo sus características físicas y su cinemática. En las categorías de simulación, según el simulador, esta información es conocida o calculable aunque el ruido que introduce el servidor puede alterar el modelo. En las categorías de robots reales se debe modelar el mundo físico donde no siempre las cosas se comportan de igual manera. Como ejemplo se puede citar el movimiento de la pelota que, dependiendo de que tan parejo sea el suelo o el efecto que tenga la pelota, puede variar o ser impredecible. Otro problema que se presenta en las

categorías de robots reales es modelar el comportamiento de los oponentes pues no son conocidas sus características físicas.

6.6 Predicción

En un juego de fútbol de robots, la habilidad de detectar sólo la localización de los objetos en el campo de juego a menudo no es suficiente. Como en fútbol real, frecuentemente es fundamental para el robot predecir la futura localización de la pelota (o incluso de otros jugadores) [VEL2]. También debido al ruido en la observación se torna difícil realizar mediciones de velocidad confiable, siendo necesario un seguimiento de trayectorias [HAN]. La predicción futura consiste en, a partir de un modelo actual del entorno y a información histórica, construir un posible modelo del entorno en un tiempo futuro. Esta predicción se utiliza para determinar trayectorias para interceptar la pelota, tomar decisiones sobre acciones a tomar en base a un estado futuro del juego, construir el modelo actual en base a información incompleta y un estado anterior, prediciendo las posiciones de los objetos no encontrados.

El filtro de Kalman provee una solución al método de cuadrados mínimos y al mismo tiempo es computacionalmente eficiente. Su naturaleza recursiva es adecuada para sistemas de tiempo real, es robusto contra ruidos del sistema y es capaz de producir buenas estimaciones incluso cuando se desconoce la precisa naturaleza del sistema modelado. El filtro de Kalman básico puede ser aplicado sólo a sistemas lineales. El Filtro Extendido de Kalman (EKF) permite su aplicación a sistemas no lineales mediante la linearización del sistema de ecuaciones centrado en la mejor estimación [ROS].

El método involucra una iteración de dos pasos llamados *Actualización* y *Propagación*. La mejor estimación del estado del sistema y la covarianza del error son calculados en cada iteración. Durante la actualización la observación es usada

para refinar la estimación actual y se recalcula la covarianza. En la propagación, el estado y la covarianza del sistema en el próximo paso es calculado usando las ecuaciones del sistema. El proceso luego se repite: el paso de actualización refina la estimación usando la observación y así sucesivamente.

La predicción se lleva a cabo mediante repetidas aplicaciones de las ecuaciones de propagación al estado actual estimado. Para obtener la predicción del estado del sistema N pasos adelante, se deben aplicar N veces las ecuaciones de predicción.

Para modelar la pelota por ejemplo, se puede capturar el estado de la misma a través de cinco variables: las coordenadas x e y de su localización, la velocidad, la dirección y un parámetro de fricción. Con estas variables se arma un conjunto de ecuaciones diferenciales.

A través de un cuidadoso ajuste de los parámetros del filtro, se puede obtener exitosamente el seguimiento y predicción de la trayectoria de la pelota [VEL2]. Esta técnica ha sido utilizada por ejemplo por los equipos CMUnited en RoboCup 97 [VEL1], Rogi [ROS] y FutBotIII [GAL] en RoboCup 99.

Capítulo 7
Acciones

7.1 Introducción

Se definen acciones en fútbol de robots como aquellos comportamientos atómicos que realizan los robots. Son los "primeros pasos" que se deben aprender para poder jugar al fútbol. Sin ellos no es posible implementar comportamientos o jugadas necesarios para el juego del fútbol.

A su vez, las acciones dependerán en gran medida de la arquitectura del robot, en el caso de los robots reales, o de las facultades ofrecidas por el servidor, en el caso de simulación. En el caso de robots reales, los actuadores del robot determinarán la forma de implementación de las acciones en gran medida. No será lo mismo moverse con ruedas que con dos o cuatro patas o apéndices. En el caso de simulación sucede algo similar. Algunos simuladores ofrecen un conjunto de acciones nativas que el jugador posee mientras que en otros cada una de estas acciones requiere un desarrollo específico.

La siguiente lista enumera las acciones utilizadas en fútbol de robots, las cuales son ampliadas a continuación:

- ***Mover:*** desplazar al agente sobre la superficie del campo de juego.
- ***Girar:*** cambiar de orientación sin desplazarse de su posición actual.
- ***Impulsar:*** separar un objeto del robot mediante un impulso.
- ***Atrapar:*** retener un objeto cerca del robot.
- ***Rotar percepción:*** cambiar el ángulo de la percepción sin cambiar la orientación del robot.

Las acciones no tienen particularmente en cuenta el entorno como un ambiente de fútbol, pues podrían aplicarse a otros ambientes. Asimismo, no importa distinguir si un objeto es un robot del mismo equipo o del equipo contrario.

7.2 Mover

La acción de mover se refiere a desplazarse en el campo de juego con una determinada dirección y un determinado sentido. Esto que puede ser tan sencillo en algunos ambientes, puede involucrar mucho esfuerzo y trabajo en otros. El software de simulación en la liga RoboCup [ROB] provee la primitiva *dash* para desplazarse en forma recta con una cierta potencia en la dirección y sentido actuales, simplificando la acción al cálculo de esta potencia.

Otra variante muy utilizada es el desplazamiento de robots con ruedas laterales. Se pueden encontrar robots con dos o más ruedas. En el primer caso el desplazamiento se realiza asignando una potencia a cada una de las ruedas de forma que el robot describe una trayectoria en forma de arco. En este caso hay que tener en cuenta que la velocidad lineal y angular tendrán una relación directa con estas potencias. Estas características pueden ser observadas generalmente en la liga de robots pequeños (F180) de RoboCup [ROB] como es el caso del equipo CMUnited-98 [VEL3], y en la liga MiroSot de FIRA [FIR], como por ejemplo UBASot [SAN]. En el caso de robots con más de dos ruedas el comportamiento depende aún más de la arquitectura del robot. Un ejemplo puede verse en RoboCup-NAIST [NAK2] donde el robot con cuatro ruedas y dos motores independientes puede desplazarse hacia ambos lados e inclusive puede rotar en el mismo lugar.

Una alternativa para un desplazamiento más libre es el omnidireccional, donde el robot se desplaza a través de ruedas que pueden moverse en cualquier

dirección. Este tipo de desplazamiento provee menos restricciones en cuanto a la dirección. El equipo Cornell en RoboCup 2000 [DAN] implementó un mecanismo de desplazamiento que consiste en tres pares de ruedas. Cada uno de los pares de ruedas tiene un grado activo de libertad en el sentido de rotación del motor y uno pasivo cuando es perpendicular a él. Dado que las direcciones de desplazamiento s*on line*almente independientes de a pares, la traslación y rotación del robot pueden ser controladas con una eficiente elección de las velocidades de cada motor.

Finalmente, las variantes más complejas son las que implementan robots bípedos y cuadrúpedos. En el caso de robots bípedos, cada pierna del robot posee una gran cantidad de actuadores que le proveen grados de libertad al movimiento de la misma. La cantidad de actuadores y estructura de las piernas varía en cada implementación. Para desplazarse, el robot deberá ir moviendo una pierna a la vez teniendo en cuenta no perder el equilibrio al hacerlo. El robot humanoide PINO [YAM] posee 6 grados de libertad en cada pierna que le permiten desplazarse, ayudándose con otros 14 grados de libertad que posee en el resto del cuerpo con los que mantiene un balance adecuado. Cada grado de libertad es implementado a través de un servomotor. La secuencia de movimientos de un ciclo de caminar consiste en 6 fases (Fig.7.1). La Fig.7.2 muestra el primer paso del robot PINO.

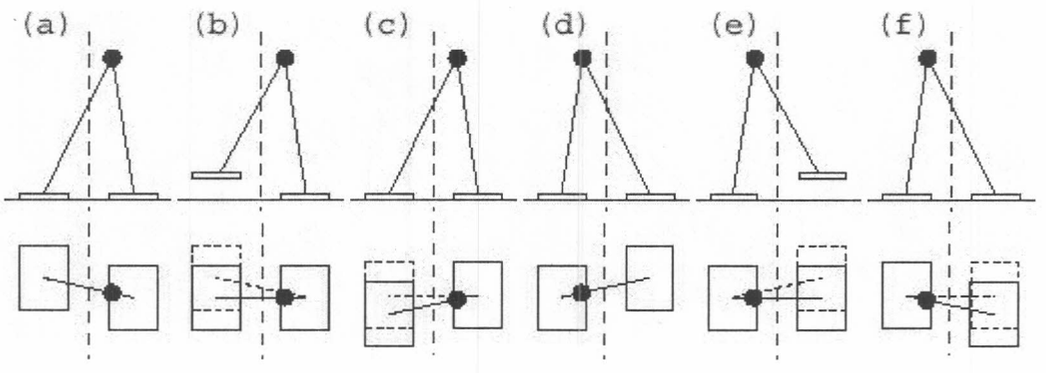


Figura 7.1: Las 6 fases del ciclo para cada paso de PINO
El círculo negro indica la ubicación del centro de gravedad.

Como ejemplo de robots cuadrúpedos se puede destacar la liga de robots comerciales SONY de RoboCup [ROB]. Los detalles de este tipo de desplazamiento pueden verse en la implementación realizada por el equipo UNSW [HEN]. La implementación tuvo en cuenta tres objetivos al momento de diseñar el módulo de acción:

- Mover al robot con 3 grados de libertad: adelante y atrás, de costado hacia la derecha e izquierda, y el giro sobre si mismo en el sentido de las agujas del reloj y al revés.
- Mover el robot a una velocidad constante para no esforzar los motores al acelerar y desacelerar.
- Mantener la cámara, situada en la cabeza del robot, lo más estable posible.

La solución implementada fue mover las patas del robot en ciclos rectangulares (Fig.7.3). En el trote utilizado en la competencia las patas diagonalmente opuestas tocaban el piso simultáneamente. Si las patas que tocan el piso se mueven a velocidad constante, entonces el movimiento del robot debería ser constante. Esto requiere que el tiempo en que la pata se mantiene en el piso (borde inferior del rectángulo) sea el mismo que el empleado en recorrer los restantes bordes del rectángulo.

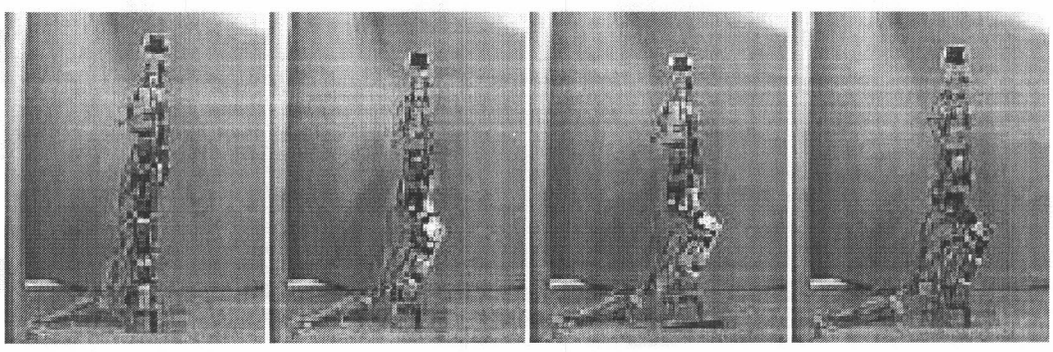


Figura 7.2: El primer paso de PINO.

El segundo objetivo se cumple mientras las dimensiones del rectángulo y la frecuencia de pasos se mantiene constante. La cámara se mantiene estable gracias a que la parte inferior del rectángulo es recta por la superficie en la que se desplaza. El movimiento producido por un ocasional desbalanceo del robot a causa del trote es corregido posando sobre el suelo las piernas levantadas.

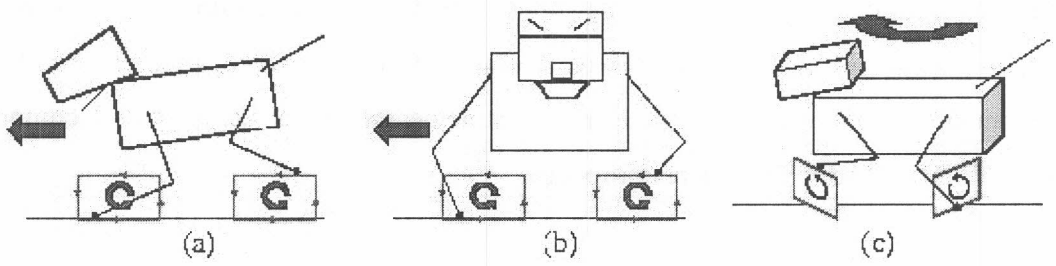


Figura 7.3: Ciclos rectangulares de movimiento de los robots de UNSW.
(a) Movimiento hacia delante. (b) Movimiento hacia el costado. (c) Giro

Para generar los tres grados de libertad se trabaja sobre el ciclo rectangular de desplazamiento. Este rectángulo forma un plano perpendicular al suelo. Cambiando el ángulo de este plano relativo a los costados del robot se determinan los movimientos hacia delante, atrás y hacia los costados. Por ejemplo, si el plano es paralelo a los lados del robot, éste se desplaza hacia delante o hacia atrás (Fig.7.3.a). Si se establece el rectángulo perpendicular a los lados, se obtiene un desplazamiento hacia los costados (Fig.7.3.b). En la Fig.7.3.c se puede observar como el robot gira determinando los planos del rectángulo tangencialmente a cada hombro.

Los tres movimientos pueden componerse para que el robot se pueda mover hacia delante, el costado y girar simultáneamente.

7.3 Girar

En este caso, la acción deseada es que el robot gire sobre su propio eje para cambiar su orientación actual. Nuevamente, esta acción depende del

dispositivo de movimiento implementado como se describe en el punto anterior.

Para esta acción el simulador de RoboCup [ROB] ofrece la primitiva *turn* que implementa un giro de la cantidad de grados pasados por parámetro. Por otro lado, el robot descripto en [JAM] posee cuatro ruedas que rotadas a 45 grados con respecto a la orientación del robot permiten girar sobre su eje. Finalmente, el mayor trabajo es requerido para los robots bípedos o cuadrúpedos que requieren de una secuencia de movimientos de sus actuadores para completar el giro. En particular, el punto anterior muestra la implementación realizada por el equipo UNSW [HEN] sobre robots cuadrúpedos

7.4 Impulsar

La acción de impulsar es la base para patear la pelota y se refiere a las primitivas o dispositivos provistos para este comportamiento. Esta acción es implementada de diversas formas de acuerdo a la categoría, la arquitectura del robot y las restricciones sobre éstos. Al respecto, las ligas de fútbol de robots fijan reglas respecto a los dispositivos impulsores y de contención de la pelota. En FIRA, por ejemplo, los apéndices desarrollados en el robot no pueden ocultar más del 30% de la pelota desde la vista superior o lateral.

Nuevamente la tarea se encuentra simplificada en el simulador de RoboCup [ROB] que busca centrarse en la estrategia y las tácticas más que en las acciones básicas. La primitiva *kick* permite impulsar la pelota, si se encuentra dentro de un radio de acción, con la dirección y fuerza deseadas.

Algunos robots, en lugar de diseñar un dispositivo de impulso, desarrollan el frente del robot de forma que pueden lograr diversos efectos sobre la pelota. Para ello se diseñan brazos fijos en el frente del robot, generalmente con forma curva, o brazos móviles.

En las categorías que lo permiten, una buena opción es proveer al robot de mecanismos impulsores especiales. Uno de los más comunes es un impulsor frontal que se activa neumáticamente, mediante un tanque de aire [DEM] o por medio de un resorte [BRU2] liberado por un solenoide, válvula eléctrica o dispositivo similar a partir de la activación de sensores de proximidad. Otro mecanismo implementado por el equipo FU-Fighters [BEH] consta de una placa giratoria que acumula la energía cinemática producida por un motor y la libera al contacto con la pelota.

Un caso particular la acción se da en la categoría Sony Legged, donde los jugadores pueden optar por impulsar la pelota por medio de alguna de sus patas delanteras o con la cabeza. En [HEN] se implementa la acción de impulsar consistente en flexionar ambas patas delanteras en paralelo, teniendo la pelota en contacto con el pecho y la cabeza junto por encima de ella. Con esto, la pelota sale impulsada hacia delante.

7.5 Atrapar

En algunas categorías se le permite al arquero retener la pelota mediante algún dispositivo durante cierto tiempo. En el caso de la categoría de simulación de RoboCup existe la primitiva *catch* que realiza esta acción. El agente toma control de la pelota y ningún otro jugador puede quitársela.

En los robots reales, esta acción debe ser implementada mediante algún dispositivo que permita sujetar la pelota. En general, el diseño está limitado por las restricciones que impone la categoría sobre la oclusión de la pelota.

7.6 Rotar Percepción

Esta acción consiste en cambiar el ángulo de orientación sin rotar al robot, es decir, sin cambiar su dirección. Esto es posible cuando el sistema de percepción posee algún grado de libertad.

El simulador de RoboCup [ROB] provee la primitiva *turn neck* que permite girar el ángulo de percepción una cierta cantidad de grados. En este caso, la diferencia máxima entre el ángulo de percepción y la orientación del agente está dada por un parámetro del servidor.

En los robots reales, algunos equipos implementan cámaras móviles que pueden variar la percepción. Esto ha sido implementado, por ejemplo, en el jugador TotTino del equipo Art-00 [ADO] y en el robot humanoide PINO [YAM] cuyo cuello tiene dos grados de libertad.

Capítulo 8
Comportamientos Elementales

8.1 Introducción

Se definen comportamientos elementales en fútbol de robots a aquellos comportamientos que representan el primer contacto con el entorno y que sirven de base para el desarrollo de comportamientos más complejos.

Los comportamientos elementales tienen una cierta dependencia con la arquitectura del robot, en el caso de los robots reales, o de las primitivas ofrecidas por el servidor en el caso de simulación.

A continuación se enumeran los comportamientos elementales utilizados en fútbol de robots:

- ***Navegar:*** moverse de un lugar a otro teniendo en cuenta el entorno.
- ***Patear:*** impulsar la pelota hacia un determinado lugar, generalmente el arco contrario o hacia un compañero.
- ***Rastrear un objeto:*** seguir el movimiento de un objeto sin moverse del lugar de manera de no perderlo de vista.

La implementación de cada comportamiento elemental puede utilizar distintas técnicas. Algunos son codificados "a mano", otros utilizan algoritmos de optimización, mientras que otros utilizan técnicas de aprendizaje. El equipo Karlsruhe Brainstormers [RIE], por ejemplo, utiliza aprendizaje por refuerzo en la implementación de los comportamientos elementales. La motivación detrás de este enfoque es lograr un aprendizaje en un dominio altamente complejo. Define *Movimiento* como una secuencia de comandos que transforma la situación actual

s(0) en una nueva situación *s(t)* algunos intervalos de tiempo más adelante. La nueva situación es una de un conjunto de posibles estados terminales S^t, que puede ser positivo (deseado) S^+ o negativo (no deseado) S^-. El *Movimiento* termina cuando se alcanza un estado terminal $s(t) \in S^t$, o cuando el tiempo excede un cierto límite $t > t_{max.}$ Para el aprendizaje, en cada paso el agente selecciona una de las posibles acciones dependiendo de la situación. El aprendizaje significa mejorar incrementalmente la política de decisión de forma que el objetivo se cumpla de una forma cada vez más eficiente. En este caso, una función es optimizada incrementalmente mediante intentos utilizando un red neuronal *feedforward.*

8.2 Navegar

Con la combinación de las acciones descriptas en el capítulo 7 el robot se puede desplazar de un lugar a otro con un método sencillo consistente en intercalar acciones de giro y avance. En cada paso se elige entre avanzar y girar dependiendo de la diferencia entre el ángulo hacia donde está el objetivo y la orientación actual del robot. Si esta diferencia es menor a una cota definida se opta por avanzar; si en cambio la diferencia es mayor el robot gira hasta que pueda seguir avanzando.

La principal diferencia entre el algoritmo descripto en el párrafo anterior y la navegación es que esta última tiene en cuenta el entorno de forma tal de desplazarse de una posición a otra en forma segura, es decir, sin colisionar con ningún obstáculo. Para ello se debe utilizar la información que provee la percepción sobre los objetos presentes en el campo de juego.

Al momento de diseñar un algoritmo de navegación, hay muchos factores y consideraciones a tener en cuenta. Uno es el punto de llegada. Este puede ser un punto determinado o tratarse de un área con cierto radio en el cual debe mantenerse el robot. Este punto de llegada u objetivo puede ser estacionario o

móvil debiendo, en este último caso, tener en cuenta su trayectoria y posición a lo largo del tiempo.

Otro factor es si se requiere llegar al objetivo con una determinada orientación que facilite la siguiente tarea. Si lo que se desea es ubicarse en un punto de la cancha para realizar una marcación o quedarse a la espera de una próxima acción, no importará con que ángulo se llega a ese lugar. Por el contrario, si el robot se dirige a patear la pelota al arco, será fundamental llegar con una orientación adecuada y precisa, a menos que el robot posea algún dispositivo de pateo específico. Este es el caso del simulador de RoboCup [ROB] en el cual el jugador puede patear hacia un amplio rango de direcciones sin cambiar su orientación. En cambio, en la liga MiroSot de FIRA [FIR] el pateo se produce por colisión del robot con la pelota. En este caso será de suma importancia desarrollar un algoritmo que le permita al robot calcular la trayectoria adecuada.

El comportamiento debe tener en cuenta, también, a los obstáculos. Es importante que la trayectoria se desarrolle sin colisionar a los mismos de forma que se considere segura. El plan trazado debe considerar zonas de seguridad alrededor de los obstáculos de forma de contemplar errores en la percepción y/o en el desplazamiento del robot. El algoritmo puede complicarse aún más si los obstáculos a considerar son móviles como es el caso del fútbol donde, dependiendo de la liga y/o categoría el ambiente puede ser altamente dinámico. No obstante, algunas implementaciones como CS Freiburg [GUT2] consideran a los obstáculos fijos para simplificar el problema aunque deban replanificar constantemente para adaptarse a los cambios de entorno.

Un problema, similar al anterior, se relaciona con los límites del espacio donde se desenvuelve el robot. En este caso particular son los límites del campo de juego y las zonas de exclusión de acuerdo a las reglas de juego de cada categoría.

Finalmente, hay tres características que deben tener los algoritmos de navegación: ser precisos, óptimos y rápidamente calculables. La primer característica tiene que ver con lograr el objetivo deseado con la mayor eficacia. La segunda, tiene que ver con obtener el camino más corto o rápido de transitar de forma de llegar al punto destino en el menor tiempo. Esto es particularmente necesario cuando el robot se dirige hacia la pelota porque algún otro robot podría llegar allí antes de él. La tercer característica tiene que ver con calcular y poder aplicar el plan de acción antes de que éste se vuelva obsoleto porque los objetos en el campo de juego han cambiado su posición.

8.2.1 Algoritmos de Planificación de Trayectorias

Existen dos grandes grupos de algoritmos de navegación de acuerdo a su mecánica. Un primer tipo de algoritmos son los llamados de planificación en los cuales se elabora un plan completo para desplazarse del punto origen al destino. En estos casos el robot debe precalcular el movimiento a desarrollar en cada tramo de la trayectoria completa. Muchas veces se precalcula el plan completo aunque luego se utilice sólo la primer parte del mismo para volver a reelaborar el plan con el nuevo estado del partido. Este es el caso del algoritmo de planificación utilizado por UBASot [SAN] para algunos de los movimientos del robot. Este método utiliza una función de costo que debe ser minimizada por una técnica de optimización no lineal. La clave para definir esta función es considerar un pequeño número de puntos intermedios a lo largo de la trayectoria. En cada punto intermedio el robot asigna velocidades fijas a ambas ruedas describiendo así un arco. De esta forma es fácil saber donde está el robot en cada instante de tiempo y cuanto tiempo le llevará el recorrido. Adicionalmente la función de costo permite introducir restricciones para evitar colisiones, posiciones del robot fuera del campo y cambios de velocidades extremos. Como el entorno es muy dinámico el plan debe ser computado rápidamente para no quedar obsoleto pronto. El plan calculado consiste en una secuencia de ternas compuestas por las velocidades de cada rueda y el tiempo de cada arco. Si el robot toma las velocidades del primer

arco estará en condiciones de reaccionar rápidamente siguiendo el plan trazado. De esta forma el resto del plan es descartado pues al momento de aplicarlo sería obsoleto.

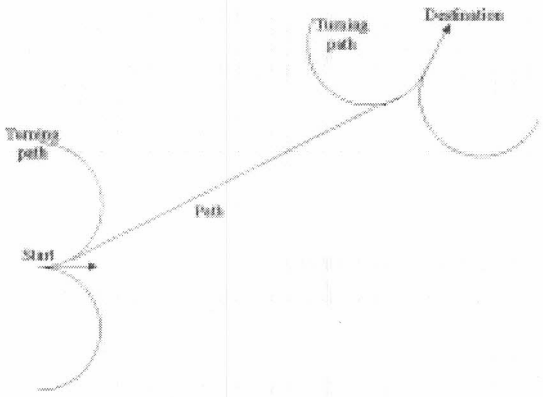


Figura 8.1: Camino simple según el planificador del equipo McGill RedDogs.

Otro ejemplo de planificación de trayectoria es la implementada por el equipo de robots cuadrúpedos McGill RedDogs [UNG]. El módulo de planificación utiliza curvas de tres tipos para construir las trayectorias: líneas rectas, giro a la izquierda de 200 mm. de radio y giros a la derecha de 200 mm. de radio. La forma de construir la trayectoria comienza colocando dos círculos en el punto de partida y dos más en el punto de llegada. Estos círculos son tangenciales a las direcciones de comienzo y fin de la trayectoria deseada. El siguiente paso consiste en dibujar las 16 líneas tangentes a los cuatro círculos, de las cuales sólo 4 resultan en un camino continuo. La trayectoria elegida es la más corta de las cuatro.

En el caso en que un obstáculo intersecta el camino simple elegido, el planificador recalcula la trayectoria teniendo en cuenta el obstáculo. Para esto utiliza caminos intermedios hacia un círculo de 200 mm. de radio centrado en el obstáculo, obteniendo dos trayectorias. En el caso de la Fig.8.2, el camino superior es elegido por ser el más corto.

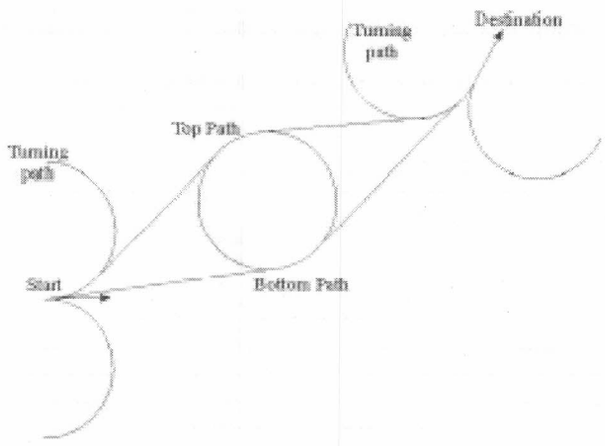


Figura 8.2: Camino complejo.
El camino superior es elegido por ser el más corto.

En el algoritmo de planificación propuesto por el equipo CS Freiburg [GUT2], el enfoque propone aproximar la solución considerando obstáculos fijos pues el problema de planificación con obstáculos móviles es computacionalmente muy complejo y, al ser el ambiente tan dinámico, la experiencia muestra que los oponentes pueden ser aproximados con objetos estáticos. Para elaborar el plan se utiliza el método gráfico de visibilidad extendida. Aquí, los objetos en el modelo del mundo son agrandados, y los limites de la cancha achicados.

El planeamiento está hecho por un algoritmo A^* que encuentra el camino libre de colisiones más corto mediante líneas rectas y arcos desde la posición actual hasta la deseada. Para incrementar la velocidad de planificación, se utiliza un enfoque iterativo. Comenzando con los nodos inicial y final, se agregan objetos sólo si interfiere con el camino encontrado (Fig.8.3). Para evitar oscilaciones, se penaliza a los caminos que requieren mayor cambio en la orientación del robot.

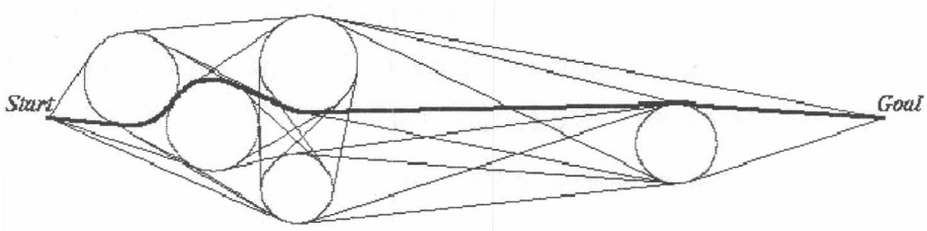


Figura 8.3: Planificación de CS Freiburg.

Algunas implementaciones no reelaboran desde cero el plan, sino que reutilizan el plan anterior y lo adaptan para la nueva configuración del espacio de estados. Baltes y Hildreth [BAL] proponen adaptar el plan realizado en el paso anterior en lugar de elaborar un nuevo plan desde cero. Las ventajas de replanificar son:

- La planificación es un proceso costoso por lo cual debería ser reusado si es posible.
- El resultado de un planificador es un plan parcial o completo.
- Asumiendo que los cambios de estado son mínimos entre dos episodios consecutivos de planificación, el nuevo plan debería ser estructuralmente similar al utilizado para la situación anterior.

8.2.2 Algoritmos Heurísticos

El segundo grupo de algoritmos es el de los llamados métodos heurísticos. En este tipo de algoritmos, a veces denominados como reactivos, en base a heurísticas particulares se decide cada uno de los pasos a realizar a cada momento. Si bien este tipo de algoritmo puede presentar situaciones donde las decisiones tomadas lleven al robot a un camino sin salida por no ver más allá de lo inmediato, tiene como ventaja el hecho de ser calculado mucho más rápido y de adaptarse a los cambios constantes del espacio de estados. En cada paso se presenta una nueva situación en base a la cual se "reacciona" en una forma independiente al paso anterior. Un ejemplo de este tipo de algoritmos es el implementado por el equipo Guaraná [REA]. Los robots tratan de evitar colisiones planeando una trayectoria alrededor de los obstáculos. El robot comienza intentando desplazarse desde su posición s hacia el objetivo g en una línea recta sg entre los dos puntos. Si a lo largo de esta trayectoria encuentra un obstáculo o como se muestra en la Fig.8.4.a, intenta buscar un punto intermedio g' como

puede verse en la Fig.8.4.b teniendo en cuenta el punto P_i donde las trayectorias de r y o se interceptan.

El algoritmo inicialmente calcula el punto de intercepción de los robots r y o, y define el tiempo t_c para la colisión en base a las velocidades de los objetos. Si el tiempo t_c es menor a t_g, que representa el tiempo en que el robot r tarda en llegar al objetivo g, y el punto de intercepción P_i de las trayectorias de r y o está en el camino de s a g, entonces se define un punto intermedio g'. Este punto se ubica a una distancia d perpendicular al camino sg. El algoritmo tiene en cuenta también las paredes de la cancha al momento de determinar los puntos intermedios.

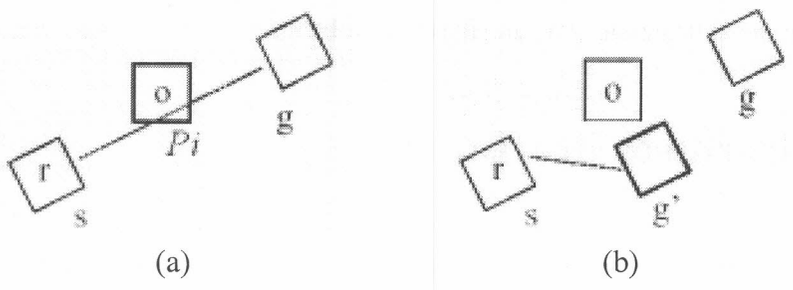


Figura 8.4: Algoritmo de navegación del equipo Guaraná.
(a) El robot precalcula su recorrido como un línea recta entre él y el objetivo. (b) Ante la presencia de un obstáculo busca un nuevo objetivo intermedio.

El algoritmo de navegación puede también tener en cuenta la predicción realizada sobre los objetos considerados para adecuar la trayectoria a sus posiciones futuras. De esta forma se puede anticipar a lo que sucederá y tomar, en base a ello, mejores decisiones.

8.2.3 Ejemplos de Implementaciones

Un ejemplo de implementación que tiene en cuenta varios de los aspectos descriptos en este punto es el desarrollo del equipo CMUnited-98 [VEL3]. Aquí

se implementa un mecanismo de control reactivo que tiene en cuenta tanto objetos estáticos como móviles para poder interceptar la pelota en movimiento, utiliza predicción para la pelota mediante el filtro de Kalman-Bucy y considera a los obstáculos inmediatos en cada replanificación cambiando su curso por uno que pase por la tangente a la zona de exclusión del obstáculo. El mecanismo de control de movimiento consiste en una serie de ecuaciones para determinar las velocidades de las ruedas derecha e izquierda v_i y v_d para llegar a una posición destino (x^*,y^*):

$$\begin{aligned} &\Delta = \theta - \Phi \\ &(t,r) = (\cos^2 \Delta \,.\, sgn(\cos \Delta)\,,\, \sin^2 \Delta \,.\, sgn(\sin \Delta)) \\ &v_i = v\,(t - r) \\ &v_d = v\,(t + r) \end{aligned} \qquad (8.1)$$

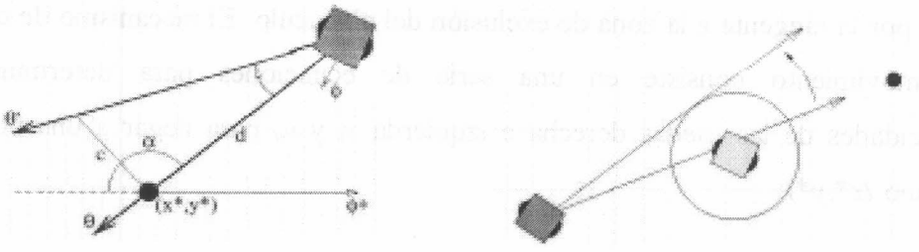
donde θ es la dirección de la posición destino (x^*,y^*), Φ es la orientación del robot y v es la velocidad deseada (Fig.8.5.a). Esta ecuación es extendida para la configuración de la posición destino de la forma (x^*,y^*, Φ^*), donde el objetivo es alcanzar la posición (x^*,y^*) teniendo una orientación Φ^*. Esto es logrado con el siguiente ajuste:

$$\theta' = \theta + min(\alpha, tan\text{-}1(c/d)) \qquad (8.2)$$

donde θ' es la nueva dirección , α es la diferencia de ángulo entre el ángulo entre la orientación del robot y la posición destino, y Φ^*, d es la distancia al punto destino, y c es el parámetro de claridad (Fig.8.5.a). Esto hace mantener al robot a distancia c del punto destino mientras lo rodea hasta alcanzar la orientación final Φ^*. Esta nueva orientación θ' es reemplazada en la ecuación 8.1 para calcular las velocidades.

Para evitar obstáculos se ajusta la dirección destino del robot basada en cualquier obstáculo cercano en el camino. Este ajuste puede verse en la Fig.8.5.b.

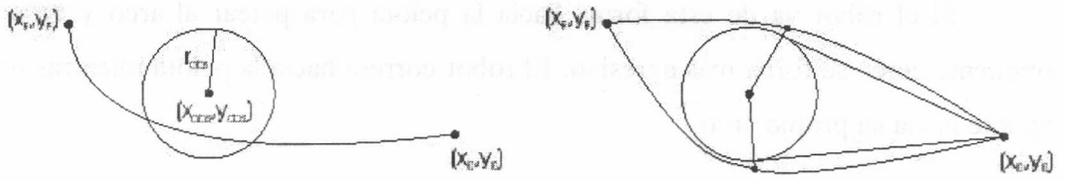
Si la dirección destino pasa cerca de un obstáculo, la dirección es ajustada para pasar tangencialmente a un círculo de seguridad predefinido. Dado que el algoritmo se repite continuamente en cada ciclo, se van replanificando caminos libres de obstáculos, adaptándose al entorno dinámico.



(a) (b)

Figura 8.5: Navegación heurística de CMUnited'98.
(a) El ajuste de θ a θ' para alcanzar la configuración final (x,y*,Φ*). (b) El ajuste para evitar obstáculos cercanos.*

El equipo Cornell [DAN] desarrolló un algoritmo para desplazamiento omnidireccional que evita obstáculos circulares considerando así al área de influencia de los robots (círculo que los circunscribe). El generador de trayectorias genera dos tipos de planes: de tiempo mínimo, utilizado cuando se debe llegar a una posición con la mayor velocidad, y de esfuerzo de control mínimo, utilizado cuando el robot debe llegar a una determinada posición, con determinada orientación en un tiempo especificado. Si bien el algoritmo contempla obstáculos fijos funciona bien dado que es llamado en cada paso. Si la trayectoria desde el punto inicial (x_0,y_0) hasta el punto final (x_f,y_f) pasa por un obstáculo circular (Fig.8.6.a), el camino debe ser modificado. En este caso se generan dos nuevas trayectorias (Fig.8.6.b) pasando por puntos en las líneas perpendiculares a las tangentes del círculo que pasan por el punto inicial. Luego, el mejor camino es elegido. Una extensión de este algoritmo se aplica a múltiples obstáculos.



(a) (b)

Figura 8.6: Navegación del equipo Cornell.
(a) Trayectoria intersectando un obstáculo. (b) Evitando un obstáculo.

El equipo UNSW [HEN] de robots cuadrúpedos desarrolló un método de navegación para ir desde cualquier punto del campo hasta una posición detrás de la pelota. Esta técnica circular es utilizada tanto por el arquero como los robots de campo e involucra transiciones no agresivas en los movimientos del robot, mantiene siempre en vista la pelota y el cuerpo alineado en esta dirección.

El rodeo de la pelota se especifica con dos puntos (Fig.8.7). El punto destino (target point) representa el punto a donde se pretende llegar, y el punto de rodeo (circling point) define el círculo de rodeo de la pelota. Para llevar a cabo la acción el robot va hacia estos puntos manteniendo el cuerpo orientado hacia la pelota.

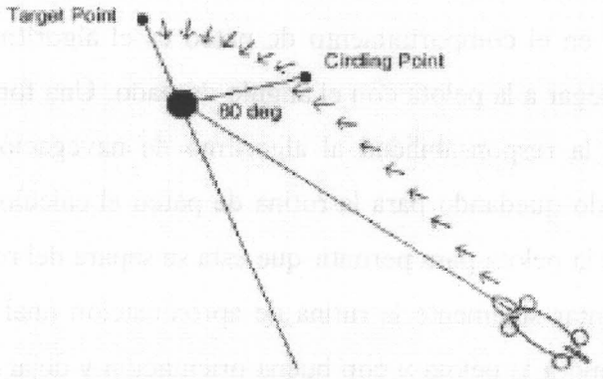


Figura 8.7: Acción de rodeo de UNSW.
El robot rodea la pelota hasta llegar al punto destino siempre manteniendo el cuerpo orientado hacia ella.

Si el robot va de esta forma hacia la pelota para patear al arco y ve un oponente cerca se torna más agresivo. El robot correrá hacia la pelota mientras no apunte hacia su propio arco.

8.3 Patear

Junto con la navegación, el pateo es una de los principales comportamientos a desarrollar en un robot pues sin ello no se podrá enviar la pelota dentro del arco rival para cumplir con uno de los objetivos del juego: hacer goles. Este comportamiento es implementado de diversas formas de acuerdo a la categoría, la arquitectura del robot y las restricciones sobre éstos.

La opción de pateo más utilizada hasta el momento es colisionar a la pelota para generar un desplazamiento en ella por la diferencia de masa. Esto se realiza con alguno de los frentes del robot. En las categorías que lo permiten, una opción utilizada es proveer al robot de mecanismos especiales de impulsión y desarrollar el comportamiento de pateo mediante la utilización de la acción de impulsar.

Un aspecto en el comportamiento de pateo es el algoritmo utilizado para desplazarse hasta llegar a la pelota con el ángulo deseado. Una forma de encarar el problema es darle la responsabilidad al algoritmo de navegación de llegar a la pelota bien orientado quedando para la rutina de pateo el cálculo del frenado una vez que se impacta la pelota para permitir que ésta se separe del robot. Una opción similar es implementar solamente la rutina de aproximación final cuando el robot se encuentra próximo a la pelota y con buena orientación y dejar que el algoritmo de navegación sólo aproxime al robot a la pelota.

En la categoría de simulación de RoboCup [ROB] el pateo acelera la pelota en una cierta dirección. El equipo CMUnited-98 [STO4] utiliza una

secuencia de pateos para enviar a la pelota en cualquier dirección, o para acelerar aún más la pelota. Por otro lado, el equipo Erika [MAT] utiliza algoritmos genéticos para aprender las velocidades y direcciones parciales para mover la pelota en rectas tangentes al cuerpo del robot de forma de poder patear en cualquier dirección.

El equipo Karlsruhe Brainstormers [RIE], también en la categoría de simulación de RoboCup, utiliza una red neuronal entrenada con aprendizaje por refuerzo para aprender a patear la pelota en una determinada dirección y velocidad. Para esto, se construyó un conjunto de 500 instancias de la acción de pateo (discretizando la dirección en 100 pasos, y la potencia de pateo en 5), más 36 instancias del comando Girar. Entre estas 536 acciones el jugador debe optar en cada ciclo de ejecución. El estado objetivo S^+, se alcanza si la pelota abandona al jugador en la dirección y con la velocidad deseadas. Si esto ocurre, se obtuvo costo 0 y la secuencia termina. Una situación negativa, S^-, ocurre cuando el jugador pierde la pelota. Esto devenga en un costo máximo 1. Dado que es razonable tener como objetivo utilizar la menor cantidad de comandos posibles en la secuencia, cada transición intermedia causa un pequeño costo (0.002). La política de pateo obtenida alcanza un grado de éxito del 95 %. Para considerar distintas velocidades objetivo de la pelota, se entrenaron tres redes neuronales: para velocidad deseada baja, media y alta. Cada red está formada por 4 neuronas de entrada, 20 en la capa oculta y 1 de salida.

8.4 Rastrear un Objeto

Este comportamiento es utilizado generalmente en los momentos donde el agente permanece en una posición del campo de juego a la espera de poder disparar algún comportamiento activo. En esta espera es conveniente, por ejemplo, no perder de vista la pelota para poder entrar en acción cuando ésta esté al alcance. Este comportamiento es empleado generalmente por los arqueros para

tener control continuo de la posición de la pelota y, de esa forma, no ser sorprendidos.

En los sistemas de visión global, cada robot tiene constantemente una vista general del estado de juego. Aquí, rastrear un objeto es una tarea trivial pues solamente consiste en mantener un registro de la posición del objeto. Aún así puede suceder que, por variaciones en la luz, reflejos, posiciones solapadas, etc. alguno de los objetos no sea reconocido en algún instante de tiempo. En este caso el sistema debe recurrir a información previa y a modelos respecto del movimiento del objeto para predecir la posible posición actual del mismo.

En la mayoría de los sistemas de visión local, este problema toma mayor dimensión. En cada instante, el agente tiene información parcial del estado de juego pues sólo puede visualizar un cierto sector de la cancha. Aquí, rastrear un objeto requiere de la utilización de información adicional o de ciertas técnicas para mantener un registro de la posición del objeto. Un enfoque posible es compartir información con los demás miembros del equipo para intentar completar el mapa actual del campo de juego. Aún así pueden darse casos donde queden regiones sin cubrir en el campo. El rastreo es generalmente llevado a cabo por un robot girando sobre su eje de forma de tener siempre el objeto rastreado en el campo de visión.

Capítulo 9
Comportamientos Complejos

9.1 Introducción

En el presente capítulo se exponen los comportamientos complejos básicos para el desarrollo de jugadas de fútbol. Estos comportamientos son implementados a partir de los comportamientos elementales descriptos en el capítulo anterior.

Estos comportamientos actúan como primitivas para las jugadas descriptas en el capítulo siguiente.

La lista siguiente enumera los comportamientos complejos que son ampliados a continuación:

- ***Interceptar la pelota:*** desplazarse de forma de colisionar la pelota en movimiento.
- ***Bloquear la pelota:*** ubicarse en algún punto futuro de la trayectoria de la pelota.
- ***Posicionarse***: elegir e ir hacia un determinado punto de la cancha.
- ***Llevar la pelota***: desplazarse de un lugar a otro sin perder contacto con la pelota.
- ***Perseguir un objeto:*** Mantenerse cerca de un objeto móvil conservando la distancia.

9.2 Interceptar la Pelota

Este comportamiento complejo es fundamental para la implementación de gran parte de las jugadas de fútbol y requiere de todos los comportamientos elementales: navegar, patear y rastrear un objeto.

Interceptar la pelota supone el cálculo de un punto en la trayectoria de la pelota que puede ser alcanzado en un tiempo t_r igual al tiempo t_p en que la pelota llega a ese punto. Una vez calculado el punto de intercepción, el robot deberá desplazarse a dicho punto e impactar a la pelota. En la mayoría de las veces será necesario llegar a la pelota con un determinado ángulo para poder darle a la pelota una nueva trayectoria deseada. Esta acción supone tener un modelo preciso del movimiento de la pelota y del propio agente.

El equipo CMU [VEL4] considera la línea desde el robot hacia el punto destino deseado para la pelota y determina el punto de intersección de esta línea con la trayectoria de la pelota. El robot se posiciona en esta línea de manera de acelerar hacia el punto de intersección en el momento en que pase la pelota (Fig.9.1).

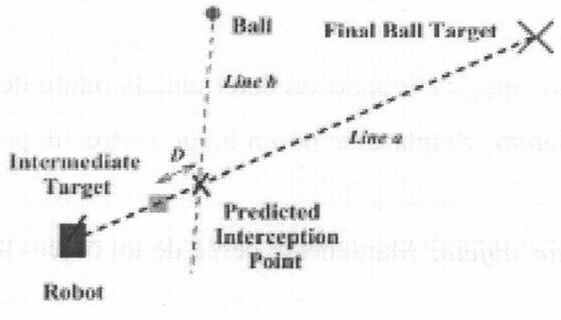


Figura 9.1: Intercepción de la pelota en CMU

AT-Humboldt [BUR] utiliza un algoritmo de simulación para calcular el punto de intercepción óptimo de la pelota. Para esto utiliza un modelado del

entorno en tiempo futuro. El agente intenta mentalmente alcanzar la pelota en un paso, luego en dos y así sucesivamente. Este procedimiento encuentra un cierto número de pasos en los cuales se puede alcanzar la pelota. Realizando la misma simulación con los demás agentes es posible saber quién está en mejores condiciones de alcanzar la pelota. La Fig.9.2 muestra gráficamente este proceso.

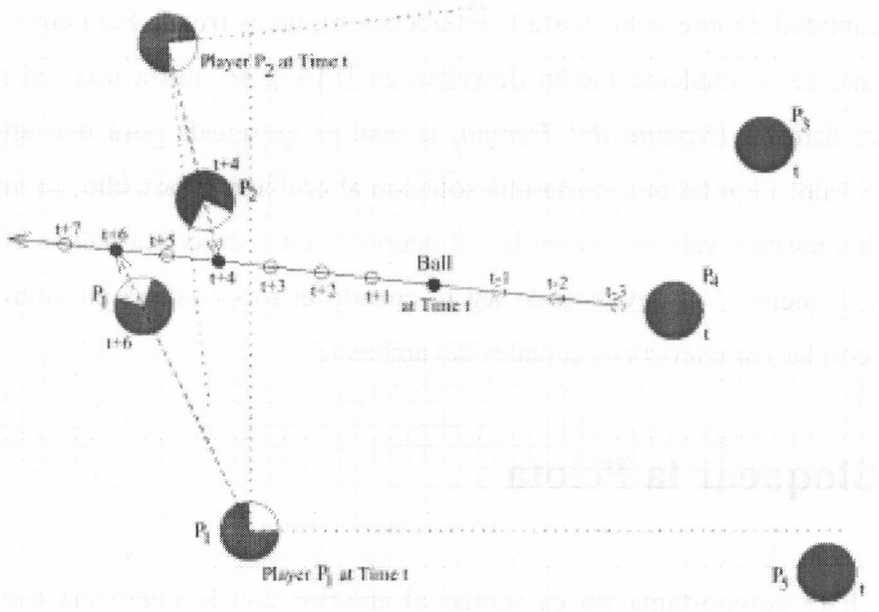


Figura 9.2: Simulación para interceptar la pelota.
La línea sólida muestra la trayectoria de la pelota con su posición en cada intervalo de tiempo $t_{\pm i}$. Las líneas punteadas representan los ángulos de visión de los jugadores P_1 y P_2 en el tiempo t. P_1 calcula 6 pasos para alcanzar la pelota mientras que P_2 necesita 4 pasos para interceptarla.

En [MAI] se presenta el problema de interceptar una pelota en movimiento y dispararla en una determinada dirección, como un problema de optimización de programación cuadrática. Para aprender el comportamiento, se entrena a una red neuronal mediante aprendizaje por refuerzo y en base a un cómputo *off line* de la trayectoria óptima mediante un optimizador de problemas de programación cuadrática. La tarea que se aprende es la intercepción y pateo en determinada dirección, con el mínimo costo de energía y con la restricción que el viaje hasta la pelota insumirá *n* intervalos de tiempo. Los parámetros de entrada del

comportamiento son: los vectores de posición y velocidad de la pelota y el jugador y la cantidad de intervalos de tiempo hasta la intercepción deseada. Se asume, que las posiciones son relativas a la posición objetivo, que se considera *(0,0)*. El resultado del módulo es un vector de aceleraciones para los *n* pasos siguientes.

Como en la mayoría de los problemas de estas características, la elección de *n* (cantidad de intervalos hasta la intercepción), no es trivial. Para enfrentar este problema, en la implementación descripta en [MAI], se utiliza una red neuronal auxiliar, llamada *Oráculo del Tiempo*, la cual es entrenada para que aprenda el mínimo valor de *n* tal que exista una solución al problema. Para ello, se ingresan a la red los mismos valores que en la red neuronal antes descripta menos el valor *n*. La red devuelve *1/n*, devolviendo así *0*, cuando el robot fallará en interceptar la pelota con las características actuales del ambiente.

9.3 Bloquear la Pelota

Este comportamiento es similar al anterior con la diferencia que aquí el robot se intenta ubicar en el punto de la trayectoria de la pelota en un instante t_r previo al instante t_p correspondiente a la llegada de la pelota, con lo cual no utiliza el comportamiento elemental de pateo. De esta manera la pelota colisiona al robot deteniendo su movimiento o generando una nueva trayectoria a causa del rebote.

9.4 Posicionarse

La implementación de este comportamiento, en general, consiste en ir a un lugar utilizando el comportamiento de navegación. Generalmente este comportamiento es utilizado por los mecanismos de decisión para ubicar al jugador que, debido a la participación en un jugada, se alejó de su zona o posición por defecto, aunque también es utilizado por las jugadas que requieren que el jugador se ubique en un lugar determinado: recibir un pase, marcar a un jugador,

liberarse de una marca o tapar a un jugador.

También se considera un comportamiento de posicionamiento el que ejecutan los defensores o delanteros en estado pasivo. Es decir, que no están participando de la jugada actual por lo cual deben ubicarse en algún lugar estratégico de la cancha para futuras intervenciones.

En [STO4] se propone un método de posicionamiento llamado SPAR (*Strategic Position by Attraction and Repulsion*), mediante el cual los agentes tratan de posicionarse a sí mismos de manera de incrementar la probabilidad de ser útiles al equipo en el futuro, favoreciendo una jugada cooperativa. El método consiste en maximizar una función que genere *Repulsión* (maximice distancia) hacia los rivales y compañeros; y *Atracción* (minimice distancia) hacia la pelota y el arco rival. También se tienen en cuenta restricciones adicionales a la hora de determinar la posición a ocupar:

- 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.

Algunas de estas restricciones dependen de la categoría (la segunda y tercera restricción sólo tienen sentido en simulación de RoboCup). Para cumplir con la cuarta restricción, se evalúa que en un cono con vértice en la pelota y en dirección a la posición calculada, no haya jugadores rivales que puedan interceptar o bloquear la pelota.

9.5 Llevar la Pelota

El comportamiento *Llevar Pelota* es la habilidad del jugador de poder

desplazarse por el campo de juego transportando consigo a la pelota. Esta jugada es útil no sólo para llevar la pelota hacia el arco rival, sino también para evitar al oponente cuando se está en posesión de la pelota. Su implementación, generalmente, requiere de los comportamientos elementales de pateo, navegación y rastreo de objetos.

La complejidad de este comportamiento está íntimamente relacionada con la morfología del robot. La tarea es mucho más costosa para robots cilíndricos (por ejemplo, robots Khepera) que para aquellos que tengan sus caras planas o, eventualmente, tengan en alguno de su lados una concavidad que mantenga la pelota. Más aún, algunos equipos desarrollan dispositivos especiales de forma que el robot no pierda la pelota. La mayoría de estos dispositivos tienen que ver con brazos de contención laterales -CS-Freiburg 2000 [WEI]-, barras de pateo curvas o mediante un rodillo que giran en sentido contrario a la rotación de la pelota. De este último caso se puede citar al equipo Cornell [DAN] que ubicó un rodillo con una cobertura de látex por encima del mecanismo de pateo. Al contacto con la pelota, la rotación del rodillo imparte una rotación hacia atrás a la pelota; la ubicación de la barra es estratégica para generar una fuerza en la pelota en sentido hacia el robot sin contradecir la regla de cobertura de la pelota.

Otro enfoque posible es trabajar sobre la acción mediante un algoritmo que permita al robot movilizarse sin que la pelota se separe del frente del mismo. De esta forma, CS-Freiburg [WEI] apoya el mecanismo implementado con un algoritmo que busca, en cada ciclo de ejecución, direcciones dentro de su ángulo de orientación que estén próximas al arco contrario. De estas direcciones elige la mejor en base a distancia a objetos, menor cambio de orientación y proximidad al arco contrario, y ajusta las velocidades de las ruedas.

En cuanto al comportamiento en sí, en [ARS] se expone el aprendizaje del mismo mediante la utilización del algoritmo Q-TD (Temporal Diference Q-learning). El agente es entrenado para llevar la pelota enfrentándose a un rival

adiestrado para interceptar la pelota. En este ambiente, un estado está definido por tres valores: la dirección hacia donde está el arco rival, la distancia al oponente y la dirección hacia la cual éste se encuentra. El agente que está siendo entrenado, debe elegir entre cinco direcciones hacia donde dirigirse llevando la pelota. Los premios asignados al agente se realizan tanto en el estado final, como en los intermedios en base a la distancia hacia el arco ganada y el tiempo consumido. Una vez aprendido el comportamiento, durante un juego real, el agente con posesión de la pelota lo ejecuta teniendo en cuenta el rival más cercano visible. Si ningún rival se visualiza, el último recordado se tiene en cuenta. Una observación interesante sobre este comportamiento aprendido, es que, a la vez que el agente aprendió a llevar la pelota, también emerge un comportamiento de evitación de obstáculos.

9.6 Perseguir un Objeto

El comportamiento de perseguir un objeto resulta generalmente útil en los casos de marcación de los robots contrarios realizada por los defensores. La misión del robot, en este caso, es mantener una posición cercana a la de otro objeto evitando colisionar con el mismo. Básicamente consiste en rastrear al objeto y calcular una posición próxima al objeto donde se llega mediante un plan de navegación que permita ubicarse rápidamente en esa posición.

Capítulo 10
Jugadas

10.1 Introducción

Se definen jugadas en fútbol de robots como aquellos comportamientos de alto nivel que utilizan uno o más comportamientos complejos para realizarse. Las jugadas son el conjunto de tareas que dispara el mecanismo de decisión del robot. Sin ellas no es posible el juego del fútbol pues los robots no podrían llevar adelante la táctica y estrategia elegidas.

La lista siguiente enumera algunas de las jugadas más utilizadas en fútbol de robots, las cuales son ampliadas a continuación:

- ***Atajar***: acción del arquero tendiente a evitar que la pelota ingrese al arco.
- ***Despejar la pelota***: impulsar la pelota fuera del área defensiva reduciendo el peligro para el arco propio.
- ***Patear al arco***: intentar convertir un gol.
- ***Realizar un pase***: enviar la pelota a una posición donde, potencialmente, quede bajo el control de un robot compañero.
- ***Recibir un pase***: posicionarse para recibir un pase de un compañero.
- ***Marcar a un jugador***: 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.
- ***Liberarse de una marca***: ubicarse de forma que los defensores del equipo contrario no interfieran en posibles recepciones de la pelota o de tiros al arco.
- ***Tapar un jugador***: molestar a un robot para que no pueda tomar control de la pelota.

10.2 Atajar

En algunas categorías se le permite al arquero retener la pelota mediante algún dispositivo durante cierto tiempo. En el caso de la categoría de simulación de RoboCup existe la primitiva *catch* que realiza esta acción. El agente toma control de la pelota y ningún otro jugador puede quitársela.

Cuando no se cuenta con la facilidad antes expuesta, la jugada de atajar se implementa en base a la intercepción y el bloqueo de la pelota para evitar que ésta ingrese al arco defendido.

10.3 Despejar la Pelota

Este comportamiento es netamente defensivo ya que tiene por objetivo enviar la pelota a una zona de la cancha con menos peligro para el arco propio que la que tiene actualmente. Esta jugada es normalmente implementada en defensores y arquero mediante el uso de los comportamientos complejos de intercepción. Los mecanismos de decisión disparan este comportamiento, normalmente cuando el jugador tiene momentáneamente el control de la pelota pero no está en una situación que le permita realizar un pase o llevar la pelota, por lo que, sin bien con el despejar pelota pierde el control de la misma, se gana en seguridad.

La principal consideración a tener en cuenta cuando se decide despejar la pelota es hacia donde enviarla. Para ello se deberá elegir un ángulo que envíe la pelota hacia el campo rival, en lo posible, hacia un lateral y no hacia el centro de la cancha y además, donde menos rivales haya.

En CMUnited-98 [STO4] la elección de este ángulo se logra multiplicando el resultado de una función de utilidad y una función de grado de éxito esperado.

La primera función, toma valores entre 0 y 1, donde 1 es un ángulo hacia un lateral en dirección al arco rival y 0 cuando el ángulo va hacia el centro de la cancha. La función de grado de éxito mide la posibilidad de que un rival intercepte la pelota despejada. Para ello se consideran las distancias *w* y *d* que se muestran en la Fig.10.1 realizando el cociente entre ambos. Multiplicados los coeficientes de todos los rivales dentro del triángulo a evaluar, se obtiene el grado de éxito esperado para el ángulo considerado. Una vez calculados el producto entre utilidad y grado de éxito de cada ángulo, se elige el de mayor valor.

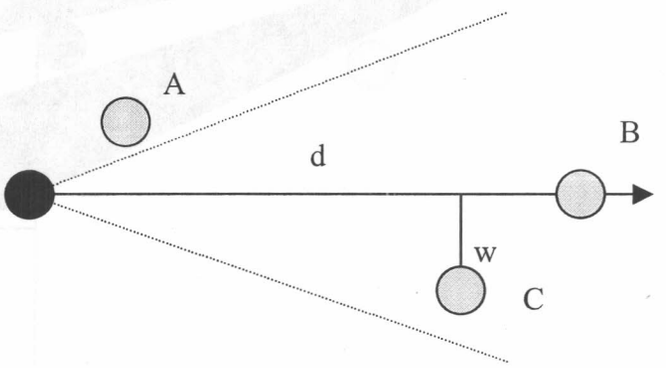


Figura 10.1: Cálculo de Grado de Éxito esperado para un ángulo de despeje.
El rival A no será tenido en cuenta. Para el rival C se calculará w/d. El rival B llevará a 0 el grado de éxito ya que se encuentra sobre la dirección evaluada.

10.4 Patear al Arco

El hecho de patear al arco, además del comportamiento que involucra el conseguir que la pelota se dirija a determinado punto, involucra la elección del punto a donde enviar la pelota de manera de maximizar las posibilidades de que dicha pelota ingrese en el arco. Más allá de las consideraciones normales del pateo con respecto a trayectos libres de obstáculos, aquí se debe prestar especial atención al comportamiento del arquero cuyos comportamientos están

exclusivamente orientados a evitar que la pelota entre al arco. También se deben tener en cuenta a los defensores que intentarán interceptar la pelota antes de que ésta llegue al arco.

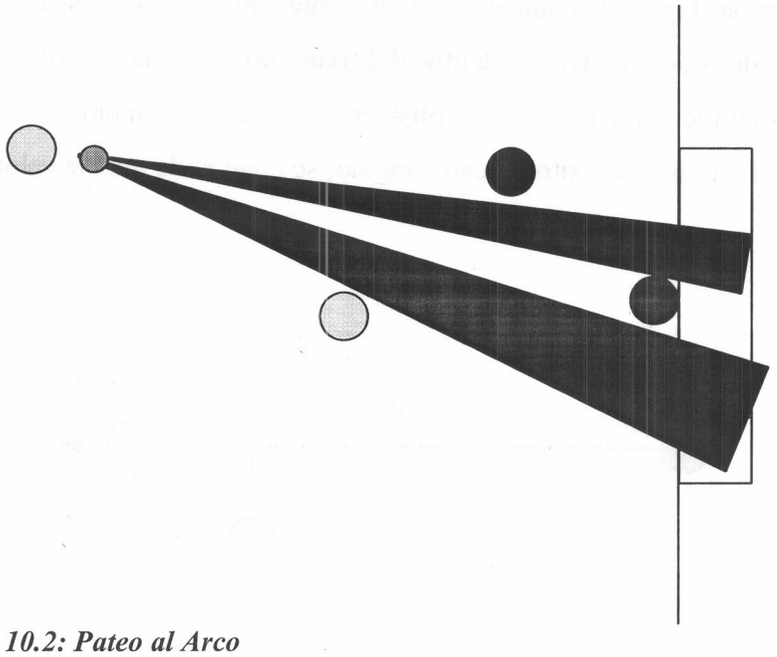


Figura 10.2: Pateo al Arco
Se elige el mayor ángulo libre entre la pelota y el arco y, si hay espacio suficiente para que pase la pelota, se establece su bisectriz como dirección del tiro.

El mecanismo más sencillo para determinar el mejor ángulo hacia donde enviar la pelota, consiste en elegir como dirección de pateo la bisectriz del mayor ángulo libre de obstáculos con vértice en la posición actual de la pelota en dirección al arco. (Fig.10.2)

Nótese que se deben considerar como obstáculos, tanto a los jugadores rivales como a los compañeros. También se debe tener en cuenta que a lo largo de la trayectoria elegida, siempre la pelota tiene espacio suficiente para pasar, siendo conveniente que este camino contemple un margen de seguridad a ambos lados de la pelota.

En cuanto al pateo en sí, puede ser implementado en base a los comportamientos de intercepción de pelota (cuando la pelota está detenida, se puede considerar un caso particular de intercepción) y de llevado de la pelota ya que, a veces, el robot llega hasta el arco con la pelota.

10.5 Realizar un Pase

El pase es una de las jugadas más complejas del fútbol ya que requiere un alto nivel de coordinación entre dos jugadores. Dado un jugador elegido como receptor, el pateador debe tener en cuenta la dinámica actual de la pelota y del jugador destinatario del pase de manera que éste pueda atrapar exitosamente a la pelota. Una vez elegida la dirección del pase, la jugada se limita al uso de los comportamientos complejos de pateo e intercepción expuestos en los capítulos anteriores.

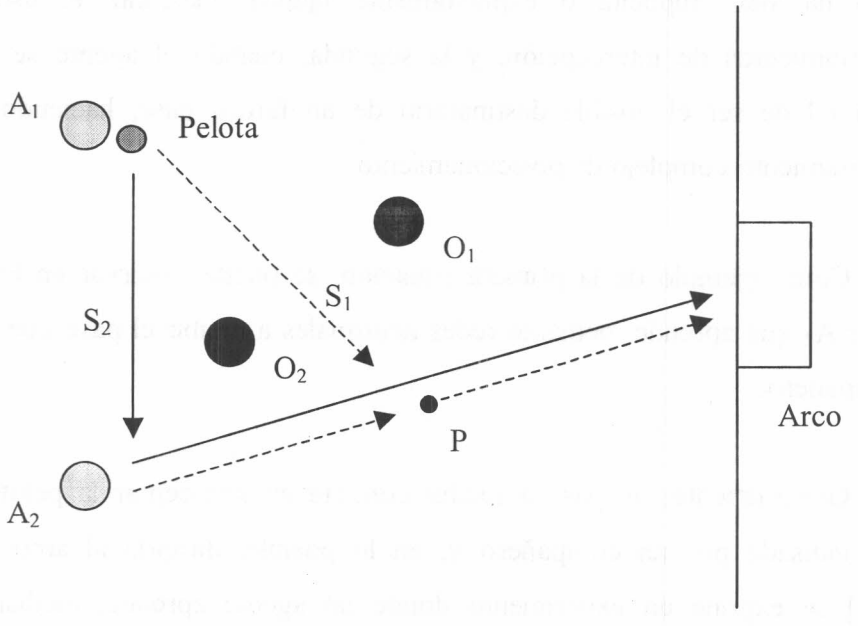


Figura 10.3: Situación de Experimento de Aprendizaje de Pase.
El agente A_1 debe aprender en que ángulo patear la pelota y A_2, cuantas veces debe avanzar para recibir la pelota.

En [OHT] se expone un experimento de aprendizaje de jugada de pase donde dos jugadores A_1 y A_2, enfrentados a dos rivales O_1 y O_2 aprenden mediante dos redes neuronales artificiales, el ángulo hacia donde enviar la pelota, y el número de veces que tiene que avanzar para recibir la pelota, respectivamente. Como se puede observar en la Fig.10.3 A_1 puede optar por hacer el pase entre O_1 y O_2 (S_1), para lo cual A_2 deberá avanzar de manera de interceptar la pelota en el punto P; o bien, pasar la pelota por detrás de O_2, para lo cual A_2 deberá detenerse y esperar el pase (S2).

10.6 Recibir un Pase

El comportamiento que lleve a cabo el jugador potencialmente destinatario de un pase tiene tanta o más influencia en el éxito de la jugada, que el propio pateador. Hay dos situaciones relacionadas a la presente jugada: la primera, tiene que ver con la recepción de un pase específicamente dirigido al agente (cuyo destino ha sido implícita o explícitamente fijado), mediante el uso de los comportamientos de intercepción, y la segunda, cuando el agente se ubica de manera tal de ser el posible destinatario de un futuro pase, haciendo uso del comportamiento complejo de posicionamiento.

Como ejemplo de la primera situación, se puede observar en Fig.10.3 al jugador A_2 que aprende mediante redes neuronales a recibir el pase que le realiza su compañero.

Generalmente, un pase a recibir consiste en interceptar la pelota que ha sido impulsada por un compañero y, en lo posible, dirigirla al arco rival. En [STO1] se expone un experimento donde un agente aprende, mediante redes neuronales artificiales, a interceptar una pelota cuya trayectoria pasa entre el jugador y el arco, y a impulsarla dentro del arco.

En [ARS] se propone una implementación para simulación de RoboCup, donde un jugador que no tiene posesión de la pelota trata de ubicarse en una posición que le permita ser potencialmente receptor de un pase (Fig.10.4). Para ello, en un juego real, el jugador trata de ver al campo de juego desde el punto de vista del compañero que tiene la pelota. En el algoritmo propuesto, el agente construye este modelo, primero, llevando todos los jugadores que ve en su campo de visión a coordenadas globales, y luego, a coordenadas relativas al campo de visión del jugador que tiene la pelota. Luego, el campo visual de 90° que tiene el jugador, es discretizado en arcos de 15° cada uno, a los cuales se les asigna un peso en base al jugador más cercano. Este concepto de modelado visual es llamado *Cono de Visión* . La asignación de pesos se realiza de la siguiente manera: +1 si el jugador más cercano es un compañero, -1 si es un rival y 0 si no hay jugadores en el cono de visión.

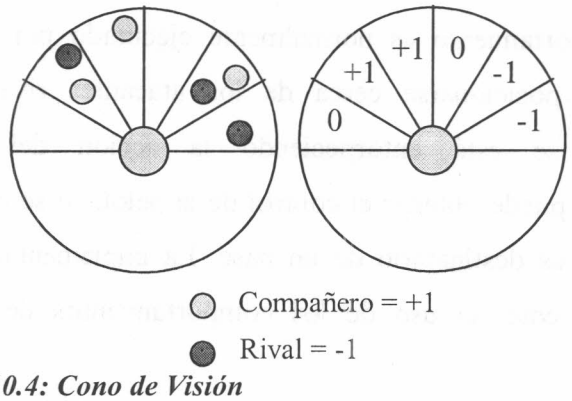


Figura 10.4: Cono de Visión
El jugador del centro es quien tiene la pelota y cuyo punto de vista los compañeros reconstruyen para poder convertirse en potenciales destinatarios de un pase.

Estos valores de peso son ingresados a un esquema de aprendizaje Q-TD para determinar la mejor acción a llevar a cabo para poder recibir la pelota. También se ingresa un valor que indica si el pateador tiene o no una visión obstruida del jugador que está realizando el análisis. Las cuatro posibles acciones son, permanecer en el lugar, avanzar hacia el compañero que tiene la pelota, girar hacia la izquierda y avanzar o girar hacia la derecha y avanzar. Para incentivar al jugador a que se desplace a un buen lugar como para recibir un pase, la máxima

recompensa se obtiene cuando no hay obstrucción de la visión entre el pateador y el posible receptor y, ambos lados del arco donde se encuentra este jugador, tienen peso 0, por lo cual no hay una inminente amenaza de rivales. El premio disminuye si en alguno de estos laterales hay un compañero y se penaliza al jugador si está junto a un oponente, el compañero con la pelota tiene obstruida la visión hacia él o hay visualización libre pero está detrás de un rival.

El método de posicionamiento SPAR [STO4] expuesto con anterioridad, también puede considerarse como una jugada de ubicación para la recepción de un pase ya que cada agente intenta estar libre y lo más cerca de la pelota y el arco rival.

10.7 Marcar a un Jugador

Este comportamiento es normalmente ejecutado por los defensores, los cuales intentarán posicionarse cerca de los atacantes rivales. Mediante este comportamiento, se está entorpeciendo la acción del jugador rival y, eventualmente, se puede obtener el control de la pelota si se consigue anticipar al rival cuando éste es destinatario de un pase. La implementación de esta jugada requiere, generalmente, el uso de los comportamientos de posicionamiento y persecución.

En CMUnited-98 [STO4] el defensor que ejecuta el comportamiento de marcar se sitúa cerca del rival sobre la bisectriz del ángulo formado por la línea que une la pelota y el rival marcado, y la que une a dicho rival y el arco propio. Así, el jugador marcado tendrá dificultades tanto para recibir un pase, como para patear hacia el arco.

10.8 Liberarse de una Marca

Este comportamiento consiste en realizar la acción inversa a la del comportamiento *Marcar Jugador* y, generalmente, lo deben implementar los atacantes utilizando el comportamiento complejo de posicionamiento.

Aquí la idea es alejarse de los jugadores rivales cuando se está en estado pasivo de manera de convertirse en un buen candidato para recibir un pase, por lo tanto, los métodos expuestos en la jugada *Recibir un Pase*, son también aplicables para el presente comportamiento. También es necesario librarse de la marca que ejerce un rival cuando se tiene posesión de la pelota y se desea llevarla o patearla. En estos casos, el rival que está marcando al robot se convierte en un obstáculo que el comportamiento de pase, llevado de pelota o pateo al arco deberá tener en cuenta.

10.9 Tapar un Jugador

Mediante esta jugada, se intenta evitar que un jugador rival tome el control de la pelota. Normalmente, esta jugada consiste en interponerse en línea que une al rival y la pelota, mediante el comportamiento de posicionamiento.

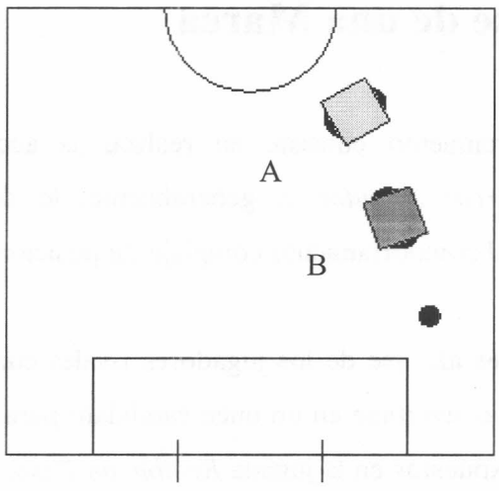


Figura 10.5: Tapar un jugador
El jugador B se interpone entre la pelota y el robot A de manera que éste no pueda llegar a la pelota.

Cabe destacar que se debe ejecutar en casos en los cuales no se puede o no conviene que el jugador propio vaya hacia la pelota. En [VEL3], por ejemplo, se ejecuta esta jugada, bajo la denominación de "molestar", cuando la pelota está más cerca del arco propio que el jugador defensor (Fig.10.5). Si el defensor fuera hacia la pelota corre el riesgo de, accidentalmente, impulsarla hacia el arco propio. Ejecutando este comportamiento se gana tiempo de manera de posibilitar que un compañero mejor posicionado pueda tomar control de la pelota o impulsarla a un lugar más seguro.

Capítulo 11
Tácticas

11.1 Introducción

En el presente capítulo se abordarán los temas relacionados al diseño de cada jugador o agente. y como lleva adelante su táctica de juego. Los problemas de este nivel tienen que ver, principalmente, con que tareas debe llevar a cabo el jugador en base al rol que le toca desempeñar dentro del equipo y las decisiones que debe tomar en cuanto a que comportamiento ejecutar en cada momento, y seleccionar las variables que estos comportamientos puedan tener. Por ejemplo, determinar con que compañero cooperar, a que rival marcar, donde ubicarse, etc. El nivel de jugador es identificado en algunos diseños como capa individual, o capa táctica.

Este nivel recibirá de niveles inferiores la información sensorial que le brinda el entendimiento del ambiente y, en base a esta información y las directivas dadas o decisiones tomadas por niveles superiores, es decir por la estrategia, deberá decidir que hacer. La correcta ejecución del comportamiento en sí, lo consideraremos en un nivel inferior del diseño del agente.

Los temas a ser tratados en este capítulo se dividen, principalmente, en dos grupos: roles y mecanismos de decisión. El concepto de rol está íntimamente relacionado a los sistemas multiagentes en general y al fútbol en particular. Ambos enfoques serán tratados. En cuanto a los mecanismos de decisión, se exponen distintos mecanismos comúnmente utilizados para la selección de los comportamientos o jugadas a ser ejecutados por el agente, así como también los parámetros de estos comportamientos.

11.2 Roles en Sistemas Multiagentes

Los roles dentro de un sistema multiagente asignan responsabilidades y tareas a los miembros del mismo. Además de la conveniencia obvia de tener roles definidos dentro de un equipo de fútbol, desde el punto de vista de los sistemas multiagentes su uso también resulta beneficioso. Considérese un sistema multiagente en tiempo real donde los agentes trabajan en pos de un objetivo mutuo. Cada uno debe analizar la situación del ambiente constantemente. Si no se utilizan roles, para un agente hay muchas reacciones y comportamientos para elegir en un determinado momento. Mientras más opciones, mayor complejidad y consumo de tiempo habrá para la decisión. Adicionalmente, nada garantiza que el agente no elija una acción que entre en conflicto con la acción de otros agentes. El peor caso de este accionar en un entorno de fútbol sería que todos los jugadores decidan correr detrás de la pelota. Una solución aquí sería tener comunicación entre los agentes. Desafortunadamente, la comunicación consume tiempo, incrementa la complejidad y no siempre está disponible [ABE].

Como se dijo anteriormente, un agente con un determinado rol tendrá menos comportamientos entre los cuales decidir. Lo ideal sería tener sólo unos pocos posibles comportamientos para cada situación, de manera tal que el agente pueda, fácilmente, decidir que hacer. En el caso de un rol determinado, muchas situaciones pueden ser excluidas. Por ejemplo, un arquero nunca se encontrará en el área penal rival, por lo tanto no necesita comportamientos para esta situación.

En síntesis, los roles reducen complejidad en los sistemas multiagentes. La reducción de complejidad implica que es necesario tomar menor cantidad de decisiones. Los agentes con roles son más simples que los que no tienen roles: de alguna manera son también menos flexibles, pero en un sistema multiagente, un combinación adecuada de roles puede hacer al sistema muy efectivo, rápido y robusto.

Un tema importante a debatir es cuánto tiene que saber un agente de un equipo acerca de los roles de los otros agentes. Un agente que conozca exactamente el comportamiento de distintos roles puede hacer predicciones acerca del comportamiento de su compañeros y usar esta información para su accionar cooperativo. Con este conocimiento sobre el comportamiento de otros agentes, la necesidad de comunicación explícita entre los jugadores puede ser minimizada.

Un agente puede sacar ventaja del conocimiento del rol de los otros agentes o bien por usar tácticas fijas o bien por predecir el comportamiento del otro agente. Una táctica fija podría ser: "El delantero izquierdo siempre pateará la pelota hacia el punto *(x,y)* si el agente está a *N* metros del arco". El delantero derecho con conocimiento de esto podría tener la siguiente regla "Posicionarse en el punto *(x,y)* y esperar por la pelota cada vez que el delantero izquierdo tenga la pelota en su poder". Esta esquema fijo es poco flexible y susceptible a eventos inesperados. Por otro lado, si se utiliza predicción, el agente modela su compañero intentando ponerse en la situación del otro agente y determinar que acción tomará. Esto implica que el agente debe tener implementado los comportamientos de todos los otros roles. Existen varias dificultades en intentar predecir el comportamiento de otro agentes, en particular debido a que no siempre los agentes tienen la misma información sobre el ambiente, en base a la cual toman decisiones. Por ejemplo consideremos el caso que un agente A_1 que intenta predecir las acciones del agente A_2, pero que no está seguro de que objetos su compañero percibe y cuales no; además pueden existir objetos que A_2 ve pero que A_1 no, lo cual conducirá a una predicción incorrecta [ABE].

11.3 Roles en el Fútbol

El fútbol es un deporte en el cual existen, por naturaleza, roles. Muchos equipos de fútbol de robots implementan algunos o todos los roles de los equipos humanos de fútbol, los cuales se describen a continuación:

- ***Arquero***: responsable de no dejar ingresar la pelota al arco propio.
- ***Defensor***: su tarea principal es evitar que la pelota llegue hasta el arco propio y que los oponentes generen jugadas de peligro cerca del mismo.
- ***Mediocampista***: actúa como nexo entre las funciones ofensivas y defensivas del equipo.
- ***Delantero***: su tarea es intentar hacer goles en el arco rival.

Estos roles generales pueden ser especializados en sub-roles de acuerdo a la región donde el agente juega y su actitud hacia el juego. El campo se puede dividir en tres regiones: derecha, izquierda y centro permitiendo subdividir los roles como izquierdos, derechos o centrales. En cuanto a la actitud hacia el juego, un defensor puede ser prudente (estar siempre en su posición defensiva) o agresivo (tratar siempre de patear la pelota). Esta especialización de roles por actitud permite fácilmente construir equipos que jueguen diferentes estrategias sin modificar las formaciones [MOR].

11.3.1 Rol Arquero

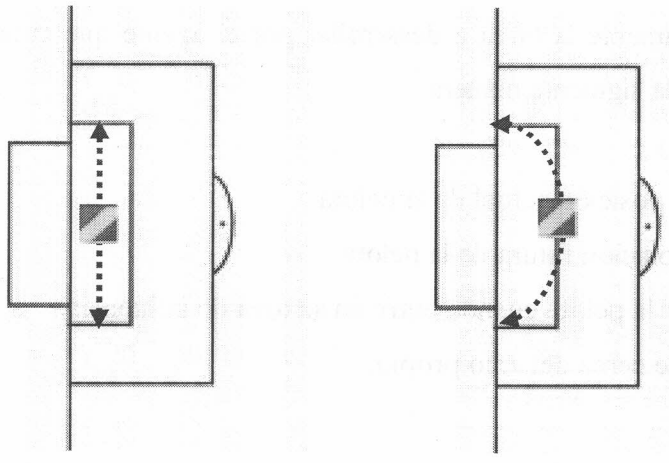
Dependiendo de la categoría, existen distintas condiciones o restricciones sobre la defensa del arco: en algunas, por ejemplo MiroSot y SimuroSot de FIRA, existe sólo una limitación en cuanto a la cantidad de jugadores que defienden dentro del área chica, por lo cual el arquero puede ser cualquier jugador del equipo. En otras, por ejemplo Large League o Simulation de RoboCup, el jugador que defiende el arco tiene atribuciones especiales, como por ejemplo puede retener la pelota, lo cual está penalizado para otros jugadores. Más allá de esto, la mayoría de los equipos designan a un jugador cuya tarea es defender permanentemente al arco. Con lo cual, como mínimo, un equipo debe considerar dos roles bien definidos: el rol de los jugadores de campo, cuya cantidad varía según la categoría, y el rol de arquero.

La táctica del arquero está altamente condicionada por la arquitectura del robot, pero básicamente la tarea a desarrollar por el agente que cubra este rol se puede resumir de la siguiente manera:

- Identificar posición actual de la pelota.
- Predecir posición futura de la pelota.
- Interceptar la pelota cuando entre en su área de influencia.
- Mantenerse cerca del arco propio.

El primer punto sólo es relevante en los ambientes con visión local, donde el robot debe tratar de detectar donde está la pelota en cada momento ya que si el arquero "pierde de vista" a la pelota puede cometer un error fatal dejándola ingresar libremente al arco. La predicción sobre la trayectoria de la pelota, si bien es beneficiosa para cualquier jugador, es fundamental para el arquero ya que él debe poder anticiparse en sus movimientos para cubrir el posible camino de la pelota hacia el interior del arco. Si sólo se maneja con la observación actual de la pelota es probable que no llegue a tiempo a interceptarla ya que la velocidad de la pelota es potencialmente mayor a la del robot. El tercer punto es la función por excelencia del arquero: evitar que la pelota ingrese al arco. Dependiendo de decisiones estratégicas, esta tarea se puede limitar a sólo bloquear el camino de la pelota, tratar de anticiparse al rival despejando la pelota, limitarse a actuar sólo frente al arco o, dependiendo de la circunstancia, salir a más distancia, etc.

El último punto tiene que ver con conservar su posición defensiva frente al arco. En ciertas ocasiones, al despejar una pelota o por efecto de otros robots puede que el arquero quede lejos del arco, lo cual resulta muy peligroso. El arquero debe rápidamente retomar su posición y orientación de defensa.



(a) (b)

Figura 11.1: Tácticas de Arquero.
(a) Trayectoria recta y paralela al arco. (b) trayectoria sobre arco.

En la figura Fig.11.1 se muestran dos de las tácticas más comunes de los arqueros. En la primera, el arquero se desplaza sobre una línea paralela al arco, en la segunda, el arquero se desplaza sobre una curva. En ambos casos, el arquero intentará ubicarse en el punto de intercepción entre la trayectoria proyectada de la pelota y la suya.

El primer caso es principalmente utilizado por los robots que no tienen movimientos laterales, ya que se ubica al arquero con orientación paralela al arco, avanzando o retrocediendo sobre la línea de manera de ubicarse rápidamente para bloquear a la pelota. La trayectoria curva del arquero, puede ser útil tanto por los agentes con visión local no omnidireccional, puesto que con este desplazamiento curvo pueden a la vez cubrir el arco y no perder de vista a la pelota, o bien para aquellos robots con movimiento omnidireccional, que podrían desplazarse sobre el arco y cuando el arquero decide que la pelota puede ser atrapada o debe ser despejada, avanzar hacia ella [NIS].

11.3.2 Rol Defensor

Un jugador con la función de defensor deberá evitar que su arco corra peligro, actuando así como primer barrera para los oponentes. En un equipo con buenos defensores, la tarea del arquero debería ser mínima.

Un defensor, en general, realiza su tarea manteniendo la pelota fuera del área penal propia y quitando la pelota al oponente. Cuando un defensor está en posesión de la pelota, la pasa a un compañero en lo posible que esté más arriba dentro de la cancha, a un compañero que esté libre de manera de asegurar la pelota, o bien la despeja enviándola a un lugar seguro lejos de los rivales y del arco propio, evitando el centro de la cancha. El defensor debe mantener una posición defensiva dentro del campo de su equipo entre la pelota y el arco propio, aunque no siempre esto es posible. Cuando un defensor ha sido superado por la pelota, debe tener especial cuidado de no generar una jugada peligrosa que desemboque en un gol en contra. En esta situación, algunos defensores optan por sólo obstruir el paso de los oponentes, sin acercarse a la pelota, lo que permite que un compañero mejor posicionado pueda ir a controlar la pelota [VEL3].

Las tácticas de desplazamiento descriptas para el arquero son aplicables a los defensores. El desplazamiento en línea recta es ampliamente utilizado como táctica de defensa. Se pueden organizar variantes teniendo en cuenta la coordenada *X* sobre la cual actúa cada robot y la amplitud de desplazamiento en coordenada *Y*. Es decir, se puede tener a más de un defensor desplazándose en forma paralela al arco, pero a distintas distancias del mismo, y además puede que estos defensores recorran todo el ancho de la cancha o sólo el lateral que les toca defender. La trayectoria curva sólo puede ser ejecutada con centro en el arco por uno o más defensores. El radio de esta curva debe ser lo suficientemente amplio de manera de no entorpecer al arquero. Una posible implementación puede ser tener un defensor en trayectoria curva y uno o más defensores, o mediocampistas, en trayectoria recta, por delante del primero formando así dos niveles de defensa.

Cuando un equipo tiene más de un jugador cumpliendo con el rol de defensor, aparecen distintos subroles o modos de comportamiento dependiendo de la situación actual. En el equipo de simulación CMUnited-98 [STO4] se definen los siguientes modos de comportamiento para los defensores:

- ***Defensa activa***: el agente irá al encuentro de la pelota, decidiendo luego si la pasa, la lleva o la despeja, dependiendo de sus posibilidades.
- ***Defensa auxiliar***: el jugador marca a un rival ubicándose cerca de él en una posición que le dificulte recibir o pasar la pelota.
- ***Defensa pasiva***: el defensor se interpone en la línea entre un rival y el arco propio a una distancia mayor a la de marcado, de manera de evitar un posible tiro al arco.

11.3.3 Rol Mediocampista

Los agentes con roles de mediocampistas se sitúan por detrás de los delanteros, y usualmente están en el medio de la cancha, aunque se mueven por casi todo el campo si es necesario. Un mediocampista que obtiene la pelota intentará llevarla hacia arriba, pasársela a un compañero o intentará patear al arco si la situación es adecuada.

La presencia de este rol en la formación de un equipo depende principalmente de la categoría. Generalmente en categorías en las cuales los equipos tienen cinco o menos jugadores, el rol de mediocampista no se implementa.

11.3.4 Rol Delantero

Los jugadores con rol delantero intentarán obtener la pelota y hacer que

ésta ingrese al arco rival, para lo cual deberán sortear exitosamente a la defensa y al arquero rival.

Juegan en frente del resto del equipo y permanecen en el lado rival, salvo restricciones reglamentarias como jugadas especiales o regla de posición adelantada, tratando de mantenerse alejados de los rivales para conservar la pelota, si la poseen, o para recibirla en forma segura de un compañero que la tiene en su poder.

Dentro de los roles del fútbol, los delanteros son quizás los que más coordinación necesitan entre sí para obtener su objetivo. Un jugador sólo llevando la pelota o intentando patearla al arco es fácilmente neutralizado por una buena defensa o un arquero eficaz. Sin embargo, una buena coordinación de pases entre delanteros, puede vulnerar defensas y arqueros ya que, al cambiar bruscamente la dirección de la pelota, las predicciones realizadas por el rival pierden vigencia y, potencialmente, puede que no lleguen a adaptarse a la nueva situación.

En forma similar a lo que se detalló para los defensores, cuando hay más de un delantero, existen subroles o modos de comportamiento. Por ejemplo en [STO4] se proponen los siguientes modos:

- ***Ataque activo***: el agente intenta ir a la pelota lo más pronto posible. Es utilizado cuando ningún compañero puede llegar a la pelota antes.
- ***Ataque auxiliar***: el atacante intenta ubicarse en un lugar libre y adecuado para recibir un pase. Se utiliza cuando un compañero tiene el control de la pelota.
- ***Ataque pasivo***: el jugador intenta ubicarse en un lugar que, probablemente, sea ofensivamente útil en el futuro.

En aquellas categorías donde intervienen pocos jugadores, por ejemplo Sony Legged de RoboCup donde participan tres robots por equipo, el rol de

atacante generalmente lo cubre un único robot, por lo cual no tiene a un compañero para realizar pases. En otros casos, como por ejemplo en el equipo CMTrio-98 [VEL5], los dos jugadores de campo son atacantes, y por lo tanto no hay defensores, aunque tampoco implementan jugada de pase. Las tareas del rol atacante son:

- Encontrar la pelota.
- Encontrar el arco rival.
- Posicionarse detrás de la pelota alineado con el arco.
- Patear

Cada uno de los atacantes ocupa uno de los laterales de la cancha, considerándose un pequeño solapamiento de las regiones.

11.4 Mecanismos de Decisión

Cuando se está desarrollando el juego, el jugador a cada instante debe decidir sus acciones, debido al cambio dinámico del ambiente. Estas decisiones involucran, básicamente dos aspectos: que comportamientos ejecutar y, dentro de este comportamiento seleccionado, que valores tomarán los eventuales parámetros de éste. Estas decisiones, dependiendo de la arquitectura del agente, pueden ser realizadas en cualquier orden y con técnicas heterogéneas.

Una solución trivial a este problema, es realizar una partición del espacio de estados y asignar a cada elemento, un comportamiento. Desafortunadamente, el espacio de estado es, conceptualmente infinito, y en la práctica tan grande que hace imposible el mapeo entre estados y acciones, lo cual hace necesario utilizar mecanismos que permitan al agente reaccionar a tiempo y seleccionando el mejor comportamiento para cada situación.

La selección de acciones, como a menudo se llama al mecanismo de decisión de un agente, puede involucran aprendizaje, ya sea *on line* u *off line*. El aprendizaje puede utilizarse en cualquiera de las dos etapas de selección: se puede aprender, en base al estado del ambiente, que comportamiento conviene llevar a cabo y/o se puede aprender a, dado un comportamiento, que parámetros dar: a que jugador hacer un pase, hacia donde patear la pelota, en que punto es más ventajoso posicionarse, etc.

A continuación se exponen distintos mecanismos y técnicas comúnmente utilizadas para abordar el problema de decisión a nivel jugador.

11.4.1 Árboles de Decisión

Los árboles de decisión pueden ser utilizados tanto en la selección de acciones, como en la elección de los parámetros de los mismos. En el primer caso, las hojas serán los comportamientos a ejecutar, y en el segundo, los valores o parámetros elegidos. En ambos casos los nodos corresponderán a condiciones o consultas sobre el estado del ambiente.

Los árboles de decisión pueden ser construidos manualmente u obtenidos en base a un conjunto de datos de entrenamiento. La manera más sencilla es armarlo manualmente, consultando condiciones sobre el ambiente en base a la experiencia o las observaciones del diseñador. Por ejemplo, dado un defensor que tiene tres comportamientos:

- ***Bloquear:*** ubicarse entre la pelota y el arco propio.
- ***Despejar***: impulsar la pelota hacia el campo rival eligiendo el mayor ángulo libre.
- ***Molesta***r: ubicarse entre un rival y la pelota (la pelota está más cerca del arco propio que el defensor),

un árbol de decisión muy sencillo que elija el comportamiento a ejecutar se muestra en la Fig.11.2, donde la función *Pelota_Arriba*, testea si la pelota está en dirección al arco rival con respecto al jugador (*Pelota_Arriba* devuelve falso cuando la pelota está más cerca del arco defendido que él mismo) y la función *Despeje_Seguro* determina si hay un ángulo lo suficientemente amplio en dirección al arco rival para impulsar la pelota hacia allí [VEL3].

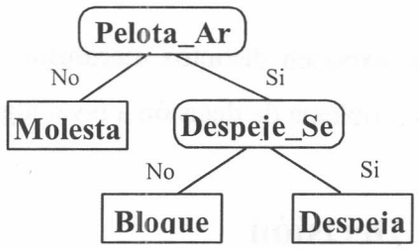


Figura 11.2: Arbol de decisión para seleccionar un comportamiento.

Cuando las variables o condiciones a evaluar son muchas o no es sencillo determinar la estructura del árbol de decisión, éstos pueden ser construidos mediante algoritmos de entrenamiento. Dado que en el entorno del fútbol muchas veces puede suceder que no siempre se cuenta con la información completa del ambiente, se necesita trabajar con herramientas que soporten incertidumbre, tanto en el entrenamiento, como en el uso. Por esto, generalmente se utilizan los árboles de decisión denominados C4.5 los cuales permiten trabajar con información incompleta. El resultado de un árbol C4.5 es un conjunto de resultados con una determinada probabilidad asociada a cada uno.

Otro uso posible de un árbol de decisión dentro de la selección de acciones, es el que se expone a continuación: el árbol no devuelve la acción ni un valor para un parámetro, sino que devuelve la probabilidad de que el jugador *i* que tiene la pelota, consiga un pase exitoso al compañero *j*. Es decir, se intenta predecir el resultado de una potencial jugada. Esta información deberá ser utilizada por una instancia superior del mecanismo de decisión y determinar si

pasar la pelota a este jugador, a otro, o no realizar un pase. Esta implementación, corresponde al equipo de simulación CMUnited [STO3].

Para entrenar el árbol, se recolectó información a partir de experimentos con el siguiente escenario limitado:

- Se ubica en forma aleatoria el pateador.
- Éste anuncia su intención de patear.
- Los compañeros informan al pateador si están en condiciones de recibir la pelota.
- El pateador elige en forma aleatoria a que compañero pasar.
- Se recolecta información del estado del escenario (necesaria para el entrenamiento).
- El pateador anuncia a quien está realizando el pase.
- El destinatario del pase, y cuatro rivales intentan interceptar la pelota (con el comportamiento de intercepción implementado en niveles inferiores).
- Se clasifica al experimento como *Exitoso* si el compañero a quien se envió la pelota, la intercepta; *Fracasado*, si la pelota es interceptada por un rival y *Perdido* si el pase no es interceptado por ningún robot.

La cantidad de patrones de entrenamientos fue de 5000, cada uno de los cuales contiene 174 características del ambiente. De este conjunto de entrenamiento, el 51% son casos exitosos, el 42% casos fracasados y el 7% restante, perdidos. El resultado del entrenamiento produjo un árbol C4.5 con 87 nodos. En etapa de testeo, se realizaron otros 5000 experimentos, ahora en vez de elegir el destinatario del paso en forma aleatoria, se utilizó el resultado del árbol, pasando al jugador con mayor probabilidad de éxito. Así, se pasó del 51% de éxito con elección aleatoria a un 65% utilizando al árbol de decisión. Cabe destacar que en el testeo, el jugador está siempre obligado a realizar un pase, a pesar de que la probabilidad más alta de éxito sea pobre. En un circunstancia real

de un partido, el jugador podría optar por no pasar la pelota y ejecutar otro comportamiento.

11.4.2 Precondiciones

La selección de acciones mediante precondiciones consiste en asociar a cada comportamiento que el agente tiene capacidad de ejecutar, un conjunto de precondiciones, las cuales deben ser satisfechas para poder disparar el comportamiento. El conjunto de precondiciones asociado a cada comportamiento está formado por precondiciones elementales que evalúan condiciones del ambiente. Por ejemplo: ¿La pelota está en campo propio? ¿Hay algún adversario con chances de patear la pelota al arco propio?, etc.

Tomando, por ejemplo, el comportamiento *Patear al Arco Rival*, su posible precondición de activación podría ser: "(¿Hay algún camino libre entre la pelota y el arco rival en el intervalo de tiempo *[t1, t2]*?) y (¿Es el robot *i* el mejor posicionado para alcanzar la pelota y patearla hacia el arco rival dentro del intervalo de tiempo *[t1, t2]*?) y no (¿Está la pelota en el campo propio en el instante *t*?)"

Para desarrollar un método de selección de acciones en base a precondiciones se deben, primero, definir el conjunto de precondiciones elementales, luego, escoger la combinación adecuada de estas precondiciones elementales para conformar la precondición de cada uno de los comportamientos de cada agente. Este proceso es, normalmente, llevado a cabo en forma manual y en base a observaciones y experiencia del diseñador. Este mecanismo de precondiciones ha sido implementado en el equipo UBASot, categoría MiroSot [SAN].

Un modelo más complejo y a la vez flexible es el que se expone a continuación. Éste divide a las precondiciones en dos categorías: Esenciales y

Benéficas. Las primeras, son precondiciones que deben cumplirse obligatoriamente para ejecutar un comportamiento. Las segundas, simplemente aumentan las probabilidades de éxito del comportamiento al cual están asociadas.

Las precondiciones son evaluadas en base a su *Grado de Satisfacción* del entorno actual. Las precondiciones esenciales al ser evaluadas, devuelven *1* (se cumple) ó *0* (no se cumple). Las precondiciones benéficas pueden devolver un valor dentro del rango continuo *[0,1]*.

A cada acción, además de asociarle el conjunto de precondiciones esenciales y benéficas, se les asocia una *prioridad.* Esta prioridad permite que, según la estrategia que se quiera llevar a cabo, los comportamientos puedan reordenarse en forma dinámica.

En base a las precondiciones y la prioridad de cada acción, se calcula una *medida de confianza* para cada acción. Esta medida está dada por una función sobre las variables anteriores. Por ejemplo:

$$Confianza(a_k) = \text{Pr}ioridad(a_k) \times \prod_i^N Satisfacción(a_k, e_i) \times \sum_j^N Satisfacción(a_k, b_j) \qquad (11.1)$$

donde a_k es una acción, e_i es una de las N precondiciones esenciales y b_j una de las M precondiciones benéficas.

El algoritmo de selección de acciones a implementar debe calcular esta medida de confianza para cada uno de los comportamientos y luego elegir el que devuelva el máximo valor de confianza [LOB].

11.4.3 Arquitectura BDI

El modelo BDI (belief-desire-intention, creencia-deseo-intención), es

ampliamente usado como técnica de planificación. Aunque en general es utilizado a nivel equipo para coordinar los jugadores entre sí, también puede ser utilizado para controlar los comportamientos del jugador

La arquitectura BDI está relacionada con el razonamiento procedural y permite estructurar los estados mediante categorías mentales:

- ***Creencias (Belief):*** modelos internos del mundo. A nivel jugador, se trata de lo que percibe más lo que predice.
- ***Deseos (Desires):*** también pueden ser denominados objetivos. Son estados a ser alcanzados en el futuro. Por ejemplo un deseo puede ser *Interceptar la Pelota*, *Patear hacia el Arco Rival*, etc.
- ***Intenciones (Intentions):*** compromisos con ciertas acciones. En particular, es un compromiso con el plan necesario para satisfacer el deseo. A nivel jugador, estos planes son los comportamientos.

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

- El robot recolecta tanto información del ambiente como le es posible y la almacena en su modelo del mundo (fase de creencia)
- Luego, busca entre sus comportamientos (por ejemplo, interceptar pelota, patear al arco, etc) y determina cual de ellos tiene la mayor prioridad (fase de deseo)
- Finalmente, el robot ejecuta el comportamiento deseado (fase de intención)

La elección del deseo se lleva a cabo mediante lo que se llama deliberación. Esta comienza una vez completado el modelo del mundo o cuando finaliza el plan actual. Primero, se busca un plan existente y se evalúan las condiciones para su continuación. Si no se decide continuar con el plan, evalúa

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 de plan del deseo mejor rankeado. Si es factible, se elige como intención; sino el siguiente deseo es chequeado. La nueva intención es comparada con la anterior, si la hubiera, estimando las ventajas y desventajas. Luego de adherir a una nueva intención, un nuevo plan o comportamiento debe ser calculado [GUG].

Algunas implementaciones introducen ciertas variantes en la implementación de la arquitectura BDI relacionadas al ambiente de fútbol. Por ejemplo, al conjunto de deseos se lo particiona en dos grupos: deseos ofensivos y deseos defensivos. La intenciones también son clasificadas en dos grupos: intenciones reactivas e intenciones tácticas.

La clasificación de deseos en ofensivos y defensivos, permite contar con distintas estrategias de juego, las cuales pueden ser decididas a nivel equipo en forma estática o dinámica, así como también, permite una elección de deseos dependiente del rol que el jugador tenga en un determinado momento. El concepto de intenciones reactivas y tácticas, introduce en el modelo BDI la consideración de comportamientos reactivos. Existe un mapeo directo entre el estado del ambiente y la intenciones reactivas, las cuales pueden reemplazar a las intenciones tácticas que se seleccionan mediante el proceso de deliberación [HHU].

11.4.4 Redes Neuronales Artificiales

La redes neuronales artificiales, por naturaleza, permiten aprender en base a ejemplos, lo cual es muy similar a la forma de aprender que tiene un jugador de fútbol. Dentro del nivel de jugador, las redes neuronales se pueden aplicar perfectamente para decidir que comportamiento ejecutar o con que variantes realizarlo. A continuación de exponen algunos ejemplos de implementaciones que utilizan redes neuronales en sus mecanismos de decisión.

Dado un conjunto de comportamientos que el agente es capaz de realizar, se puede asociar a cada uno de ellos una red entrenada de tal manera que estime la probabilidad de éxito si se desea ejecutar este comportamiento. Cada vez que se deba decidir entre los distintos comportamientos, se evalúan los resultados de las redes y, dentro de aquellos comportamientos cuya probabilidad de éxito supera cierto umbral, se debe elegir al de mayor prioridad. Por ejemplo, es preferible un pateo al arco que un pase, un pase hacia delante que un pase hacia atrás, etc.

Tomando como ejemplo al comportamiento *Realizar un Pas*, de entre los posibles comportamientos que puede ejecutar un agente en posesión de la pelota, a continuación se muestra el mecanismo implementado en [BUK]. Para estimar la probabilidad de éxito de un pase, hacia un determinado compañero, se deben tener en cuenta a los rivales que pueden hacer fracasar dicho pase. Por lo tanto, la red que estima el coeficiente de éxito a obtener, toma en cuenta al destinatario del pase y a los rivales (de a uno por vez). La red utilizada ha sido un perceptrón multicapa. En vez de *backpropagation* se utilizó *Rprop*. Los *patterns* de entrenamiento tienen la forma: $((d_o,\Delta_a,d_c),Y_t)$, donde d_o es la distancia al rival, Δ_a es la apertura entre la línea hacia el destinatario y la línea hacia el rival, d_c es la distancia al destinatario candidato y Y_t es el resultado del pase: 1 si el pase fue exitoso, 0 si no. Una vez entrenada la red, para obtener el coeficiente de éxito de un pase hacia un determinado compañero j, se consulta a la red para cada uno de los posibles rivales (X_{op}). El menor valor devuelto, constituye el coeficiente de éxito del pase:

$$P(pase_j) = min_{op}\{Red_{pase}(j, X_{op})\} \qquad (11.2)$$

Si este coeficiente supera determinado umbral, el pase es factible. Este umbral si bien está fijo, podría ser adaptado en forma *on line* según la fortaleza del rival.

Otra aplicación de redes neuronales artificiales dentro de los mecanismos de decisión del agente es realizada por el equipo AndHill 97 [AND]. Aquí se utiliza una RN para determinar a donde es conveniente enviar la pelota. Esta función se llama función de seguridad y recibe las siguientes entradas:

- Distancia al arco propio
- Distancia al arco rival
- Distancia al jugador propio más cercano
- Distancia la jugador rival más cercano

La red neuronal es ajustada "a mano" de manera que la posición más peligrosa es cerca del arco propio y la más segura es cerca de un compañero.

11.4.5 Aprendizaje por Refuerzo

Aprendizaje por refuerzo es la máquina de aprendizaje en la cual un agente intenta adaptarse al ambiente reforzando la acción que le da mayor recompensa (señal de refuerzo).

El aprendizaje por refuerzos resulta aplicable a la cuestión de cómo un agente que sensa y actúa en un ambiente, puede aprender a elegir la acción óptima para obtener su objetivo. La principal ventaja del aprendizaje por refuerzo es que provee un mecanismo para programar agentes mediante premios y castigos sin necesidad de especificar como llevar a cabo la tarea. En cada paso de interacción el agente recibe una entrada, la cual generalmente provee indicaciones del estado del ambiente. El agente luego elige una acción para generar una salida. La acción cambia el estado del ambiente y también provee un premio o un castigo al agente según cómo la tarea se llevó a cabo. El agente debe elegir acciones de manera de maximizar la sumatoria de sus recompensas a largo plazo [HHU].

Otro posible uso de aprendizaje por refuerzos en el nivel de jugador, puede ser en la decisión que el jugador debe tomar con respecto a donde ubicarse dentro de la cancha. El equipo AndHill97 utilizó este mecanismo, el cual se expone a continuación [AND].

Para determinar donde debe ubicarse el agente, se implementó una red neuronal que recibe las siguientes entradas:

- Distancia al arco propio
- Distancia al arco rival
- Distancia al jugador propio más cercano
- Distancia la jugador rival más cercano
- Seguridad de la localización actual de la pelota.

La función de seguridad de la pelota se explicó en la sección 11.4.4 sobre redes neuronales. La salida de la red es la posición a donde el agente se debe ubicar. Para entrenar a esta red se utilizó aprendizaje por refuerzo, donde se refuerza la posición donde el jugador pudo obtener la pelota con éxito. Como es usual que un jugador patee varias veces la pelota al obtenerla, se limita a un premio por segundo. Otra restricción para evitar que los jugadores se amontonen es que, para recibir premio, debe haber una cierta distancia al compañero más próximo. Dado que la exploración es costosa en términos de energía, se introdujo una exploración observacional. Es decir, los jugadores refuerzan aquellas posiciones a donde ha ido la pelota y ningún jugador la recibió.

11.4.6 Redes de Comportamientos

Una red de comportamientos en un mecanismo de selección de acciones en dominios dinámicos e impredecibles. Sus principales ventajas son: reactividad, capacidad de planificación, robustez, consideración de múltiples objetivos y cálculo distribuido y de bajo costo [MAE].

Un versión extendida del algoritmo original ha sido utilizada para el dominio de fútbol de robots. Esta versión, agrega a las características originales, la explotación de la información proveniente de un dominio continuo como el del fútbol y permite la ejecución concurrente de comportamientos [DOR].

Una red de comportamientos extendida está compuesta por los objetivos del agente, sus comportamientos (llamados módulos de capacidad) y su percepción.

Un objetivo está representado por la *Importancia,* que indica el valor del objetivo (valor estático); las *Condiciones del Objetivo,* las cuales indican cuando se considera satisfecho y las *Condiciones de Relevancia,* cuyo valor de verdad dinámico indica cuan relevante es el objetivo en cuestión. Las condiciones de relevancia sirven para modelar distintos tipos de objetivos: por ejemplo, el mantenimiento de objetivos, es decir la motivación para mantener cierto estado (ej: *Tener Energía*), se puede obtener introduciendo una condición de relevancia (ej: *Energía Baja*), que incremente la relevancia del objetivo a medida que el estado diverge del objetivo. Es decir, mientras más se aleja del objetivo, más relevante este objetivo es. Por otro lado, cuando se desea alcanzar un objetivo, mientras más cerca se está del mismo, más relevante el objetivo es. Es decir, la motivación para llegar a cierto estado. Esto se puede obtener introduciendo una condición de relevancia cuyo valor aumenta a medida que el objetivo está más cerca. Por ejemplo, la condición de relevancia *Pelota Cerca*, para el objetivo *Obtener Pelota*. La importancia estática y la relevancia dinámica, se combinan multiplicándolas para obtener la *Utilidad* del objetivo.

Un *Módulo de Capacidad* consiste del *Comportamiento* que será ejecutado una vez que el módulo es seleccionado; una lista de *Precondiciones,* que determinan la ejecutabilidad del módulo; una lista de *Efectos* que indican los

valores esperados luego de ejecutar al módulo y el valor de *Actividad,* el cual indica la contribución del módulo para alcanzar objetivos.

La *Percepción* del agente está dada por valores reales que representan el estado de elementos del ambiente y del agente mismo. La percepción es utilizada para calcular la ejecutabilidad de los módulos, la relevancia de un objetivo y controlar la propagación de la activación.

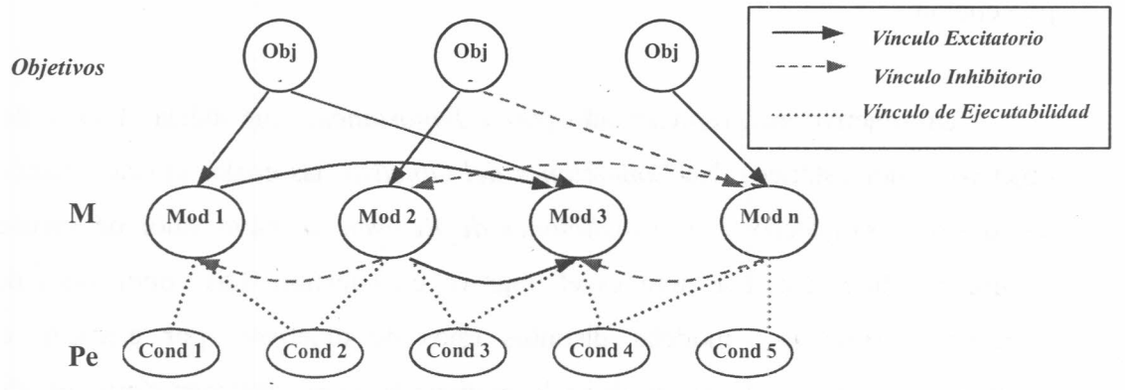


Fig.11.3: Red de comportamientos.

Los módulos de capacidad de la red están conectados con los objetivos y con otros módulos para recibir activación. Un alto valor de activación mejora las probabilidades del módulo de ser ejecutado. Un módulo recibirá activación de un objetivo, si contribuye al cumplimiento de dicho objetivo. Por otra parte, recibirá inhibición por parte de un objetivo, si el efecto del módulo es no deseado por el objetivo. Cuando un módulo no es ejecutable, propaga activación a los módulos que pueden hacer que sus precondiciones se vuelvan verdaderas. Mientras menos precondiciones se satisfacen, más activación propaga hacia otros módulos. Por otro lado, un módulo puede propagar inhibición a otros módulos, cuando éstos pueden deshacer precondiciones que el módulo ya tiene satisfechas.

El ciclo de ejecución de una selección de comportamiento usando estas redes de comportamientos extendida, consiste de los pasos del algoritmo detallado en la Fig.11.4.

Selección de Comportamiento
Comenzar
Calcular el valor de activación de cada módulo de capacidades.
Calcular la ejecutabilidad de cada módulo mediante una conjunción difusa de sus precondiciones.
Multiplicar activación y ejecutabilidad.
***Si** el producto es mayor a un umbral de selección de comportamiento*
Entonces
    Ejecutar el comportamiento
    Reinicializar el umbral con el valor original
    Volver al principio
Sino
    Decrementar levemente el umbral de selección de comportamiento
    Volver al principio
FinSi
Fin.

Figura 11.4: Algoritmo de selección de comportamientos.

Para permitir la ejecución de acciones puramente reactivas, en la selección de acciones del equipo MagmaFreiburg [DOR], se consideran las Acciones Reflejas. Estas acciones son siempre ejecutadas cuando se presentan los estímulos correspondientes. Otro tema que se tiene en cuenta es que cada comportamiento, generalmente, es una cadena de acciones, las cuales son ejecutadas una a una hasta completar el plan. En base a lo anterior, el motor de decisión del equipo funciona según el pseudocódigo de la Fig.11.5.

Motor de Decisión
Comenzar
Chequear si existen acciones reflejas para ejecutar
Si** existen **Entonces
    Ejecutar acción refleja
Sino
    Chequear si existen acciones pendientes de comportamientos en ejecución
    Si** existen **Entonces
        Ejecutar acción pendiente
    Sino

```
                    Seleccionar Comportamiento utilizando la Red de
                    Comportamientos Extendida
            FinSi
FinSi
Fin.
```

Figura 11.5: Algoritmo de decisión del equipo MagmaFreiburg.

11.4.7 Lógica Difusa

El control de los comportamientos de un agente puede ser implementado mediante los que se llama *Comportamientos Difusos*, es decir mediante módulos de control que se disparan en base a condiciones formadas por predicados difusos. A continuación se detalla la implementación de [BON] que utiliza este concepto.

Se consideran dos conjuntos de precondiciones difusas: las precondiciones de *Poder Hacer*, que habilitan al comportamiento, y las precondiciones de *Querer Hacer*, que dan la cantidad de motivación para llevar a cabo el comportamiento. Por ejemplo, si el robot tiene la pelota, puede patearla; pero se querrá hacer sólo si tiene sentido hacerlo. Los predicados difusos hacen posible clasificar la información proveniente de sensores en clases de alto nivel, dándole un significado. Así, es posible razonar con alto grado de abstracción, con un relativamente pequeño conjunto de conceptos, obteniendo alta velocidad y robustez. Más aún, una interpretación difusa da la posibilidad de razonar sobre clasificaciones solapadas, lo cual parece ser exactamente lo que los seres humanos, y ciertos animales, hacen en la mayoría de las situaciones [BON]. Por ejemplo, en la Fig.11.6 se muestra las funciones de pertenencia definiendo tres conjuntos difusos (cerca, medio, lejos) usadas para implementar los predicados difusos que clasifican la distancia a un objeto. En el ejemplo, la distancia medida (1 m) es clasificada como *Cerca* con un valor de verdad de 0.0 y *Medio,* con un valor de verdad de 0.6.

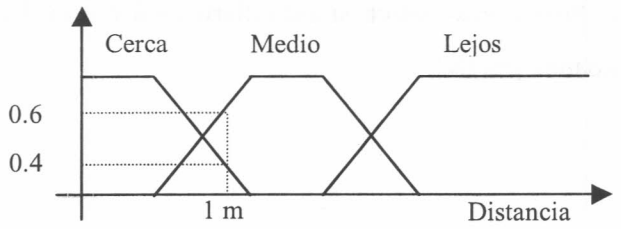


Figura 11. 6: Ejemplo de clasificación difusa.
La distancia 1 m. es considerada Cerca con un valor de verdad de 0.4 y se considera distancia media, con un valor de verdad de 0.6.

En la implementación presentada por los autores, se asocian a los comportamientos dos parámetros adicionales: la relevancia estática y la relevancia dinámica. La primera es implementada a priori, como un ordenamiento parcial entre los comportamientos, permitiendo establecer, por ejemplo, que evitar un choque es siempre mejor que obtener la pelota. La relevancia dinámica también establece un orden parcial, pero éste puede ser modificado de acuerdo a situaciones enfrentadas por el agente, y es utilizada para implementar estrategias y mecanismos de aprendizaje.

En cada ciclo de control se calcula para cada comportamiento cuyas precondiciones de *Poder Hacer* son verdaderas por encima de determinado umbral, su *Nivel de Ejecutabilidad*, componiendo sus dos valores de relevancia, la motivación proveniente de sus precondiciones de *Querer Hacer* y la posibilidad proveniente de su precondiciones de *Poder Hacer*. Como harían la mayoría de los seres vivos y en contraste con la mayoría de los agentes artificiales difusos, el comportamiento con el mayor nivel de ejecutabilidad es activado y sus acciones son realizadas. Se obtiene así una activación del tipo "El ganador toma todo", en vez de una combinación de acciones típica de los sistema difusos, dado que esto da más coherencia a la selección de comportamientos. Si se decide que es mejor ir a la pelota en vez de esperar un pase, no tiene ningún sentido realizar un comportamiento híbrido cuya posibilidad de éxito resulta muy cuestionable. Es preferible tomar una decisión y actuar coherentemente con ella. Si la decisión fue errónea, sus condiciones de activación deben ser ajustadas, pero esto sólo puede

llevarse a cabo en forma automática si está claro cual fue la decisión responsable de las acciones tomadas [BON].

Capítulo 12
Estrategias

12.1 Introducción

Cuando en fútbol se habla de cómo sale un equipo a jugar, se está hablando de su estrategia de juego. La estrategia no es más que el plan que lleva a cabo el equipo durante el partido: la distribución de sus jugadores en el campo, el rol que desempeña cada uno, si éste lo mantiene durante todo el partido o cambia de acuerdo a las circunstancias, los códigos para la comunicación, las jugadas prearmadas, los comportamientos individuales, etc. Como se mencionó en la introducción a la Parte II, Aspectos Generales de Diseño, la estrategia es un plan global y de alcance amplio. La estrategia se aplica porque se considera que un juego ordenado redunda en mejores resultados para el equipo.

Para la definición de la estrategia en el ambiente de fútbol de robots, además de las características particulares que tendrá el plan de juego, se deben tener en cuenta las condiciones y limitaciones propias de este ambiente. A continuación se describen los puntos que se deben considerar para cada enfoque.

12.2 Características del Juego

En la definición de una estrategia es primordial determinar la actitud o mentalidad de juego que tendrá el equipo. Si se prioriza el rol del atacante se está ante un juego ofensivo, donde las posiciones de los jugadores se fijan más hacia el campo contrario que al propio. Las jugadas privilegian los pases hacia adelante y los tiros al arco, sobre mantener y asegurar el control de la pelota. Por el contrario, un equipo podría preferir un juego más defensivo donde se trabaje con una

defensa sólida para evitar los goles del adversario. Evidentemente este tipo de juego ofrece menos oportunidades de convertir goles. La decisión sobre el tipo de juego a implementar debe estar basada en las características propias del equipo y de sus jugadores, y puede estar influenciada por las características del oponente.

Una opción válida entre estas dos posturas es la del juego variado, donde el equipo no se restringe a un solo tipo de juego. En este enfoque, se puede pasar de una actitud ofensiva a defensiva y viceversa de acuerdo con la evolución del juego. Se requiere para ello una buena coordinación entre los jugadores que integran el equipo. En cada instante deben acordar el estilo de juego y el rol que cada uno tiene en él.

Las formaciones representan la distribución de los jugadores en la cancha. La cantidad de jugadores en cada campo (propio o rival) denota la actitud de la estrategia que usa la formación. En la notación (*a-b-c*), *a* corresponde a la cantidad de jugadores en campo propio, *b* a la cantidad de jugadores en el sector central de la cancha y *c* a la cantidad de jugadores en campo rival. En formaciones con reducida cantidad de jugadores, alguna de las variables de la formación (*a-b-c*) seguramente será igual a cero.

Un ejemplo de juego variado es el presentado por el equipo CMUnited 97 [STO2] en RoboCup. En este equipo las formaciones pueden pasar de muy defensivas *(8-2-0)* a muy ofensivas *(2-4-4)* de acuerdo con el resultado del partido y el tiempo restante de juego: el equipo pasa a una formación ofensiva si va perdiendo y el final del partido está cerca; o pasa a una formación defensiva cuando está ganando. Las diferentes formaciones están preacordadas a través de *Acuerdos de Vestuario* y mediante acuerdos en los momentos de sincronización.

La estrategia aplicada por el equipo UBASot presentado en la categoría *SimuroSot* en FIRA 2002 y que se describe en forma detallada en el capítulo 13 de este trabajo, también combina ambas posturas, ofensiva y defensiva. En este caso,

no se alterna entre las dos posturas, sino que se adoptan ambas en forma simultánea. El equipo presenta una formación fija *(2-0-2)*, donde la tarea asignada a los dos defensores consiste en cubrir el arco de la manera más eficiente posible mientras que los dos delanteros, en forma coordinada, se mantienen en el campo rival atentos para iniciar el ataque.

La actitud de juego es una de las características más destacables cuando se observa un partido de fútbol, pero también es importante mencionar que existen otras características para tener en cuenta a la hora de la definición de la estrategia y que pueden tener mucha influencia en el resultado final: el ritmo que se aplica al juego, la presión que ejerce el equipo para recuperar la pelota, la elección de los espacios libres para el ataque, el estilo de ataque, el estilo de marcación de la defensa, el intercambio de posiciones o roles, etc. [REI3].

El ritmo de juego está asociado con la velocidad que impone el equipo y es independiente de la actitud planteada: ofensiva o defensiva. Los equipos pueden utilizar velocidades máximas para todos los movimientos, o aplicar velocidades bajas tanto para mover la pelota como para el desplazamiento de los robots. Los movimientos lentos permiten mayor dominio y control de los robots, pero pueden redundar en un bajo rendimiento por no adaptarse a la velocidad y dinamismo impuesto por el juego. Por el contrario, los movimientos rápidos generan un juego dinámico pero de menor precisión. Combinando ambos planteos, se puede optar por que los jugadores sólo se muevan en forma rápida cuando su participación en la jugada es decisiva y aplicar movimientos lentos, o no moverse, dependiendo de su cercanía de la pelota. En el caso particular de la categoría de simulación de RoboCup, los jugadores tienen energía (*stamina*) y ésta se consume con sus movimientos. Por este motivo, hacer movimientos innecesarios puede generar que los jugadores estén cansados cuando se requiere su participación en el juego. De la misma manera, en los robots reales, se debe considerar el consumo de sus baterías.

También se puede tener en cuenta la presión a ser ejercida por el equipo cuando ha perdido la posesión de la pelota. Para incrementar las posibilidades de recuperar la pelota, una alternativa es que los jugadores tiendan a concentrarse en la región donde se encuentra la pelota. En este caso, más de un jugador trata de presionar al rival y sus compañeros les dan soporte cubriendo posibles pases o tiros hacia el arco. En ocasiones la defensa propia se puede ver debilitada, pero la probabilidad de recuperación de la pelota se puede ver incrementada.

Como parte de la estrategia también se puede considerar la utilización de distintas regiones de la cancha para el desarrollo de los movimientos de ataque. Se puede plantear un sistema de ataque balanceado, que considere los espacios libres en forma indistinta, o un sistema que priorice los movimientos que se desarrollen utilizando sectores particulares de la cancha, como pueden ser los laterales. Para sus movimientos de ataque, el equipo de simulación AndHill 97 [AND], identifica y utiliza los sectores libres de la cancha que brindan mayor seguridad para que el ataque sea efectivo. El método aplicado por este equipo se describen en la sección 11.4.4.

El estilo de ataque está relacionado fundamentalmente con el tipo de pases utilizados por los robots durante el ataque. Los pases pueden ser cortos a otros compañeros o a si mismo con la finalidad de llevar la pelota (juego posesivo). También pueden ser largos, hacia adelante, ganando distancia y haciendo que los robots se desplacen rápidamente para alcanzar la pelota (juego directo). La elección de estilo, depende en gran medida de las características físicas de los robots. Los robots que poseen dispositivos especiales para pateo tienen mayores posibilidades de realizar pases largos, puesto que el dispositivo les puede facilitar el desprendimiento de la pelota.

El intercambio o reasignación de posiciones o roles hace referencia a la posibilidad que poseen los jugadores de cambiar su posición con un compañero o variar el rol asignado inicialmente.

El intercambio de roles está asociado directamente con la asignación dinámica de roles, haciendo que el jugador adopte un rol en función al estado del juego. El intercambio implica que dos jugadores en forma recíproca tomen el rol del otro, y sin la asignación dinámica esta alternativa sería imposible. La principal ventaja de la asignación dinámica es una mayor flexibilidad ante los cambios, permitiendo hacer variaciones de acuerdo con los resultados de los partidos, el tiempo de juego restante y las modificaciones en el planteo oponente.

Los roles básicos en el juego de fútbol, como se ha expuesto en detalle en la sección 11.3, son: Delantero, Mediocampista, Defensor y Arquero. Todos los jugadores están habilitados para cambiar sus roles. Por ejemplo, para un equipo de cinco jugadores, se plantea como formación inicial *(1-2-1)*. Durante el partido, la política de asignación de roles puede tomar como referencia la posición actual del jugador: por ejemplo si tres jugadores se encuentran en el campo propio, uno asume el rol de defensor y los otros dos, de mediocampistas [NOR].

Otra posibilidad es asignar los roles en función del modo de juego y la situación de los jugadores con respecto a la pelota y a los demás jugadores. Si se está en un modo ofensivo, los roles que pueden adoptar los jugadores son, por ejemplo: primer atacante (jugador que posee la pelota), segundo atacante (jugador que brinda apoyo al primer atacante) y tercer atacante (el resto de los jugadores). Si se está en un modo defensivo: primer defensor (jugador más cercano a la pelota), segundo defensor (jugador que apoya al primer defensor) y tercer defensor (resto de los jugadores). Cada jugador debe seleccionar autónomamente el rol más adecuado. En el momento que un jugador atrapa la pelota, los roles de todos los jugadores automáticamente deben cambiar [KIN].

Como se mencionó anteriormente, las características descriptas se pueden tener en cuenta durante la definición de una estrategia, pero esto no significa que todas ellas deben formar parte de la estrategia final o que se debe optar solamente

por una de las alternativas presentadas para cada punto. De hecho, para algunas de estas características aún no se han encontrado referencias donde las mismas hayan sido tratadas en profundidad. Fundamentalmente esto se debe a su complejidad, en combinación con las limitaciones que presenta el ambiente de fútbol de robots. Las principales cartacterísticas y limitaciones de este ambiente se describen en la próxima sección.

12.3 Características del Ambiente

En muchos casos la posibilidad o no de considerar una característica determinada en la estrategia depende de las facilidades o limitaciones que posee el ambiente donde se desarrollará el juego. En este aspecto, los dos elementos de mayor importancia son: la *Información* con la que contará cada uno de los jugadores a la hora de tomar las decisiones y actuar; y el *Tipo de Comunicación* que existirá entre los jugadores.

Se dice que los jugadores cuentan con información completa cuando conocen el estado de todos los elementos que participan del juego en cada instante: la posición y la orientación de todos los jugadores (tanto propios como rivales), la ubicación de la pelota, las velocidades de los objetos en movimiento y su aceleración, los límites de la cancha y cualquier otro elemento que sea referencia en el juego. El hecho de tener disponible todo el entorno hace que la estrategia utilizada dispare las acciones tomando como base condiciones reales y no supuestas.

En otras oportunidades los jugadores no tienen acceso a toda la información y su conocimiento sobre el juego es acotado. No siempre los jugadores saben la ubicación de sus compañeros, de sus rivales o la posición de la pelota, por lo que es posible que esta información incompleta presente

inconvenientes en el momento de decidir y actuar. Esto es consecuencia de contar con parte del estado actual.

En las categorías de robots reales que implementan visión global o en la categoría de simulación de FIRA, los jugadores cuentan con información completa sobre el estado de juego [FIR]. Por otro lado, en las categorías que utilizan visión local o en la categoría de simulación de RoboCup, los jugadores cuentan con información parcial, basada solamente en los datos que les brinda su campo visual [ROB][NOD].

Contando cada jugador con información local solamente, pero existiendo comunicación entre los mismos jugadores o entre los jugadores y algún agente externo (habitualmente denominado *Coach*), es posible completar el estado del juego. De esta forma cada jugador tiene la posibilidad de conformar una percepción global, sumando su propia percepción y las percepciones locales de sus compañeros. Esta suma puede traer aparejados problemas de compatibilidad entre las percepciones, pues éstas pueden ser inconsistentes entre sí. Por este motivo se deben prever mecanismos para resolver las diferencias. Para el intercambio de información mencionado, tanto entre los jugadores como con un *coach*, es necesario que el ambiente cuente con algún mecanismo de comunicación explícita. En caso de existir este mecanismo, además de ser utilizado para el envío de información sobre posiciones y velocidades, puede ser utilizado para la transmisión de las directivas definidas por la estrategia empleada.

En términos generales, los mecanismos de comunicación permiten coordinar las comportamientos de los jugadores y sincronizar las acciones de los mismos. La comunicación se puede utilizar para acordar objetivos, tomar decisiones, compartir información o recibir órdenes de agentes externos. Existen básicamente dos formas de comunicación: *comunicación explícita* y *comunicación implícita*.

En la comunicación implícita, a diferencia de la comunicación explícita presentada previamente, no existe flujo de información entre jugadores sino que la comunicación se da a través de preconceptos o creencias comunes. Cada jugador conoce su función y las acciones que debe ejecutar ante cada situación. Dependiendo de la implementación de la estrategia, también puede conocer la reacción de cada uno de sus compañeros ante la misma situación y actuar en consecuencia. Ante este tipo de mecanismo de comunicación, la estrategia de equipo definida debe ser lo suficientemente robusta como para que los jugadores, trabajando en forma individual, cooperen entre sí y funcionen como un equipo.

Existe un tercer elemento en el ambiente que se debe considerar en el momento de determinar si la estrategia que se está definiendo será dinámica o estática. Se dice que la estrategia es dinámica cuando el plan tiene la capacidad de adaptarse automáticamente de acuerdo con las condiciones de juego. En cambio, la estrategia es estática cuando conserva las mismas características durante todo el partido. En categorías de simulación y también de robots reales, por ejemplo SimuroSot y MiroSot en FIRA, los reglamentos permiten interrumpir el juego y reemplazar el software activo por un nuevo software. Esta acción permite variar la estrategia de juego sin necesidad de haber desarrollo una estrategia dinámica, sino varias estrategias estáticas que tienen su aplicación ante distintas circunstancias. Se puede decir que las estrategias dinámicas tienen una adaptación *on-line* al juego y las estrategias estáticas, adaptación *off-line*.

12.4 Mecanismos de Decisión

Cuando se aplican estrategias dinámicas, durante el desarrollo de un partido, los responsables de llevar adelante la estrategia deben decidir ante cada circunstancia el tipo de plan más conveniente. Los denominados responsables de llevar adelante la estrategia pueden ser tanto agentes externos (*coach*) como los mismos jugadores.

En esta instancia, cuando se habla de la toma de decisiones de los jugadores, no se hace referencia a la determinación de una acción y a la selección de sus parámetros (detallado en la sección anterior), sino que se está haciendo referencia a los cambios de política que el equipo implementa para mejorar su rendimiento.

En los equipos que cuentan con *coach*, este agente externo es el encargado de analizar el juego e indicar a los jugadores de su equipo el plan adoptado. Cuando los mismos jugadores son los encargados de seleccionar el plan, existen dos alternativas para su implementación: cada jugador en forma individual hace este análisis y lo aplica o el equipo tiene definidos jugadores específicos que toman las decisiones y las comunican a un subconjunto de compañeros para su aplicación. Este segundo esquema introduce el concepto de líderes dentro del equipo.

Habitualmente la decisión de hacer variantes sobre la política de juego aplicada se basa en alguno de los siguientes elementos:

- ***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 su 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.

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

A continuación se exponen técnicas comúnmente utilizadas para abordar el problema de decisión a nivel equipo.

12.4.1 STEAM

STEAM [TAM] es un modelo general de trabajo en equipo. Este modelo general está motivado por la necesidad de flexibilidad en las actividades de equipo, así como también la reusabilidad de capacidades de equipo entre distintos dominios. STEAM usa el marco formal de *Intenciones conjuntas* como piedra fundamental, pero con mejoras claves para reflejar las limitaciones del dominio del mundo real. STEAM requiere que los miembros individuales del equipo representen explícitamente sus objetivos de equipo, planes y creencias mutuas. Luego permite a los miembros del equipo razonar autónomamente sobre coordinación y comunicación en el trabajo de equipo, proveyendo una mejorada flexibilidad.

En la categoría de simulación de RoboCup en 1997, el equipo ISIS [TAM] implementó, en lo que hace al trabajo en equipo, la comunicación entre los agentes mediante el razonamiento de propósitos generales de STEAM. Dada su independencia del dominio, STEAM también permite la reusabilidad entre dominios. El fútbol de robots provee un desafiante dominio de testeo, dado su disimilitud con el dominio original de STEAM: equipos de pilotos para simulación de combates para entrenamiento militar. Un 35 % del código original de STEAM fue reutilizado y no se requirió otro código adicional para trabajo en equipo.

Un agente ISIS está desarrollado en una arquitectura de dos niveles. El nivel más bajo, comunica los datos recibidos del simulador (después de un preprocesamiento) al nivel superior. Todas las decisiones las toma el nivel superior, implementado en la arquitectura de inteligencia artificial integrada *Soar* [RSN][SOA]. Una vez que el nivel superior basado en *Soar* toma una decisión, éste se comunica con el nivel inferior, el cual entonces envía la información relevante al simulador.

La arquitectura *Soar* involucra la ejecución dinámica de una jerarquía de operadores (plan reactivo). Estos operadores consisten de (i) Reglas de precondición; (ii) Reglas de aplicación; y (iii) Reglas de terminación. Las reglas de precondición ayudan a seleccionar operadores a ejecutar basado en los objetivos / tareas y creencias sobre su ambiente de alto nivel actuales del agente. La selección de operadores abstractos de alto nivel a ejecutar llevan a subobjetivos, donde nuevos operadores son seleccionados para la ejecución, y así resulta una expansión jerárquica de operadores. Los operadores activados son ejecutados por las reglas de aplicación. Si las actuales creencias del agente coinciden con las reglas de terminación de un operador, el operador termina.

En la jerarquía de operadores de los agentes de ISIS, necesaria para soportar el razonamiento de trabajo en equipo de STEAM, se incluyen *Operadores de equipo* (planes reactivos de equipo). Los operadores de equipo explícitamente expresan actividades conjuntas de un equipo, a diferencia de los operadores individuales comunes los cuales expresan las actividades propias de un agente.

Como con los operadores individuales, los operadores de equipo también consisten en; (i) Reglas de precondición; (ii) Reglas de aplicación; y (iii) Reglas de terminación. Sin embargo, mientras que un operador individual se aplica al estado privado de un agente (las creencias privadas), un operador de equipo se

aplica al estado de equipo de un agente. Un estado de equipo es el modelo (abstracto) de las creencias mutuas del equipo sobre el mundo, por ejemplo, las estrategias actuales del equipo mutuamente creídas. El estado de equipo es normalmente inicializado con información a cerca del equipo, tal como los miembros del equipo, posibles subequipos, canales de comunicación disponibles para el equipo, el líder predeterminado del equipo, etc. STEAM también puede mantener estados de subequipos. Por este motivo, cada miembro del equipo mantiene su propia copia del estado del equipo, y de los subequipos en los que participa. Para preservar la consistencia de un estado de equipo o subequipo, una restricción clave es impuesta para su modificación: sólo los operadores de equipo ejecutados por el equipo o subequipo pueden modificarlo, respectivamente.

Una intención conjunta de un equipo está basada en su compromiso conjunto, el cual es definido como un *Objetivo Persistente Conjunto o Joint Persistent Goal (JPG)*. Un *JPG* para obtener una acción de equipo *P*, requiere que todos los miembros del equipo mutuamente crean que *P* es actualmente falso y quieren que *P* sea eventualmente verdadero.

Un *JPG* garantiza que los miembros del equipo no puedan descomprometerse hasta que *P* sea mutuamente conocido como alcanzado, inalcanzable o irrelevante. Si un agente privadamente cree que *P* es alcanzado, inalcanzable o irrelevante, el *JPG* es disuelto, pero el agente mantiene un compromiso de hacer que esta creencia sea una creencia mutua de todo el equipo. Para esto, el agente típicamente se deberá comunicar con sus compañeros.

Para establecer un *JPG* los agentes se deben sincronizar. Para esto se adoptó el protocolo de *Request-confirm.* Un agente envía un mensaje intentando establecer *JPG*, los demás contestan mediante *Confirm* o *Refuse*. Si todos confirman, el *JPG* está establecido.

En la teoría de intenciones conjuntas, para establecer y para terminar una intención se produce comunicación. Sin embargo, la comunicación para establecer y terminar cada intención puede ser altamente ineficiente. STEAM incluye decisión teórica de comunicación selectiva. En particular, los agentes explícitamente razonan acerca de costos y beneficios de la comunicación. Por ejemplo, evitan comunicación si hay una alta probabilidad de que información relevante puede ser obtenida por los otros compañeros vía observación.

Un ejemplo de comunicación se puede ver cuando tres jugadores en un subequipo de defensa ejecutan el operador de equipo DEFENDER-ARCO. Para servir a este operador, los jugadores de este subequipo normalmente ejecutan el operador de equipo DEFENSA-SIMPLE para posicionarse a sí mismos adecuadamente en el campo y tratar de estar atentos a la posición de la pelota. Por supuesto, cada jugador puede sólo ver lo alcanzado por su campo de visión, y particularmente mientras se reposiciona, puede estar desatento a una pelota que se acerca. Aquí es donde se destaca lo beneficioso del trabajo en equipo. En particular, si cualquiera de estos jugadores ve la pelota acercarse, declara al operador DEFENSA-SIMPLE, irrelevante. Sus compañeros ahora focalizan la defensa del arco en una manera coordinada vía el operador de equipo DEFENSA-CUIDADOSA. Si cualquiera de los jugadores del subequipo de defensa ve la pelota moverse lo suficientemente lejos, avisa a sus compañeros: DEFENSA-CUIDADOSA es irrelevante. Los jugadores del subequipo una vez más ejecutan DEFENSA-SIMPLE para intentar posicionarse cerca del arco. De esta manera, los agente intentan coordinar la defensa del arco, mientras también intentan posicionarse cerca de él [MAR].

12.4.2 Redes Neuronales Artificiales (RNA)

En el nivel de equipo, las variaciones y ajustes sobre las estrategias de juego tendientes a mejorar el desempeño de los equipos, técnicamente se pueden resolver aplicando RNA. Las RNA, aplicadas en este ambiente, pueden ser

entrenadas para tomar decisiones estratégicas sobre la formación, la actitud, la velocidad de juego u otros aspectos claves en el desarrollo de un partido. A continuación se presenta la implementación realizada por el equipo Virtual Werder, participante de la categoría de simulación de RoboCup 2000. Este equipo ha utilizado esta técnica para determinar la formación más conveniente para su equipo, en base a la formación de su oponente [VIS].

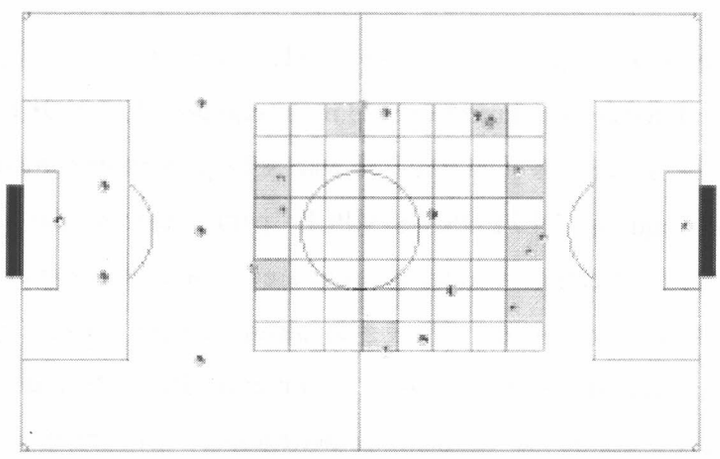


Figura 12.1: Posicionamiento de jugadores oponentes.
Cuadrícula para determinar los valores de entrada a la RNA.

La información de entrada a la RNA corresponde a la distribución de los jugadores rivales en el campo de juego. Esta red está compuesta por 64 neuronas de entrada con valores binarios. Cada uno de ellas representa una celda en una cuadrícula de 8 casilleros de lado *(8 x 8)* que se posiciona dentro de la cancha (Fig.12.1). Cuando se detecta al menos un jugador en la celda, el valor asignado es *1*. En caso contrario, se asigna *0*.

La RNA clasifica la distribución de jugadores informada como entrada en uno de los 16 sistemas de juego que tiene identificados para su adversario. Como consecuencia la red tiene 16 neuronas de salida y cada una corresponde a una formación rival. Si la neurona de salida de mayor valor supera un umbral preestablecido, la distribución actual de los jugadores rivales está dada por la

formación que esta neurona representa. Este resultado, sumado a información adicional del tamaño y posicionamiento de la cuadrícula mencionada en el párrafo anterior o el resultado parcial del partido, determinan la formación propia que se debe adoptar. Para el análisis de esta información se utiliza árboles de decisión.

El conjunto de *patterns* de entrenamiento se generaron desde archivos de logs de partidos jugados en la Liga RoboCup 1999. Los diseñadores del equipo Virtual Werder desarrollaron una herramienta que, tomando como entrada los archivos de logs, pudiera extraer las posiciones de los rivales de acuerdo con las 64 celdas definidas. Para el entrenamiento de la RNA y la generación de su código fuente se aplicó *Sttutgart Neural Network Simulator* (SNNS v4.1). La RNA fue entrenada con 612 *patterns* y testeada con 68 *patterns*. Como método de aprendizaje se utilizó *backpropagation* con parámetro de aprendizaje $\eta = 1.0$ y diferencia máxima $d_{max} = 0.3$ entre el valor de aprendizaje y la salida.

12.5 Modelado del Oponente

En la definición o aplicación de una estrategia se evalúan, en primera instancia, capacidades del equipo propio y condiciones externas básicas, como pueden ser, la cantidad de goles anotados y recibidos o el tiempo que resta para finalizar el juego. Estos elementos son relativamente estáticos, sin embargo el ambiente donde se desarrolla el partido es sumamente dinámico.

Este dinamismo lo genera el juego propiamente dicho. La incertidumbre ante una situación, que indefectiblemente deriva en la toma de una decisión para los jugadores, se origina fundamentalmente por la intervención de los jugadores del equipo contrario. Sin su participación, no sería un partido de fútbol, pero además se convertiría en un juego determinístico donde siempre se conoce el camino para encontrar la solución.

Contar con un modelo del desenvolvimiento del equipo adversario es de suma utilidad para el equipo propio. Modelar y razonar sobre los objetivos, los planes, el conocimiento o las capacidades de los jugadores rivales, tanto en forma individual como grupal, permite la optimización de los planes y estrategias propios, aprovechando los puntos débiles del oponente y tomando ventaja de ellos.

El modelado del equipo oponente se puede encarar mediante tres enfoques. Los mismos pueden ser combinados o aplicados individualmente:

- ***Análisis de partidos jugados:*** se trata de la recopilación de partidos jugados por el equipo adversario y, mediante la observación de un "experto", determinar las fortalezas y debilidades del equipo evaluado.
- ***Reconocimiento de estrategias:*** esta alternativa consiste en observar el juego desde afuera y descubrir las estrategias de alto nivel empleadas por el adversario. Esta función la cumple un *coach* y se realiza en tiempo real, a diferencia del análisis planteado en el enfoque anterior.
- ***Rastreo:*** se trata de un seguimiento dinámico del oponente, sus intenciones y objetivos, para predecir su juego y actuar en consecuencia. El desafío consiste en rastrear al adversario en tiempo real a pesar de las ambigüedades, y rastrear al equipo más que a cada jugador en forma individual.

En el ámbito de fútbol de robots, el modelado de adversarios es una disciplina en desarrollo. En las categorías de RoboCup las investigaciones y los principales avances están focalizados en las categorías de robots de mayor porte, con sistemas de visión y mecanismos de movilidad. Por su complejidad, existen más proyectos de modelado en categorías de robots pequeños o en simulación [STE].

12.5.1 Análisis de Partidos Jugados

Los diseñadores de la estrategia analizan los características de los próximos rivales observando su desenvolvimiento en partidos jugados anteriormente con otros equipos o con el propio. De acuerdo con el modelo obtenido, se adapta la estrategia existente o se selecciona la estrategia desarrollada que más se ajusta al modelo.

Esta alternativa se basa en la observación y la lógica humana. Los diseñadores son los encargados de ajustar los parámetros o realizar los nuevos desarrollos, no actúan procesos automatizados que realicen el análisis. Aunque los resultados pueden ser satisfactorios, esta metodología no cumple con la premisa de autonomía que persigue el fútbol de robots [STE].

Dependiendo de la categoría, el software de la estrategia se mantiene durante todo el partido o puede ser reemplazado por otro software en los tiempos muertos. En el primer caso, el modelado del oponente también se mantiene durante todo el partido. En el segundo caso, mediante observación humana y en base a la evolución del partido, se puede optar por otro esquema de juego desarrollado previamente.

12.5.2 Reconocimiento de Estrategias

En esta alternativa se introduce el concepto de análisis y modelado en tiempo real, impactando en forma directa sobre la posibilidad de adaptar en forma automática la estrategia propia en base a la política aplicada por el equipo oponente. Para este análisis se plantea un módulo externo (*coach*) que, evaluando el contexto y el juego adversario, envía directivas a los jugadores para actuar en consecuencia. En primera instancia, esta es una alternativa intermedia entre el esquema anterior y el modelado totalmente *on-line*. Se trata de realizar la

evaluación en forma centralizada y luego distribuir los resultados.

El equipo Virtual Werder, participante de la categoría de simulación de RoboCup 2000 ha utilizado el reconocimiento de la estrategia oponente y, en base a la formación del equipo rival, determina la formación más conveniente para su equipo [VIS]. El método utilizado ha sido detallado en este capítulo.

12.5.3 Rastreo

La opción de modelado más compleja es el rastreo. Como consecuencia de su complejidad, es la alternativa menos desarrollada en el ámbito de fútbol de robots, pero es el modelado más completo, con mayor dinamismo y que más se ajusta a la realidad. Cada agente debe tener la posibilidad de adaptar sus tareas si, en base a las acciones de los rivales, la estrategia aplicada por su equipo en forma global varía. Cada jugador debe estar capacitado para decidir sin recibir directivas de un *coach*.

A modo de ejemplo se menciona la implementación realizada por FC Portugal, participante en la categoría de simulación de RoboCup 2000 [REI1]. Un módulo interno en cada agente analiza la evolución del juego y a través de mecanismos de aprendizaje intenta determinar, entre otros, los siguientes parámetros:

- El agente que tiene asignado el rol de arquero
- La posición del agente arquero en la ejecución de un tiro libre
- La dirección y velocidad del arquero
- La dirección y velocidad del saque de arco
- La formación global del equipo oponente
- El posicionamiento de los jugadores rivales ante cada situación

En el caso particular de FC Portugal 2000, también existe intervención humana. La selección de la estrategia aplicada inicialmente es definida por un *coach* humano en base a información previa sobre el oponente. Pero a diferencia la primera alternativa planteada para modelado *Análisis de Partidos Jugados*, FC Portugal ajusta el modelado y la estrategia durante el transcurso del partido [REI2].

12.6 Ejemplos de Estrategias Aplicadas

A continuación se muestran ejemplos de estrategias a nivel equipo donde la decisión de variar la estrategia se toma en forma automática y su coordinación se realiza a distintos niveles: *coach* como un agente externo (no humano), subequipos con agentes líderes y agentes individuales con coordinación de equipo implícita.

12.6.1 Coordinación de Agente Externo (*Coach*)

La funcionalidad de *coach* no es aplicable a todas las categorías de fútbol de robots, es utilizada fundamentalmente en la categoría de simulación de RoboCup. Para su implementación se han desarrollado lenguajes especiales a través de los cuales el *coach* se comunica con los agentes. El lenguaje utilizado en la competencia se denomina *CLang*. Otro lenguaje desarrollado con esta finalidad es *COACH UNILANG*.

Para la implementación de la figura de *coach* resulta necesario que el hardware utilizado (robots, computadores centrales, etc.) permita la instalación de un módulo especial, independiente de los módulos tácticos de los jugadores. Con este esquema también son primordiales los mecanismos de comunicación para distribuir las órdenes.

Existen distintas arquitecturas para llevar a cabo la implementación de esta funcionalidad:

- Dos módulos externos, uno corresponde al *coach* y otro, a un asistente. En esta arquitectura se introduce el concepto de asistente de *coach*. Este módulo tiene asignada la responsabilidad de analizar la información estadística y modelar al equipo oponente, proporcionando estos datos al *coach* para la toma de decisiones.

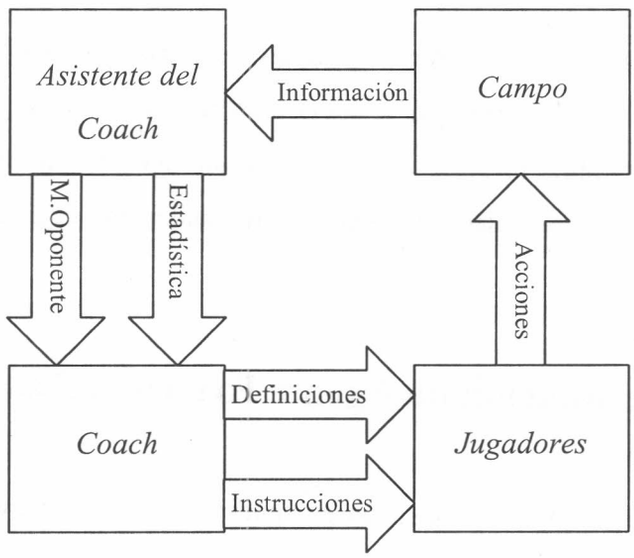


Figura 12.2: Arquitectura de dos módulos: coach y asistente de coach.

- Un módulo externo, donde se unifica el *coach* y el asistente. El mismo agente se encarga de observar, analizar y decidir.

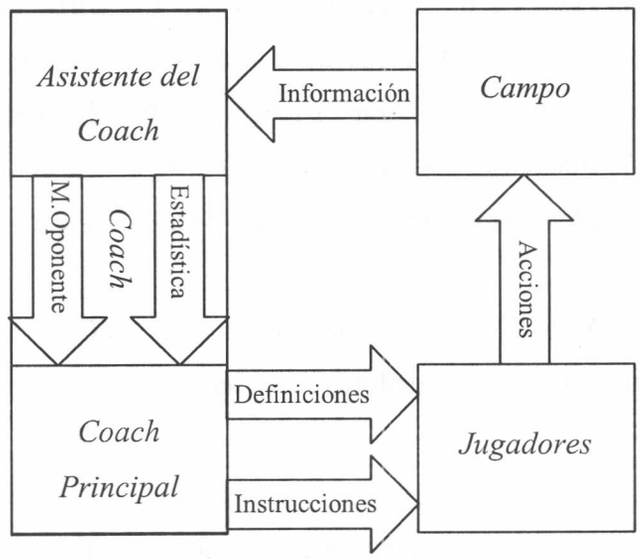


Figura 12.3: Arquitectura de un módulo unificando coach y asistente.

- Un módulo externo que cumple la función de asistente. Este agente brinda los resultados del análisis realizado a los jugadores y cada agente actúa como un *coach* individual.

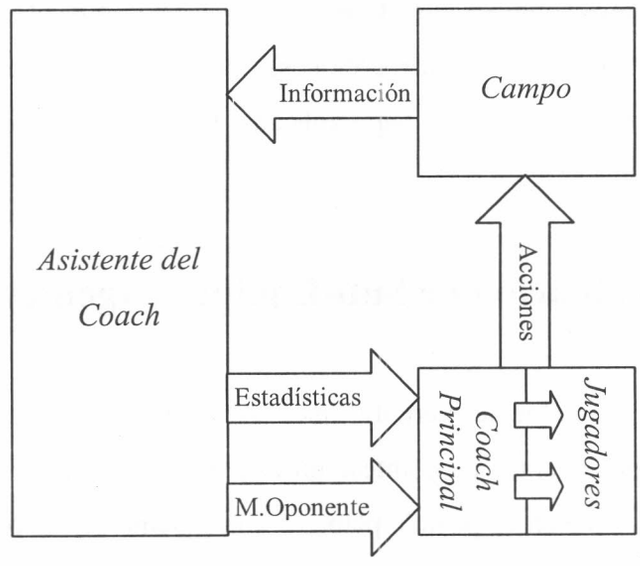


Fig.12.4: Arquitectura de un de asistente.

El equipo FC Portugal participó en la categoría de simulación de RoboCup, tanto en el año 2000 como 2001. En su primera participación eran los jugadores los encargados de analizar el juego y definir la estrategia que debían aplicar. En el año 2001, FC Portugal incorporó la figura de *coach* a su esquema. La arquitectura empleada responde a la segunda de las arquitecturas presentadas.

COACH UNILANG es el lenguaje utilizado por FC Portugal 2001 para la transmisión de mensajes a los jugadores. Este lenguaje cuenta con cuatro tipos de mensajes:

- Mensajes de definición, que le permiten al *coach* informar a los jugadores nuevos conceptos, como ser: regiones, períodos de tiempo, tácticas, formaciones, jugadas especiales o roles.
- Mensajes estadísticos, que posibilitan la distribución de información estadística sobre el juego para ser utilizada en la toma de decisiones por parte de los jugadores.
- Mensajes sobre el modelo del oponente, que le permiten al *coach* informar a los agentes sobre las características del equipo oponente.
- Mensajes instructivos (instrucciones), mediante los cuales el *coach* indica los cambios sobre la estrategia aplicada: la formación utilizada, el ritmo de juego a aplicar, etc.

12.6.2 Coordinación de Sub-Equipos (Agentes Líderes)

Cuando el ambiente permite que los jugadores se puedan comunicar durante el partido, además de plantear un esquema de coordinación con un solo líder (el *coach*), también es posible plantear un esquema de coordinación con sub-equipos y más de un líder. En este segundo esquema, el equipo completo es fragmentado en varios sub-equipos y se designa un agente líder para cada grupo. El líder (o capitán) tiene la responsabilidad de decidir la estrategia adoptada y de

comunicarla a los jugadores que conforman su grupo para que, en conjunto, alcancen los resultados deseados.

Uno de los equipos que, a través de sus sucesivas participaciones en RoboCup, ha presentado este esquema de coordinación es CMUnited (Carnegie Mellon University). CMUnited asigna a cada agente un área de influencia. De esta manera, los agentes tienen una posición dinámica pero controlada, que aporta flexibilidad para adaptarse a los cambios pero que, con una asignación adecuada de áreas, garantiza que el equipo cubra toda la cancha [STO2].

El equipo puede adoptar distintas formaciones y cada formación está compuesta por un conjunto de unidades. Las unidades pueden contener defensores, mediocampistas, delanteros, laterales izquierdos, centrales y derechos. Además cada uno de estos jugadores tiene definida una posición, es decir un área de influencia. Cada unidad posee un capitán.

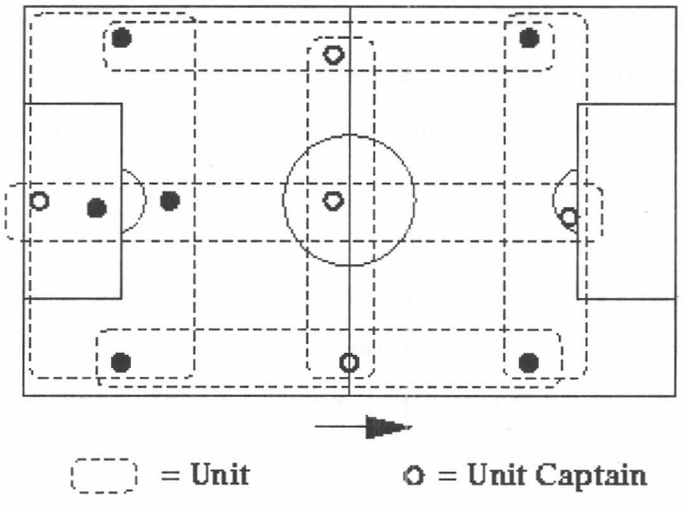


Figura 12.5: Esquema de unidades y capitanes por unidad.

Estas formaciones son conocidas por todos los integrantes y es el capitán quien comunica a los integrantes de su sub-equipo los cambios de formación. Por este motivo, en períodos de autonomía total o baja comunicación, puede suceder

que los jugadores no conozcan la formación que el resto está utilizando sino la formación recibida en el último contacto.

CMUnited cambia su formación considerando las diferencias en el resultado: el equipo pasa a una formación ofensiva si está perdiendo y el final del partido es cercano; y a una formación defensiva si está ganando.

12.6.3 Coordinación Implícita

Cuando el ambiente no cuenta con mecanismos de comunicación explícita entre los jugadores o entre los jugadores y un agente externo, como ser el *coach*, cada jugador debe tomar sus propias decisiones sin recibir directivas ni referencias de las decisiones tomadas por sus compañeros.

Cada jugador puede conocer la estrategia de su equipo en forma parcial o total. En el caso de los equipos que aplican un esquema de roles, cada integrante solo conoce las funcionalidades que le corresponde a su rol. En los equipos donde todos los jugadores cumplen las mismas funciones, cada uno de ellos conoce la estrategia de equipo en forma completa y actúa infiriendo cuales son las acciones aplicadas por sus compañeros.

Los equipos UBASot presentados en el campeonato FIRA 2002 utilizan coordinación implícita. El equipo de robots simulados aplica un esquema con tres roles diferenciados: uno corresponde al arquero y los dos restantes a jugadores de campo. El equipo presentado en la categoría MiroSot aplica un esquema de dos roles: arquero y jugador de campo [SAN].

Parte III
Implementación

En los anteriores capítulos de esta tesis se ha presentado el proyecto de fútbol de robots a nivel mundial, la motivación de la elección de este ámbito para la investigación, sus objetivos, sus distintas componentes, los avances en su desarrollo, los organismos internacionales abocados a fomentar su crecimiento. En síntesis, el *Estado del Arte* de este proyecto.

Para la presentación de este trabajo se ha recopilado gran cantidad de información sobre este tema. Tomando como punto de partida los campeonatos organizados entre el año 1997 y el año 2001, se ha investigado sobre las características de las categorías, las estrategias aplicadas por los equipos, los comportamientos desarrollados y los métodos utilizados. Se han evaluado los resultados obtenidos por los mismos y las ventajas y desventajas de cada uno de estos desarrollos. El análisis de esta información ha sido una de las etapas más ricas de este trabajo.

Finalizada la etapa de análisis y conclusiones sobre el proyecto, la propuesta fue avanzar en el desarrollo de una estrategia de equipo donde se pudiera aplicar el conocimiento adquirido.

En los comienzos se trabajó en una estrategia para la categoría de simulación de RoboCup donde, como se mencionó en capítulos anteriores, los agentes representan a humanos jugando fútbol. Habiendo hecho algunos avances en este desarrollo, se presentó la posibilidad de que la Universidad de Buenos Aires participara en el Campeonato Mundial de Fútbol de Robots FIRA 2002, a través del Grupo de Inteligencia Computacional Aplicada a Robótica Cooperativa

en Sistemas Multirobots del Departamento de Computación de la Facultad de Ciencias Exactas y Naturales. Esta posibilidad motivó que se volcaran todos los esfuerzos en el desarrollo de una estrategia para que un equipo pudiera participar en esta liga, optando por la categoría de simulación *Middle League SimuroSot* (5 vs. 5).

En los dos capítulos que componen esta tercera parte, se describe en forma detallada el ambiente donde se ha trabajado y la implementación que se ha llevado a cabo para alcanzar el equipo de simulación (UBASot) que ha participado en la categoría *Middle League SimuroSot* en el campeonato FIRA 2002, obteniendo el tercer puesto entre 59 equipos participantes de 12 países.

En el capítulo 13 se detalla inicialmente el entorno donde opera el equipo UBASot, las características de este ambiente (el simulador), la interfase de información entre ambos, el esquema de equipo implementado, la estrategia y los comportamientos individuales desarrollados.

El capítulo 14 describe la evolución de esta implementación así como también la experimentación y sus resultados.

Capítulo 13
Implementación del Equipo UBASot 2.10

13.1 Introducción

El equipo de fútbol de robots UBASot 2.10 se ha desarrollado utilizando la plataforma brindada por el simulador de FIRA para la categoría *Middle League SimuroSot*. Las características de este simulador han sido descriptas en el capítulo 5 de este trabajo.

En el ambiente de simulación, de la misma manera que para la categoría *Middle League MiroSot* de robots físicos, el campo de juego es un rectángulo de 220 cm x 180 cm.

En un partido de fútbol de robots simulados se enfrentan dos equipos de cinco robots cada uno. Antes de comenzar el partido, se determina mediante azar el color que identificará a cada equipo (azul o amarillo) durante el juego. El tamaño de cada robot simulado es 7.5 cm x 7.5 cm x 7.5cm. La pelota es color naranja y su diámetro es de 42.7 mm.

Un partido consta de dos períodos de 5 minutos cada uno, con un intervalo de 10 minutos. Un gol se marca cuando la pelota ingresa en el arco pasando en forma completa la línea de arco. El ganador de un partido se determina en base a la cantidad de goles marcados. En caso de terminar un partido empatado (ambos períodos), el ganador se decide por muerte súbita. Después de un descanso de 5 minutos, se reanuda el partido por un período máximo de 3 minutos. El primer

equipo que marque un gol es declarado ganador. Si el empate persiste pasados los 3 minutos de descuento, el ganador se decide mediante tiros penales.

El arco defendido por el equipo de color azul es siempre el ubicado sobre el lado derecho de la cancha. Como consecuencia, el arco sobre el lado izquierdo es el defendido por el equipo amarillo. Esta condición no se modifica con el inicio del segundo tiempo.

El detalle completo del reglamento de la categoría *SimuroSot* se presenta en el Apéndice B.

A continuación se exponen los lineamientos generales seguidos en el proceso de diseño y luego se brindan los detalles de implementación de la versión con la cual se ha competido en el campeonato mundial llevado a cabo en Corea durante el mes de Mayo del año 2002 (UBASot Versión 2.10).

13.2 Lineamientos Generales

En los capítulos previos se han mencionado distintas opciones en el diseño de un equipo de fútbol. Para el equipo implementado se ha optado por trabajar con roles, con formación fija, con un modelado sencillo del movimiento del oponente y sin comunicación explícita entre los jugadores.

Al momento de diseñar y durante todo el proceso de evolución del desarrollo, se ha tenido en cuenta un conjunto de premisas para obtener el estilo de juego que se deseaba respetando la filosofía de implementación asumida. Ambos aspectos se detallan a continuación.

En cuanto al estilo de juego deseado, las características fundamentales son i) lograr una defensa sólida del arco propio, para lo cual el arquero debe ser

preciso y los defensores, eficaces; y ii) atacar el arco rival siempre que sea posible aunque la situación no sea óptima y priorizando velocidad por sobre precisión.

Otra de las características de juego buscada ha sido la de dinamizar al máximo el juego. Para esto se intenta: i) mantener la pelota en movimiento la mayor parte del tiempo, incluso sacrificando precisión, pero siempre tratando de desplazarla hacia el campo rival y alejándola así del arco propio; y ii) mover a los robots a la mayor velocidad posible, aunque esto ocasione, a veces, colisiones o imprecisiones no deseadas.

En cuanto a la disposición de los jugadores en la cancha, se desea que los mismos i) se sitúen en puntos estratégicos de manera de no dejar descubiertas zonas hacia donde la pelota pueda dirigirse de manera inminente, ii) que los jugadores que no participan de la jugada no interfieran negativamente en el accionar de sus compañeros y iii) que se ubique con respecto a la jugada de manera tal de favorecer una futura jugada cooperativa.

Debido a la limitación que presenta la duración del ciclo de simulación, se ha tomado como filosofía de implementación la simplificación al máximo de modelados y cálculos complejos que puedan comprometer el tiempo de respuesta dentro del ciclo de simulación. Por ello, en algunos casos se utilizan aproximaciones o se ignoran ciertos datos irrelevantes. Teniendo en cuenta cuan dinámico es el ambiente del juego, la mayoría de las veces la precisión no mejora el desempeño del equipo (salvo en jugadas críticas de defensa) ya que todos los cálculos y modelos pueden perder vigencia en unos pocos ciclos de simulación, siendo suficiente con trabajar con tendencias aproximadas.

13.3 Arquitectura de la Implementación

La implementación del equipo se divide en las siguientes partes: *Ambiente*, *Control* y *Roles* (Fig.13.1). El módulo de ambiente es el encargado de registrar y suministrar información sobre el estado de cada objeto partícipe del juego, a los demás módulos. El módulo de control es el responsable de proveer las acciones de los jugadores y por último, los roles quienes dan el perfil de juego a cada jugador.

Todos estos módulos dependen de un módulo principal que es el responsable de administrar la comunicación con el simulador, recibiendo y enviándole información, la cual es procesada y producida, por los demás módulos y, además, es el responsable de organizar al equipo según el estado de juego. Esto es, distribuir los roles y realizar designaciones que más adelante se detallan, tanto en el estado de juego normal, como para cada una de las jugadas especiales. A la información recibida del simulador se la denomina *Percepción* y a la información enviada, *Comandos*.

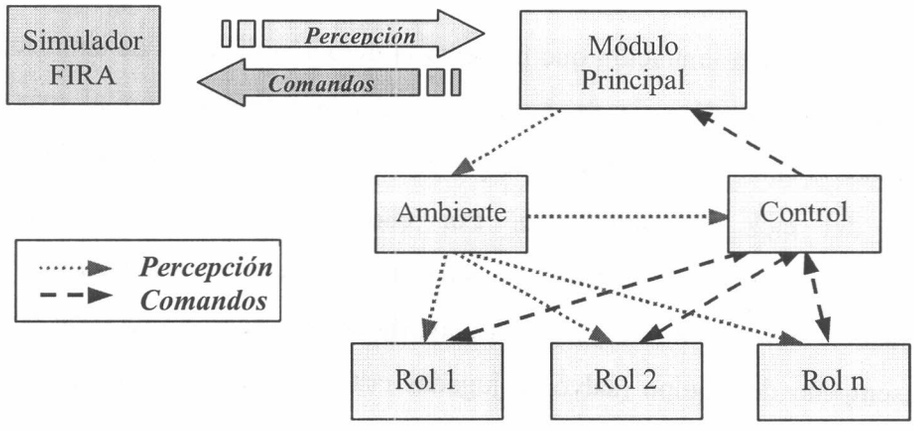


Figura 13.1: Arquitectura de la Implementación.
Intercambio de información entre los módulos relacionada a percepción y comandos. Se puede observar intercambio de comandos entre control y los roles ya que ciertas funciones de control son directamente quienes determinan los comandos a enviar.

Cada robot, conceptualmente, decide y actúa en base a su rol. Sin embargo, ciertos datos son calculados en forma global a los efectos de evitar la

replicación de cálculos. Esta información es calculada al recibirse la percepción, en forma previa a que se ejecute el rol de cada robot. Dichos roles, directamente, consultan sobre las condiciones calculadas en vez de calcularlas nuevamente. Esta información la almacena, para cada robot, el módulo de ambiente. Los datos precalculados en forma global son:

- Condición de atascado.
- Condición de estar llevando la pelota.
- Condición de tener que ir a la pelota.

Las dos primeras condiciones se calculan al recibir la percepción. En el caso de atascado, se calcula si la distancia recorrida en los últimos tres ciclos es menor al mínimo que debería haberse registrado según las potencias asignadas a las ruedas. En caso positivo, se aumenta un contador que lleva la cuenta de los ciclos consecutivos en los cuales se registra esta situación. Cuando la distancia recorrida es mayor a un mínimo, se inicializa este contador.

Para detectar si el robot está llevando la pelota, se calcula en cada ciclo si la pelota está muy próxima al robot (distancia entre robot y pelota menor a $AnchoRobot/2 + 2 \times AnchoPelota$). Cada vez que esto sucede, se incrementa un contador. Cuando la pelota se aleja, el contador es llevado a cero.

La decisión de quien es el más apto para ir a la pelota se toma también en forma global pero según el estado de juego ya que hay jugadas especiales que requieren la intervención de ciertos jugadores en particular. Sin embargo, en la mayoría de las jugadas especiales y, obviamente, durante el juego propiamente dicho, la decisión se toma en base a cuan bien posicionado está el jugador con respecto a la pelota. Como el módulo principal es el que organiza las jugadas, valiéndose de servicios brindados por el ambiente, determina cuales son los jugadores elegidos para acudir a la pelota. Esta información es administrada por el módulo de control. Cabe destacar que, si bien se ha centralizado esta decisión, los

jugadores son agentes autónomos que podrían observar en forma independiente las condiciones del ambiente y decidir si les corresponde o no, ir a la pelota.

En la presente implementación, para determinar el mejor posicionado, básicamente se considera la distancia lineal entre los jugadores y la pelota. Por este motivo el jugador elegido es, simplemente, el que está más cerca de la pelota, aunque con ciertas restricciones en los candidatos. De esta selección se excluye a los siguientes jugadores:

- ***Arquero:*** no puede abandonar el arco para ir por la pelota y por otro lado, sea quien sea el seleccionado, si la pelota representa peligro, el arquero acude a ella para evitar que el rival convierta gol.
- ***Atascados:*** se excluye de la selección a estos jugadores para evitar que se elija a un jugador que está momentáneamente imposibilitado de acudir a la pelota dejando así que vaya otro a pesar de estar más lejos, pero que está con libertad de movimiento.
- ***Marcados:*** cuando un jugador está en contacto físico (distancia entre robots < *(DiagonalRobot + AnchoRobot)/2*) con uno o varios rivales y alguno de ellos está más cerca de la pelota, se está frente a un jugador marcado. Se elige a otro que, aunque más lejano, pueda tener más oportunidad de llegar a la pelota.

En el caso particular detallado más adelante donde se quiera elegir un segundo jugador para acudir a la pelota, éste es directamente seleccionado por distancia lineal sólo excluyéndose al arquero y al primer seleccionado.

13.4 Ambiente

Como se mencionó previamente, el módulo de ambiente es el encargado de registrar y suministrar información sobre el estado de cada objeto partícipe del

juego; es decir jugadores y pelota. El módulo no sólo almacena el estado actual, sino que también un historial de los últimos estados de manera de poder realizar predicciones en base a la tendencia de los objetos. Esta historia también es útil para modelar el comportamiento del equipo rival, en base al análisis de comportamientos pasados.

La información que el módulo almacena es la siguiente:

- Para cada robot (propios y rivales):
    - Posición (X, Y)
    - Orientación
    - Potencias de las ruedas (comandos enviados)
    - Condición de bloqueado
    - Condición de llevar la pelota.
- Para la pelota:
    - Posición (X, Y)
    - Dirección (Indica si la pelota se dirige hacia el arco propio o contrario)
- Estado de Juego (*GameState*)
- Equipo propietario de la pelota en jugadas especiales (*WhosBall*)

El módulo brinda también todos los servicios necesarios para poder consultar y actualizar esta información.

13.5 Control

Este módulo contiene la implementación de las acciones de los jugadores, así como también, el registro de atributos de juego de cada robot. Las acciones que implementa son todas aquellas relacionadas a los métodos de pateo, de navegación y de movimientos en general del robot. También en este módulo se

mantienen las potencias de las ruedas que se han determinado en el presente ciclo de simulación y que son enviadas, vía el módulo principal, al simulador.

En cuanto a los atributos de juegon el módulo de control mantiene para cada robot la siguiente información:

- Rol
- Condición de elegido
- Posición base del robot

El *Rol* describe cual es el perfil de juego que desarrolla el robot y, por lo tanto, cual de los módulos de rol controla el comportamiento del robot en cuestión. La *Condición de Elegido*, como se mencionó con anterioridad, es la marca que en base a una decisión global, indica si los jugadores deben acudir en busca de la pelota. Por último, la *Posición Base* es el lugar dentro de la cancha al cual los robots acuden cuando no tienen una participación activa en la jugada. Es decir, es su posición por defecto durante el transcurso normal del juego.

13.6 Roles

El esquema implementado en el equipo consta de tres roles: *Arquero, Defensor* y *Delantero*. La asignación de roles es estática, por lo tanto cada robot mantiene su rol durante todo el partido sin importar que, circunstancialmente, se encuentre en mejores condiciones para cubrir otro rol. El hecho antes mencionado redunda en que también la formación del equipo es fija. La formación utilizada es un arquero, dos defensores y dos delanteros.

Los módulos de roles utilizan los servicios ofrecidos tanto por el módulo de ambiente como de control y son quienes, en definitiva, determinan el accionar de cada robot dentro de la cancha. A continuación, se detallan en profundidad

cada uno de los tres roles implementados en el equipo.

Nro. Robot	Rol Asignado	Posiciones Base			
		Equipo Azul		Equipo Amarillo	
		X	Y	X	Y
1	Arquero	FrightX – (AR + AP) / 2	Centro Arco.Y	FleftX + (AR + AP) / 2	Centro Arco.Y
2	Defensor	FrightX – 20	FbotY + 25	FleftX + 20	FtopY – 25
3	Defensor	FrightX – 20	FtopY – 25	FleftX + 20	FbotY + 25
4	Delantero	FleftX + 20	FbotY + 20	FrightX - 20	FtopY – 20
5	Delantero	FleftX + 20	FtopY – 20	FrightX - 20	FbotY + 20

Tabla.13.1: Roles y posiciones base.
Las posiciones base se establecen relativas a las coordenadas de la cancha. FrightX, FleftX, FtopY y FbotY son los límites derecho, izquierdo, superior e inferior de la cancha, respectivamente. AR y AP hacen referencia al ancho de los robots y al ancho de la pelota, respectivamente.

13.6.1 Rol Arquero

Dentro de un equipo de fútbol, el puesto de arquero es crítico debido a que es quien debe evitar que la pelota ingrese al arco cuando ésta no ha podido ser controlada por los defensores propios.

Si bien las reglas de la categoría habilitan a cualquier miembro del equipo a ocupar el área chica para atajar, siempre y cuando no haya más de un jugador del equipo dentro de la misma, se optó por diseñar un rol específico de arquero y se lo asignó a un jugador en forma permanente.

Debido a la imposibilidad que tienen los robots simulados de desplazarse en forma lateral, en el rol del arquero se hace que el jugador tenga un desplazamiento paralelo al arco, en un recorrido que cubre el ancho del área chica. De esta manera, el arquero rápidamente puede avanzar o retroceder pudiendo así interceptar la pelota, la cual impactará sobre su lateral. Para conservar esta línea de desplazamiento del arquero, se debe cuidar la posición del robot en la coordenada X y la orientación del mismo.

La coordenada X que se tratará de mantener debe ser lo suficientemente alejada del arco de manera que el cuerpo del arquero no colisione con los postes del arco (es decir, los rincones formados por las paredes de la línea del fondo de la cancha y las paredes laterales internas del arco), pero a su vez lo suficientemente cercana como para que no pase la pelota entre el arquero y la pared de fondo. En cuanto a la orientación, el ángulo a conservar debe ser lo más cercano posible a 90° o –90° para conservar un recorrido paralelo al arco.

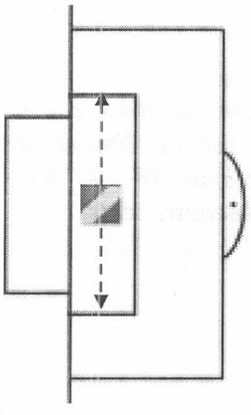


Figura 13.2: Arquero
Trayectoria en la que el arquero se desplaza para atajar.

La acción de atajar propiamente dicha se lleva a cabo mediante el avance o retroceso del robot sobre la línea antes mencionada. La idea implementada es que el robot tienda a ubicarse, sobre su recorrido, en el punto en el cual se espera que la pelota intercepte a la recta determinada por su lateral defensivo (cara externa del robot que se encuentra del lado opuesto al arco). Para esto, en cada ciclo de simulación, se calcula en base al movimiento actual de la pelota si en un determinado número de ciclos la pelota interceptará la línea de defensa mencionada. En base a este cálculo se pueden determinar los siguientes casos:

- La pelota no interceptará la línea, ya sea por su recorrido o porque la cantidad de ciclos requeridos para que esto ocurra excede los analizados.
- La pelota interceptará a la línea en un punto por fuera de los límites del arco.

- La pelota interceptará a la línea en un punto dentro de los límites del arco.

En el primer caso, no se toma ninguna acción destinada a defender el arco. Más adelante se describen las acciones tendientes a posicionamiento que se realizan en estos momentos en los cuales no hay peligro para el arco. La cantidad de ciclos analizados a futuro se estableció en la cantidad de ciclos que la pelota requiere para recorrer media cancha a la velocidad máxima calculada por observación.

El segundo caso se puede dividir en dos situaciones dependiendo del tiempo que resta para que, en base al movimiento actual de la pelota, se produzca la intercepción entre la pelota y la línea de defensa. Si bien en la circunstancia actual no habría peligro directo de que la pelota ingrese al arco, si la pelota llegara en forma inminente a la línea de defensa, significa que la jugada se está desarrollando sobre un lateral del arco y, por lo tanto, es probable que la pelota pueda ser impulsada por un rival hacia el arco en dirección paralela al mismo. En este caso se procede con la acción de posicionarse para atajar, con lo cual el arquero tiende a ubicarse sobre el extremo de su recorrido del lado en el que la pelota se proyecta. La segunda situación se presenta si el momento en el que se produce la intercepción es lejano (empíricamente se estableció este tiempo en la cantidad de ciclos de simulación que tarda la pelota en recorrer un cuarto de cancha a velocidad máxima). Aquí, la pelota puede todavía cambiar su trayectoria por lo cual el arquero permanece en su posición base.

El tercer caso resulta el más crítico ya que, sin la intervención del arquero, el rival convierte un tanto en forma segura. En este caso se calcula el punto intermedio entre la posición proyectada de la pelota sobre la línea de defensa y la posición actual del robot. A este punto se envía al robot, de manera tal que el mismo tiende a ubicarse exactamente en el punto de intercepción. Se optó por elegir este punto intermedio en vez del punto exacto de contacto para evitar bruscos movimientos del arquero que, ante cambios abruptos de dirección de la

pelota, lo dejen imposibilitado de llegar a tiempo al nuevo punto de intercepción. Si la trayectoria de la pelota se mantiene, el robot se mueve hacia el punto intermedio antes mencionado, con lo cual la próxima vez que se calcule dicho punto, estará más cerca del punto donde llegará la pelota y así sucesivamente hasta quedar exactamente parado en ese punto.

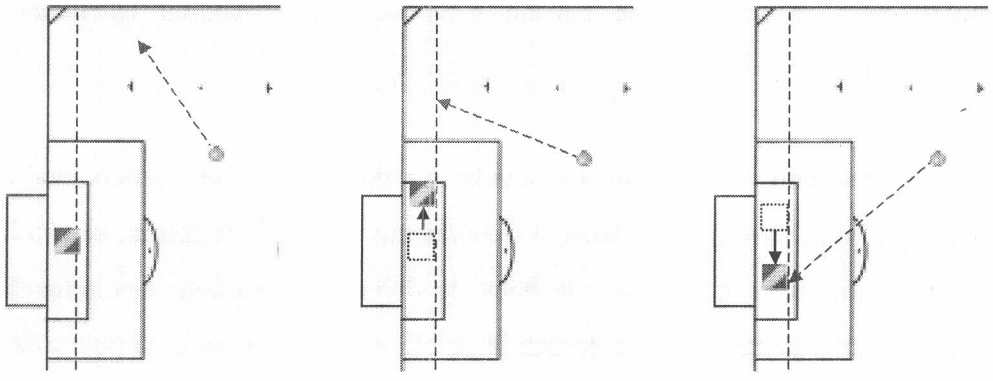


Figura 13.3: Casos de proyección de pelota.
(a) La trayectoria de la pelota no intercepta la línea de defensa o la cantidad de pasos excede los analizados (b) la pelota intercepta la línea de defensa fuera de los límites del arco (c) la trayectoria de la pelota intercepta la línea de defensa del arquero dentro de los límites del arco.

Un caso particular que ha requerido un tratamiento especial es aquel en el que la pelota se desplaza hacia el arco en contacto con alguna de las paredes de fondo. De utilizarse el comportamiento general antes expuesto, el arquero tiende a ir hacia el centro del arco ya que hacia allí se proyecta la pelota a futuro. Esta acción deja a la pelota desplazarse libremente frente al arco con un alto riesgo de que se convierta un gol por más que el movimiento actual es paralelo al mismo, ya que un jugador rival o simplemente un rebote, pueden hacer que se desvíe la pelota hacia el interior del arco. Para tratar este caso particular, al detectarse la situación planteada, se ubica al arquero por fuera del arco y en contacto con la pared de manera de evitar que la pelota pueda llegar al arco.

```
Rol_Arquero()
Comenzar
Si estoy en contacto con la Pelota Entonces
   Si tengo la pelota sobre un lateral de mi cuerpo Entonces
      Si pelota de lado exterior Entonces
         Hacer Pateo Lateral para alejar la pelota
      FinSi
   Sino
      Si estoy fuera del Area Grande Entonces
         Hacer Pateo Lateral girando de manera tal que la pelota sea impulsada hacia
         el interior de la cancha.
      Sino
         Empujar la pelota hacia la línea lateral con fuerza levemente dispar de
         manera que la pelota tienda a ir hacia el interior de la cancha.
      FinSi
   FinSi
Sino
   Calcular si la pelota llegaría a la línea paralela al arco, donde se ubica el arquero,
   en base a su desplazamiento actual.
   Si la pelota puede llegar a mediano plazo y se proyecta dentro del arco, o bien,
   puede llegar en un corto plazo Entonces
      Si no estoy demasiado desfasado en cuanto a orientación Entonces
         Posicionarse_para_Atajar()
      Sino
         Rotar para quedar paralelo al arco
      FinSi
   Sino
      Si estoy desfasado de mi línea de defensa Entonces
         Si la pelota se proyecta dentro del arco Entonces
            Ir a Posición donde pelota interceptará mi línea de defensa
         Sino
            Ir hacia mi línea de defensa levemente desplazado hacia el lado donde se
            proyecta la pelota.
         FinSi
      Sino
         Si no estoy demasiado desfasado en cuanto a orientación Entonces
            Posicionarse_para_Atajar()
         Sino
            Rotar para quedar paralelo al arco
         FinSi
      FinSi
   FinSi
FinSi
Fin
```

Figura 13.4: Pseudocódigo del rol arquero

Una vez que el arquero ha interceptado con éxito la pelota, ya que no tiene forma de retenerla, aparece el problema de qué hacer con ella. En el caso general

en el cual la pelota viene con dirección perpendicular al arco, la pelota hace contacto con un lateral del robot. De no tomar acción alguna, la pelota queda allí corriéndose el riesgo de que un rival la pueda impulsar al interior del arco. Tampoco en esta situación se puede contar con los defensores ya que, según las reglas de la categoría, no puede haber más de un jugador dentro del área chica.

Para sacar de esta posición de peligro a la pelota, si se detecta que ésta se encuentra en contacto con el lateral del robot que da hacia el centro de la cancha, el arquero hace un giro sobre sí mismo lo que impulsará a la pelota hacia el interior de la cancha posibilitando así que los defensores del equipo puedan alcanzarla en forma segura.

En el caso particular de la pelota desplazándose por la pared de fondo, la pelota hace contacto con la parte delantera o trasera del robot. En este caso, se avanza o retrocede de manera de alejar a la pelota del arco. Cuando se alejó a la misma lo suficiente (fuera del área grande) se intenta realizar un giro en dirección al interior de la cancha de manera de despegar a la pelota de la pared. Sólo se hace esto fuera del área grande para evitar el peligro que representaría que la pelota quede liberada cerca del frente del arco posibilitando que otro jugador rival la impulse convirtiendo un gol.

Durante el período en el que se empuja a la pelota, esto se hace con fuerza levemente dispar de manera que el robot tienda a impulsar a la pelota hacia el interior de la cancha. Este último movimiento es particularmente útil cuando un jugador rival se encuentra empujando a la pelota hacia el arco y el arquero la empuja del lado contrario. Con el movimiento de giro, se intenta desequilibrar esta puja entre arquero y rival al hacer que la pelota se zafe de la presión de ambos jugadores saliendo impulsada hacia el interior de la cancha.

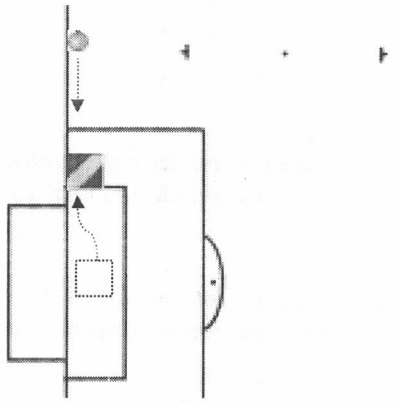


Figura 13.5: Comportamiento particular.
Con el comportamiento general, el arquero retrocedería hacia donde se proyecta la pelota, en vez de avanzar para impedir su avance.

Como se explicó anteriormente, el arquero realiza su tarea dentro de un recorrido paralelo al arco, y entre los límites del mismo, salvo casos excepcionales. Si bien por su propia voluntad no sale de esta posición, existen factores externos, en particular los demás robots, que pueden causar que el arquero sea desplazado ya sea en posición como en orientación.

Cuando existe una desviación en la orientación, en los casos que no se está frente a una situación crítica, se gira levemente de manera de quedar nuevamente más o menos en 90° o –90°. Algo similar sucede cuando se está alejado de la línea de defensa natural del arquero que es a una distancia no mayor a una pelota del frente del arco.

Si se está en una situación peligrosa, no importa donde el arquero se encuentre, intenta atajar. Caso contrario, se invoca al comportamiento de navegación para ubicarlo nuevamente en su línea de defensa en un punto levemente desplazado desde el centro hacia el lateral donde se encuentre la pelota, si ésta no se proyecta dentro del arco. Si fuese así, el punto de destino es la intercepción de la línea de defensa y la proyección de la pelota.

```
Posicionarse_para_Atajar()
Comenzar
Si la pelota está por detrás de la línea de mi lateral, contra la pared Entonces
    Posicionarse pegado al poste del lado donde se encuentra la pelota de manera de
    esperar la pelota.
Sino
    Avanzar en línea recta hacia el punto intermedio entre mi Posición actual y Posición
    proyectada de la pelota al llegar a mi línea de desplazamiento sin salir del arco.
FinSi
Fin
```

Figura 13.6: Pseudocódigo de rutina de posicionamiento del arquero

13.6.2 Rol Defensor

Los defensores son aquellos jugadores que intentan evitar que la pelota llegue al arco propio cuando la misma está en poder del adversario. En el equipo implementado, se optó por un formación en la que dos jugadores cubren el rol de defensores en forma permanente. En general, uno de estos dos jugadores será el defensor principal y el otro, el defensor secundario.

La tarea de defensa puede ser llevada en forma activa o pasiva (Fig.13.7). Se considera activa la defensa cuando el jugador va al encuentro de la pelota, ya sea para apoderarse de ella ejecutando acciones tendientes a alejarla del arco propio y, de ser posible, acercándola al arco rival (*Defensa Activa de Despeje*: DAD) o bien cuando intenta obstruir su paso yendo a un punto sobre la trayectoria actual de la pelota para que no siga avanzando en campo propio (*Defensa Activa de Encuentro*: DAE).

Es una defensa pasiva la que lleva a cabo un jugador cuando su comportamiento tiende a cubrir una zona con su presencia. La estrategia de defensa pasiva también presenta dos posibilidades en la implementación actual: una es ubicarse sobre la línea que va desde la posición actual de la pelota y el

centro del arco, evitando así dejar libre el camino hacia el arco, sin importar hacia donde se mueve actualmente la pelota (*Defensa Pasiva en Trayectoria*: DPT); la otra acción posible es ubicarse frente al arco, ya sea para cubrir el centro o un lateral. Aquí tampoco se tiene en cuenta donde está la pelota, ya que ésta esta siendo cubierta, supuestamente, por el otro defensor. De esta manera se trata de dejar menos espacio libre frente al arco, ya que un pase o rebote puede hacer que la pelota cambie bruscamente de dirección y quedar así el defensor principal mal ubicado (*Defensa Pasiva en Arco*: DPA).

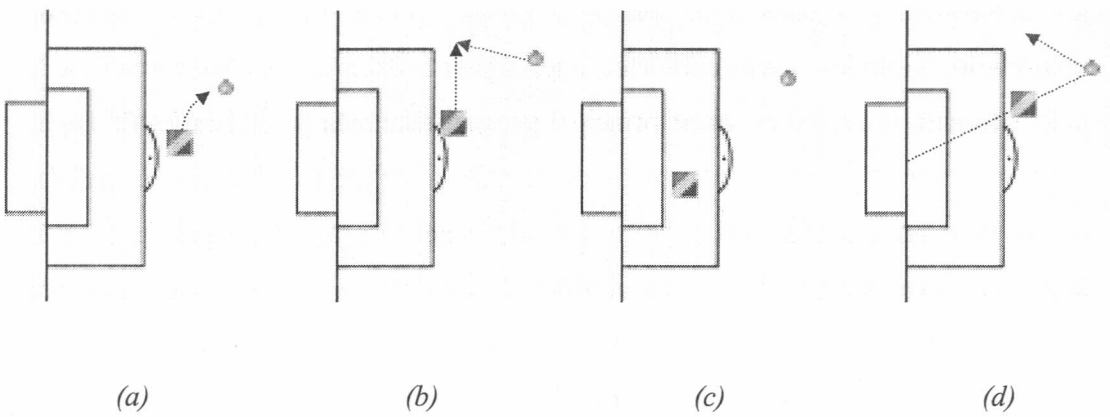


Figura 13.7: Estrategias de defensa.
(a) Defensa activa de despeje (DAD) (b) Defensa activa de encuentro (DAE) (c) Defensa pasiva en arco (DPA) (c) Defensa pasiva en trayectoria (DPT).

A diferencia del arquero, donde un único jugador cubre el rol y por lo tanto éste es el responsable de todas las acciones que el mismo requiere, al haber dos defensores sus tareas deben ser complementarias y sus acciones coordinadas. Por ello, cada uno debe analizar que parte de la defensa le toca llevar a cabo. El caso más sencillo es aquel en el que la pelota está en poder del equipo propio y en ataque. Aquí ambos defensores ejercen una defensa pasiva posicionándose a ambos lados del frente del arco (DPA) por fuera del área grande. Cuando la pelota está dentro del campo propio o viene hacia él, ya los defensores deben tomar acciones acordes a la circunstancia.

En el caso que uno de los defensores resulta el mejor ubicado para llegar a la pelota, lleva a cabo un comportamiento llamado *Despejar Pelota* convirtiéndose así en el defensor principal. El segundo defensor deberá ejecutar el comportamiento de *Cubrir Arco* convirtiéndose así en el defensor secundario.

Cuando ninguno de los dos defensores es el responsable de ir por la pelota, ambos defensores desarrollan el comportamiento de *Cubrir Arco*, siendo aquí más compleja la decisión de cual será el defensor principal y cual el secundario. En este caso se toman en cuenta las siguientes condiciones: si uno de los defensores está más lejos del arco propio que la pelota, es el defensor secundario. Si ambos están más lejos que la pelota del arco, el más cercano a la pelota tomará el rol de defensor principal ya que tiene más posibilidades de llegar a interceptarla. Si ambos defensores están más cerca de su arco que la pelota, entonces se analiza cual está sobre el lateral al cual se proyecta la pelota. Si sólo uno está sobre este lateral, esto da el orden de los defensores. En el caso que, por alguna circunstancia, ambos están sobre el mismo lateral es defensor principal aquel que esté más cerca del punto de intercepción entre su trayectoria (en dirección paralela al arco) y la trayectoria actual de la pelota.

Cuando el defensor tiene que acudir a la pelota como representante del equipo, se pueden dar distintas situaciones. Las posibilidades contempladas en la implementación tienen en cuenta los siguientes factores: cercanía de rivales a la pelota y posición relativa del defensor con respecto a la pelota. El caso más simple es aquel en el que no hay rivales cerca y por lo tanto se cuenta con tiempo suficiente para controlar de la mejor manera posible la pelota. Para hacer esto se utiliza el comportamiento de patear la pelota al arco, el cual hace que el defensor lleve la pelota hacia el arco rival, o al menos la deje mejor posicionada para este fin. Esta jugada es una DAD ideal.

```
Rol_Defensor()
Comenzar
Si Estoy Bloqueado y (no tengo la pelota o estoy fuera del radio del área grande)
Entonces
    Girar sobre si mismo para intentar desbloquearse
Sino
    Si Estoy dentro del Arco Entonces
        Ir en dirección al medio de la cancha
    Sino
        Si no pude hacer pateo lateral Entonces
            Si Pelota en campo propio o se mueve hacia mi campo o yo la vengo llevando
        Entonces
                Si soy el elegido Entonces
                    Despejar_Pelota()
                Sino
                    Cubrir_Arco()
                FinSi
            Sino
                Si Estoy en mi posición base Entonces
                    Girar para quedar paralelo al arco
                Sino
                    Ir a mi posición base
                FinSi
            FinSi
        FinSi
    FinSi
FinSi
Fin
```

Figura 13.8: Pseudocódigo del rol defensor.

Un caso más complejo y más habitual durante el juego, es cuando el defensor debe acudir a la pelota y hay rivales cerca. Aquí no puede perder tiempo en conseguir una buena ubicación. En este caso lo que se trata de hacer es alejar la pelota de la zona de peligro para el arco propio. Si el defensor está más abajo que la pelota, es decir más cerca del arco propio en coordenada *X,* o la pelota superó la línea de defensa propio en cordenada *Y* pero por fuera del ancho del arco (cercano a las paredes laterales de la cancha), entonces su tarea es ir directamente hacia el punto donde está la pelota intentando colisionarla y así alejarla del arco propio, ambos casos son variantes de defensa activa de despeje (DAD).

```
Despejar_Pelota()
Comenzar
Si hay algún rival cerca de la pelota Entonces
   Si estoy detrás de la pelota o bien la pelota está más hacia los laterales que yo y
   fuera del ancho del arco más un jugador Entonces
      Ir hacia la Pelota
   Sino
      Si la distancia entre el centro del arco y la pelota es mayor al radio del área
      chica más dos jugadores Entonces
         Ir hacia el punto (X intermedio entre arco y pelota, Y de la pelota)
         Si el punto anterior cae dentro del area chica Entonces
            Reajustar la coordenada X, llevándola fuera del área chica
         FinSi
      Sino
         Ir a la Posición base de manera de no molestar al arquero
      FinSi
   FinSi
Sino
   Patear_al_Arco()
FinSi
Fin
```

Figura 13.9: Pseudocódigo de rutina para despejar pelota.

Cuando ninguna de las situaciones antes mencionadas se presentan, significa que hay una situación de alto riesgo ya que, si el defensor en cuestión es el mejor ubicado para ir a la pelota y no se cumplieron las condiciones anteriores, o bien la pelota está muy cerca del arco o todos los jugadores del equipo propio están muy mal ubicados. En este caso lo que se intenta hacer es ir a un punto intermedio entre la pelota y el arco propio, siempre y cuando haya lugar para tomar esta ubicación sin entrar en el área chica, lo cual constituye una defensa pasiva en trayectoria (DPT). Si esta estrategia tampoco se puede ejecutar, el defensor vuelve a su posición dejando ya la tarea de defensa en manos del arquero evitando así entorpecer su tarea y que, eventualmente, sea cobrado un penal por la presencia de más de un jugador en el área chica o que genere un rebote que descoloque al arquero y termine en gol. Aquí el defensor principal se ve forzado por la situación a ejecutar una defensa pasiva en arco (DPA).

Cuando, una vez determinado el defensor principal y el defensor secundario según se detalló con anterioridad, tanto el uno como el otro deben

cumplir con el comportamiento *Cubrir Arco*, las acciones ejecutadas son las siguientes:

- ***Defensor Principal:*** si la pelota y él mismo están por delante del área grande propia, se ejecuta el comportamiento DAE. Caso contrario, si el jugador está más cerca del arco que la pelota, ejecuta DAD. Si no, se queda donde está sin ejecutar acción alguna.
- ***Defensor Secundario:*** ejecuta DPA, decidiendo el lugar donde se sitúa en base a donde está el defensor principal. Si éste último se encuentra fuera del radio del área grande, el defensor secundario se ubica frente al arco y en el centro del mismo. Caso contrario, se ubica desplazado hacia el lado de la cancha contrario al que se encuentra el defensor principal.

En cualquiera de los casos, cuando un defensor ya está en el lugar que le corresponde ocupar por el estado de juego actual, trata de orientarse paralelo al arco, 90° o –90°, de manera que sea rápida la ejecución de la acción DAE, la cual es muy similar al comportamiento que implementa el arquero salvo que a lo ancho de toda la cancha. Otra diferencia en comportamiento con el arquero es que, generalmente, la acción de defensa activa de encuentro desemboca en una defensa activa de despeje ya que, al acercarse la pelota, el defensor pasa a ser el mejor ubicado para controlar la pelota.

Otros comportamientos que poseen los defensores y que no están directamente relacionados con la tarea de defender son los siguientes:

- ***Desbloquearse:*** cuando se detecta que el jugador está bloqueado, es decir que en varios ciclos consecutivos se le ha dado velocidad a las ruedas y sin embargo no se obtuvo desplazamiento, el jugador intenta girar sobre si mismo para poder destrabarse. Esto lo hace sólo si no tiene en su poder la pelota o sí la tiene pero se está fuera del área grande. Se implementan estas

restricciones para evitar perder el control de la pelota estando muy cerca del arco propio.

- ***Salir del Arco Propio:*** dado que desarrollan su tarea muy cerca del arco muchas veces, ya sea por haber sido empujados o porque el algoritmo de navegación al evitar obstáculos los llevó por allí, terminan entrando al arco propio. Cuando se detecta esta situación se activa un comportamiento que los envía con dirección hacia el centro de la cancha. Dicho comportamiento se desactiva cuando ya han salido del arco restableciéndose el desempeño normal del defensor.
- ***Pateo Lateral:*** cuando se está en contacto con la pelota bajo las condiciones de pateo lateral, se ejecuta el mismo.

```
Cubrir_Arco()
Comenzar
Si soy el único defensor o el otro defensor es el elegido pero la pelota está más cerca
del arco propio que él Entonces
   Debo Cubrir
Sino
   Si la pelota está más cerca de mi arco que el otro defensor Entonces
      Si yo también estoy más lejos del arco que la pelota Entonces
         Si estoy más cerca de la pelota que el otro defensor Entonces
            Debo Cubrir
         FinSi
      Sino
         Debo Cubrir
      FinSi
   Sino
      Si yo también estoy más cerca del arco que la pelota Entonces
         Si la pelota se proyecta hacia mi lado de la cancha Entonces
            Si el otro defensor también está de mi lado de la cancha Entonces
               Si estoy más cerca que el otro defensor del punto donde la pelota
               interceptará mi línea de defensa que él del punto donde llegará a su
               línea de defensa Entonces
                  Debo Cubrir
               FinSi
            Sino
               Debo Cubir
            FinSi
         Sino
            Si el otro defensor no está del otro lado de la cancha Entonces
               Si estoy más cerca que el otro defensor del punto donde la pelota
               interceptará mi línea de defensa que él del punto donde llegará a su
               línea de defensa Entonces
```

```
                    Debo Cubrir
                FinSi
            FinSi
        FinSi
    FinSi
FinSi
Si Debo Cubrir Entonces
    Si estoy más cerca del arco que la pelota Entonces
        Si la pelota y yo estamos más arriba del área grande Entonces
            Ir hacia el punto de intercepción entre mi línea de defensa y la trayectoria de
        la pelota.
        Sino
            Ir hacia un punto intermedio en la línea entre la pelota y el arco con un límite
        máximo de alejamiento del arco.
        FinSi
    Sino
        Quedarse en el mismo lugar
    FinSi
    Si el punto destino está dentro del área chico Entonces
        Corregirlo llevándolo fuera del área chica
    FinSi
Sino
    Si el primer defensor está más allá del radio del área grande Entonces
        Ir hacia un punto a dos cuerpos del área chica a la altura del centro del arco
    Sino
        Ir hacia un punto a dos cuerpos del área chica y desplazado dos cuerpos del
        centro de arco hacia el lado contrario al que se encuentra el otro defensor.
    FinSi
    Si ya llegué a mi destino Entonces
        Girar de manera de quedar paralelo al arco
    FinSi
FinSi
Fin
```

Figura 13.10: Pseudocódigo de rutina para cubrir arco.

13.6.3 Rol Delantero

Los jugadores que tienen a cargo el rol de delanteros son los responsables de convertir goles. En el equipo implementado, se le asignó a dos robots el rol de delanteros y, normalmente, se sitúan en campo rival. Al igual que en el caso de los defensores, en general existe un delantero principal y uno secundario.

El delantero principal es, en general, aquel que por su ubicación es el mejor posicionado para acudir a la pelota y ejecutar el comportamiento de *Patear al Arco*, mediante el cual intenta lograr su objetivo de hacer que la pelota ingrese al arco rival.

Cuando algún defensor es quien tiene la pelota o está mejor ubicado para obtenerla, uno de los delanteros es el delantero principal ubicándose de manera tal de poder recibir la pelota y el delantero secundario queda fuera de la jugada volviendo a su posición.

Dado que el delantero no está capacitado para la defensa, no es conveniente dejar el control de la pelota a un delantero cuando la misma está cerca del arco propio. Por esto, cuando un jugador con rol delantero es el mejor ubicado para ir a la pelota pero se está relativamente cerca del arco propio, en vez de ir a ella comienza a volver a su posición de manera de dejar en libertad de acción a los defensores los cuales poseen comportamientos especiales para la defensa.

Otra situación en la que el delantero mejor posicionado no acude de inmediato a la pelota es cuando detecta que quedándose donde está, pronto tendrá una mejor posición con respecto a la pelota y el arco. Lo que analiza para esperar a la pelota es lo siguiente:

- La pelota ya está relativamente cerca de él.
- Por algunos ciclos más, él está más cerca del arco rival que la pelota.
- Dentro de algunos ciclos más, la pelota estará más cerca de él y más cerca del arco de lo que está en la actualidad.

Mientras se queda esperando la pelota, el delantero gira de manera de quedar mirando al arco rival lo cual, posiblemente, le sea útil cuando deba ir hacia allí con la pelota cuando ésta llegue a él.

Cuando el delantero principal es el responsable de llevar o acudir a la pelota, el delantero secundario cumple una tarea de acompañamiento, manteniendo cierta distancia. Cuando el delantero secundario está muy alejado de su compañero en cuanto a coordenada *X*, se dirige a una posición ligeramente retrasada con respecto a la pelota (actualmente, el equivalente al ancho de dos robots) y, en cuanto a coordenada *Y*, desplazado hacia el eje central de la cancha a la distancia definida a tal fin (establecida en estos momentos en el equivalente a un cuarto del ancho de la cancha).

El desplazamiento en coordenada *X* es controlado de manera tal que no se acerque demasiado a la línea de fondo del campo rival, quedando así sin ángulo hacia el arco, ni tampoco que se retrace demasiado en área propio entorpeciendo, eventualmente, la tarea de los defensores.

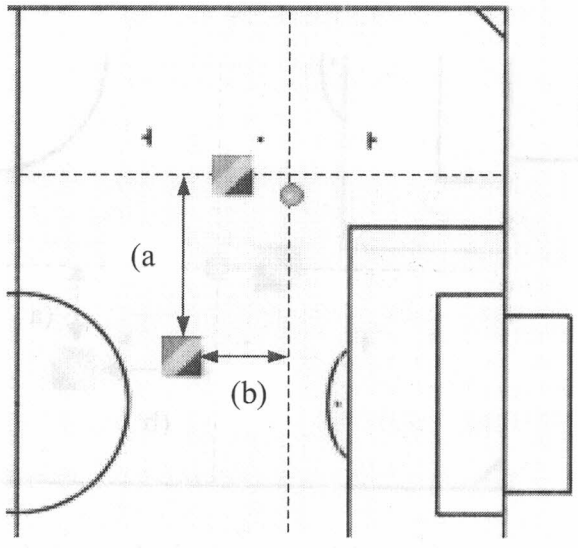


Figura 13.11: Delantero secundario.
Posición de acompañamiento que toma el delantero secundario. (a) Desplazamiento desde la coordenada Y del delantero principal (actualmente establecida en un cuarto de cancha). (b) Desplazamiento desde la coordenada X de la pelota (actualmente, el equivalente a dos robots).

Algo similar a lo antes expuesto ocurre con el delantero principal cuando la pelota está bajo la responsabilidad de un defensor. En este caso el delantero intenta ubicarse a una cierta distancia por delante de la pelota en cuanto a coordenada X y ligeramente desplazado hacia los laterales de la cancha con respecto al Y de la pelota. Si no hay lugar hacia los laterales debido que la pelota está muy cerca de la pared, el robot se ubica desplazado hacia el centro de la cancha. Este comportamiento del delantero principal hace que, eventualmente, pueda dejar pasar la pelota si ésta se dirige al arco rival y poder comenzar a llevarla.

El retroceso del delantero en la cancha es controlado de manera que no se acerque demasiado al área propia entorpeciendo así las tareas de defensa. Como ya se mencionó con anterioridad, en esta circunstancia donde un defensor tiene el control de la pelota, el delantero secundario no participa en la jugada hasta que la pelota pase a ser responsabilidad del delantero principal.

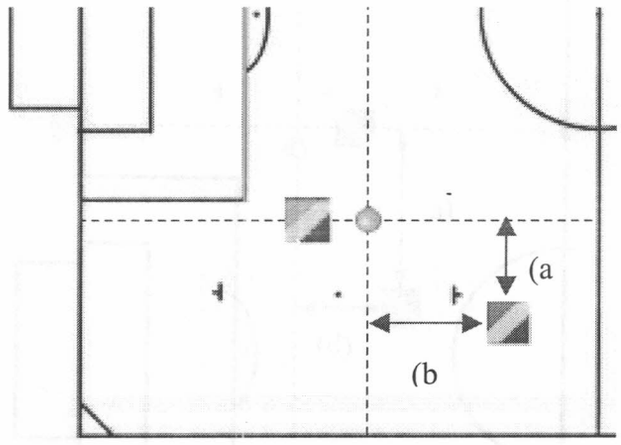


Figura 13.12: Posición del delantero principal cuando la pelota está en poder de un defensor.
(a) Desplazamiento desde la coordenada Y de la pelota (actualmente establecida en el ancho de dos robots). (b) Desplazamiento desde la coordenada X de la pelota (actualmente, el equivalente a un octavo del largo de la cancha).

Cada vez que un delantero está en su posición destino y, por lo tanto, no tiene que ejecutar ninguna acción, gira de manera de quedar orientado hacia el arco rival.

Existe una situación en la que ambos delanteros se consideran principales y es cuando la pelota está dentro del área grande del rival. En este caso, a ambos jugadores se los considera los mejores ubicados del equipo para ir por la pelota. De esta manera se obtiene un juego altamente ofensivo ya que ambos jugadores intentan obtener la pelota y conducirla hacia el arco rival.

Como en el caso del rol defensor, el rol delantero también tiene implementado los comportamientos de desbloqueo y de salir del arco rival (donde a veces entra en una jugada de ataque y puede quedar trabado. También se lo envía en dirección al centro de la cancha de manera que salga del arco). El comportamiento de pateo lateral no lo tiene explícitamente implementado ya que está desarrollado dentro del comportamiento de *Patear al Arco*.

```
Rol_Delantero()
Comenzar
Si Estoy Bloqueado y (no tengo la pelota o estoy fuera del radio del área grande)
Entonces
    Girar sobre si mismo para intentar desbloquearse
Sino
    Si Estoy dentro del Arco Entonces
        Ir en dirección al medio de la cancha
    Sino
        Si soy el elegido Entonces
            Si estoy a más de un 1.5 × radio área grande de mi propio arco Entonces
                Si conviene esperar la pelota Entonces
                    Girar de manera de quedar mirando al arco rival
                Sino
                    Patear al arco
                FinSi
            Sino
                Ir hacia posición base de manera de no molestar a defensores y a mi
                arquero.
            FinSi
        Sino
            Según el rol del elegido Hacer
```

```
                También es Delantero:
                    Si Estoy más lejos en coordenada X del otro atacante que lo
                    parametrizado o estoy más adelante que la línea de área chica rival
                    Entonces
                        Establecer X destino en dos cuerpos detrás de la pelota controlando
                        los topes de avance y retroceso permitidos.
                    FinSi
                    Si Estoy más lejos en coordenada Y del otro atacante que lo
                    parametrizado Entonces
                        Establecer Y destino a distancia parametrizada desde el Y del otro
                        atacante, hacia el centro de la cancha.
                    FinSi
                Es Defensor:
                    Si Estoy del mismo lateral que la pelota Entonces
                        Si no hay otro delantero o el otro delantero está más lejos de la
                        pelota que yo Entonces
                            Establecer X destino en distancia parametrizada por delante de
                            la pelota controlando los topes de retroceso permitidos.
                            Establecer Y destino de manera de estar dos cuerpos más afuera
                            de la posición de la pelota, si hay lugar, sino a dos cuerpos más
                            adentro.
                        Sino
                            Establecer posición base como destino
                        FinSi
                    Sino
                        Establecer posición base como destino
                    FinSi
                Es Arquero:
                    Establecer posición base como destino
            FinSegún
            Si ya estoy en mi destino Entonces
                Girar de manera de quedar mirando al arco rival
            Sino
                Ir a la posición destino
            FinSi
        FinSi
    FinSi
FinSi
Fin
```

Figura 13.13: Pseudocódigo del rol delantero.

13.7 Comportamientos y Jugadas

El conjunto de comportamientos y jugadas a desarrollar en un robot cuando se encara la formación de un equipo de fútbol comprende aquellas tareas

que debe realizar cada robot independientemente de la estrategia que desarrolle el equipo. Para lograr la estrategia descripta anteriormente mediante roles, posiciones, jugadas prearmadas, se debe pensar como los robots navegarán por el campo de juego, como harán para patear la pelota hacia el destino elegido, detenerse, quedarse en una zona de la cancha, girar sobre su eje, patear de costado o girando, manipular la pelota cerca de la pared, etc.

En esta sección se verá cada uno de estos puntos para analizar la técnica elegida y la implementación de los mismos. Cada uno de estos comportamientos o jugadas son disparados de acuerdo a los roles que adquiere el robot en cada momento según las precondiciones que se detallaron en la sección anterior. Los comportamientos y jugadas a implementar son:

- Navegar
- Ir a una zona
- Parar
- Girar
- Patear
- Patear recto
- Patear de costado
- Patear girando
- Patear al arco

Previo a describir estos comportamientos y jugadas hay que detenerse a analizar algunos conceptos que son utilizados en ellos. Se mencionan en este caso algunas características que se presentan en los robots y que requieren una atención especial.

13.7.1 Predicción

Para predecir la posición futura de un objeto se utiliza información de las posiciones que tuvo en determinados momentos anteriores. Por consiguiente en la implementación del equipo se almacenan en una estructura del ambiente las observaciones de las posiciones de los objetos móviles en la cancha: pelota, jugadores propios y jugadores contrarios.

El movimiento de la pelota ha sido modelado linealmente, sin tener en cuenta posibles trayectorias curvas debido a efectos particulares. Por otra parte, estos efectos no han sido observados en la simulación. Si bien se ha obtenido empíricamente, en base a mediciones realizadas con el simulador, un modelo del rozamiento que hace disminuir la velocidad de la pelota, no ha sido implementado en el desarrollo. Este factor de rozamiento hace que la velocidad de la pelota disminuya progresivamente aunque en muy baja proporción. Dado que es muy poco probable que la pelota siga en su movimiento actual sin ser modificado por un golpe o rebote, las predicciones deben ser realizadas a futuros cercanos y, por lo tanto, el factor de decrecimiento de la velocidad es despreciable.

Para la predicción de posiciones futuras de la pelota, se utiliza la velocidad observada en el último ciclo de simulación proyectándola linealmente a la cantidad de ciclos a futuro deseados:

$$\begin{aligned} x_{t+i} &= x_t + \Delta x \times i \qquad \text{donde } \Delta x = x_t - x_{t-1} \\ y_{t+i} &= y_t + \Delta y \times i \qquad \text{donde } \Delta y = y_t - y_{t-1} \end{aligned} \tag{13.1}$$

Cuando la proyección termina fuera de la cancha, se devuelve la coordenada correspondiente al límite de la cancha (desplazado en media pelota, ya que las coordenadas de los objetos referencian al centro de los mismos).

A diferencia del movimiento de la pelota, los robots simulados gran parte del tiempo se mueven describiendo curvas. Esto se debe a la característica que poseen de impulsarse mediante dos ruedas laterales con movimiento independiente en cada una. Salvo el caso en el que se le envíe exactamente la misma potencia a cada rueda, el robot describirá un arco.

Para calcular la posición futura de un robot, es necesario saber la velocidad angular (variación en su orientación) y el radio del círculo sobre el cual el robot se está desplazando. Con esta información, suponiendo que seguirá describiendo el mismo arco, podemos predecir posición y orientación, puesto que:

$$\begin{aligned} \Delta x &= radio \times (sen(\alpha + Va \times t) - sen(\alpha)) \\ \Delta y &= radio \times (-\cos(\alpha + Va \times t) + \cos(\alpha)) \\ \Delta \alpha &= \alpha + Va \times t \end{aligned} \qquad (13.2)$$

Para obtener esto dos datos, se usó el promedio de las dos últimas observaciones, obteniéndose así, por un lado la ***Va*** (en radianes por ciclo) y por otro, ***Δx*** y ***Δy***. Con lo cual se puede deducir el radio ya que:

$$radio = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{2 \times \cos\left(arctg\left(\frac{-\Delta x}{\Delta y} \right) - \alpha \right)} \qquad (13.3)$$

Sin embargo, para evitar este cálculo y dado que en cada ciclo es muy pequeña la distancia recorrida, sabiendo que:

$$radio = \frac{Vl}{Va} \qquad (13.4)$$

se pude aproximar la velocidad lineal (longitud del arco) por la distancia euclideana entre las posiciones inicial y final (longitud de la cuerda), es decir:

$$radio \approx \frac{\sqrt{\Delta x^2 + \Delta y^2}}{Va} \tag{13.5}$$

Otra simplificación que se ha realizado, ha sido considerar a las velocidades angulares muy pequeñas como nulas, con lo cual, en vez de hacer los cálculos en base a un movimiento curvo, se realizan en base a un movimiento recto.

En base a lo antes expuesto, para predecir la posición del robot a una determinada cantidad de pasos en el futuro, se utilizaron las siguientes ecuaciones:

$$\begin{aligned}
\Delta x_i &= \frac{\sqrt{\Delta x^2 + \Delta y^2}}{Va} \times (sen(\alpha_t - Va \times \Delta t) - sen(\alpha_t + i \times Va \times \Delta t)) \\
\Delta y_i &= \frac{\sqrt{\Delta x^2 + \Delta y^2}}{Va} \times (-cos(\alpha_t - Va \times \Delta t) + cos(\alpha_t + i \times Va \times \Delta t)) \\
\Delta \alpha_i &= Va \times i \\
y_{t+i} &= y_t \pm \Delta y_i \\
x_{t+i} &= x_t \pm \Delta x_i \\
\alpha_{t+i} &= \alpha_t \pm \Delta \alpha_i
\end{aligned} \tag{13.6}$$

donde i es la cantidad de pasos futuros que se quiere predecir y Δt es 1, y el signo depende del sentido en el que se esté moviendo el robot.

Esto es así si, como se mencionó anteriormente, la *Va* es mayor a una determinada cota. Caso contrario, se aplican las ecuaciones lineales expuestas para el caso de la predicción de la pelota.

Cabe destacar que, si bien se ha modelado la aceleración y la desaceleración que experimentan los robots al registrar cambios de potencia debido a la inercia, esto no ha sido utilizado en las predicciones. Por esto, la predicción de los robots propios es similar a la de los robots rivales, de los cuales se desconoce la potencia que le ha sido enviada sus motores.

13.7.2 Inercia

El simulador utilizado, al modelar el movimiento de los objetos, tiene en cuenta la inercia al momento de cambiar de trayectoria. Así, si un robot lleva una trayectoria determinada y decide cambiar la misma o detenerse deberá esperar algunos ciclos antes de que el nuevo estado se haga efectivo. Mientras tanto el robot intentará seguir con el rumbo anterior hasta que la nueva fuerza aplicada sobre él supere a la inercia generando estados intermedios.

Este hecho que parece tan común debe ser tenido en cuenta al momento de modelar los movimientos de los robots. Por ejemplo si un robot se encuentra detenido y decide correr hacia la pelota deberá tener en cuenta los ciclos que le llevará vencer a la inercia cuando calcule el momento de empezar a correr.

En la implementación del equipo se optó por un modelo sencillo que tiene en cuenta la velocidad actual del robot y la distancia a la pelota o al punto elegido como destino. Lo que se hace es calcular la cantidad de ciclos que la inercia retardará la efectivización de la nueva trayectoria. Este valor es tenido en cuenta para la predicción de la posición futura del robot. Como se dijo antes, a mayor velocidad actual mayor será la cantidad de ciclos de inercia obtenidos. Al no contar con la descripción del modelo utilizado por el simulador, se establece un modelo empírico donde la inercia varía entre 0 y 4 ciclos de simulación dependiendo de la velocidad actual del robot. Finalmente este valor es adaptado si la pelota o el destino están próximos a la posición actual del robot.

13.7.3 Obstáculos

Cuando un robot debe desplazarse hacia un punto determinado no siempre se puede elegir el camino recto y más corto. Posiblemente tenga obstáculos que saltear que demanden la planificación de una trayectoria. Nuevamente, al ser el

fútbol un ambiente dinámico, hay que contemplar que los obstáculos se mueven y que en ciertos casos se deberá trabajar con predicciones de sus posiciones futuras. Así mismo, no se puede planificar una trayectoria y seguirla ciegamente porque al cabo de un par de ciclos el estado de juego habrá cambiado y la trayectoria elegida no será más útil. Es por esto que se debe readaptar la trayectoria elegida paso a paso de acuerdo a como evoluciona el estado de juego.

13.7.4 Descripción de los comportamientos básicos

Una vez aclarados los conceptos previos, se describirá en forma detallada la implementación de cada uno de los comportamientos y de las jugadas antes enumeradas.

13.7.4.1 Navegar

La navegación se refiere a la acción de trasladarse de un punto a otro en el campo de juego mediante una trayectoria definida. Esto es utilizado por los robots como medio de desplazamiento a los puntos de destino o como una primera etapa de aproximación para el pateo de la pelota. En este último caso, el ajuste final se hace por otros mecanismos que le dan mayor exactitud.

Para poder navegar hacia un destino hay que tener varios factores en consideración, como la diferencia entre la orientación actual y el destino, el sentido en el que se mueve, la velocidad que trae el robot y la inercia que ésta genera, el ángulo de llegada a la posición destino esperado, los bordes de la cancha y los obstáculos –en movimiento- que se encuentran en el camino.

Existen básicamente dos grandes alternativas al momento de elegir un algoritmo de navegación. Una opción es la planificación anticipada de toda la trayectoria entre el robot y la posición destino antes de cada movimiento. Este

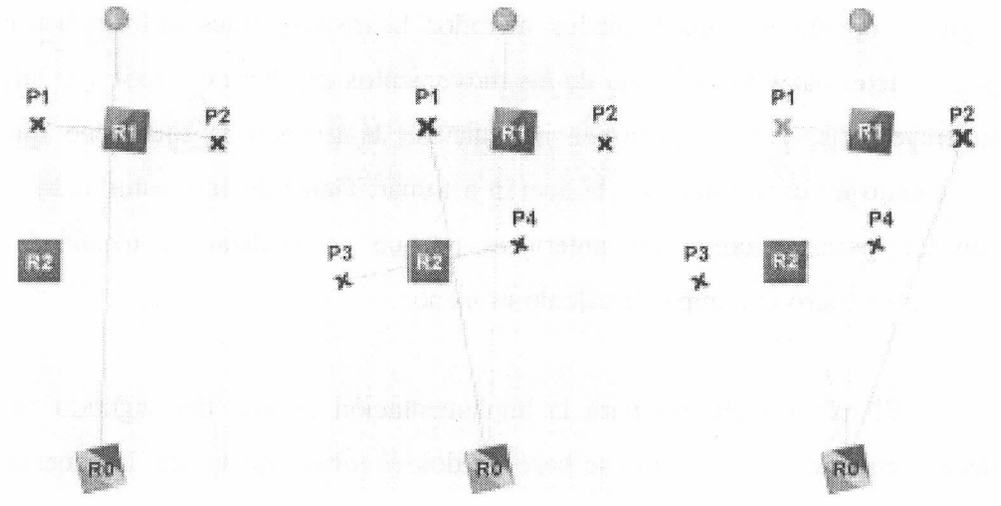
algoritmo debe tener en cuenta la posición de cada uno de los elementos que están en el campo de juego a cada instante para evitar colisiones. En general son métodos con los que se obtiene una buena precisión para llegar al destino. La segunda opción son los llamados métodos heurísticos pues utilizan heurísticas para ir determinando cada uno de los movimientos que hará el robot a lo largo de su trayectoria. Estos algoritmos no calculan la trayectoria total sino que van decidiendo en cada momento la acción a tomar. Generalmente estos métodos no son tan precisos como los anteriores porque no analizan la totalidad de la trayectoria, pero el tiempo de cálculo es menor.

El método elegido para la implementación es uno del segundo tipo, es decir, heurístico. La elección se basó en dos factores principales: la necesidad de una respuesta rápida debido al corto tiempo que involucra un ciclo de juego, y al gran dinamismo que tiene el fútbol por lo que cualquier trayectoria calculada quedaría obsoleta unos pocos instantes después.

El algoritmo implementado, basado en la implementación del equipo Guaraná [REA], utiliza una rutina de detección de obstáculos que analiza en cada paso de la navegación si existe algún elemento entre la posición actual y la deseada, teniendo en cuenta el ancho del camino que determina el ancho del robot. Puntualmente se verifica la distancia perpendicular de cada robot a la recta que determina la trayectoria deseada sea menor que la longitud de la diagonal del robot. Si el camino está libre, se inicia el desplazamiento. En caso de tener el camino obstruido, la rutina realiza un proceso iterativo de selección de alternativas.

El procedimiento consiste en detectar el obstáculo más cercano y seleccionar dos puntos alternativos hacia los costados de este obstáculo para poder sortearlo. Estos puntos se agregan, junto con un valor de ponderación dado por el sentido de movimiento, a una lista de destinos a analizar. En este momento se elige de esta lista una nueva posición deseada para analizar en base a la que esté a

mayor distancia y con menor valor de ponderación (sentido de movimiento del obstáculo).



(a) (b) (c)

Figura 13.14: Cálculo de obstáculos.
En éste gráfico se muestra parte del cálculo de trayectoria del robot amarillo (R0) para llegar a la pelota: a) En la trayectoria recta se encuentra R1 como obstáculo. A ambos lados de él se calculan los puntos P1 y P2. b) Se toma el punto P1 por ser el más lejano al R0 y más cercano a la trayectoria original. En esta nueva trayectoria se encuentra R2 por lo que nuevamente se designan dos puntos en la línea perpendicular a R0-P1, a la altura de R2. Estos puntos son P3 y P4. c) En el tercer paso se elige entre los puntos no analizados el más lejano (P2). En este caso no hay obstáculos con lo cual el robot tomará como objetivo este punto.

Función IraPosicion(Robot, x, y)
Comenzar
Ajustar posición del robot de acuerdo a la inercia
Ajustar posición destino dentro de los límites de la cancha
Si** Hay Obstáculos entre mi posición actual y el destino **Entonces
Cambiar el destino por un punto intermedio en una trayectoria libre
FinSi
Calcular sentido de desplazamiento a partir de velocidad actual y diferencia de orientación
Si** la diferencia de ángulo es grande y la velocidad es alta **Entonces
Disminuir la velocidad
SiNo
Si** el ángulo es muy grande **Entoces
Asignar velocidades opuestas a las ruedas interna y externa

```
        SiNo
            Si la velocidad es baja Entonces
                Asignar velocidad máxima a la rueda externa
            SiNo
                Asignar velocidad media a la rueda externa
            FinSi
            Calcular velocidad de la rueda interna en base a la longitud del eje
        FinSi
        Asignar velocidades calculadas a cada ruedas según el sentido y la velocidad
         angular
    FinSi
Fin.
```

Figura 13.15: Pseudocódigo de rutina de navegación.

Una vez seleccionada una posición destino (final o intermedia) sin obstáculos, el algoritmo calcula, en base a diferencia de ángulos y a velocidad actual, la velocidad necesaria para cada una de las ruedas de forma que el robot se dirija al destino. En la Fig.13.15 se puede ver el pseudocódigo de alto nivel para el algoritmo de navegación.

13.7.4.2 Ir a una zona

Este comportamiento se utiliza para mantener un robot en una zona determinada de la cancha. La zona está definida por un punto de coordenadas fijas y un valor delta recibido como parámetro que indica el desplazamiento máximo en cualquiera de las dos coordenadas que puede tener el robot respecto de este punto. Si el robot está dentro de este punto se detiene utilizando el comportamiento *Parar*, si no invocará al algoritmo de navegación para dirigirse al punto destino.

Ubicar el robot en un punto determinado resulta a veces un tanto complicado pues los ciclos de ejecución a intervalos de tiempo fijos y la inercia modelada por el simulador hacen que no pueda detener al robot en una posición fija, más aún al estar las coordenadas en la cancha representadas con valores reales. Otras veces, la posición destino del robot requiere poca precisión y no es

necesario establecer una posición puntual. Es por eso que cuando se quiere enviar a un jugador a su posición base (posición de espera) o a un determinado punto sea para bloquear la pelota, atajar o esperar un pase se debe utilizar este comportamiento, adecuando el valor delta de acuerdo a cada circunstancia.

13.7.4.3 Parar

Este comportamiento se utiliza cuando se quiere dejar inmóvil a un jugador. Consiste en asignarle potencia 0 a las ruedas produciendo la detención del robot en un número reducido de ciclos a causa de la inercia que se desprenda de la velocidad actual.

13.7.4.4 Girar

Cuando se quiere orientar a un robot, se debe realizar una rotación en alguno de los dos sentidos posibles (velocidad angular positiva o negativa). El comportamiento *Girar* posiciona a los robots con un ángulo determinado realizando el giro más corto. Para esto toma en cuenta la orientación actual y la predicha para algunos ciclos en el futuro y decide cuál es el menor ángulo de rotación hacia el destino elegido.

En el caso del arquero, este comportamiento es utilizado para despegarse la pelota cuando ésta se encuentra en el lateral. El defensor también utiliza el giro pero para buscar una orientación efectiva para interceptar la trayectoria de la pelota, es decir, poniéndose perpendicular a la cancha de manera de moverse rápidamente hacia un lado u otro. Finalmente, el delantero utiliza el comportamiento de giro para ubicarse mirando hacia el arco cuando acompaña la jugada del otro delantero.

13.7.4.5 Patear

El comportamiento *Patear* es el que permite realizar la aproximación final de la navegación y darle precisión a un pateo hacia el arco contrario. La función básica de este algoritmo es chequear si existe un arco basado en la posición y orientación del robot que describa una trayectoria hacia la pelota de manera que llegue a ésta con el ángulo necesario para dirigirla hacia el arco. Si éste existe, calcula y establece las potencias necesarias para cada rueda del robot. Este algoritmo es invocado y utilizado por el comportamiento *Patear al Arco*.

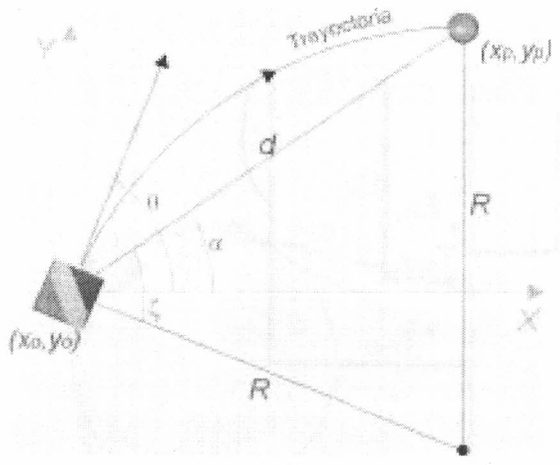


Figura 13.16: Curva de pateo.
La figura muestra el robot en la posición (x_o,y_o) y el arco de circunferencia de radio R que describe hasta llegar al punto (x_p,y_p). El arco es calculado en base a la orientación del robot θ, el ángulo α hacia donde se encuentra la pelota y la distancia d a la misma

El algoritmo utiliza las posiciones del robot y de la pelota calculadas a futuro en base a la inercia estimada para el robot y la distancia de éste a la pelota. La posición del robot puede tomarse, según el parámetro recibido, del centro del mismo o de alguno de sus costados pues no siempre es necesario que el robot impacte con el centro de su parte frontal a la pelota.

Figura 13.17: Centros del robot.
Este gráfico muestra los tres centros que se toman en el robot para el cálculo de las curvas de trayectoria hacia la pelota: (a) Centro convencional. (b) Centro desplazado hacia la rueda izquierda. (c) Centro desplazado hacia la rueda derecha.

Tomar los puntos laterales como centro del robot amplia las posibilidades de llegar a la pelota con un ángulo apropiado, como se puede apreciar en el siguiente gráfico.

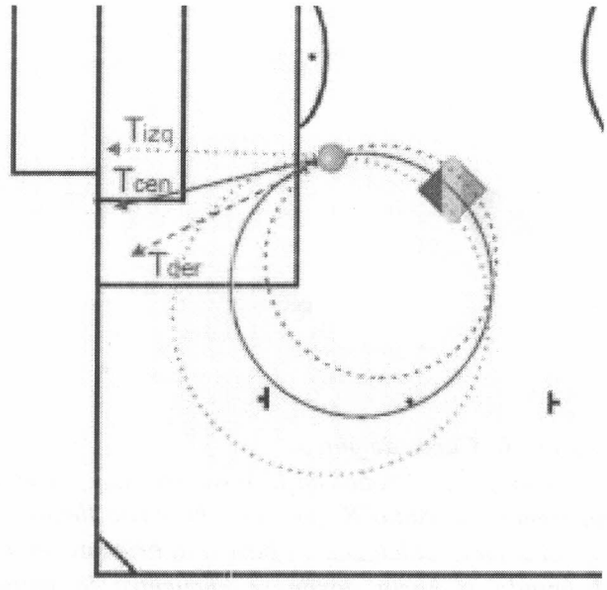


Figura 13.18: Pateo con distintos centros.
Este gráfico muestra la dirección resultante de patear la pelota calculando la trayectoria en base a la curva que pasa por cada uno de los 3 centros: convencional (Tcen), izquierdo (Tizq) y derecho (Tder). Si bien tomando el centro convencional el destino final de la pelota no será el arco, esto se consigue tomando el centro izquierdo.

Inicialmente, el algoritmo decide si el robot debería avanzar o retroceder para llegar a la pelota, en base a la diferencia de ángulo entre la orientación del robot (θ) y el ángulo de la pelota relativo al robot (α). También calcula si el

recorrido a realizar será recto o curvo y, en este último caso, si la velocidad angular será positiva o negativa, como se muestra en (13.7).

Si $(\alpha \cong \theta)$ *o* $(\alpha \cong \theta \pm \pi)$ *la trayectoria es recta, sino es curva*
Si $(\theta - \pi < \alpha < \theta)$ *la velocidad angular es negativa, sino es positiva* (13.7)

Aquí ya se puede calcular el ángulo de llegada a la pelota y chequear si éste se encuentra dentro del rango calculado en base a la posición del robot. El cálculo del ángulo de llegada esta dado por la fórmula

$$\alpha final = \theta \tag{13.8}$$

para el caso en que la trayectoria es recta, donde θ es la orientación del robot. En cambio, si la trayectoria es curva, el ángulo final surge de la siguiente asignación

$$\alpha final = 2 \times \alpha + \theta \tag{13.9}$$

donde θ es la orientación del robot y α es el ángulo de la pelota relativo al robot. Si el robot retrocede hay que sumarle o restarle π para ajustar al cambio relativo de frente del robot respecto de θ.

En este punto el algoritmo ya calculó si existe una trayectoria curva hacia la pelota que la impulse en la dirección deseada. Si no existe, el algoritmo termina. Si existe hay que revisar otros aspectos de la trayectoria antes de mover al robot. El análisis es distinto si la trayectoria calculada es recta o curva. Si es recta y no hay obstáculos en el camino entonces se establecen potencias máximas para cada rueda. En el caso que la trayectoria sea curva se calcula el radio del círculo que contiene al arco determinado (Fig.13.16). El primer análisis realizado es el tamaño del radio. Si este es muy pequeño se corre el riesgo de que el robot golpee a la pelota con un lateral al girar. Luego se calcula la longitud del arco y se establecen condiciones sobre la apertura y longitud del arco a recorrer para evitar

que se describa un gran círculo mientras que se puede trazar un recorrido de menor distancia y, por consiguiente, llegar más rápido a la pelota. Finalmente se observa si el arco pasa por algún punto fuera del campo de juego que imposibilite seguir totalmente su trayectoria o si hay obstáculos en el camino.

El cálculo de obstáculos se realiza de dos formas distintas dependiendo, nuevamente, de si la trayectoria es recta o curva. En el caso de una trayectoria recta se utiliza el mismo algoritmo de la navegación para determinar si existe un camino del tamaño de la diagonal del robot entre la posición actual y la pelota sin obstáculos. Si, en cambio, la trayectoria es curva, se toman n puntos sobre el arco a distancia $2 \times AnchoRobot$ y se calcula si hay algún robot cerca de estos puntos. Se establece como condición de cercanía al doble del ancho del robot en distancia a cada punto.

Finalmente, si la trayectoria cumple con todas las condiciones, se calculan las potencias de las ruedas interna y externa y se envían al simulador. La potencia de las ruedas se obtiene de las siguientes asignaciones

$$\begin{aligned} potExterna &= Potencia\_Máxima \\ potInterna &= potExterna \times \frac{Radio - EJE/2}{Radio + EJE/2} \end{aligned} \tag{13.10}$$

donde *EJE* es la longitud del eje de ruedas del robot.

13.7.4.6 Patear Recto

Este comportamiento implementa la acción de pateo propiamente dicha. Aquí el robot impacta la pelota dirigiéndola hacia el arco contrario. Generalmente este comportamiento se da como culminación del comportamiento anteriormente descripto (Patear) en el momento en que el robot llega al final del arco que lo llevaba hacia la pelota.

El algoritmo implementado evalúa ciertas condiciones antes de decidir que se puede realizar la acción que consiste en poner potencia máxima en ambas ruedas, ya sea con magnitud positiva o negativa, para alcanzar la pelota y dirigirla en la dirección deseada. La primera de éstas es que la pelota debe estar a una distancia mínima del robot de $2 \times AnchoRobot$. Esto se debe a que, como se mencionó antes, es el último instante del pateo y, además, porque el algoritmo no realiza un chequeo de obstáculos en la trayectoria a la pelota.

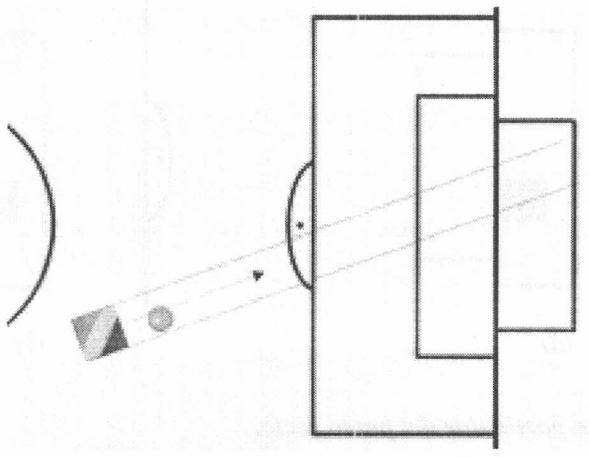


Figura 13.19: Pateo recto.
La pelota se encuentra en la banda formada por la prolongación de las líneas de los costados del robot. El robot avanza a máxima velocidad en el sentido actual.

Nuevamente, las posiciones del robot y de la pelota se toman en base a predicciones realizadas en base a la inercia calculada y a la distancia del robot a la pelota respectivamente. La segunda condición es que la pelota esté entre el robot y el arco contrario. Aquí se calcula si la trayectoria que describe el robot con la orientación actual llega al arco contrario y si la pelota se encuentra en ese camino que describe el robot. Finalmente se chequea que el jugador no entre en el área chica contraria para evitar colisionar con el arquero.

Un caso particular contemplado en este comportamiento es cuando la pelota se encuentra pegada a la línea de fondo del arco contrario y el robot está en

la misma zona, más desplazado hacia los laterales y orientado hacia uno de los costados de la cancha (Fig.13.20). Esta configuración permite que el robot se desplace hacia el centro, en referencia a la coordenada *Y*, de la cancha enviando a la pelota hacia el arco.

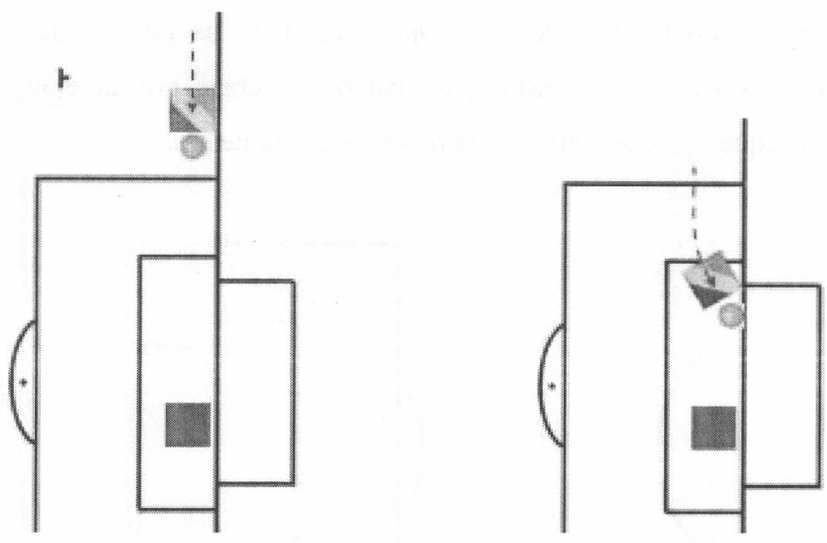


Figura 13.20: Caso particular del pateo recto.
En el gráfico a) el robot se aproxima con pateo recto por el lateral del arco contrario. En b) el robot realiza un pequeño giro hacia el interior del arco para lograr que la pelota entre en él.

La acción tomada para este comportamiento es avanzar a velocidad máxima empujando la pelota hacia el eje central de la cancha (en coordenada Y). Cuando la pelota llega a la zona ubicada frente al arco, el robot realiza un pequeño giro hacia el interior del mismo intentando convertir un gol.

13.7.4.7 Patear de Costado

El pateo de costado es un comportamiento que resuelve la situación que se presenta cuando el robot se encuentra con la pelota cerca de uno de los laterales de su cuerpo por lo cual no puede empujarla desplazándose hacia delante o hacia atrás. La acción consiste en realizar un giro sobre su eje, asignándole potencias

máximas a las ruedas pero en sentido contrario. La única condición para activar el comportamiento es que la pelota no esté en el lado del robot que apunta hacia el arco propio. Si la pelota se encuentra entre el robot y el arco contrario, el giro puede realizarse en cualquier dirección. En cambio si el robot se encuentra mirando hacia alguno de los arcos, el sentido de giro deberá depender del lado en el que se encuentre la pelota.

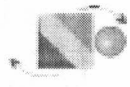


Figura 13.21: Pateo de costado.
El robot gira sobre su eje impulsando la pelota con el lateral.

13.7.4.8 Patear Girando

Este comportamiento es similar al de *Patear de costado* para el caso en que la pelota está cerca del frente o reverso del robot, pero el ángulo no permite empujar la pelota hacia el arco contrario. Aquí también se realiza un giro sobre el eje del robot, asignándole potencias máximas en sentido contrario a las ruedas.

Figura 13.21: Pateo girando.
El robot gira asignando velocidad máxima en una rueda y velocidad cero en la otra.

13.7.4.9 Patear al Arco

Este comportamiento se inicia analizando la posibilidad de realizar un pateo de costado o girando. Si esto no es factible intenta un pateo convencional. El primer paso para esto es intentar un pateo recto hacia el arco. Si ninguno de

estos intentos es exitoso, entonces el algoritmo calcula el rango de ángulos en donde se orienta el arco contrario respecto del robot, para invocar al comportamiento de pateo con los tres centros desplazados del robot (Fig.13.23).

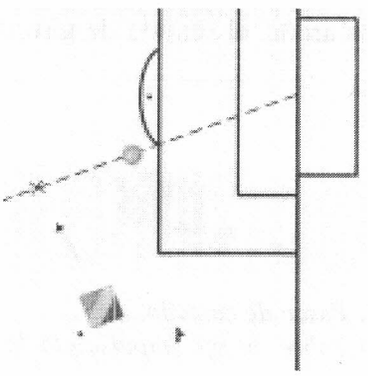


Figura 13.22: Posicionamiento para patear.
Si el robot no puede utilizar ninguno de los comportamientos de pateo, se posiciona detrás de la pelota.

Si hasta este punto no se ha podido utilizar ninguno de los comportamientos, el robot intenta posicionarse detrás de la pelota para poder tener una futura posibilidad de pateo (Fig.13.22). Para esto se define un punto en la continuación de la recta que une el centro del arco contrario con el robot y a una cierta distancia de éste. Si el robot se encuentra cerca de este punto o está yendo hacia la pelota con un ángulo aceptable es dirigido directamente hacia la pelota; si no se dirige hacia el punto especificado.

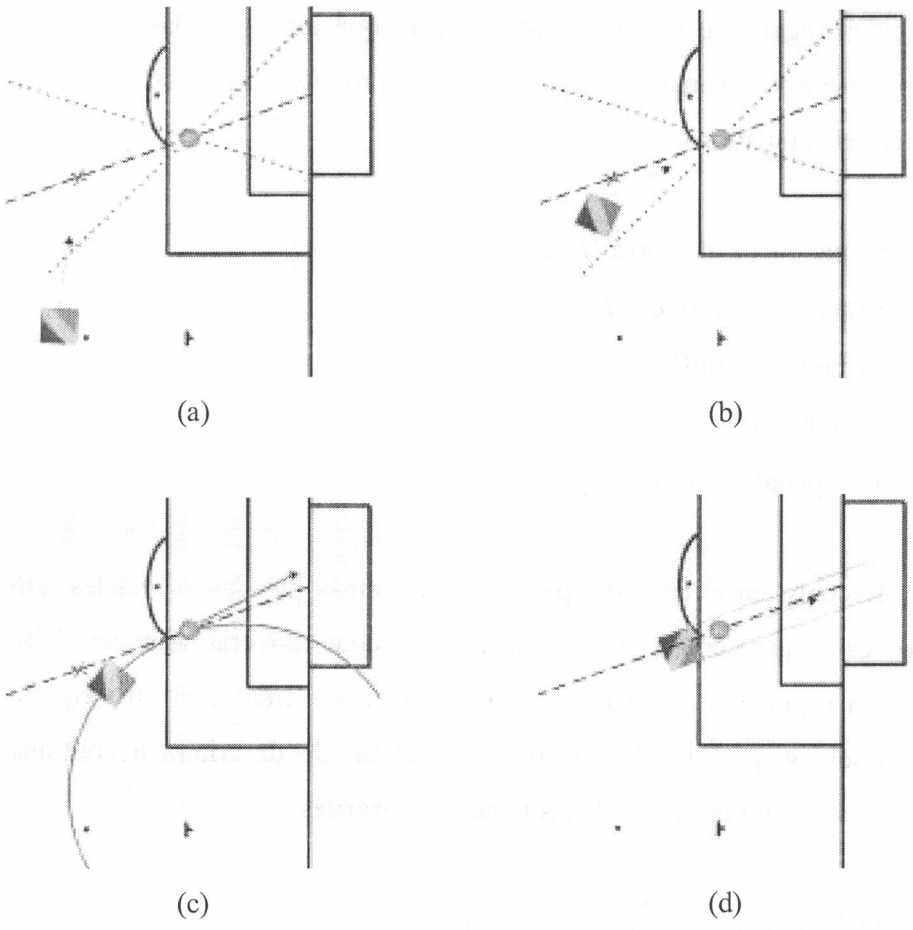


Figura 13.23: Secuencia de pateo.
a) El robot se aproxima a una posición detrás de la pelota. b) El robot se encuentra dentro del cono que generan las rectas que pasan por la pelota y los postes del arco, entonces va hacia la pelota. c) Existe un arco que pasa por el centro del robot y de la pelota de forma que el robot llegue a la pelota con un buen ángulo. d) El robot se encuentra alineado con la pelota en dirección al arco para un pateo recto.

13.8 Jugadas Especiales

Las jugadas especiales son formaciones específicas o acciones particulares a realizarse en las jugadas con pelota detenida. En estos casos hay ciertos aspectos a tener en cuenta que hacen que el esquema de juego normal no pueda ser utilizado. Un ejemplo de esto es el saque desde el centro de la cancha (*Place Kick*) donde uno de los robots debe impulsar la pelota hacia el propio campo de juego. Previo a estas jugadas el representante de cada equipo puede posicionar a cada

robot en el lugar que tenga previsto, por lo que estas jugadas no requieren de posicionamientos automáticos por parte de los robots. Las jugadas especiales que considera el simulador son:

- Saque desde el centro (*Place Kick*)
- Saque de arco (*Goal Kick*)
- Pique (*Free Ball*)
- Tiro libre (*Free Kick*)
- Tiro penal (*Penalty Kick*)

Excepto en el caso del pique, en las demás jugadas especiales uno de los equipos es el que tiene control de la pelota y debe moverla. Aún así, debe buscar una forma rápida de realizar la acción pues si se demora, el otro equipo puede adelantársele y quitarle el control de la pelota. Se describirá a continuación la implementación de cada una de las jugadas especiales.

Debido a que el servidor no indica cuando el estado de juego pasa de una jugada especial al juego normal, se utiliza el movimiento de la pelota para pasar al estado de juego normal.

13.8.1 Saque desde el Centro (*Place Kick*)

Debido a que en el saque del centro, el robot que realiza el saque debe estar en el campo contrario para sacar hacia su propio campo, esta condición se utiliza para identificar al encargado de realizar el saque. Una vez detectado el robot ubicado en campo contrario, se detiene al resto y se lo envía a éste hacia la pelota para que colisione con ella y la dirija hacia su campo. Una vez que se mueve la pelota, el juego continúa en forma normal.

Si el encargado de realizar el saque es el equipo contrario, los robots asumen su rol habitual para el estado de juego normal.

13.8.2 Saque de Arco (*Goal Kick*)

El saque de arco es realizado por uno de los defensores para no sacar al arquero de su línea de movimiento habitual y, de esta forma, permitir que se posicione rápidamente para poder atajar. El elegido para realizar el saque es el más cercano a la pelota y la acción que realiza es ir hacia la pelota y colisionarla.

Si el encargado de realizar el saque es el equipo contrario, los robots asumen su rol habitual para el estado de juego normal.

13.8.3 Pique (*Free Ball*)

Para seleccionar el pateador en este estado de juego se elige aquel jugador que tenga la menor distancia a la pelota en el eje de coordenadas *Y*, pues es el único que puede estar en el cuadrante del pique, y la posición elegida para los demás jugadores no coincide con esta coordenada respecto de la pelota. La acción a realizar es ir hacia la pelota y, cuando se encuentra próximo a la pelota, realiza un pequeño giro para orientar el pateo. Si es un pique en zona defensiva, intenta patear hacia los laterales y, si está en campo contrario, intenta patear hacia el centro de la cancha. El resto del equipo desarrolla su rol habitual.

13.8.4 Tiro Libre (*Free Kick*)

Para ejecutar el tiro libre, el robot más cercano a la pelota debe patear al arco mientras que los demás jugadores implementan sus roles habituales.

Si el encargado de realizar el tiro libre es el equipo contrario, los robots asumen su rol habitual para el estado de juego normal.

13.8.5 Tiro Penal (*Penalty Kick*)

En el caso que el equipo deba patear el penal, el jugador más cercano a la pelota es el encargado de ejecutar el tiro al arco. El resto del equipo permanece en su rol habitual.

Ahora, si el equipo contrario es el que patea el penal, hasta que la pelota se mueva el arquero se posiciona en la coordenada Y de la pelota. Una vez que el contrario patea la pelota el arquero asume su rol y comportamiento habitual. El resto del equipo también toma su rol habitual

Capítulo 14
Experimentación y Resultados

14.1 Introducción

En el presente capítulo se exponen las experimentaciones realizadas y los resultados obtenidos así como también el detalle de la forma en la que fue evolucionando cada parte de la implementación hasta llegar a la versión descripta en el capítulo 13.

En primer término se expone la evolución registrada por la estrategia del equipo y sus roles; luego, como han ido evolucionando los comportamientos y acciones individuales. A continuación, se explican brevemente ciertos aspectos relacionados al simulador que generaron cambios en el desarrollo. Se sigue con la descripción de un conjunto de herramientas desarrolladas para el testeo y análisis de comportamientos y por último se muestran los resultados de un conjunto de pruebas y mediciones realizadas durante una serie de partidos contra equipos rivales tendientes a evaluar en detalle el comportamiento del equipo UBASot.

14.2 Evolución de Estrategia

Habiendo definido e implementado las primeras versiones de los comportamientos individuales básicos e indispensables para jugar un partido de fútbol, se estuvo en condiciones de comenzar con la definición de la estrategia de juego del equipo, es decir, la forma en que se debían combinar estos comportamientos individuales para que los partidos se desarrollen en forma ordenada, siendo un juego en conjunto y no un juego individual.

Si bien desde un principio se contó con un conjunto de lineamientos generales basados en el estilo de juego y la filosofía de implementación deseados, los cuales se describen en 13.2, muchas veces la estrategia y las jugadas diseñadas no se comportan según lo esperado, lo cual se observa luego de haber sido implementadas y observando el desarrollo de los partidos. Por ello, el desarrollo del equipo fue evolucionando en base al siguiente algoritmo:

1. *Diseñar modificaciones a la estrategia de acuerdo a los lineamientos generales.*
2. *Implementar las modificaciones en la estrategia.*
3. *Evaluar nueva estrategia en base a lineamientos generales observando el desempeño del equipo en partidos.*
4. *Volver a 1.*

Este ciclo puede ser repetido casi indefinidamente, ya que el amplio dominio del juego siempre contiene alguna situación donde la estrategia puede ser mejorada. La implementación del equipo UBASot tuvo como cota de este ciclo, la realización del campeonato FIRA 2002. Cabe aclarar que, previo a la participación en el campeonato, los partidos de test solamente se podían realizar con los equipos básicos provistos por el simulador o las versiones anteriores del equipo en desarrollo ya que, durante la etapa de desarrollo, no fue posible contar con equipos participantes en campeonatos anteriores o que fueran a presentar su estrategia en el campeonato 2002 que permitiera evaluar en forma más efectiva el nivel competitivo de UBASot.

A continuación se detalla la evolución del equipo en base al proceso descripto anteriormente. La formación del equipo y el mecanismo de elección del robot mejor posicionado no se tratan en este capítulo dado que no han sufrido variaciones desde la primera estrategia planteada y el detalle de su implementación ha ido desarrollado en el capítulo 13 de este trabajo.

14.2.1 Rol Arquero

En primera instancia el arquero recorría el frente de su arco, entre las líneas que delimitan el área chica, siguiendo la posición de la pelota. De esta manera, el robot siempre mantenía la misma posición X ajustando su posición Y de acuerdo a la ubicación de la pelota. En las observaciones de juego, se detectaron dos inconvenientes que comprometían la eficiencia del arquero:

- La velocidad aplicada y la inercia de los robots hacía que los mismos no alcanzaran la posición deseada, sino que oscilaran continuamente alrededor de la posición final. Esta oscilación ocasionaba, en ciertos casos, que el arquero, por el desplazamiento continuo, no detuviera la pelota permitiendo el gol.
- La utilización de la posición Y de la pelota como posición destino del arquero mostraba un comportamiento débil del arquero cuando el recorrido de la pelota hacia el arco se alejaba de la línea perpendicular al mismo ya que, al ser la pelota más veloz que el robot, potencialmente el arquero podía no llegar al punto de intecepción a tiempo.

Para resolver ambos inconvenientes se estableció que el arquero intentara ubicarse, dentro de su recorrido, en el punto en el cual se esperaba que la pelota interceptara a la recta determinada por su lateral defensivo (cara externa del robot que se encuentra del lado opuesto al arco). La velocidad aplicada en cada ciclo de simulación para alcanzar la posición final sería menor en cada paso de manera que el robot llegara la posición deseada, evitando que la inercia lo desplace a otra ubicación. El método implementado se describe en forma minuciosa en el capítulo anterior, en la sección 13.6.1.

Habiendo mejorado notablemente el desempeño del arquero, durante los partidos de prueba se pudieron observar goles generados fundamentalmente por dos motivos:

- Jugadas sobre las paredes laterales, en las que el robot oponente empujaba la pelota hacia el arco desplazando al arquero hacia atrás hasta que parte del arco quedara libre y le permitiera concretar el gol.
- Debido a roces con otros robots o con esquinas del arco propio, el arquero perdía la orientación correcta (90° o –90°) y, sin retomar su trayectoria paralela al arco, continuaba moviéndose de acuerdo con la posición de la pelota. Esta situación generaba espacios no cubiertos por el arquero, debilitándose así la defensa del mismo.

Para solucionar el primero de estos problemas se implementó que, cuando el arquero detectara la cercanía de la pelota viniendo sobre una de las paredes laterales, ejerciera fuerza opuesta al sentido que trajera la pelota. La fuerza ejercida por el arquero se implementó asignando máxima potencia a ambas ruedas, otorgando mínimamente mayor potencia a la rueda interna (sobre la pared) de manera que, si la pelota supera la posición del arquero, sea sobre su lateral exterior y no sobre la pared, lo cual resulta peligroso. De esta manera ambos robots, el arquero propio y el robot oponente, presionaban la pelota pero la misma no avanzaba, perdiéndose así el dinamismo de juego deseado. Para mejorar esto y, en caso que esta situación se produjera fuera del área grande, se optó por aplicar *Patear Girando* donde el pelota se desbloquea con mayor facilidad y la lejanía del arco no aporta riesgo de gol.

En cuanto al segundo punto, se estableció que si la desviación sobre la orientación inicial (90° o –90°) era mayor a 5°, el arquero debía reacomodarse para recuperar su orientación. Para ello se incorporó dentro de los comportamientos invocados por el arquero el comportamiento individual *Rotar*. La variación permitida (5°) se determinó mediante sucesivas observaciones concluyendo que mientras el arquero mantenía una orientación entre 85° y 95° o entre -85° o –95°, la trayectoria recorrida por el robot cubría el arco, sin resultar

significativa la variación con respecto a la trayectoria ideal (con orientación de 90° o –90°) por lo cual no se comprometía la eficiencia del arquero.

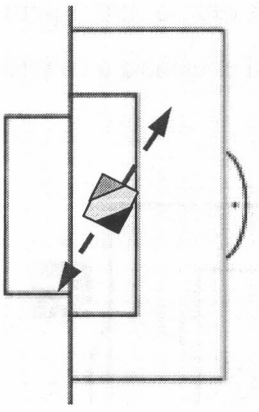


Figura 14.1: Trayectoria errónea del arquero.
Ejemplo de trayectoria que sigue el arquero después de ser desplazado por otro robot o haber colisionado con la pared.

Aunque el arquero se posicionaba en forma correcta e interceptaba la trayectoria de la pelota, cumpliendo así eficientemente su tarea, en ocasiones el arquero no podía desprenderse de la pelota atajada, quedando ésta detenida sobre el lateral externo del robot, comprometiendo tanto la seguridad del arco, como la dinámica de juego deseada. Para estos mejorar esto, se incorporó a las acciones de este rol *Patear de Costado*, haciendo que el robot gire e impulse la pelota despegándola de su lateral externo, propiciando la puesta en juego de misma.

14.2.3 Rol Defensor

En el rol defensor siempre se ha mantenido un esquema de primero y segundo jugador en función a la cercanía de los robots con la pelota, según se detalla en 13.6.2.

Desde el primer esquema planteado para este rol, cuando la pelota se encuentra en el campo rival o avanza en dirección al arco contrario, ambos robots defensores se ubican en sus posiciones base. En primera instancia, no se

consideraba la orientación que tenían los robots estando en sus posiciones base. Pero habiendo observado las maniobras que debían hacer éstos para posicionarse convenientemente cuando entraban en acción, se determinó que se ubicaran con la misma orientación que el arquero (90° o –90°) permitiendo, de esta manera, una reacción más rápida, aportando así eficiencia a su tarea.

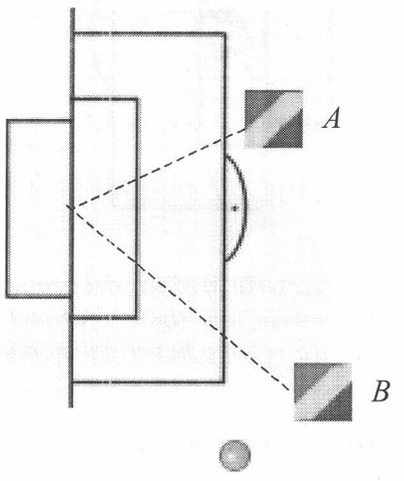


Figura 14.2: Selección de defensor.
En primera instancia acudía a la pelota el robot A. Con los cambios implementados acude el robot B.

Cuando la pelota se encuentra en campo propio y está avanzando hacia el mismo arco, los robots defensores abandonan su actitud pasiva para intervenir en la jugada. Como se mencionó en el párrafo anterior la orientación resultó de gran importancia para la capacidad de reacción de los robots, pero de todas maneras, no era posible descartar las situaciones donde el delantero oponente sorteaba al defensor propio y la pelota quedaba posicionada, en coordenadas X, entre el arco y ambos defensores. En estos casos, en primera instancia, el robot defensor más cercano al centro del arco era el que se posicionaba entre el arco y la pelota. Dado que la ubicación de los defensores puede variar (debido a que cuentan con un rango dentro del cual posicionarse), como se muestra en Fig.14.2, el defensor más cercano al arco no siempre era el más cercano a la jugada. Esto ocasionaba que el robot que acudiera a la pelota para interceptarla o cubriera el arco, fuera el más lejano, quedándose inmóvil el robot más cercano, el cual podría llegar a cubrir el

arco correctamente, quedando ambos jugadores, eventualmente, muy cerca y dejándose libre una amplia zona frente al arco propio. Observando estos casos, se optó por que el robot defensor que tomara acción fuera el robot más cercano a la pelota y no el más cercano al arco, lográndose así mejorar la táctica de defensa y, a su vez, cumplir con los lineamientos relativos a la no interferencia de los jugadores entre sí y la cobertura de áreas de la cancha.

También relativo al hecho de ejercer una sólida defensa y cubrir eficientemente las zonas de la cancha, se detectó que el primer defensor se encontraba sobre uno de los sectores laterales, alejado de su posición base, yendo por la pelota o cubriendo su trayectoria, dejaba sin protección la mitad de arco que debía cubrir. Para estar mejor posicionado ante un cambio de dirección en la jugada, el segundo defensor, que estaba cubriendo solamente una mitad de arco, comenzó a ubicarse en el centro para poder dominar el arco completo. Como en la mayoría de estas situaciones el arquero propio se encontraba sobre uno de los extremos de su arco, la posición del segundo defensor resulta de gran utilidad.

Un caso particular se daba cuando un defensor recuperaba la pelota en su campo e ingresaba en el campo rival. Al pasar la mitad de la cancha, y según las reglas de su accionar, soltaba la pelota considerando que el delantero principal estaba ubicado para recibirla y continuar la jugada. En muchos casos este traspaso le permitía al equipo rival interceptar la pelota y atacar nuevamente o, aunque el equipo propio continuara con la pelota, se volvía más lento el juego y se perdía una buena oportunidad de ataque. Para aprovechar esta situación se habilitó al robot defensor a avanzar con la pelota en campo contrario hasta que la misma salga de su alcance.

14.2.3 Rol Delantero

Desde la primera estrategia planteada, el equipo mantiene un esquema de dos robots delanteros, asignando a cada uno de ellos distintas funciones

dependiendo de su posicionamiento con respecto a la pelota, según se detalla en la sección 13.6.3.

El delantero acompañante o secundario en primera instancia se ubicaba más adelante que su compañero. La línea máxima de avance definida era nueve décimos de cancha, dejando el último décimo libre para poder maniobrar. La movilidad del segundo delantero acompañando al delantero principal, abarcaba casi toda la cancha, dejando solamente un décimo libre sobre ambas paredes del fondo. Con respecto a la distancia vertical, el segundo delantero se posicionaba a lo sumo a un cuarto de cancha del delantero principal. Los problemas observados con respecto a este accionar fueron los siguientes:

- Ubicándose delante de su compañero, el segundo delantero demoraba en posicionarse para patear la pelota, dado que en la mayoría de las situaciones el robot se encontraba entre el arco y la pelota y para poder patearla debía acomodarse detrás de la misma. Estas maniobras consumían tiempo haciendo que el equipo perdiera la posesión de la pelota. En este accionar se observó también que no se cumplía con la premisa de propiciar jugadas cooperativas que se hubiesen presentado de estar mejor posicionado.
- Con respecto a los espacios sobre el fondo de la cancha que los delanteros mantenían libres en su avance (un décimo de cancha), se observó que este espacio era reducido y los robots se encontrarían mejor posicionados para intervenir si esperaban la pelota sobre el borde del área grande.
- Con la distancia vertical entre robots definida en un cuarto de cancha, existían posibilidades que ambos robots delanteros ocuparan solamente uno de los cuadrantes de la cancha (campo rival superior o inferior) dejando descubierto el otro cuadrante.

Los cambios introducidos al rol delantero tendientes a solucionar estos inconvenientes fueron los siguientes: para mejorar el posicionamiento y evitar la

pérdida de la pelota y propiciar jugadas de pase, el segundo delantero comenzó a acompañar al delantero principal más retrasado que éste, como mínimo a dos robots de distancia. De esta manera, cuando el delantero principal soltaba la pelota, su compañero ya estaba detrás, debiendo recorrer una trayectoria menos compleja para interceptarla. En cuanto al segundo punto, la distancia referida fuer ampliada a un octavo. Este espacio les daba más tiempo para tomar carrera e intentar interceptar la pelota. Relativo al último punto, se comenzó a tomar como referencia la línea imaginaria que separa en forma transversal el campo y, cuando el delantero principal estaba debajo de esta línea, su acompañante se posicionaba a un cuarto de cancha sobre dicha línea, evitando el arrinconamiento sobre la pared inferior. El mismo tratamiento se aplicó si el delantero principal estaba por encima de la línea imaginaria.

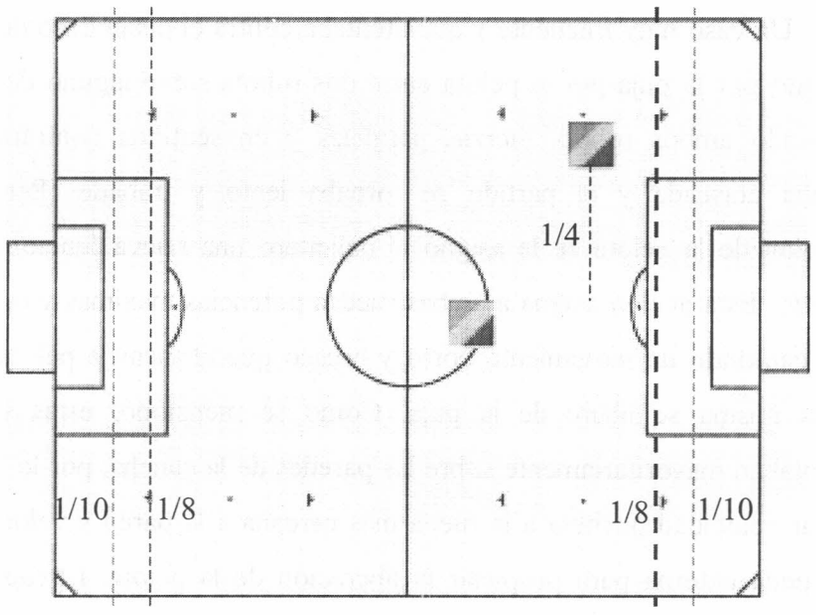


Figura 14.3: Límites de avance de los delanteros en la cancha.
Variación de 1/10 a 1/8 de cancha en los límites de avance de los delanteros. También se visualiza la distancia máxima que deben mantener los delanteros entre sí.

Mejorados los aspectos anteriores, en nuevas observaciones se detectó que, cuando un delantero era el robot elegido, independientemente de su ubicación dentro del campo, siempre intentaba llevar o recuperar la pelota. Esto ocasionaba

que, estando en el área chica propia, el delantero entorpeciera el trabajo de su arquero. Para evitar estos inconvenientes se optó por dar dirección al robot delantero hacia el centro de la cancha, dejando de ser el robot elegido y permitiendo de esta forma que actúen libremente el arquero y los defensores.

Otra situación planteada fue que, estando el delantero principal en campo propio, por la distancia máxima que debía conservar con su acompañante, el segundo delantero también se mantenía en el mismo campo haciendo que el campo rival quedara totalmente libre y la posibilidad de ataque se viera disminuida. Para mantener la cancha más cubierta, se optó por desplazar al delantero más alejado de la pelota a su posición base cubriendo así, parcialmente el campo rival.

Un caso muy frecuente y que atentaba contra el dinamismo del juego era el generado por la puja por la pelota entre dos robots sobre alguno de los laterales. Ejerciendo ambos robots fuerzas paralelas y en sentidos contrarios, la pelota quedaba atascada y el partido se tornaba lento y trabado. Para permitir la liberación de la pelota se le asignó al delantero una nueva funcionalidad *Patear Girando*. Esta acción asigna a ambas ruedas potencias máximas y opuestas (-125, 125), causando un movimiento corto y brusco que al tocar la pelota puede hacer que la misma se libere de la puja. Como se mencionó, estas situaciones se presentaban mayoritariamente sobre las paredes de la cancha, por lo que se definió otorgar velocidad positiva a la rueda más cercana a la pared y velocidad negativa a la rueda externa para propiciar la liberación de la pelota. El robot aplica este pateo y vuelve a su orientación inicial tantas veces como sea necesario hasta que la pelota se libere o se cobre un falta. Para evitar riesgos en el arco defendido, esta acción se aplica sobre las paredes laterales y zonas no peligrosas de la cancha.

En la estrategia original, si el robot elegido era un defensor, ambos delanteros tomaban una actitud pasiva esperando la pelota para iniciar una acción de ataque. Si el defensor se encontraba en campo propio, el delantero que cubría

la misma mitad de cancha (superior o inferior) donde se encontraba la pelota se posicionaba sobre la línea central para recibirla mientras el otro delantero se mantenía en su posición base. Con este accionar, se desperdiciaban jugadas cooperativas debido a qué no podían recibir una pelota enviada por el defensor. La mejora introducida consistió en enviar al delantero principal a una cierta distancia de la jugada y situándose en un lateral con respecto al desplazamiento de la pelota, de manera que un tiro hacia delante del defensor pueda pasar y la pelota pase a ser dominada por este delantero, resultando así en una jugada de pase implícita.

Observando situaciones en las que un delantero se convertía en el elegido y la pelota estaba más atrás que él pero se dirige a este lugar, se notó que, en su afán de ir hacia la pelota, abandonaba una buena posición que le hubiese permitido directamente impulsar la pelota hacia el arco rival. Por este motivo se incorporó una nueva función *Esperar Pelota* que, si determina que las condiciones son las adecuadas, en vez de tratar de posicionarse detrás de la pelota, orienta a los robots mirando hacia el arco a la espera de ésta. Con este nuevo comportamiento se logró mayor efectividad en el ataque y control de la pelota.

Si bien uno de los lineamientos hace referencia a que los jugadores no deben interferir entre sí, a los fines de aumentar la capacidad ofensiva del equipo, se determinó que cuando la pelota estuviera dentro del área grande rival ambos delanteros sean elegidos y, por lo tanto, acudan a la pelota. Con este accionar, se consiguen mayores posibilidades de ataque al arco rival.

14.3 Evolución de comportamientos individuales

En este trabajo, los comportamientos individuales que conforman la base de los comportamientos implementados son: *Navegación* y *Pateo*. De la misma manera que para la estrategia, para el desarrollo de estos comportamientos se han definido las premisas que se deben cumplir. Cada uno de los comportamientos

mencionados tienen asociadas distintas premisas. Para *Navegación* se intenta llegar a destino i) en forma rápida y ii) evitando colisionar con los obstáculos que se puedan interponer en el camino. Para *Pateo* se busca i) interceptar la pelota aunque la misma se encuentre moviéndose en cualquier dirección y ii) que la misma salga disparada en la dirección deseada. En ambos casos, se debe tener en cuenta que el simulador impone un tiempo máximo de respuesta: 60 veces por segundo el servidor debe recibir los comandos a ejecutar sobre los robots. Por este motivo, se debe prever que la velocidad de respuesta de los algoritmos implementados sea menor a un ciclo de simulación.

En la evolución de comportamientos también se aplicó el algoritmo de refinamiento presentado en la sección 14.2. En estos casos, además de la evaluación mediante partidos, fueron desarrolladas herramientas especiales que permitían hacer un seguimiento minucioso del recorrido realizado por los robots. Estas herramientas se describen la sección 14.5 de este capítulo.

14.3.1 Navegación

Existen básicamente dos alternativas al momento de elegir un algoritmo de navegación. Una opción es la planificación anticipada de toda la trayectoria entre el robot y la posición destino antes de cada movimiento. La segunda opción son los llamados métodos heurísticos, pues utilizan heurísticas para ir determinando cada uno de los movimientos que hará el robot a lo largo de su trayectoria.

En principio se abordó la técnica de planificación anticipada. El método de planeamiento utilizado consistía en generar una partición de la trayectoria buscada en una secuencia de arcos, donde el robot debía establecer ciertas velocidades a cada rueda por un intervalo de tiempo. Este método utilizaba el algoritmo de Powell [POW] para optimizar una función de costo que evaluba las bondades de la trayectoria planeada teniendo en cuenta el error en el destino alcanzado y las colisiones producidas en el camino [SAN]. Una consideración a realizar respecto

de este algoritmo es que el mismo necesita que se le indique la cantidad de arcos que se desea obtener. Inicialmente el método proponía en forma incremental valores entre 1 y 4 para este parámetro. En una segunda instancia, se utilizó una variación de este método que asignaba intervalos fijos de tiempo para todos los arcos [SAN]. A pesar de los intentos, el método no cumplía con una de las premisas establecidas: el tiempo de respuesta obtenido resultaba ser muy superior a la duración del ciclo de simulación, haciendo que no se pudiera devolver la información al servidor en el tiempo requerido.

Con el objetivo de cumplir con la premisa anterior, se comenzó a trabajar con métodos heurísticos. La versión inicial del algoritmo de navegación era muy sencilla y consistía en girar mientras la diferencia entre la orientación necesaria para ir en forma recta al objetivo y la orientación del robot fueran menores a una cierta cota. Una vez conseguida esa diferencia mínima el robot avanzaba a máxima velocidad hasta llegar al punto destino. Este algoritmo cumplía con la premisa básica de tiempo de respuesta, pero todavía no comtemplaba el resto de los objetivos que se querían alcanzar: evitar obstáculos en la trayectoria y hacer el recorrido en forma rápida. La lentitud en el desplazamiento se originaba porque cada vez que se debía ajustar el ángulo, el robot frenaba. Este algoritmo utilizaba predicción lineal de la pelota en un ciclo de simulación.

La primera mejora que se introdujo a este algoritmo apuntaba a reducir la distancia que debía recorrer el robot para alcanzar su objetivo, disminuyendo en consecuencia, el tiempo consumido. Para ello se adaptó el algoritmo haciendo que el robot se desplazara hacia adelante o hacia atrás, acortando el espacio recorrido y minimizando el ángulo de giro.

Un aspecto lateral de este problema resultó ser que, como el servidor de simulación provee valores reales para las posiciones de los objetos, y teniendo en cuenta la inercia que lleva el robot, se hacía muy dificultosa la ubicación del robot en un punto determinado de la cancha. Esto generaba la oscilación del robot

alrededor de la posición indicada debiendo realizar giros adicionales. Habiendo observaba esta situación y, considerando que no en todos los casos se requiere un punto exacto de llegada, cuando el robot no se dirigía a la pelota se distendieron los requisitos y se comenzó a indicar una zona destino, especificando el error aceptado, en lugar de un punto determinado.

La próxima premisa abordada fue tener en cuenta los obstáculos durante el desplazamiento. El enfoque utilizado fue el descrito en el capítulo 13 y consiste en un método heurístico donde el robot, al encontrar con obstáculos en la trayectoria a realizar, busca puntos adicionales a los costados de los obstáculos como destinos intermedios en la trayectoria. Con el método planteado los resultados obtenidos fueron satisfactorios, aunque mejoraron notablemente con la inclusión del método de predicción descrito en el capítulo 13.

Para incrementar la velocidad y la exactitud en la llegada, se planteó una partición en varios casos sobre el problema de navegación. Si el ángulo era menor a una cierta cota, el robot avanzaba en forma recta, estableciendo velocidad máxima en ambas ruedas. Si la diferencia de ángulos era muy grande el robot giraba sobre su eje a una velocidad proporcional a esta diferencia. En el caso intermedio entre los dos anteriores, se utilizaban velocidades distintas en cada rueda para producir un arco que fuera disminuyendo la diferencia de ángulos entre la orientación del robot y el punto destino. Un caso particular era cuando el ángulo de giro requerido era grande y el robot se desplazaba a altas velocidades. En este caso se procedía a frenar primero el robot y luego realizar el giro. Este método, con alguna pequeña variante sería el utilizado hasta el final de la implementación.

Simultáneamente a esta última implementación, se comenzó a aplicar en este algoritmo el método de predicción mencionado anteriormente y se introdujo el modelado de la inercia para calcular la cantidad de ciclos de juego a utilizar en la predicción de la posición futura de la pelota y el robot. Ambos desarrollos

fueron de suma importancia a la hora de evaluar la aplicación del algoritmo de navegación.

Las últimas modificaciones sobre el algoritmo se relacionan con el sentido de avance del robot. Hasta ese momento el robot buscaba el menor ángulo de giro, cambiando el sentido de desplazamiento si era conveniente. Esto producía que el robot se frenara e hiciera más lenta su navegación. Para mejorar esta situación se establecieron condiciones sobre el ángulo de giro y la velocidad actual del robot para considerar cambiar de sentido de desplazamiento. El robot mantendría su sentido a menos que la velocidad fuera baja o demasiado grande el ángulo de giro.

14.3.2 Pateo

Desde el inicio, el comportamento de pateo se planteó en dos etapas: una primera de aproximación, utilizando el algoritmo de navegación, y una segunda de pateo propiamente dicho.

La primera versión de este algoritmo consistía en aproximarse a un punto detrás de la pelota, muy cercano a la misma, utilizando el método de naveganción antes descripto y, una vez alcanzado el objetivo, activar alguna de las cinco rutinas de pateo: pateo hacia adelante, corto a la izquierda y a la derecha, girando a la izquierda y a la derecha. El primero consistía en ir en forma recta hacia adelante cuando existía una alineación entre el robot, la pelota y el arco. El pateo corto hacia la izquierda o derecha dejaba quieta una rueda y asignaba a la otra máxima velocidad, impulsando la pelota hacia la izquierda o la derecha respectivamente. Finalmente, el pateo girando a la izquierda o derecha se realizaba girando sobre su eje hacia uno u otro lado. La elección del método de pateo dependía de la relación entre la posición del robot, la pelota y el ángulo hacia donde se deseaba patear. Estos pateos no resultaron efectivos porque el robot se frenaba y la pelota se alejaba del mismo imposibilitando el pateo. Aún

cuando el robot lograba impactar la pelota no se conseguía darle la orientación deseada.

Para resolver esta deficiencia se estableció que el punto de aproximación a la pelota se definiera más lejano de manera que el robot tuviera espacio para maniobrar y llegar a la pelota con velocidad y la orientación adecuada. Para ello se mantuvo la utilización del algoritmo de navegación y se reemplazaron las rutinas de pateo por un nuevo algoritmo. En este nuevo planteo, el acercamiento se realizaba eligiendo un punto detrás de la pelota y dirigiéndose allí. Una vez que el robot entraba en el cono definido por la continuación de las rectas que pasan por cada poste del arco contrario y el punto elegido, se dirigía hacia la pelota. En cada paso de este proceso, el robot primero evaluaba si existía un arco de circunferencia que pasara por el centro del robot y de la pelota y que tuviera como tangentes la orientación del robot y la dirección que se le deseaba dar a la pelota. En este caso asignaba a las ruedas las velocidades necesarias para describir ese arco siempre y cuando no hubiera obstáculos en el medio y el arco estuviera inscripto en las dimensiones del campo de juego. Un caso particular se presentaba cuando el desplazamiento se realizaba por una recta (arco de circunferencia de radio infinito). Este método, con los agregados que se mencionan a continuación formaría la versión final de la rutina de pateo.

Aunque la incorporación de la predicción resultó muy positiva en muchos aspectos, combinar las posiciones futuras con la exactitud requerida por el pateo se tornaba una tarea compleja. En este caso, cuando el robot se encontraba muy cercano a la pelota, resultaba complicado mantenerse sobre algún arco apto para llegar a ella con el ángulo deseado. Cualquier pequeño error o desplazamiento hacía que el arco se curvara hacia un lado u otro y el robot oscilara en forma continua. Para solucionar este inconveniente, se implementó un pateo recto para los casos que se cumplieran dos condiciones: la primera era que el robot apuntara hacia el arco; la segunda era que la pelota estuviera en la banda comprendida entre las prolongaciones de los laterales del robot. Un caso especial se presenta cuando

el robot viene empujando la pelota sobre la pared del arco contrario. En este caso no existe una alineación entre robot, la pelota y el arco y, sin embargo, la acción que se está llevando a cabo es correcta. Para este caso particular también se habilitó el comportamiento pero agregando la condición de que al llegar al arco el robot girara empujando la pelota hacia el interior del mismo.

Otro inconveniente que se encontró fue que, tomando solamente el punto central del robot como referencia para el cálculo de la trayectoria, se desechaban casos en los que el robot hubiera llegado a la pelota con el ángulo correcto pero impactándola con un costado del frente produciendo el efecto deseado. Para ampliar las posibilidades de encontrar alguna trayectoria hacia la pelota se tomaron tres puntos interiores del robot para el cálculo de trayectorias.

Finalmente, una particularidad que se observó fue que cuando el robot se encontraba en el campo propio muchas veces no pateaba la pelota hacia el campo rival porque no podía dar dirección a la pelota hacia el arco contrario. La decisión fue distender el requisito que planteaba que el pateo debía ser en dirección al arco, ampliando el rango de pateo permitido. De esta forma, cuando la pelota se encontraba cerca de su arco, el requisito era que la pelota llegara al campo rival, restringiendo gradualmente esta condición hasta llegar a considerar sólo el arco cuando la pelota estaba a la altura del área rival. Este cambio también responde a los lineamientos generales de la estrategia que plantean el juego con dinamismo priorizando la permanencia de la pelota en campo rival.

14.4 Versiones de Simulador

El software de simulación de la categoría Middle League SimuroSot sufrió varios cambios desde el comienzo del desarrollo del equipo UBASot. Algunos de estos cambios impactaron directamente sobre la implementación, lo que obligó a llevar a cabo modificaciones en el equipo propio para adecuarse a los cambios del

simulador. En otros casos, se debieron estudiar y realizar pruebas tendientes a identificar errores e incluir rutinas que los contemplen en el mencionado software de simulación. A continuación se detallan algunos de los temas mencionados.

14.4.1 Referencia invertida de las Ruedas

Originalmente, los comandos enviados al simulador en cuanto a las ruedas izquierda y derecha, eran interpretados por el servidor en forma invertida, con lo cual se obtenían movimientos espejados a los deseados. Hasta que fue corregido en versiones posteriores, la implementación propia tuvo un módulo de adaptación de información que acomodaba el vector de potencias antes de enviarlo al simulador.

14.4.2 Coordenadas de Cancha Traspuestas

Si bien en la estructura de interfase que ofrece el simulador se incluyen las coordenadas de la cancha, dicha información está traspuesta con respecto a la que el simulador modela. En las primeras versiones del equipo UBASot se transformaba esta información para ser utilizada hasta que fuese corregido el problema en el simulador. Esta corrección no se realizó. Los responsables del servidor indicaron que en vez de usar la información de la interfase, se debían usar un conjunto de constantes que definían las coordenadas de la cancha. Se modificó, entonces, la implementación para utilizar estas constantes.

14.4.3 Estado de Juego

Si bien esta información figuraba en la interfase desde el la primera versión, la estructura no entregó los valores correctos sino en las últimas versiones del simulador. Puesto que las jugadas especiales se implementaron sobre el final del desarrollo, este inconveniente no supuso más que el esfuerzo necesario para

detectar el error. Sin embargo, un problema que nunca fue solucionado y que requirió un desarrollo especial fue el hecho que no se informa el fin de un estado especial y el pase al juego propiamente dicho. Para abordar este tema, se implementó internamente un estado de *Juego Normal* al cual se pasa cuando, estando en jugadas especiales, se detecta la puesta en movimiento de la pelota.

14.4.4 Eliminación de Potencias de Ruedas del Rival

Hasta la penúltima versión previa a la competencia, el simulador entregaba tanto para jugadores propios como para los rivales, la información relacionada a la potencia que se había dado a cada rueda del robot. En la implementación propia, se utilizaba esta información provista sobre el rival para predecir sus posiciones futuras. Eliminada esta información de la interfase, la predicción se reimplementó en base a la observación de las últimas posiciones del robot, lo que hace menos precisa la predicción debido a la inercia.

Si bien el cambio desde el punto de vista del modelo de simulación es más que justificado y razonable, se convirtió en un tema crítico debido a que el cambio en el servidor fue realizado en fecha muy cercana a la competencia. Sumado a que cada vez que se realizaron modificaciones en la interfase, la documentación de los cambios fue imprecisa y la mayoría de las veces no se correspondía con la estructura real, el tiempo invertido en los testeos de cada nueva versión del simulador y la adaptación del desarrollo propio interfirió en las tareas de desarrollo del equipo en sí.

14.5 Herramientas Utilizadas

La implementación del equipo consiste en desarrollar una dll que el simulador carga para ejecutarlo. Por esta causa no se pueden utilizar las herramientas tradicionales de depuración para encontrar y corregir los errores de

programación. Además, analizar el funcionamiento de los comportamientos implementados o la estrategia utilizada requiere conocer ciertos valores en cada ciclo de juego que no brinda el software de simulación. La simple visualización no permite darse cuenta si un determinado comportamiento se ha disparado o el cálculo de las velocidades asignadas al robot han sido calculadas eficientemente.

Para poder analizar estos aspectos se desarrollaron herramientas de análisis ad-hoc. Las mismas permitieron depurar errores de implementación y mejorar el diseño de los comportamientos básicos. Estas herramientas son programas que obtienen la información de juego a partir de archivos de log generados por la dll cargada en el simulador. Se desarrollaron dos tipos de programas: simuladores y generadores de situaciones.

El primer tipo de programa corresponde a aquellos que toman los logs generados por la dll. En base a esta información reproducen el juego grabado permitiendo visualizar ciertos datos y valores utilizados en el programa. Se han desarrollado tres programas de este tipo: sim, simpateo y simlog. Los dos primeros permitieron depurar y mejorar el algoritmo de pateo de los robots, mientras que el tercero es de uso más general y permitió mejorar la estrategia del equipo, en particular la elección del robot destinado a ir a la pelota.

El segundo tipo de programa corresponde a aquellos que se desarrollaron para generar situaciones hipotéticas para analizar la implementación de ciertos comportamientos y/o algoritmos. De este tipo de programas podemos mencionar a test, curva y curva2. El primero y el segundo están destinados a analizar las trayectorias para el pateo. El tercer programa estaba destinado a analizar una mejora al algoritmo de pateo.

14.5.1 Programas Simuladores

Los logs utilizados por estos programas varían en su contenido pero generalmente contienen información sobre la posición de cada objeto en la cancha y valores de variables utilizadas en el programa para decidir o calcular acciones y/o trayectorias. Los logs son generados por las dlls durante el transcurso del partido.

14.5.1.1 Sim y SimPateo

Estos programas tienen la misma utilidad. Son versiones distintas del mismo programa y permiten analizar con mas o menos detalle cada paso del robot que se dispone a patear la pelota. Van mostrando la posición actual y predicha del robot y la pelota, además de marcas y líneas para saber a cada paso que está haciendo el robot. Por ejemplo, si el robot ha calculado un arco hacia la pelota se dibuja en verde el círculo que contiene este arco y una línea azul que indica la futura trayectoria de la pelota luego del impacto.

Muestra, además los valores de las variables referidas al robot analizado y a la pelota como por ejemplo las velocidades actuales de las ruedas y las calculadas para el próximo ciclo, el ángulo final obtenido, etc.

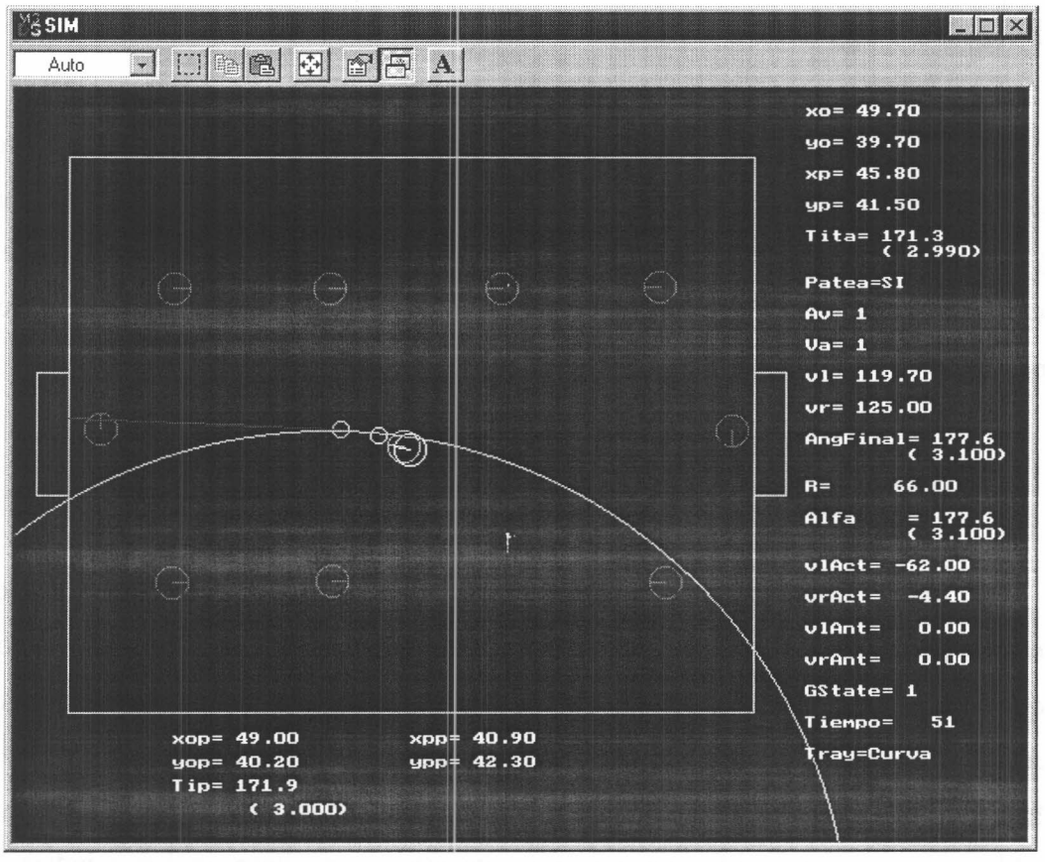


Figura 14.4: Programa Sim.
El circulo blanco indica la posición actual del robot y el circulo gris, la predicción a futuro. El circulo verde indica el arco calculado y la línea azul la dirección que tomaría la pelota luego del impacto.

14.5.1.2 SimLog

Como se dijo antes, este programa es de un propósito más general. Permite analizar la ubicación y datos de todos los robots en la cancha. Permite visualizar, para cada robot, su posición, orientación y velocidad de las ruedas. Para la pelota muestra sus coordenadas y, además, muestra el estado de juego. Cada equipo se individualiza por su color.

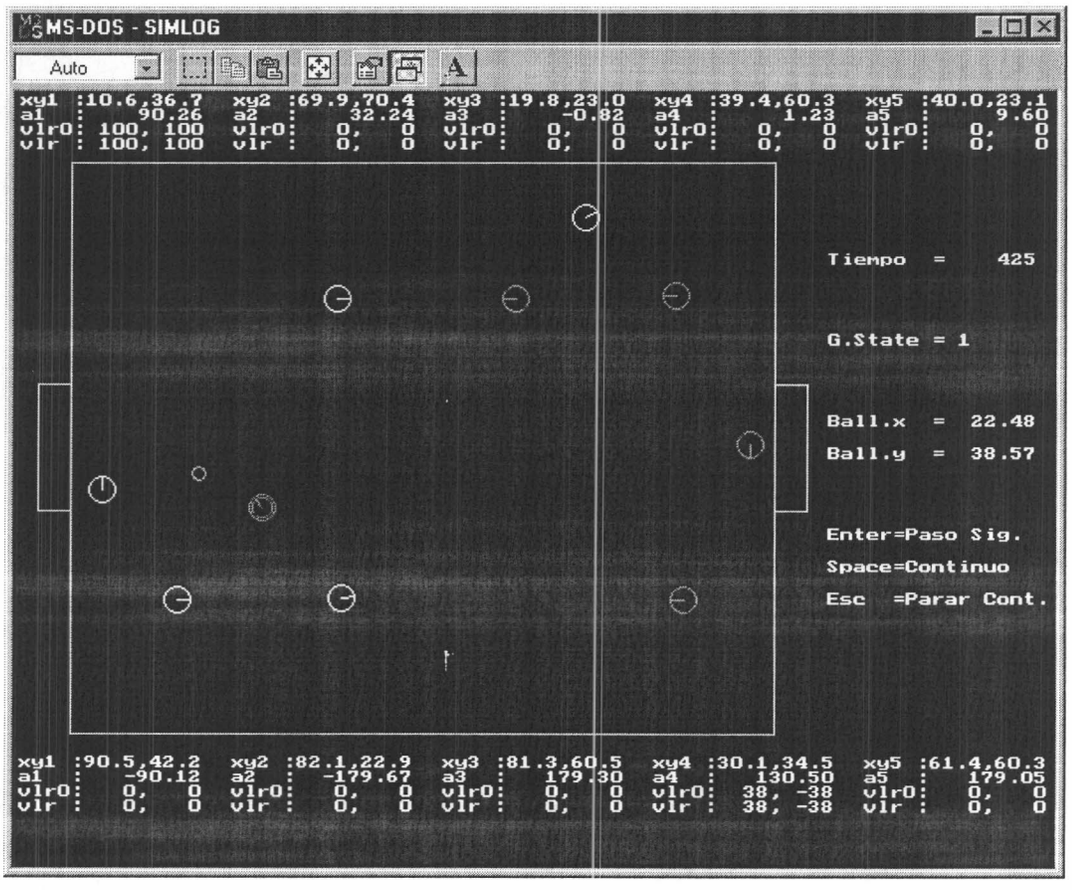


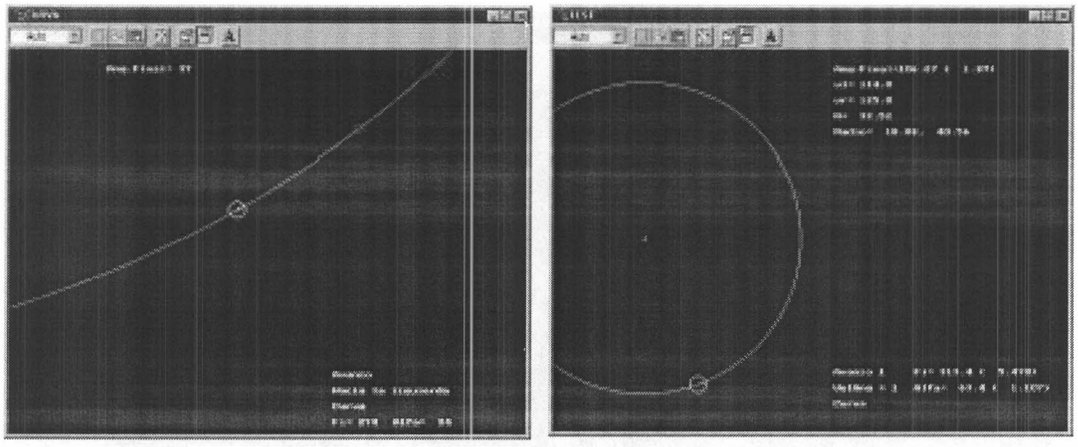
Figura 14.5: SimLog.
Aquí se ve un momento del partido. El equipo analizado es el azul y el robot elegido para ir a la pelota se individualiza con un doble círculo.

14.5.2 Programas Generadores de Situaciones

Estos programas no utilizan logs, sino que piden datos iniciales por teclado y simulan una situación hipotética.

14.5.2.1 Test y Curva

Estos dos programas testean el algoritmo de cálculo de trayectoria desde el robot hacia la pelota. Analizan todos los caminos del algoritmo para verificar su corrección.



(a) (b)

Figura 14.6: Curva y Test.
Se puede apreciar las trayectorias calculadas desde el robot hacia la pelota.

14.5.2.2 Curva2

Este programa analiza la mejora al algoritmo de pateo consistente en adaptar el ángulo del robot para conseguir un arco desde el robot a la pelota que llegue con el ángulo apropiado. El programa permite desplazar al robot por toda la cancha y cambiar su orientación, mostrando el arco descripto entre el robot y la pelota que tiene como tangente la orientación del robot en color gris, y en colores verde, rojo y azul las orientaciones que debería tener el robot para impulsar la pelota hacia el centro del arco o sus respectivos postes.

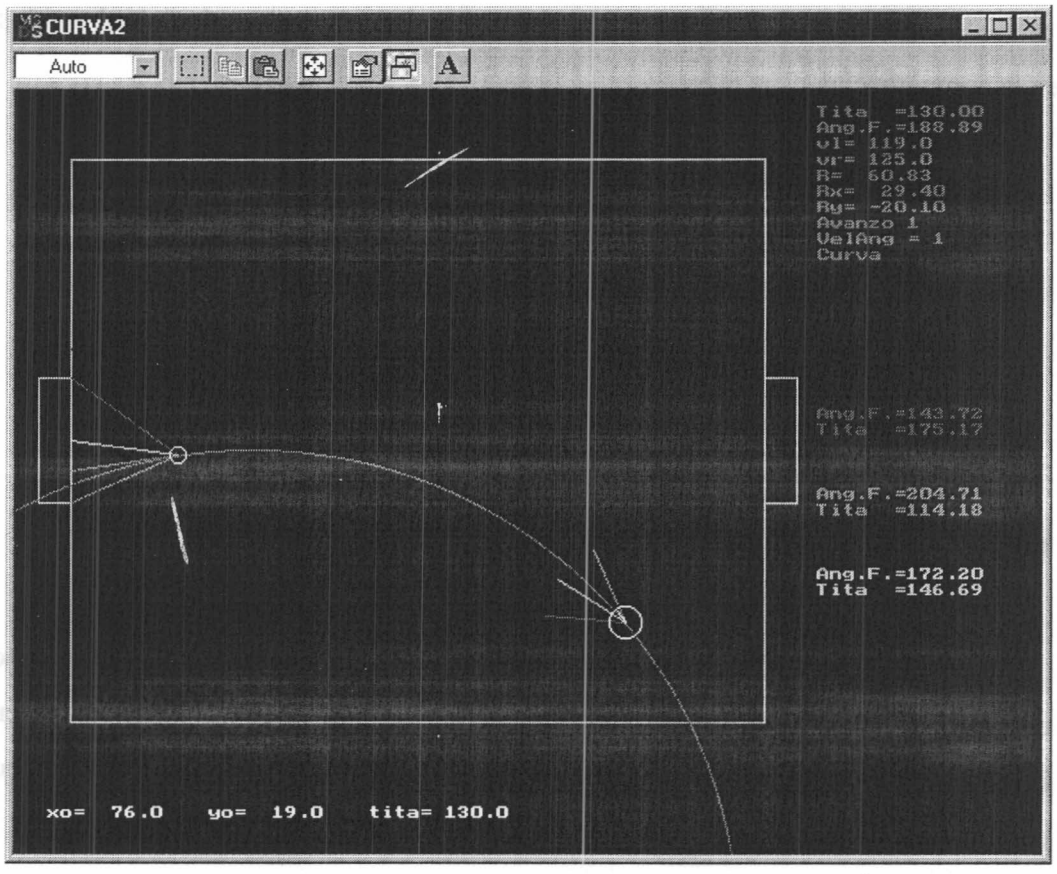


Figura 14.7: Cuva2
El programa muestra, a la derecha los valores de orientación y ángulo de llegada para cada caso.

14.6 Evaluación y Resultados

El equipo UBASot versión 2.10 participó en el campeonato mundial FIRA 2002 llevado a cabo en Corea entre el 23 y el 29 de Mayo de 2002, obteniendo el tercer puesto dentro de la categoría Middle League SimuroSot.

En esta categoría se presentaron 59 equipos provenientes de instituciones de 12 países. Cabe destacar que no existían límites de equipos por país, por lo cual países de la región presentaron varios equipos. El país organizador, Corea, tenía cuarenta equipos inscriptos en esta categoría.

14.6.1.1 Mediciones Globales

- ***Goles:*** Variable por excelencia que mide el resultado de un partido de fútbol. Se registraron los instantes en los cuales fueron convertidos los tantos de cada equipo así como también si se trató de un gol del rival o gol en contra.
- ***Tiempo de la pelota en cada campo:*** Se cronometró el tiempo acumulado en el cual la pelota estuvo en el campo rival, calculándose por diferencia el tiempo que permaneció en campo propio.

14.6.1.2 Intervenciones de Arqueros

Para ambos arqueros se midieron sus intervenciones, de manera de poder analizar el poder defensivo del equipo evaluado, como el poder ofensivo del rival. Los items cuantificados fueron:

- ***Cantidad de Pelotas Atajadas:*** Se consideró que el arquero atajó cuando realizó una acción tendiente a defender al arco y tomó contacto con la pelota.
- ***Cantidad de Cargas Recibidas:*** Se trata de jugadas en las que el rival tomó contacto con el arquero, entorpeciendo su tarea. Generalmente, estas jugadas concluyen en goles que son anulados.
- ***Cantidad de Goles Recibidos:*** Oportunidades en las cuales el arquero fracasó en la defensa del arco y fue cobrado gol.

14.6.1.3 Intervenciones de Jugadores

Para ambos equipos se midieron las intervenciones de sus jugadores según la zona de la cancha en la cual ocurrieron. Se tomó como intervención de un jugador al hecho de que éste altere el movimiento de la pelota, ya sea por impulso

u obstrucción. Las jugadas en las cuales un jugador lleva la pelota o se encuentra disputándola con un rival, donde en ambos casos existen múltiples contactos seguidos entre la pelota y el jugador, se contabilizaron como una única intervención. En estas mediciones no se incluyen las acciones de los arqueros. Los conteos sobre cada equipo se particionaron de la siguiente forma:

- Intervenciones en área propia
- Intervenciones en campo propio fuera del área
- Intervenciones en campo rival fuera del área
- Intervenciones en área rival

Cuando se habla de área o campo propio, siempre se considera desde el punto de vista del equipo UBASot. Es decir, área y campo rival es sobre el que se trata de avanzar y, área y campo propio es el que se intenta defender.

14.6.2 Resultados

Los resultados que se exponen a continuación surgen de las mediciones realizadas sobre veinte partidos jugados: cuatro contra cada rival. Los resultados obtenidos en la competencia de FIRA 2002 sólo se incluyen en la evaluación de los resultados ya que no se cuenta con las mediciones de las demás variables.

14.6.2.1 Cantidad de Goles

En la siguiente tabla se detallan los resultados obtenidos, incluidos los partidos de FIRA 2002, y el promedio de goles propios y rivales, por rival enfrentado.

Equipo Rival	FIRA 2002		Test 1		Test 2		Test 3		Test 4		Promedio	
	Goles Propios	Goles Rival	Goles Propios	Goles Rival	Goles Propios	Goles Rival	Goles Propios	Goles Rival	Goles Propios	Goles Rival	Goles Propios	Goles Rival
Furoec (Ecuador)	5	0	6	0	5	1	12	2	8	0	**7.2**	**0.6**
Superman (Corea)	5	3	2	1	4	2	5	0	6	3	**4.4**	**1.8**
AntiRats (Corea)	6	0	3	0	3	0	4	0	3	0	**3.8**	**0.0**
NewNeu (China)	1	2	1	4	0	1	0	4	0	4	**0.4**	**3.0**
Cosmos (Corea)	6	1	9	0	6	0	4	1	4	1	**5.8**	**0.6**

Tabla 14.2: Tabla de Resultados.

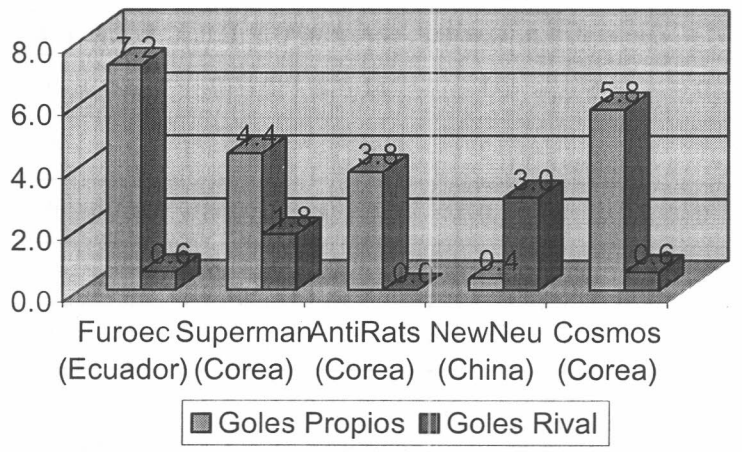


Gráfico 14.1: Cantidad promedio de goles propios y del rival por partido.

De la observación de la tabla de resultados se desprenden las siguientes particularidades:

- Con todos los rivales, siempre se mantuvo constante el vencedor de cada partido a lo largo de los cinco enfrentamientos evaluados.
- Los partidos no resultaron parejos. Se registró una diferencia de goles significativa a favor del ganador.

- Salvo contra NewNeu al cual el equipo UBASot nunca pudo ganarle, la diferencia de goles promedio obtenida por UBASot es amplia a su favor.
- El equipo Cosmos fue, después de Furoec, el equipo con el cual se obtuvo mayor cantidad de goles a favor. Si embargo, Cosmos obtuvo el cuarto lugar habiéndose convertido en el mejor de los cuarenta equipos coreanos participantes.

14.6.2.2 Goles en Contra

Para analizar los goles en contra, se discriminaron los goles de cada equipo según su origen y se totalizaron los goles de cada tipo.

Equipo Rival	Test 1				Test 2				Test 3				Test 4				Totales			
	G.P.	E.R.	G.R	E.P.	G.P.	E.R.	G.R	E.P.	G.P.	E.R.	G.R	E.P.	G.P.	E.R.	G.R	E.P.	G.P.	E.R.	G.R	E.P.
Furoec (Ecuador)	6	2	0	-	5	-	1	1	12	2	2	2	8	-	0	-	**31**	**4**	**3**	**3**
Superman (Corea)	2	-	1	1	4	-	2	-	5	-	0	-	6	-	3	-	**17**	**-**	**6**	**1**
AntiRats (Corea)	3	-	0	-	3	-	0	-	4	1	0	-	3	-	0	-	**13**	**1**	**-**	**-**
NewNeu (China)	1	-	4	2	0	-	1	-	0	-	4	2	0	-	3	1	**1**	**-**	**13**	**5**
Cosmos (Corea)	9	-	0	-	6	-	0	-	4	-	1	1	4	-	1	-	**23**	**-**	**2**	**1**

Tabla 14.3: Goles.
Goles propios (G.P.), goles en contra del rival (E.R), goles del rival (G.R.), goles en contra propios (E.P.). Los goles, tanto propios como del rival (G.P. y G.R) incluyen los goles en contra (E.R. y E.P.)

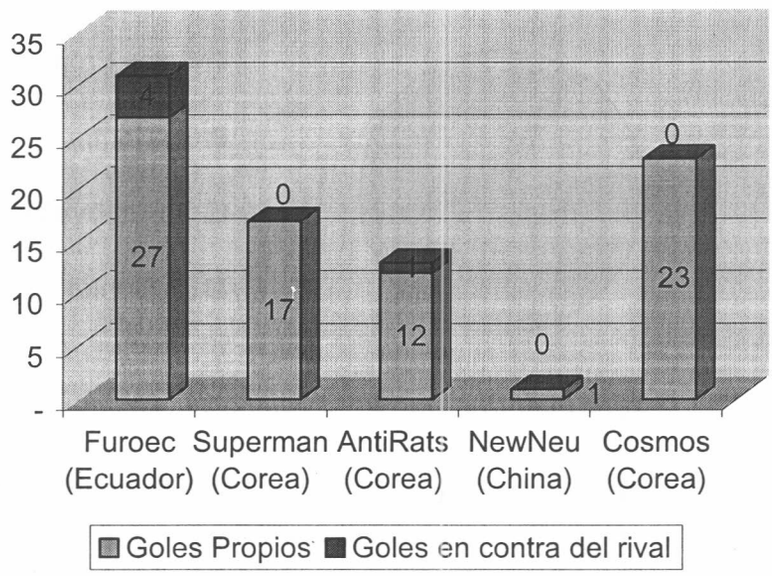


Gráfico 14.2: Relación entre goles convertidos y goles en contra del rival.

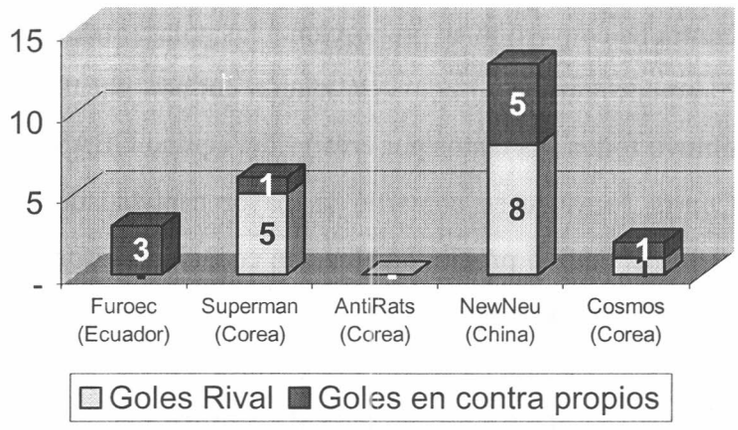


Gráfico 14.3: Relación entre goles convertidos por el rival y goles en contra propios.

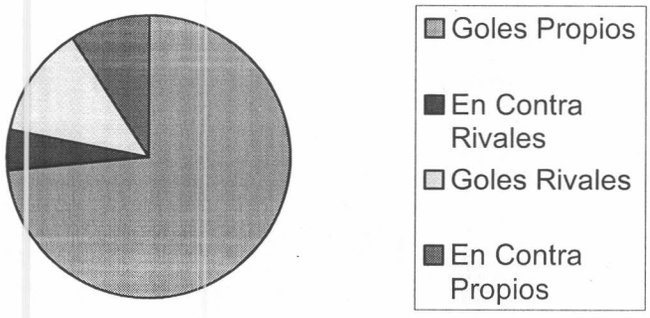


Gráfico 14.4: Distribución de Goles según origen.

Algunas particularidades a señalar en base a la información relevada sobre goles en contra observados son:

- De los goles rivales contra el equipo UBASot, gran parte han sido goles en contra (10 de 24).
- De todos los equipos evaluados, el equipo propio es el que más goles en contra ha convertido en proporción a los goles totales.
- El equipo Furoec no convirtió ningún gol con jugadores propios, sino que obtuvo todos sus tantos por errores del equipo UBASot.
- Contra NewNeu se convirtió la mayor cantidad de goles en contra por parte del equipo propio, aunque, sin considerar estos goles, el equipo rival también supera al propio

14.6.2.3 Tiempo de Permanencia de Pelota en Cada Campo

Se muestra a continuación el tiempo que la pelota permaneció en cada campo en cada partido y el promedio por rival.

Equipo Rival	Test 1		Test 2		Test 3		Test 4		Promedio	
	Campo Rival	Campo Propio	Campo Rival	Campo Propio	Campo Rival	Campo Propio	Campo Rival	Campo Propio	Campo Rival	Campo Propio
Furoec (Ecuador)	6:56	3:04	5:44	4:16	6:54	3:06	6:50	3:10	**6:36**	**3:24**
Superman (Corea)	5:32	4:28	6:34	3:26	5:54	4:06	7:06	2:54	**6:16**	**3:43**
AntiRats (Corea)	6:38	3:22	5:58	4:02	6:06	3:54	4:56	5:04	**5:54**	**4:05**
NewNeu (China)	3:04	6:56	3:28	6:32	6:04	3:56	4:40	5:20	**4:19**	**5:41**
Cosmos (Corea)	6:54	3:06	6:46	3:14	6:22	3:38	6:38	3:22	**6:40**	**3:20**

Tabla 14.4: Tiempo de permanencia de pelota en cada campo (en minutos).

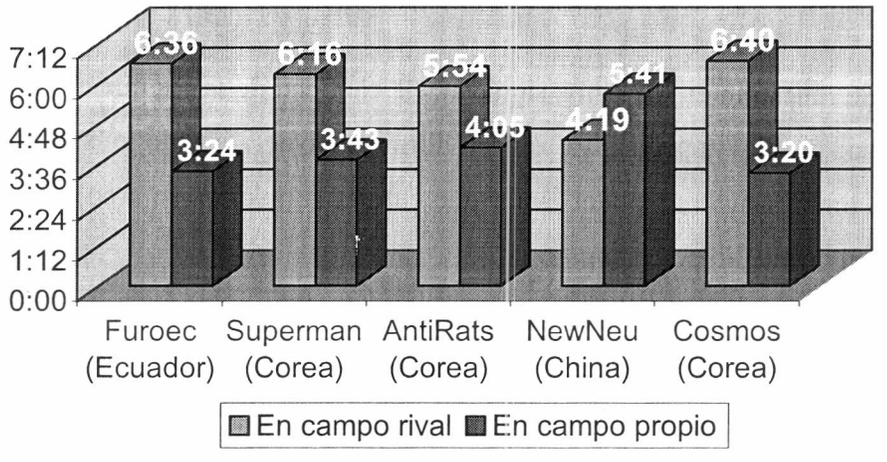


Gráfico 14.5: Tiempo de permanencia de la pelota en cada campo (en minutos).

El tiempo de permanencia de la pelota en cada campo, según las mediciones realizadas, muestra cierto semejanza con los resultados obtenidos ya que en todos los casos, la pelota ha permanecido – en promedio – más tiempo en el campo del equipo derrotado. Sin embargo, se pueden observar algunas particularidades:

- En los partidos contra los equipos vencidos por UBASot, la proporción de permanencia de la pelota en campo rival, ha sido del orden de dos tercios del tiempo total. En los partidos en los que UBASot fue dorrotado, la proporción de permanencia de la pelota en campo propio fue menor al promedio.

- Los tiempos promedio en los partidos contra FuroEc y Superman son muy similares siendo que Furoec fue el equipo con el cual se consiguió mayor diferencia de goles y, por el contrario, Superman fue, dentro de los equipos derrotados por UBASot, con el cual fueron más parejos los resultados.
- De los equipos derrotados, el que más tiempo mantuvo la pelota en el campo de UBASot fue Antirats y, sin embargo, fue el único equipo que no consiguió hacer goles.

14.6.2.4 Intervenciones de Arquero Propio

Se muestran en los datos siguientes la cantidad de veces que el arquero, ha intervenido en las jugadas, ya sea en forma activa (atajando o sufriendo una carga), como pasiva (sin poder evitar que entrara la pelota a su arco). Se muestra también el promedio por equipo rival.

Equipo Rival	Test 1				Test 2				Test 3				Test 4				Promedio			
	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot
Furoec (Ecuador)	2	0	0	2	2	0	1	3	3	0	2	5	2	0	0	2	**2.3**	**0.0**	**0.8**	**3.0**
Superman (Corea)	4	0	1	5	3	0	2	5	6	0	0	6	6	0	3	9	**4.8**	**0.0**	**1.5**	**6.3**
AntiRats (Corea)	8	0	0	8	9	0	0	9	6	0	0	6	6	1	0	7	**7.3**	**0.3**	**0.0**	**7.5**
NewNeu (China)	10	1	4	15	12	3	1	16	13	4	4	21	9	1	4	14	**11.0**	**2.3**	**3.3**	**16.5**
Cosmos (Corea)	5	0	0	5	6	1	0	7	7	2	1	10	6	0	1	7	**6.0**	**0.8**	**0.5**	**7.3**

Tabla 14.5: Intervenciones del arquero propio.
Cantidad de pelotas atajadas (Ataj), cargas recibidas (Chrg), goles no atajados (Gols) por el arquero propio e intervenciones totales (Tot).

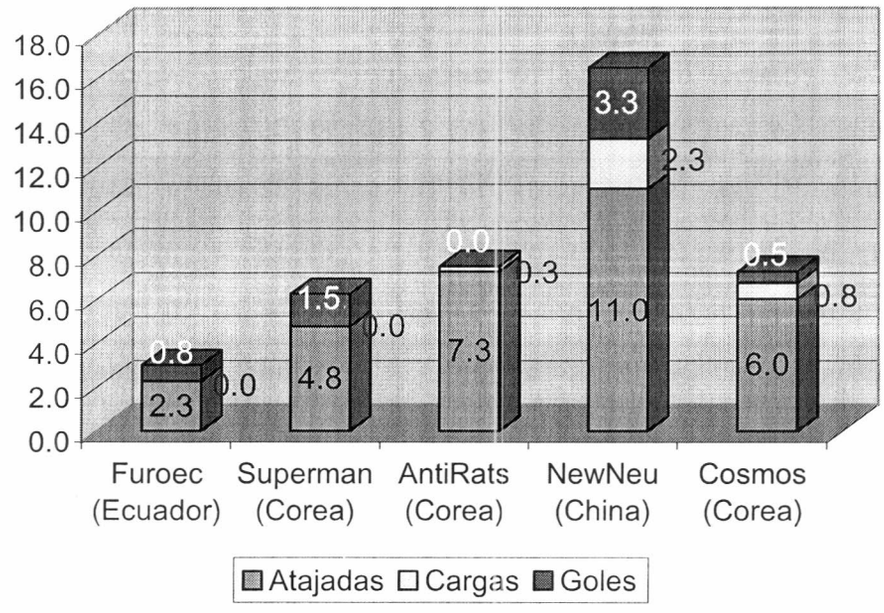


Gráfico 14.6: Cantidad Promedio de pelotas atajadas, cargas recibidas y goles no atajados por el arquero propio.

El arquero propio, como era de esperar, tuvo mayor cantidad de intervenciones frente a NewNeu que fue quien mayor cantidad de goles le convirtió al equipo propio. Las particularidades observadas son:

- Con Furoec se consiguió el menor grado de intervenciones del arquero propio.
- El equipo NewNeu llegó más del doble de veces que cualquiera de los otros equipos rivales.
- Antirats fue, después de NewNeu, el equipo que más veces llegó al arquero propio y, sin embargo, no convirtió ningún gol.

14.6.2.5 Intervenciones de Arquero Rival

Se muestran en la siguiente tabla las intervenciones del arquero rival, con iguales criterios que con el arquero propio.

Equipo Rivales	Test 1				Test 2				Test 3				*Test 4*				Promedio			
	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot	Ataj	Chrg	Gols	Tot
Furoec (Ecuador)	32	2	6	40	11	0	5	16	16	2	12	30	21	1	8	30	**20.0**	**1.3**	**7.8**	**29.0**
Superman (Corea)	18	6	2	26	19	2	4	25	20	3	5	28	22	0	6	28	**19.8**	**2.8**	**4.3**	**26.8**
AntiRats (Corea)	6	0	3	9	15	1	3	19	8	0	4	12	13	2	3	18	**10.5**	**0.8**	**3.3**	**14.5**
NewNeu (China)	1	0	1	2	4	0	0	4	3	0	0	3	0	0	0	0	**2.0**	**0.0**	**0.3**	**2.3**
Cosmos (Corea)	20	3	9	32	24	2	6	32	21	3	4	28	16	3	4	23	**20.3**	**2.8**	**5.8**	**28.8**

Tabla 14.6: Intervenciones del arquero rival.
Cantidad de pelotas atajadas (Ataj), cargas recibidas (Chrg), goles no atajados (Gols) por el arquero rival e intervenciones totales.

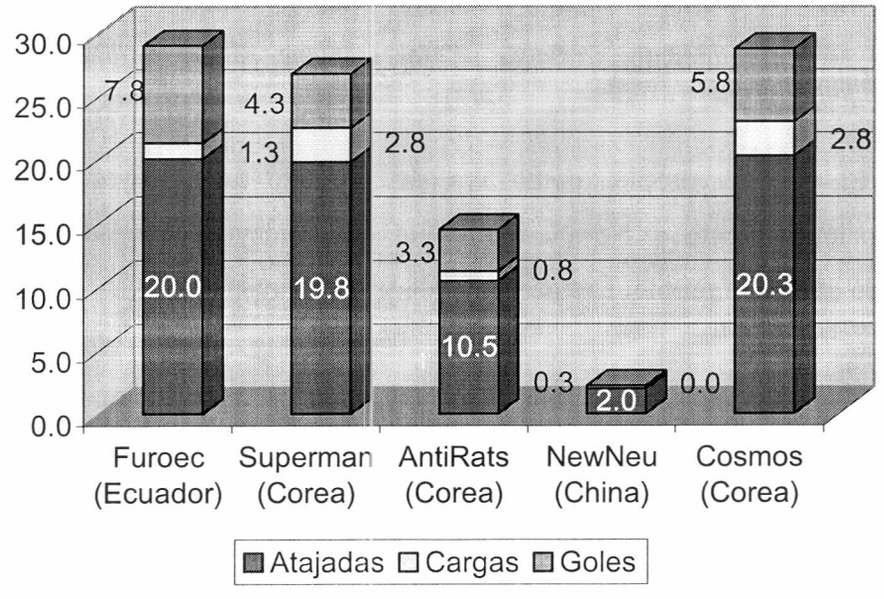


Gráfico 14.7: Cantidad Promedio de pelotas atajadas, cargas recibidas y goles no atajados por el arquero rival.

Se observa que la cantidad de intervenciones del arquero rival es proporcional a la cantidad de goles convertidos, lo que muestra que la estrategia de ataque no sufrió influencias por el comportamiento de los distintos arqueros

y/o que el desempeño de los arqueros es relativamente parejo. Algunos puntos a destacar son:

- Contra NewNeu se llegó en muy pocas oportunidades al arquero, por lo que la baja cantidad de goles convertidos se explica en las dificultades para llegar hasta el arco rival. Nótese que en el partido del Test 4 no hubo ninguna intervención del arquero chino.

14.6.2.6 Pateos de Jugadores Propios

La tabla siguiente muestra las cantidades de pateos de jugadores propios y promedio por partido, discriminado por sector de la cancha.

Equipo Rival	Test 1				Test 2				Test 3				*Test 4*				Promedio			
	AP	CP	CR	AR	AP	CP	CR	AR	AP	CP	CR	AR	AP	CP	CR	AR	AP	CP	CR	AR
Furoec (Ecuador)	0	34	56	18	1	44	42	7	0	48	39	17	0	33	44	15	**0.3**	**39.8**	**45.3**	**14.3**
Superman (Corea)	1	38	38	16	0	33	33	25	1	30	37	18	0	31	48	21	**0.5**	**33.0**	**39.0**	**20.0**
AntiRats (Corea)	1	20	24	9	0	32	19	7	0	30	18	6	2	43	21	14	**0.8**	**31.3**	**20.5**	**9.0**
NewNeu (China)	4	58	26	1	3	62	27	2	8	41	22	2	10	48	18	0	**6.3**	**52.3**	**23.3**	**1.3**
Cosmos (Corea)	0	34	45	32	1	43	31	27	5	31	40	18	3	27	31	21	**2.3**	**33.8**	**36.8**	**24.5**

Tabla 14.7: Pateos.
Cantidad de pateos realizados por jugadores propios por sector de la cancha. AP: Area propia; CP: Campo propio (excluyendo área); CR: campo rival (excluyendo área); AR: Area rival.

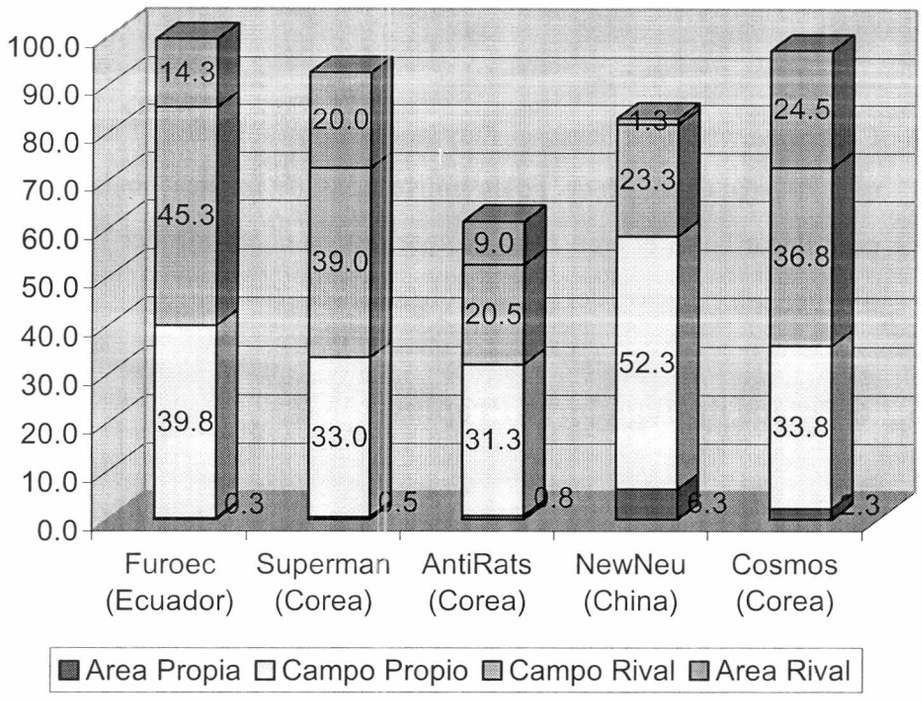


Gráfico 14.8: Cantidad Promedio de pateos realizados por jugadores propios.

Las cantidades de jugadas por sector se corresponden también con los resultados obtenidos. Se puede observas en los partidos contra NewNeu la gran cantidad de intervenciones sobre campo propio, incluso superiores a las cantidades de intervenciones del rival en su propio área (Tabla 14.8). Esto muestra una activa tarea de defensa ya que, en comparación, la cantidad de veces que el rival llegó hasta el arquero propio fueron relativamente pocas. Otras particularidades observadas son:

- Contra Antirats, con pocas jugadas sobre campo rival, se alcanzó una alta tasa de goles.
- En los partidos contra Superman, se llegó mucho al área rival y sin embargo la cantidad de goles fue relativamente baja.
- Se nota la gran dificultad que tenía el equipo propio para llegar al área rival contra NewNeu, lo hace pensar que la falta de goles contra este

equipo se debe principalmente a una sólida defensa del rival más que en fallas de ataque.

14.6.2.7 Pateo de Jugadores Rivales

La siguiente tabla contiene similares datos al anterior, con respecto a jugadores rivales. Nótese que la denominación de los sectores de la cancha se mantienen con respecto al equipo UBASot.

Equipo Rival	Test 1				Test 2				Test 3				*Test 4*				Promedio			
	AR	CR	CP	AP	AR	CR	CP	AP	AR	CR	CP	AP	AR	CR	CP	AP	AR	CR	CP	AP
Furoec (Ecuador)	19	37	22	2	10	49	33	2	6	52	19	1	9	47	24	0	**11.0**	**46.3**	**24.5**	**1.3**
Superman (Corea)	10	33	19	0	6	39	29	1	3	56	23	5	5	44	23	4	**6.0**	**43.0**	**23.5**	**2.5**
AntiRats (Corea)	13	31	14	5	11	33	23	6	8	38	14	0	3	28	36	1	**8.8**	**32.5**	**21.8**	**3.0**
NewNeu (China)	4	37	48	9	7	32	56	10	4	42	36	10	3	36	34	6	**4.5**	**36.8**	**43.5**	**8.8**
Cosmos (Corea)	6	47	25	0	13	47	28	2	3	29	19	5	4	56	16	1	**6.5**	**44.8**	**22.0**	**2.0**

Tabla 14.8: Cantidad de pateos realizados por jugadores rivales por sector de la cancha.
AR: Area rival; CR: campo rival (excluyendo área); CP: Campo propio (excluyendo área);. AP: Area propia.

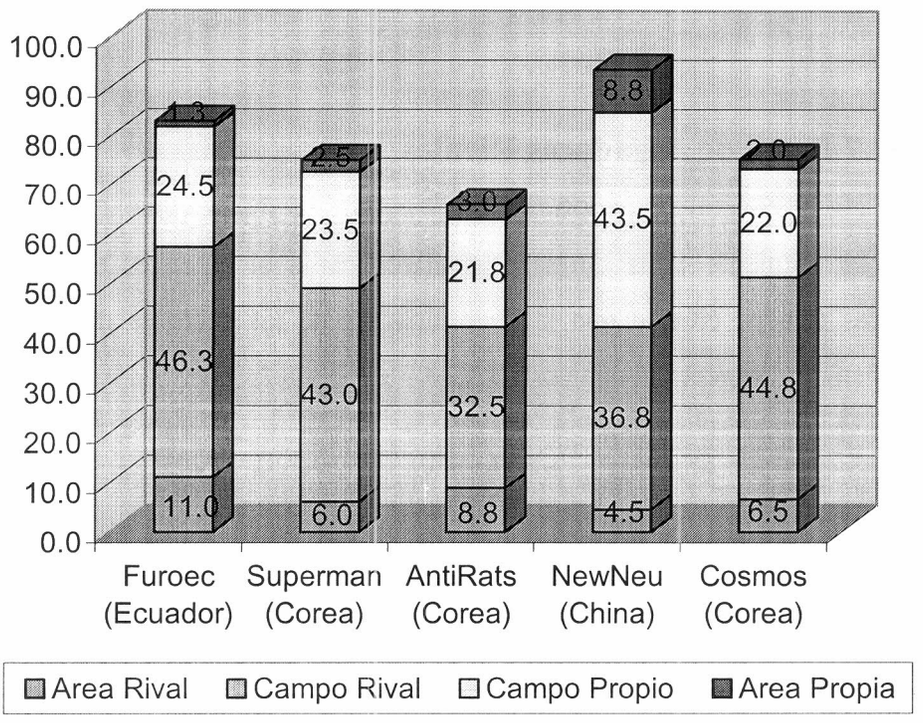


Gráfico 14.9: Cantidad Promedio de pateos realizados por jugadores rivales.

De estos datos se pueden mencionar la siguientes particularidades:

- Los jugadores rivales han tenido muchas menos llegadas al área propia que los propios a la rival.
- Las cantidades de pateos en el área central de la cancha, han sido muy similares entre los equipos derrotados por UBASot, salvo en el caso de Antirats que mostró una menor cantidad de acciones sobre el campo propio.

14.6.2.8 Total de Jugadas por Sector de la Cancha

Aquí se ha realizado una composición de las mediciones anteriores en una única tabla de la siguiente manera:

- ***Jugadas en Área Propia:*** intervenciones totales de arquero propio + Pateos de jugadores propios en área propia + pateos de jugadores rivales en área propia.
- ***Jugadas en Campo Propio:*** pateos de jugadores propios en campo propio + pateos de jugadores rivales en campo propio.
- ***Jugadas en Campo Rival:*** pateos de jugadores propios en campo rival + pateos de jugadores rivales en campo rival.
- ***Jugadas en Área Rival:*** intervenciones totales de arquero rival + Pateos de jugadores propios en área rival + pateos de jugadores rivales en área rival.

La denominación de los sectores de la cancha, como en las tablas anteriores, corresponden al punto de vista del equipo UBASot.

Equipo Rival	Test 1				Test 2				Test 3				*Test 4*				Promedio			
	AP	CP	CR	AR	AP	CP	CR	AR	AP	CP	CR	AR	AP	CP	CR	AR	AP	CP	CR	AR
Furoec (Ecuador)	4	56	93	77	6	77	91	33	6	67	91	53	2	57	91	54	**4.5**	**64.3**	**91.5**	**54.3**
Superman (Corea)	6	57	71	52	6	62	72	56	12	53	93	49	13	54	92	54	**9.3**	**56.5**	**82.0**	**52.8**
AntiRats (Corea)	14	34	55	31	15	55	52	37	6	44	56	26	10	79	49	35	**11.3**	**53.0**	**53.0**	**32.3**
NewNeu (China)	28	106	63	7	29	118	59	13	39	77	64	9	30	82	54	3	**31.5**	**95.8**	**60.0**	**8.0**
Cosmos (Corea)	5	59	92	70	10	71	78	72	20	50	69	49	11	43	87	48	**11.5**	**55.8**	**81.5**	**59.8**

Tabla 14.9: Total de jugadas por sector de la cancha.
AP: Area propia; CP: Campo propio (excluyendo área); CR: campo rival (excluyendo área); AR: Area rival.

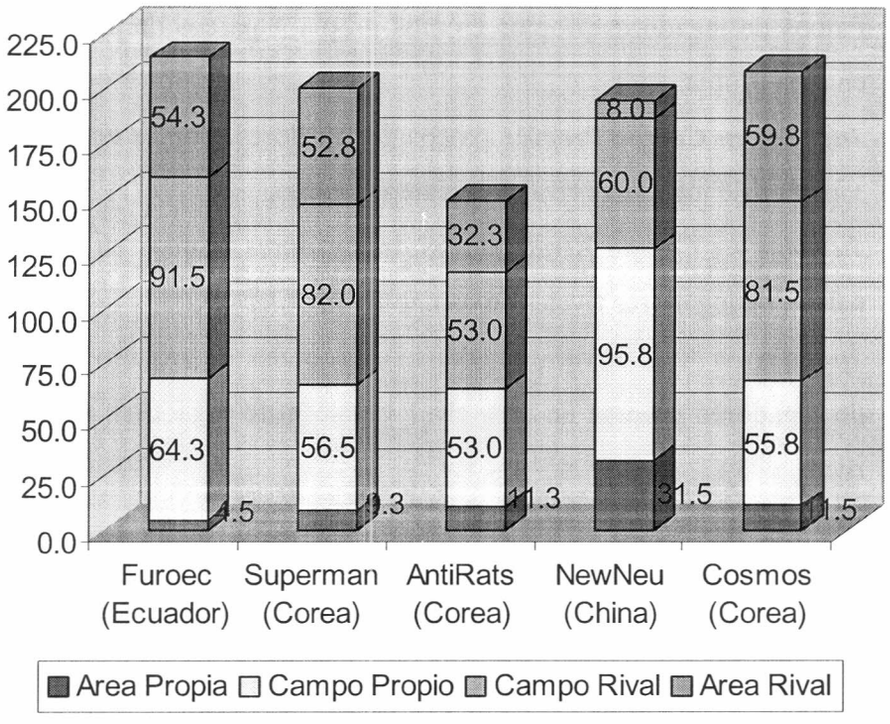


Gráfico 14.10: Cantidad promedio jugadas por sector de la cancha.

De esta información se pueden realizar las siguientes observaciones:

- Furoec ha sido el equipo que menos situaciones de riesgo le generó a UBASot.
- Contra Antirats la cantidad de jugadas en cada campo ha sido pareja, pero la diferencia se dió en llegadas a su área.
- Los partidos contre NewNeu se desarrollaron, mayoritariamente, sobre el campo de UBASot.

14.7 Conclusiones Generales de la Implementación

De los resultados obtenidos en las mediciones sobre los varios aspectos arriba mencionados, se desprenden una serie de conclusiones sobre el desempeño del equipo UBASot 2.10:

- En todas las variables analizadas, el equipo UBASot 2.10 posee un mejor desempeño que los equipos Furoec (Escuela Politécnica del Litoral - Ecuador), Superman (Konyang University Corea), AntiRats (Myongji University - Corea) y Cosmos (Konyang University - Corea) ratificándose los resultados obtenidos durante la competencia FIRA 2002.
- En nungún aspecto el equipo UBASot 2.10 ha superado al equipo NewNeu (Northeastern University - China) ratificándose el resultado obtenido en el enfrentamiento de ambos equipos durante la competencia en Corea.
- El equipo UBASot 2.10 debe mejorar la estrategia de manera tal de bajar el índice de goles en contra que actualmente registra.

Capítulo 15
Conclusiones

15.1 Contribuciones

Este trabajo de tesis ha presentado un relevamiento del estado del arte respecto del fútbol de robots y de las distintas técnicas utilizadas en referencia a sistemas multiagentes y desarrollo de habilidades especiales sobre robots, el cual constituye un marco teórico para futuros trabajos y desarrollos, así como también conforma un material de consulta para quienes deseen dar sus primeros pasos dentro del ambiente de fútbol de robots.

Adicionalmente, se propuso un enfoque de diseño para equipos de fútbol de robots en el cual los problemas a resolver han sido presentados en una estructura de niveles. Estos niveles dividen al problema en modelado del entorno, acciones, comportamientos elementales, comportamientos complejos, jugadas, tácticas y estrategias. Esta clasificación constituye uno de los principales aportes del presente trabajo.

Por último, se describió el diseño del equipo UBASot desarrollado y presentado en Corea, en el marco de la *2002 FIRA Robot Soccer Wold Championship* en la categoría *Middle League SimuroSot*. Luego, se ha expuesto la evolución del diseño a lo largo del desarrollo. Por último, se presentaron los resultados de las mediciones de performance realizadas.

15.2 Limitaciones

En este trabajo se abordó el tema del fútbol de robots desde una visión general, intentando abarcar todos los aspectos a tener en cuenta y tratando de orientar al lector sobre la forma de abordar el tema. Se han enumerado los distintos temas y niveles del problema y se han descripto algunos enfoques tomados hasta el momento.

Con el propósito de no perder en ningún momento la perspectiva global del problema se le ha dado un alcance limitado al presente trabajo. Cada uno de los temas fue abordado en forma general, en un primer nivel de profundidad, obviando detalles y mencionando sólo los enfoques que tratan al tema desde puntos opuestos o suficientemente distintos. También se abordó el tema del hardware utilizado en forma muy enunciativa, pues en todo momento se intento enfocar el tema del fútbol de robots desde el punto de vista del comportamiento cooperativo y el procesamiento de la información sensorial.

Respecto a la implementación del equipo de fútbol simulado detallada en la Parte III de este trabajo, el principal factor que limitó el desarrollo fue el hecho de tener como uno de los objetivos participar en el campeonato mundial organizado por FIRA en mayo del 2002. Este hecho significó un límite de tiempo en el cual se debía presentar un equipo competitivo que tuviera implementadas las características esenciales descriptas en este trabajo.

15.3 Trabajos Futuros

En base al alcance expuesto en la sección anterior se presentan a continuación algunos conceptos que pueden ser ampliados o tratados en forma exhaustiva en futuros trabajos.

Respecto al abordaje de la problemática del fútbol de robots, se puede analizar con mayor profundidad cada uno de los aspectos a tener en cuenta al momento de desarrollar un equipo. Este análisis deberá tener en cuenta los enfoques que, por similitud a los presentados o por no brindar más datos relevantes, no han sido incluidos en el presente trabajo. Así mismo, se puede incursionar en el relevamiento de los componentes de hardware utilizados en el desarrollo de equipos de fútbol de robots.

La implementación realizada y expuesta en la Parte III de un equipo de simulación puede ser mejorada y ampliada. Al respecto, se pueden destacar, entre otros, los siguientes puntos centrales:

- Aplicar técnicas de aprendizaje a los comportamientos y acciones
- Introducir modelado del equipo oponente
- Desarrollar estrategias dinámicas de juego
- Implementar pases entre jugadores
- Analizar nuevos algoritmos de navegación que permitan mejorar la performance del equipo.

El fútbol de robots provee un amplio campo de experimentación para el desarrollo del conocimiento y de la tecnología, que abre las puertas a una gran cantidad de trabajos que aporten soluciones innovadoras o complementarias a las existentes, brindando nuevos puntos de vista que permitan un avance en las tecnologías involucradas en el ambiente del fútbol de robots.

Referencias

[ABE] H. Aberg. *"Agent Roles in RoboCup Teams"*. Tesis, Kungliga Tekniska Högskolan School of Computer Science and Technology, 1998.

[ADO] G. Adorni, A. Bonarini, G. Clemente, D. Nardi, E. Pagello, M. Piaggio. *"Art'00 – Azzurra Robot Team for the Year 2000"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 64-75. Springer, 2001.

[AMO] C. Amoroso, A. Chelia, V. Morreale, P. Storniolo. *"A segmentation System for Soccer Robot Based on Neural Networks"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 136-147. Springer, 2000.

[AND] T. Andou. *"Refinement of Soccer Agent's Positions Using Reinforcement Learning"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 373-388. Springer, 1998.

[ARS] S. Arseneau, W. Sun, C. Zhoa, J. Cooperstock. *"Inter-Layer Learning Towards Emergent Cooperative Behavior"*. 17th Annual Conference on Artificial Intelligence, American Association of Artificial Intelligence, Austin, Texas. páginas 3-8. 2000

[ASA] M. Asada, H. Kitano, I. Noda, M. Veloso. *"RoboCup: Today and Tomorrow What We Have Learned"*. Artificial Intelligence 110: páginas 193-214. 1999.

[BAL] J. Baltes, N. Hildreth. *"Adaptative Path Planner for Highly Dynamic Environments"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 76-85. Springer, 2001.

[BAR] R. Barman, S. Kingdon, A. Mackworth, D. Pai, M. Sahota, H. Wilkinson, Y. Zhang. *"Dynamite: A Testbed for Multiple Mobile Robots"*. En Proceedings, IJCAI-93 Workshop on Dynamically Interacting Robots', Chambery, France, páginas 38-44. 1993.

[BEH] S. Behnke, B. Frötschl, R. Rojas, P. Ackers, W. Lindstrot, M. Melo, A. Schebesch, M. Simon, M. Sprengel, O. Tenchio. *"FU-Fighters Team Description"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 667-670. Springer, 2000.

[BER] A. Berry. *"Soccer Robot with Local Vision"*. 1999 Honorous Dissertation. Centre for Intelligent Information Processing Systems. Department of Electrical & Electronic Engineering. University of Western Australia. 1999.

[BON] A. Bonarini. *"The Body, the Mind or the Eye, First?"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 210-219. Springer, 2000.

[BRC1] J. Bruce, T. Balch, M. Veloso. *"Fast and Inexpensive color image segmentation for interactive robots"*. En Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00), Vol.3, páginas 2061-2066. 2000.

[BRC2] J. Bruce. *"Realtime Machine Vision Perception and Prediction"*. Undergraduate thesis, Carnegie Mellon University. 2000.

[BRU1] J. Brusey, L. Padgham. *"Techniques for Obtaining Robust, Real-Time Colour-Based Vision for Robotics"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 243-253. Springer, 2000.

[BRU2] J. Brusey, A. Jennings, M. Makies, C. Keen, A. Kendall, L. Padgham, D. Singh. *"RMIT Raiders"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 741-744. Springer, 2000.

[BRU3] J. Brusey, A. Jennings, M. Makies, C. Keen, A. Kendall, L. Padgham. *"RMIT Raiders"* M. Asada, H. Kitano (Ed.), RoboCup-98: Robot Soccer World Cup II, páginas 741-744. Springer, 1999.

[BUC] S. Buck, M. Riedmiller. *"Learning Situation Dependent Success Rates of Actions in a RoboCup Scenario"*. PRICAI 2000: 809. 2000.

[BUR] H. Burkhard, M. Hannebauer, J. Wendler. *"AT-Humboldt – Development, Practice and Theory"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 357-372. Springer 1998.

[CAS] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio. *"Coordination among heterogeneous robotic football players: ART in the F-2000 League"*. RoboCup workshop, Amsterdam. 2000.

[CHA] C. Chang, A. Ghose, J. Lipman, P. Harvey. *"Gongeroos'99"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 572-575. Springer, 2000.

[CHE] M. Cheny, E. Foroughi, F. Heintz, Z. Huangy, S. Kapetanakis, K. Kostiadis, J. Kummeneje, I. Noda, O. Obst, P. Riley, T. Steffens, Y. Wangy, X. Yiny. (2000) *"Users Manual RoboCup Soccer Server for Soccer Server Version 7.07 and later"*. RoboCup Federation. 2001.

[DAN] R. D'Andrea, T. Kalmar-Nagy, P. Ganguly, M. Babish. *"The Cornell RoboCup Team"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 41-51. Springer, 2001.

[DAS] A. Da Silva Simoes y A. Reali Costa. *"Using Neural Color Classification in Robotic Soccer Domain"*. En International Joint Conference IBERAMIA'2000 and SBIA'2000, Workshop Proceedings, Meeting on Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning. L. N. Barros, R. M. Cesar Jr., F. G. Cozman, A. H. Reali Costa (eds.). São Paulo, Atibaia, SP, Brasil, páginas 208-213. 2000.

[DEM] K. Demura, K. Miwa, H. Igarashi, D. Ishihara. *"The Concept of Matto"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 723-726. Springer, 2000.

[DIC] I. Dichiera. *"Robot Soccer with Global Vision"*. Tesis. Centre for Intelligent Information Processing Systems. Department of Electrical & Electronic Engineering. University of Western Australia. 1999.

[DOR] K. Dorer. *"Extended Behavior Networks for the MagmaFreiburg Team"*. RoboCup-99 Team Descriptions for the Simulation League, páginas 79-83. Linkoping University Press, 1999.

[EME] R. Emery, T. Balch, J. Bruce, S. Lenser, M. Osterfeld, R. Shern, K. Sicorski, A. Stroupe, J. Sweeney, M. Veloso. *"CMU Hammerheads Team Description"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 575-578. Springer, 2001.

[FIR] *The Federation of International Roboc-soccer Associations* (www.fira.net).

[GAL] G. Gallinelli, G. Gritan, J. Ramírez. *"FutBotIII: Towards a Robust Centralized Vision System for RoboCup Small League"*. Linköping Electronic Articles in Computer and Information Science Vol. 4 No. 006/07. 1999.

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

[GUT1] J. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, T. Weigel, B. Welsh. *"The CS Freiburg Team"*. M. Asada (Ed.), RoboCup-98: Robot Soccer World Cup II. Proceedings of the second RoboCup Workshop, Paris, France. 1998.

[GUT2] J. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, T. Weigel, B. Welsh. *"The CS Freiburg Robotic Soccer Team: Reliable Self-Localization, Multirobot Sensor Integration, and Basic Soccer Skills"*. M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 93-108. Springer, 1999.

[HAN] K. Han, M. Veloso. *"Physical Model Based Multi-objects Tracking and Prediction in RoboSoccer"*. Working Note of the AAAI 1997 Fall Symposium. AAAI, MIT Press, 1997.

[HEN] B. Hengst, D. Ibbotson, S. Pham, J. Dalgliesh, M. Lawther, P. Preston, C. Sammut. *"The UNSW RoboCup 2000 Sony Legged League Team"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 64-75. Springer, 2001.

[HHU] H. Hu, K. Lostiadis, M. Hunter, M. Seabrook. *"Essex Wizards'99 Team Description"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 558-562. Springer, 2000.

[JAM] M. Jamzad, A. Foroughnassiraei, T. Aaghai, V. Mirrokni, R. Ghorbani, A. Heydar Noori, M. Kazemi, H. Chitsaz, F. Mobasser, M. Maghaddam, M. Gudarzi, N. Ghaffarzadegan. *"A Goal Keeper for Middle Size RoboCup"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 583-586. Springer, 2001.

[KIM1] J. Kim, P. Vadakkepat, I. Verner. *"The FIRA Robot World Cup Initiative and Research Directions"*. En International Journal of Robotics and Autonomous Systems. 1999.

[KIM2] J. Kim, H. Shm, H. Kim, M. Jung, P. Vadakkepat. *"MICRO-ROBOT SOCCER SYSTEM: Action Selection Mechanism and Strategies"*. En Proc. of the 3rd ECPD Int. conference on Advanced Robotics, Intelligent Automation and Active Systems, European Centre for

Peace and Development - ECPD, Bremen, Germany, páginas 151-156. 1997.

[KIN] S. Kinoshita, Y. Yamamoto. *"11 Monkeys Description"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 550-553. Springer, 2000.

[KIT1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa. *"RoboCup: The Robot World Cup Initiative"*. En Proceedings of the IJCAI-95 Workshop on Entertainment and AI/ALife. 1995.

[KIT2] H. Kitano, M. Asada *"RoboCup Humanoid Challenge: That's One Small Step for a Robot, One Giant Leap for Mankind"*. En Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-98). 1998.

[KIT3] H. Kitano, M. Tambe, P. Stone, M. Voloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, M. Asada. *"The RoboCup Synthetic Agent Challenge 97"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 62-73. Springer 1998.

[LUB] J. Lubbers, R. Spaans. *"The Priority/Confidence Model as a Framework for Soccer Agents"* M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 162-172. Springer, 1999.

[MAE] P. Maes. *"How to do the right thing"*. Connection Science, 1989. 1(3): páginas 291-323. 1989.

[MAI] F. Maire, D. Taylor. *"A Quadratic Programming Formulation of a Moving Ball Interception and Shooting Behavior, and Its Application to Neural Network Control"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 327-332. Springer, 2001.

[MAR] S. Marsella, J. Adibi, Y. Al-Onaizan, A. Erdem, R. Hill, G. Kaminka, Z. Qiu, M. Tambe. (1998) *"Using an Explicit Teamwork Model and Learning in RoboCup: An Extended Abstract"*. M. Asada (Ed.), Proceedings of the second RoboCup Workshop. 1998.

[MAT] T. Matsumura. *"Description of Team Erika"*. M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 377-383. Springer, 1999.

[MEN] E. Menegatti, M. Wright, E. Pagello. *"A New Omnidirectional Sensor for the Spatial Semantic Hierarchy"*. IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '01). 2001.

[MON] F. Mondada, D. Floreano. *"Evolution and mobile Autonomous Robots"*. Towards Evolvable Hardware, The Evolutionary Engineering Approach, Papers from an international workshop, Lausanne, Switzerland, páginas 221-249. Springer, 1996.

[MOR] C. Moreno, A. Suárez, Y. Amirat, E. González, H. Loaiza. *"Real MagiCol 99: Team Description"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 735-740. Springer, 2000.

[NAK1] T. Nakamura, K. Terada A. Shibata, H. Takeda. *"The RoboCup-NAIST: A cheap Multisensor-Based Mobile Robot with Visual Learning Capability"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 326-337. Springer, 2001.

[NAK2] T. Nakamura, K. Terada, A. Shibata, H. Takeda. *"The RoboCup-NAIST: A cheap multisensor-based mobile robot with visual learning capability"*. M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 326-337. Springer, 1999.

[NIS] J. Nishino, T. Kawarabayashi, T. Morishita, T. Kubo, H. Shimora, H. Aoyagi, K. Hiroshima, H. Ogura. *"Zeng99: RoboCup Simulation Team with Hierarchical Fuzzy Intelligent Control and Cooperative Development"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 649-652. Springer, 2000.

[NOD] I. Noda, H. Matsubara, K. Hiraki, I. Frank. (1997) *"Soccer Server: a tool for research on multi-agent systems"*. Applied Artificial Intelligence, 12, páginas 233-250. 1998.

[NOR] T. Naruse, T. Takahashi, D. Murakami, Y. Nagasaka, K. Isiwata, M. Nagami, Y. Mori. *"OWARI-BITO"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 675-678. Springer, 2000.

[OHT] M. Ohta. *"Learning Cooperative Behaviors in RoboCup Agents"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 412-419. Springer 1998.

[PLA] M. Plagge, B. Diebold, R. Gunther, J. Ihlenburg, D. Jung, D. Zhedi, A. Zell. *"Design and Evaluation of the T-Team of the University or*

Tuebingen for RoboCup'98". M. Asada, H. Kitano (Ed.), RoboCup-98: Robot Soccer World Cup II, páginas 464-472. Springer, 1999.

[POW] M. Powell. *"An efficient method for finding the minimum of a function of several variables without calculating derivatives"*. Computer. J.Vol.7, página 155. 1964.

[PRI] A. Price, A. Jennings, J. Kneen. *"RoboCup97: An Omnidirectional Perspective"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 320-332. Springer, 1998.

[REA] A. Reali Costa, R. Pegoraro, G. Stolfi, J. Sichman, F. Pait, H. Ferasoli Filho. *"GUARANA Robot-Soccer Team: Some Architectural Issues"*. En Proceedings of the FIRA 98 Workshop, Paris, France, 1998.

[REI1] L. Reis, N. Lau, L. Seabra Lopes. (2000) *"FC Portugal Team Description Paper"*. En http://www.ieeta.pt/RoboCup/archive.htm. 2000.

[REI2] L. Reis, N. Lau. *"FC Portugal Team Description: RoboCup 2000 Simulation League Champion"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 29-40. Springer, 2001.

[REI3] L. Reis, N. Lau. (2001) *"COACH UNILANG - A Standard Language for Coaching a (Robo)Soccer Team"*. A. Birk, S. Coradeschi, S. Tadokoro (Eds.), RoboCup 2001: Robot Soccer World Cup V., páginas 183-192. Springer, 2002.

[RIE] M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, R. Ehrmann *"Karlsruhe Brainstormers – A Reinforcement Learning Approach to Robotic Soccer"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 367-372. Springer, 2001.

[ROB] *The RoboCup Federation* (www.RoboCup.org).

[ROS] J. de la Rosa, R. García, B. Innocenti, I. Muñoz, A. Figueras, J. Ramón, *"Rogi Team Description"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 679-682. Springer, 2000.

[RSN] P. Rosenbloom, J. Laird, A. Newell, R. McCarl. *"A Preliminary Analysis of the Soar Architecture as a Basis for General Intelligence"*. Artificial Intelligence, 47 (1-3), páginas 289-325. 1991.

[SAN] J. Santos, H. Scolnik, I. Laplagne, S. Daicz, F. Scarpettini, H. Fassi, C. Castelo. *"UBA-Sot: an approach for control and team strategy in Robot Soccer"*. En Proceedings 2002 FIRA Robot World Congress, páginas 654-659. KRSA 2002.

[SHE1] W. Shen, J. Adibi, R. Adobbati, A. Erdem, H. Moradi, B. Salerni, S. Tejada: *"Totally Autonomous Soccer Robots"*. RoboCup 97: Robot World Cup Workshop. 1997.

[SHE2] W. Shen, J. Adibi, R. Adobbati, S. Lanksham, H. Moradi, B. Salemi, S. Tejada. *"Integrated Ractive Soccer Agents"* M. Asada, H. Kitano (Ed.), RoboCup-98: Robot Soccer World Cup II, páginas 286-298. Springer, 1999.

[SIM] M. Simon, S. Behake, R. Rojas. *"Robust Real Time Color Tracking"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 239-248. Springer, 2001.

[SOA] *Soar Home Page*. http://ai.eecs.umich.edu/soar.

[STE] T. Steffens. (2002) *"Feature-based declarative opponent-modelling in multi-agents systems"*. Master thesis, Osnabrück University. 2002.

[STO1] P. Stone, M. Veloso. *"Broad Learning from Narrow Training: A Case Study in Robotic Soccer"*. Technical Report CMU-CS-95-207, Computer Science Department, Carnegie Mellon University, 1995.

[STO2] P. Stone, M. Veloso. *"The CMUnited-97 Simulator Team"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 389-397. Springer, 1998.

[STO3] P. Stone, M. Veloso *"Using Decision Tree Confidence Factors for Multiagent Control"*. H. Kitano (Ed.), RoboCup-97: The First Robot World Cup Soccer Games and Conferences. Springer Verlag, Berlin, 1998.

[STO4] P. Stone, M. Veloso, P. Riley. *"The CMUnited-98 Champion Simulator Team"*. M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 61-76. Springer, 1999.

[TAM] M. Tambe, J. Adibi, Y. Al-Onaizan, A. Erdem, G. Kaminka, S. Marsella, I. Muslea, M. Tallis. *"ISIS: Using an Explicit Model of Teamwork in RoboCup"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 123-131. Springer, 1998.

[UNG] R. Unger. *"McGill RedDogs"*. M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup-99: Robot Soccer World Cup III, páginas 774-783. Springer, 2000.

[VEL1] M. Veloso, P. Stone, K. Han, S. Achim. *"CMUnited: A Team of Robotic Soccer Agents Collaborating in an Adversarial Environment"*. Proceedings of the First International Workshop on RoboCup, Nagoya,Japan. 1997.

[VEL2] M. Veloso, P. Stone, K. Han, S. Achim. *"Prediction, Behaviors, and Collaboration in a Team of Robotic Soccer Agents"*. Proceedings of the International Conference on Multi-Agent Systems (ICMAS'98), Paris, France. 1998.

[VEL3] M. Veloso, M. Bowling, S. Achim, K. Han, P. Stone. *"The CMUnited-98 Champion Small-Robot Team"*. M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 77-92. Springer, 1999.

[VEL4] M. Veloso, P. Stone, K. Han, S. Achim. *"The CMUnited-97 Small Robot Team"*. H. Kitano (Ed.), RoboCup-97: Robot Soccer World Cup I, páginas 242-256. Springer 1998.

[VEL5] M. Veloso, W. Uther. *"The CMTrio-98 Sony Legged Robot Team"*. M. Asada, H. Kitano (Eds.), RoboCup-98: Robot Soccer World Cup II, páginas 491-497. Springer, 1999.

[VIS] U. Visser, C. Drucker, S. Hubner, E. Schmidt, H. Weland. (2000) *"Recognizing Formations in Opponent Teams"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 391-396. Springer, 2001.

[WEI] T. Weigel, W. Auerbach, M. Dietl, B. Dümler, J. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowsky, M. Thiel. *"CS-Freiburg: Doing the Right Thing in a Group"*. P. Stone, T. Balch, G. Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 52-63. Springer, 2001.

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

[YAM] F. Yamasaki, T. Matsui, T. Miyashita, H. Kitano. *"PINO The Humanoid: A Basic Architecture"*. P. Stone, T. Balch, G.

Kraetzschmar (Eds.), RoboCup 2000: Robot Soccer World Cup IV, páginas 269-278. Springer, 2001.

[YOU] F. Yong, B. Ng, K. Loh, E. Ong, K. Teo, K. Loh. *"Alpha++"*. Linköping Electronic Articles in Computer and Information Science, Vol. 4: no. 006/18. 1999

[ZHA] X. Zhang, H. Hexmoor. *"Utility-based Role Exchange"*. From Theory to Practice in Multi-Agent Systems Second International Workshop of Central and Eastern Europe on Multi-Agent Systems, CEEMAS 2001 Cracow, Poland, página 332. 2001.

Apéndice A
Modelos del Simulador RoboCup

Modelo Cinemático de los Jugadores

A través del comando Dash, se da al jugador un impulso instantáneo según la magnitud *Power*, lo que se traduce como una aceleración en dirección X e Y. Esta aceleración dura sólo un ciclo, volviendo a 0 en el instante t +1.

$$\begin{aligned} a_x &= Power \times power\_rate \times cos(\theta) \\ a_y &= Power \times power\_rate \times sen(\theta) \end{aligned} \tag{A.1}$$

donde a_x y a_y son los componentes de la aceleración sobre el eje X e Y, respectivamente y *power_rate* un coeficiente parametrizable del servidor.

Esta aceleración se aplica en t_0 para calcular la velocidad inicial, a partir del cual se aplica un coeficiente constante de reducción de la velocidad (decay). Para simplificar, se muestran los cálculos sólo sobre una de las coordenadas. Idéntico razonamiento es aplicable a la otra coordenada.

$$\begin{aligned} V_0 &= Min(V_{anterior} + a_x \times \Delta t,\ Velocidad\_max) \\ V_1 &= V_0 \times decay \\ &\vdots \\ V_n &= V_{n-1} \times decay = V_0 \times decay^n \end{aligned} \tag{A.2}$$

donde $\Delta t = 1$ y *Velocidad_max* es la máxima velocidad que puede alcanzar un robot y es parametrizable en el servidor.

Con lo que se deduce que el espacio recorrido por el jugador por efecto del comando Dash con magnitud *Power* es:

$$\begin{aligned} E_0 &= V_0 \times \Delta t \\ E_1 &= E_0 + V \times_1 \Delta t = V_0 \times \Delta t + V_0 \times decay \times \Delta t \\ &\vdots \\ E_n &= V_0 \left(\sum_{i=0}^{n} decay^i \right) \times \Delta t = V_0 \left(\frac{1 - decay^{n+1}}{1 - decay} \right) \times \Delta t \end{aligned} \tag{A.3}$$

donde $\Delta t = 1$.

Con lo anterior se puede saber el espacio recorrido a los n momentos de haber recibido el impulso. Si se quiere conocer el espacio total recorrido en base a una velocidad determinada, en definitiva se quiere saber en que momento el espacio recorrido desde el ciclo anterior es menor a determinada cota ξ:

$$\begin{aligned} & E_{n+1} - E_n < \xi \\ & V_0 \left(\frac{1 - decay^{n+2}}{1 - decay} \right) \Delta t - V_0 \left(\frac{1 - decay^{n+1}}{1 - decay} \right) \Delta t < \xi \\ & V_0 decay^{n+1} \left(\frac{1 - decay}{1 - decay} \right) \Delta t < \xi \\ & V_0 decay^{n+1} \times \Delta t < \xi \\ & decay^{n+1} < \frac{\xi}{V_0} \\ & (n+1) ln(decay) < ln(\xi) - ln(V_0) \\ & n > \frac{ln(\xi) - ln(V_0)}{ln(decay)} - 1 \end{aligned} \tag{A.4}$$

Por ejemplo, si el jugador está en reposo ($V_0 = 0$), no se aplica ruido al movimiento ni viento y se aplica Dash(100) con $\theta = 0$ (desplazamiento horizontal), usando los valores de dash_power_rate = 0.1, decay = 0.5 y

player_speed_max = 32, como usa por defecto el servidor, se obtienen los siguientes resultados:

$$\begin{aligned} a_x &= Power \times power\_rate \times cos(\theta) = 100 \times 0.1 \times 1 = 10 \\ a_y &= Power \times power\_rate \times sen(\theta) = 100 \times 0.1 \times 0 = 0 \end{aligned} \tag{A.5}$$

Como el jugador se supone en reposo, la velocidad inicial será igual en valor a la aceleración obtenida (como sólo hay velocidad en el eje X, sólo se calcula el espacio recorrido en esta dirección ya que no habrá desplazamiento en el eje Y):

$$V_0 = 10 \tag{A.6}$$

El espacio total recorrido gracias al impulso recorrido será:

$$E_n = V_0\left(\frac{1-decay^{n+1}}{1-decay}\right) = 10\left(\frac{1-0.5^{n+1}}{0.5}\right) \tag{A.7}$$

Pero para saber en que momento deja de desplazarse, se debe utilizar la inecuación antes deducida y usando una cota de 0.001:

$$n > \frac{ln(0.001)-ln(10)}{ln(0.5)} - 1 = 12.28 \tag{A.8}$$

Es decir que a partir del decimotercer ciclo, el espacio recorrido será despreciable. Con lo cual el espacio recorrido será:

$$E_{13} = V_0\left(\frac{1-decay^{14}}{1-decay}\right) = 10\left(\frac{1-0.5^{14}}{0.5}\right) = 19.999 \tag{A.9}$$

Otra manera de saber el espacio total que recorrerá el jugador, es tomando el límite de la función de espacio cuando *n* tiende a infinito:

$$E_f = \lim_{n\to\infty}(E_n) = \frac{V_0}{1-decay} \times \lim_{n\to\infty}\left(1-decay^{n+1}\right) \tag{A.10}$$

Como *decay* es un valor menor a 1, cuando *n* tiende a infinito, la potencia tiende a 0, por lo tanto:

$$E_f = \lim_{n\to\infty}(E_n) = \frac{V_0}{1-decay} \tag{A.11}$$

Con el ejemplo con los valores del ejemplo anterior, se tiene que:

$$E_f = \frac{10}{1-0.5} = 20 \tag{A.12}$$

De lo cual se puede deducir la V_0 necesaria para llegar a una determinada distancia:

$$V_0 = E_i\,(1\text{-}decay) \tag{A.13}$$

Y dado que la V_0 depende de la $V_{anterior}$ y la aceleración, la cual depende del *Power* y del parámetro power_rate,

$$Power = \frac{E_i(1-decay) - V_{anterior}}{power\_rate} \tag{A.14}$$

Modelo de Inercia en los Giros de los Jugadores

Mediante el comando Turn, el jugador puede cambiar su dirección, o punto de vista en un rango determinado por *minmoment* y *maxmoment*. Siendo por defecto este rango de 360° (entre 180° y -180°), pero si el jugador se encuentra en movimiento, la efectividad del giro se ve afectada por la inercia:

$$ángulo\_efectivo = \frac{ángulo}{1.0 + inertia\_moment \times Velocidad\_Jugador} \qquad (A.15)$$

Con parámetros por defecto, el momento de Inercia es 0.5 y dado que la velocidad máxima es 32, el rango del ángulo efectivo será:

$$\frac{ángulo}{9} = \frac{ángulo}{(1.0 + 0.5 \times 16)} \leq ángulo\_efectivo \leq ángulo \qquad (A.16)$$

Nótese que se usó 16, en vez de 32 como velocidad máxima, ya que al no poder realizarse dos acciones en el mismo ciclo, si el jugador alcanzó una velocidad de 32, recién podrá hacer un comando Turn en el siguiente ciclo, donde ya fue aplicado un decay a la velocidad.

Modelo de la Pelota

El movimiento de la pelota se obtiene a través del comando Kick(power, direction), cuyo efecto es similar al comando Dash. Las diferencias con éste último son las siguientes:

- La dirección efectiva de la pelota será la suma del ángulo que tenga el agente y la dirección pasada como parámetro.

- El comando sólo tiene efecto cuando la pelota está dentro de un *Kickable_area o Área de pateo*, la cual es la suma del radio del jugador (*player size*), el radio de la pelota (*ball size)* y el margen de pateo (*kickable margin)*:

$$Kickable\_area = player\_size + ball\_size + kickable\_margin \tag{A.17}$$

Con parámetros por defecto del servidor:

$$Kickable\_area = 1.0 + 0.15 + 1.0 = 2.15 \tag{A.18}$$

- La *Potencia* del tiro depende del ángulo y la distancia entre el jugador y la pelota. Esto se traduce en un coeficiente que se aplica para calcular el kick_power, como se muestra a continuación:

$$Kick\_power = Kick\_power\_rate\left(1 - 0.25 \times \frac{dir\_diff}{180} - 0.25 \times \left(\frac{dist\_ball - player\_size - ball\_size}{kickable\_margin}\right)\right) \tag{A.19}$$

donde d*ir_diff* es el valor absoluto del ángulo entre la dirección hacia donde mira el jugador y la posición de la pelota; y *dist_ball* la distancia entre el jugador y la pelota. Utilizando los parámetros por defecto del servidor:

$$Kick\_power = 0.1\left(1 - 0.25 \times \frac{dir\_diff}{180} - 0.25 \times \left(\frac{dist\_ball - 1.15}{1.0}\right)\right) \tag{A.20}$$

por lo tanto:

$$0.1(0.5) < Kick\_power < 0.1\ (1.29) \rightarrow 0.05 < Kick\_power < 1.13 \tag{A.21}$$

donde la máxima potencia se logra cuando la pelota está exactamente frente "a los ojos" del jugador y "pegada a sus pies", y la mínima cuando la pelota está justo detrás y en el límite de lo alcanzable.

Energía de los Jugadores

Cada jugador tiene su propia energía (stamina), la cual decrece cada vez que el jugador realiza un comando Dash, y se recupera lentamente en cada ciclo de simulación (recovery).

Otro parámetro que interviene en los movimientos es la efectividad (effort) que decrece cuando la stamina es baja y también aumenta paulatinamente hasta un máximo de 1.0 (donde Dash es totalmente efectivo).

A continuación se describe el algoritmo que se sigue para la actualización de los distintos valores intervinientes:

Para *cada Ciclo de Simulación* ***Hacer***
  Si *se realiza un Dash(Power)* ***Entonces***
    Si *Dash > 0* ***Entonces***
      Stamina ← Max(Stamina – Power, 0)
    Sino
      Stamina ← Max(Stamina – 2×Power, 0)
    FinSi
    Power_efectivo ← Effort × Min(Power, Stamina)
    Dash(Power_efectivo)
  FinSi
  Si *Stamina <= recovery_dec_Thr × stamina_max* ***Entonces***
    Recovery ← Max(recovery_min, Recovery – recovery_dec)
  FinSi
  Si *Stamina <= effort_dec_thr × stamina_max* ***Entonces***
    Effort ← Max(effort_min, effort - effort_dec)

Sino

Effort ← *Min(1.0, Effort + effort_inc)*

FinSi

Stamina ← *Min(Stamina_max, Stamina + recovery × stamina_inc_max)*

FinPara

A modo de ejemplo, a continuación se reemplazan los parámetros por los que utiliza el simulador por defecto.

Para *cada Ciclo de Simulación* ***Hacer***

Si *se realiza un Dash(Power)* ***Entonces***

Si *Dash > 0* ***Entonces***

Stamina ← *Max(Stamina – Power, 0)*

Sino

Stamina ← *Max(Stamina – 2×Power, 0)*

FinSi

Power_efectivo ← *Effort × Min(Power, Stamina)*

Dash(Power_efectivo)

FinSi

Si *Stamina <= 750* ***Entonces***

Recovery ← *Max(0.1, Recovery – 0.05)*

FinSi

Si *Stamina <= 1000* ***Entonces***

Effort ← *Max(0.1, effort – 0.05)*

Sino

Effort ← *Min(1.0, Effort + 0.05)*

FinSi

Stamina ← *Min(2500, Stamina + recovery × 20)*

FinPara

Tanto la stamina como el effort paulatinamente aumentan en cada ciclo de simulación, por lo tanto si un jugador "se cansa" sin llegar a un valor crítico, su efectividad y velocidad en la recuperación de energía vuelven a ser óptimas con el

tiempo. Sin embargo, el valor de recovery, que es el factor de recuperación de la stamina, una vez que decrece cuando la stamina baja de un determinado umbral, no vuelve a aumentar nunca. Como consecuencia, es importante monitorear que un jugador no "quede exhausto" pues esto redunda en que nunca más vuelva a tener una recuperación óptima de energía y efectividad.

Coeficientes de Error

Los movimientos de los objetos y los parámetros de los comandos, sufren pequeñas modificaciones aleatorias las cuales se rigen por los parámetros *Player_rand* y *Ball_rand*. Con respecto a la posición de los objetos, al realizarse un comando Dash o Kick, al calcularse la velocidad inicial, se suma un coeficiente aleatorio, además de la velocidad anterior y la aceleración:

$$V_0 = Min(V_{anterior} + a_x + Error\ ,\ Velocidad\_max) \tag{A.22}$$

Donde Error es un valor aleatorio dentro de un rango, el cual depende del módulo del vector de velocidad del objeto y el parámetro *Player_rand* o *Ball_rand*, según corresponda:

$$-Rand \times Vel\_Objeto <= Error <= Rand \times Vel\_Objeto \tag{A.23}$$

En los argumentos de los comandos también se introduce error aleatorio, el cual toma valor en el rango [-Rand, Rand], de la siguiente manera:

$$Argumento = (1 + Error) \times Argumento \tag{A.24}$$

En base a los parámetros por defecto, donde Rand_player = 0.1 y Rand_Ball = 0.2, se obtienen las siguientes cotas de error.

$$\begin{aligned} -0.2\ Vel_{pelota} < Error_{pelota} < 0.2\ Vel_{pelota} \\ -0.1\ Vel_{jugador} < Error_{jugador} < 0.1\ Vel_{Jugador} \end{aligned} \tag{A.35}$$

Efecto del Viento

El viento afecta la velocidad de los objetos en base a la velocidad que llevan, su peso y la dirección y fuerza del viento. A menor peso del objeto, mayor será la influencia del viento:

$$Vel_x = Vel_{x0} + Vel_{objeto} \times \frac{Viento_x}{Peso \times 10000} \tag{A.26}$$

En base a los parámetros por defecto, donde Weight_Player = 60 y Weight_Ball = 0.2, Fuerza del viento = 10 y ángulo = 0 (corre inicialmente de izquierda a derecha), se obtiene:

$$\begin{aligned} Vel_{jug\_x} &= Vel_{jug\_x0} + Vel_{jugador} \times \frac{1}{60000} \\ Vel_{pel\_x} &= Vel_{pel\_x0} + Vel_{pelota} \times \frac{1}{200} \end{aligned} \tag{A.27}$$

Como se puede observar, la influencia del viento sobre el jugador es despreciable y sobre la pelota, bastante baja.

Apéndice B
Reglamento MiroSot y SimuroSot

La categoría de simulación FIRA Middle League SimuroSot comparte las reglas de juego con la categoría de robots reales Middle League MiroSot. A continuación se describe cada una de estas reglas.

Regla 1: El campo y la pelota

Dimensiones del campo de juego

La cancha es un rectángulo de madera color negro de 220 cm x 180 cm, delimitada con paredes de color blanco de 5 cm de altura y 2.5 cm de espesor. La parte superior de las paredes (2.5 cm) debe ser color negro y los laterales color blanco. Se deben colocar triángulos isósceles de 7 cm x 7 cm en las esquinas de la cancha para evitar que la pelota quede atascada. La textura de la superficie debe ser semejante a una cancha de ping pong.

Marcas en el campo de juego

El campo de juego se marca como se muestra en la Fig.B.1.

El círculo central tiene 25 cm de radio. El arco, donde se encuentra el punto penal, tiene 25 cm de longitud y 5 cm de profundidad.

Las principales líneas y arcos (línea central, áreas y círculo central) son de color blanco y tienen 3 mm de espesor. Las posiciones de los robots para la ejecución de piques se marcan en color gris.

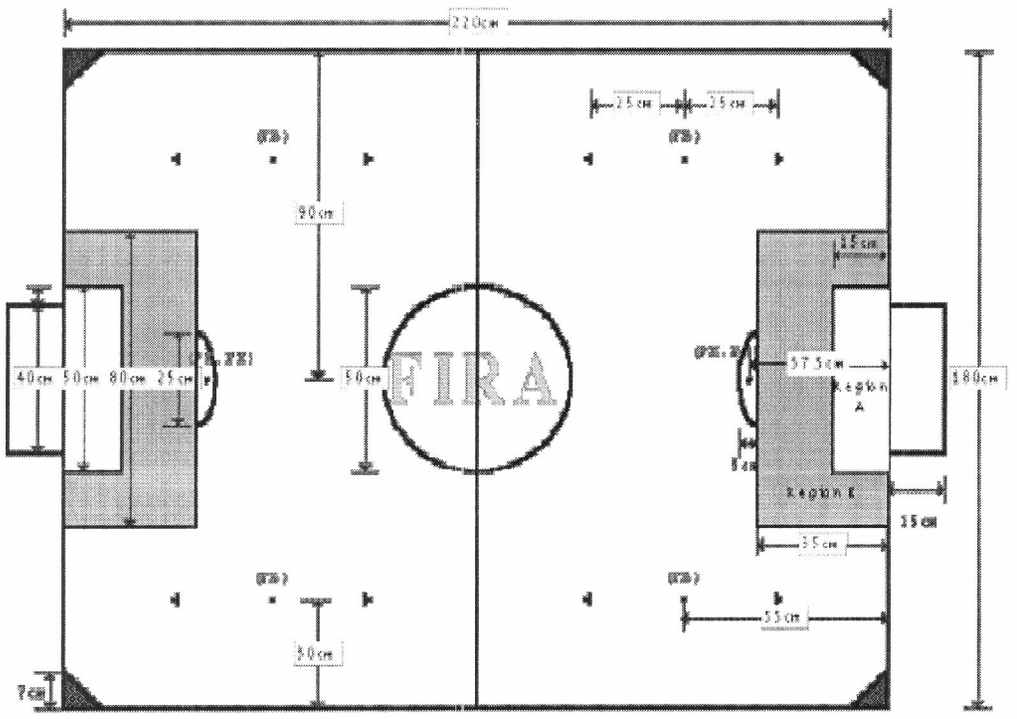


Figura B.1: Campo de juego.

El arco

Los arcos tienen 40 cm de ancho. No poseen ni postes ni redes.

La línea de arco y el área chica

La línea de arco es la línea justo en frente del arco y tiene una longitud de 40 cm.

El área chica es el área conformada por el rectángulo de 50 cm de longitud y 15 cm de ancho ubicado delante del arco. En la Fig.B.1 esta área está indicada como Región A.

El área penal

El área penal es el área conformada por el rectángulo de 80 cm de longitud y 35 cm de ancho ubicado delante del arco y el arco de ataque (arco donde se encuentra el punto penal, de 25 cm de longitud y 5 cm de profundidad). En la Fig.B.1 esta área está indicada como Región B.

La pelota

Una pelota de golf color naranja se utiliza como pelota. Su diámetro es de 42.7 mm y su peso es 46 grs.

Ubicación de la cancha

La cancha se debe ubicar en un recinto cerrado, no a la intemperie.

Condiciones de iluminación

La iluminación en el recinto de la competencia se debe fijar en aproximadamente 1.000 lux.

Regla 2: Los jugadores

Sistema global

En un partido se enfrentan dos equipos, cada uno conformado por cinco robots. Uno de los robots puede ser el arquero (regla 2.b.2). Solamente tres

personas pueden integrar el equipo humano presente en el estadio: un representante, un director técnico y un entrenador.

Los robots

El tamaño de cada robot es 7.5 cm x 7.5 cm x 7.5cm. La altura de la antena de comunicación no está considerada en el tamaño de los robots.

La parte superior de un robot no puede ser color naranja. Los robots de un equipo se deben identificar con un parche azul o amarillo (en cada juego el color es asignado por los organizadores). Todos los robots deben tener (al menos) una región visible de 3.5 cm x 3.5 cm sobre su parte superior del color asignado para su equipo. El color identificador puede variar de un partido a otro y el parche utilizado debe ser removible. Cuando a un equipo le es asignado un color, los robots no pueden tener ningún parche visible del color de su oponente.

Nota: Se recomienda a los equipos la preparación de un mínimo de 10 parches de colores diferentes para la identificación individual de los robots.

Para el sensado infrarrojo los robots pueden llevar sobre sus laterales luces de colores, con excepción de las regiones que se utilizan exclusivamente para funcionalidades, como ser los sensores, las ruedas o el mecanismo para retener la pelota. Los robots pueden llevar uniformes. En este caso su tamaño queda limitado a 8 cm x 8 cm x 8 cm.

Un robot dentro de su propia área chica (regla 1.d) es considerado como arquero. El arquero solamente tiene permitido tomar o retener la pelota cuando la misma se encuentra dentro de su área chica o su área penal.

Cada robot debe ser totalmente independiente, con mecanismos de potencia y motricidad propios. Solamente las radio-comunicaciones están habilitadas para establecer la interacción entre los robots y la computadora central.

Los robots pueden estar equipados con brazos, piernas u otros apéndices, pero deben cumplir con las restricciones de tamaño establecidas (regla 2.b.1) también después de desplegar los mismos. Ninguno de los robots, exceptuando el arquero, tiene permitido tomar o retener la pelota cuando más de un 30 % de la misma se puede visualizar observando el robot desde arriba o desde uno de sus laterales.

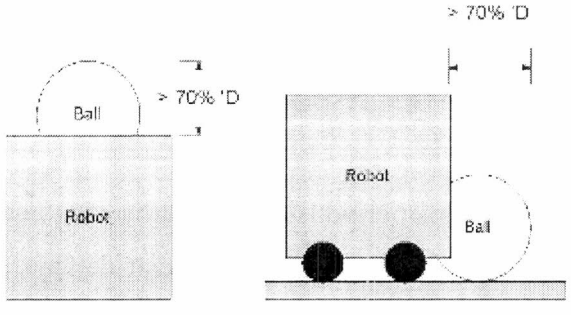


Figura B.2: Oclusión de la pelota

Durante un partido, el árbitro puede indicar al operador humano la detención de todos los robots usando la comunicación entre los robots y la computadora central.

Sustituciones

Durante el desarrollo de un partido solamente se pueden realizar dos sustituciones. En el entretiempo la cantidad de sustituciones es ilimitada. Cuando un equipo necesita hacer una sustitución durante el juego, su representante debe solicitar un "tiempo muerto" al árbitro y éste detiene el partido en un momento que lo considera apropiado. El partido se reanuda cuando todos los robots y la

pelota están nuevamente ubicados en las posiciones que se encontraban en el momento de interrumpir el juego.

Tiempo muerto

Cada equipo cuenta con dos tiempos muertos en un partido de dos minutos cada uno. El operador humano debe indicar "tiempo muerto" al árbitro.

Se omiten las reglas 3 y 4 debido a que las mismas hacen referencia al control de los robots físicos y los sistemas de visión utilizados por ellos. Estas reglas no son aplicables a la categoría SimuroSot.

Regla 5: Duración del juego

Un partido consta de dos períodos de 5 minutos cada uno, con un intervalo entre ambos de 10 minutos. Un cronometrista oficial detiene el reloj cuando se realizan las sustituciones, cuando se debe retirar de la cancha un robot por averías, durante los tiempos muertos o ante otras situaciones en que se considere necesario.

Si un equipo no está listo para reanudar el partido después del intervalo, se le pueden otorgar 5 minutos adicionales. Si pasado este tiempo no se encuentra en condiciones de continuar, el equipo es descalificado.

Regla 6: Comienzo del juego

Antes de comenzar un partido, el color para el equipo (azul o amarillo) o el saque inicial se decide lanzando una moneda. El equipo que acierta el resultado de este lanzamiento elige el color (azul o amarillo) o la pelota (para comenzar el

partido). El equipo que recibe la pelota tiene la posibilidad de optar por la banda de frecuencia a usar.

En el comienzo del partido, el equipo atacante puede posicionar libremente sus robots en su propio campo y dentro del círculo central. En cambio, el equipo defensor puede ubicar sus robots dentro de su campo sin considerar el círculo central.

En el inicio del primer o segundo tiempo, o después de haber marcado un gol, la pelota se ubica en el círculo central y el equipo atacante debe patear o pasar la pelota hacia el campo propio. Con una señal del árbitro, el juego comienza y los robots pueden moverse libremente.

En el comienzo del partido o después de haber marcado un gol, el juego comienza / continúa, posicionando los robots como se indica en la regla 6.2.

Después del entretiempo, los equipos deben cambiar de campo.

Regla 7: Sistema de marcación

El ganador

Un gol se marca cuando la pelota ingresa en el arco pasando en forma completa la línea de arco. El ganador de un partido se determina en base a la cantidad de goles marcados.

El desempate

En caso de haber finalizado el partido empatado (ambos períodos), el ganador se decide por muerte súbita. Después de un descanso de 5 minutos, se

reanuda el partido por un período máximo de 3 minutos. El primer equipo que marque un gol es declarado ganador. Si el empate persiste pasados los 3 minutos de descuento, el ganador se decide mediante tiros penales. Cada equipo tiene tres tiros penales, que difieren de la regla 11 en que sólo se permite un pateador y un arquero en el campo de juego. El arquero debe estar dentro del área chica y el pateador y la pelota deben ubicarse como lo indica la regla 11. Después que el árbitro pita, el arquero puede salir de su área. En caso que se mantenga el empate después de los tres tiros penales, se ejecutan tiros penales adicionales (uno para cada equipo) hasta que se marca una diferencia y se declara un ganador. Todos los penales deben ser ejecutados por un solo robot y el mismo está habilitado para patear cuando el árbitro hace sonar el silbato. Un tiro penal se considera finalizado cuando:

- El arquero toma la pelota con sus apéndices (si tiene) dentro del área chica
- La pelota sale del área chica (hacia el arco o hacia el campo)
- Pasan 30 segundos desde que el árbitro da la señal para ejecutar el tiro

Regla 8: Faltas

Un robot comenta una falta en los siguientes casos.

Chocar con un robot oponente, sea intencional o no: el árbitro sanciona aquellas faltas que afectan en forma directa el desarrollo del juego o que parecen potencialmente perjudicar al robot oponente. Cuando un robot defensor empuja intencionalmente a un robot oponente, se otorga un tiro libre a favor del equipo oponente. Está permitido empujar la pelota y a un jugador oponente ubicado detrás de ella, mientras que el jugador que empuja esté siempre en contacto con la pelota.

Está permitido empujar al arquero en el área chica, siempre que la pelota se encuentra entre el robot y el arquero. Sin embargo, si el robot atacante empuja al arquero con la pelota dentro del arco o si lo empuja directamente (sin pelota entre ambos), el árbitro beneficia al arquero cobrando un saque de arco (por carga al arquero).

Atacar con más de un robot en el área chica oponente es penalizado con un saque de arco a favor del equipo defensor. Un robot se considera dentro del área chica si está dentro de ella en más de un 50 %, a juzgar por el árbitro.

Defender con más de un robot en el área chica propia es penalizado con un tiro penal a favor del equipo atacante. Un robot se considera dentro del área chica si está dentro de ella en más de un 50 %, a juzgar por el árbitro. Una excepción a esta regla se presenta cuando un robot adicional se ubica dentro del área chica pero no encuentra en posición de defensa ni afecta directamente el desarrollo del juego. El árbitro es quien juzga las situaciones para determinar si ameritan la ejecución de un tiro penal.

Se denomina *"Retención"*, a juzgar por el árbitro, cuando un robot que no cumple la función de arquero toma la pelota. También se considera *"Retención"*, si un robot se abalanza firmemente contra la pelota de manera que ningún otro robot pueda manipularla.

El arquero puede mantener la pelota dentro del área chica (regla 1.d) hasta 10 segundos. De excederse, puede ser sancionado con un tiro penal a favor del equipo oponente.

El bloqueo intencional al arquero en el área chica se penaliza con un saque de arco.

Sólo el árbitro y uno de los integrantes humanos de los equipos (representante, entrenador o director técnico) están autorizados a tocar los robots. La manipulación de los robots sin el permiso del árbitro se penaliza con un tiro penal.

Defender con más de tres robots en el área penal se sanciona con un tiro penal a favor del equipo atacante. Un robot se considera dentro del área penal si está dentro de ella en más de un 50 %, a juzgar por el árbitro.

Regla 9: Interrupciones en el juego

El juego se interrumpe, y un operador humano puede reubicar los robots, cuando:

- Se debe reemplazar un robot.
- Se cae un robot quedando fuera de juego.
- Se marca un gol o se comete una falta.
- El árbitro señala un tiro libre (regla 10), un tiro penal (regla 11), un saque de arco (regla 12) o un pique (regla 13).

Regla 10: Tiro libre

Cuando un robot empuja intencionalmente a un robot oponente, se sanciona un tiro libre a favor del equipo contrario (regla 8.1). La pelota se debe ubicar en el punto para tiro libre (FK) que se muestra en la Fig.B.3. El robot que ejecuta el tiro libre se debe ubicar detrás de la pelota. El equipo atacante puede posicionar los otros robots libremente. Dos robots defensores deben ubicarse tocando la línea frontal de su área chica y fuera de ella, sobre los extremos izquierdo y derecho de dicha línea. Los otros dos robots de la defensa se deben

posicionar tocando las líneas laterales de su área penal, pero también fuera de ella. Con la señal del árbitro, todos los robots pueden moverse.

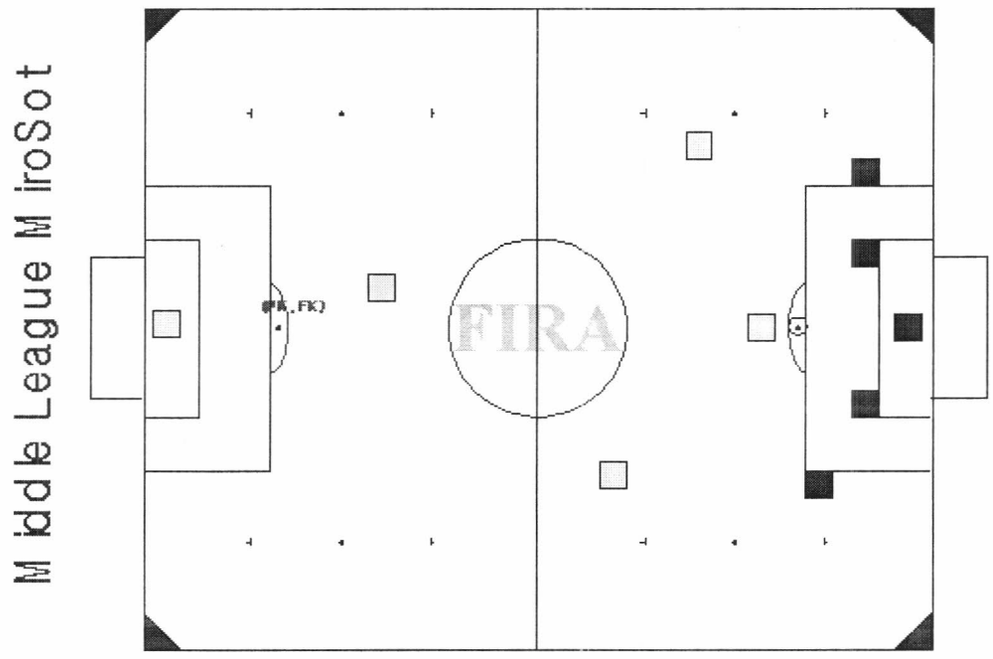


Figura B.3: Tiro libre.

Regla 11: Tiro penal

Un tiro penal se sanciona en las siguientes situaciones:

- Un equipo defiende con más de un robot en el área chica propia (regla 8.4).
- El arquero retiene la pelota dentro del área chica por más de 10 segundos (regla 8.6).
- Un integrante humano toca un robot durante el juego sin el permiso del árbitro (regla 8.8).
- Un equipo defiende con más de tres robots en el área penal (regla 8.9).

Cuando el árbitro sanciona un tiro penal, la pelota se debe ubicar en el punto para tiro penal (PK) que se muestra en la Figura 4. El robot que ejecuta el tiro penal se debe ubicar detrás de la pelota. El arquero debe posicionarse tocando la línea de arco con alguno de sus laterales. Su orientación puede ser cualquiera. Los restantes robots se pueden ubicar libremente detrás de la línea central. El equipo atacante tiene prioridad para el posicionamiento de sus robots. Con la señal del árbitro, todos los robots pueden moverse.

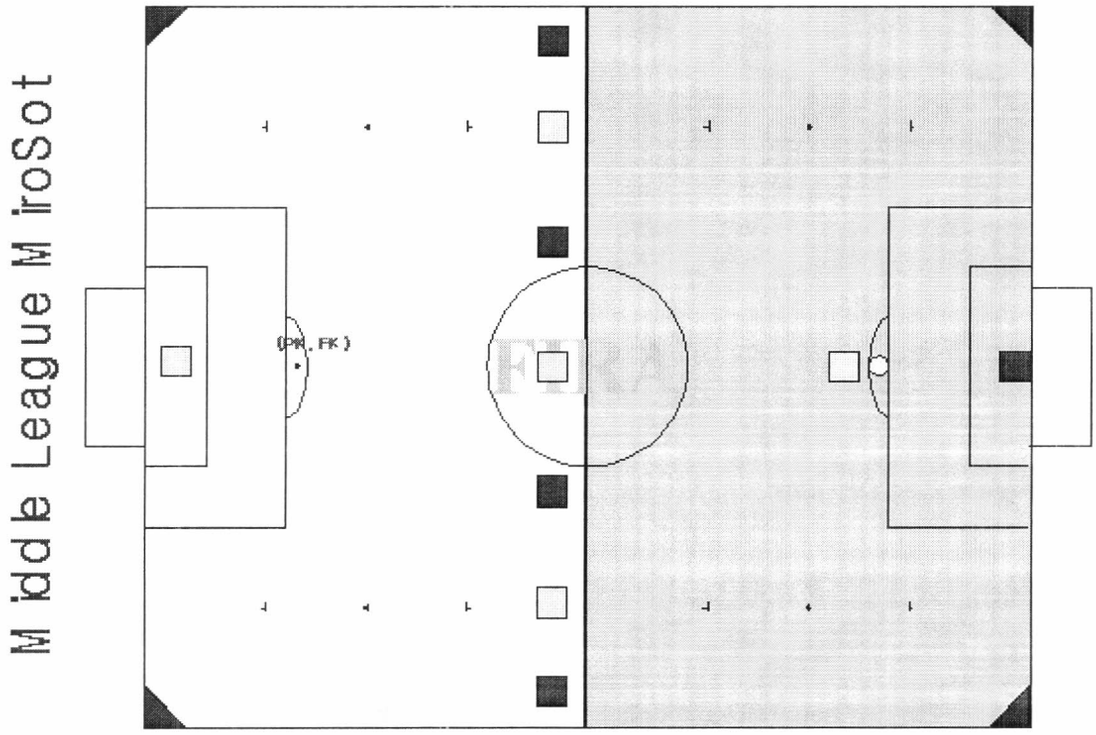


Figura B.4: Tiro penal

Regla 12: Saque de arco

Un saque de arco se sanciona en las siguientes situaciones:

- Un robot atacante empuja al arquero en el área chica (regla 8.2).
- Un equipo ataca con más de un robot en el área chica oponente (regla 8.3).
- Un robot bloquea intencional al arquero en su área chica (regla 8.7).

- El arquero toma la pelota con sus apéndices (si tiene) dentro del área chica.
- El juego se paraliza durante más de 10 segundos dentro del área chica.

Durante un saque de arco solamente el arquero puede ubicarse dentro del área penal y la pelota debe colocarse en cualquier punto dentro del mismo área. Los restantes robots se pueden ubicar libremente fuera del área penal. El equipo atacante tiene prioridad para posicionar sus robots, siempre y cuando considere la regla 8.3. El equipo defensor puede ubicar sus robots en su campo. El juego se reanuda cuando el árbitro hace sonar el silbato.

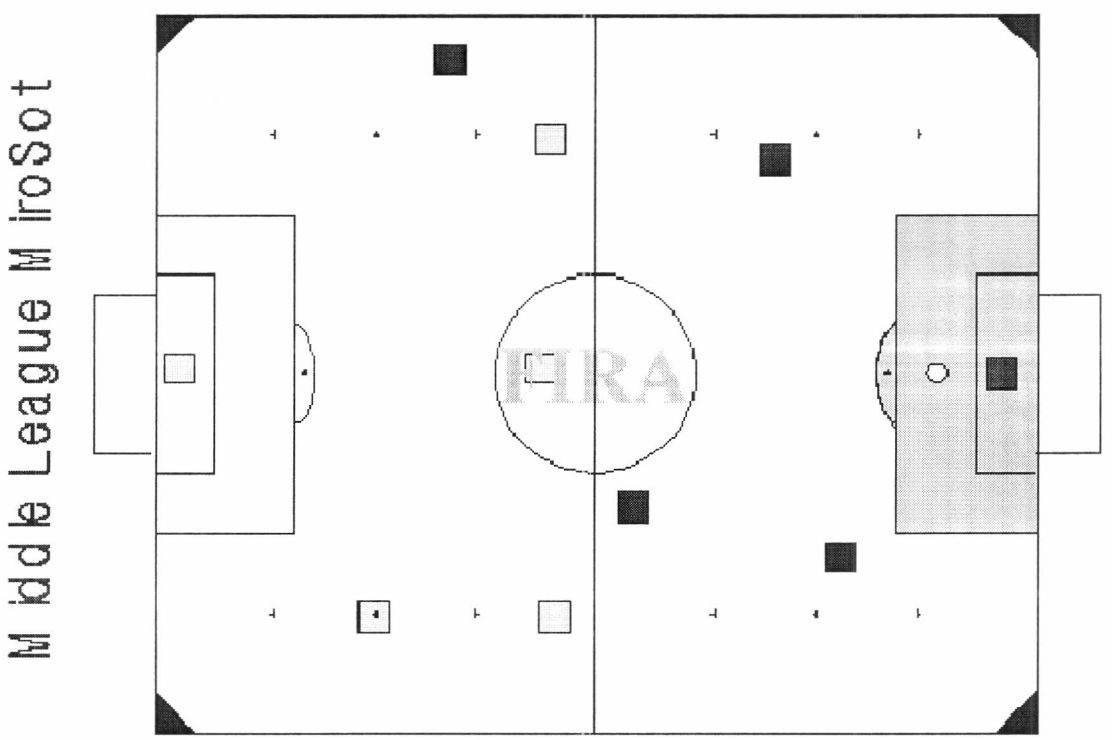


Figura B.5: Saque de arco.

Regla 13: Pique

El árbitro sanciona un pique cuando el partido se paraliza fuera del área chica durante más de 10 segundos

Cuando se ejecuta un pique, la pelota se debe ubicar sobre el punto de pique (FB) en el cuadrante de la cancha donde se sancionó el mismo. Un robot de cada equipo se debe posicionar a 25 cm de la pelota en dirección longitudinal. Los restantes robots se ubican libremente fuera del cuadrante donde se ejecuta el pique. El equipo defensor tiene prioridad para posicionar sus robots. El juego se reanuda cuando el árbitro hace sonar el silbato y los todos robots comienzan a moverse.

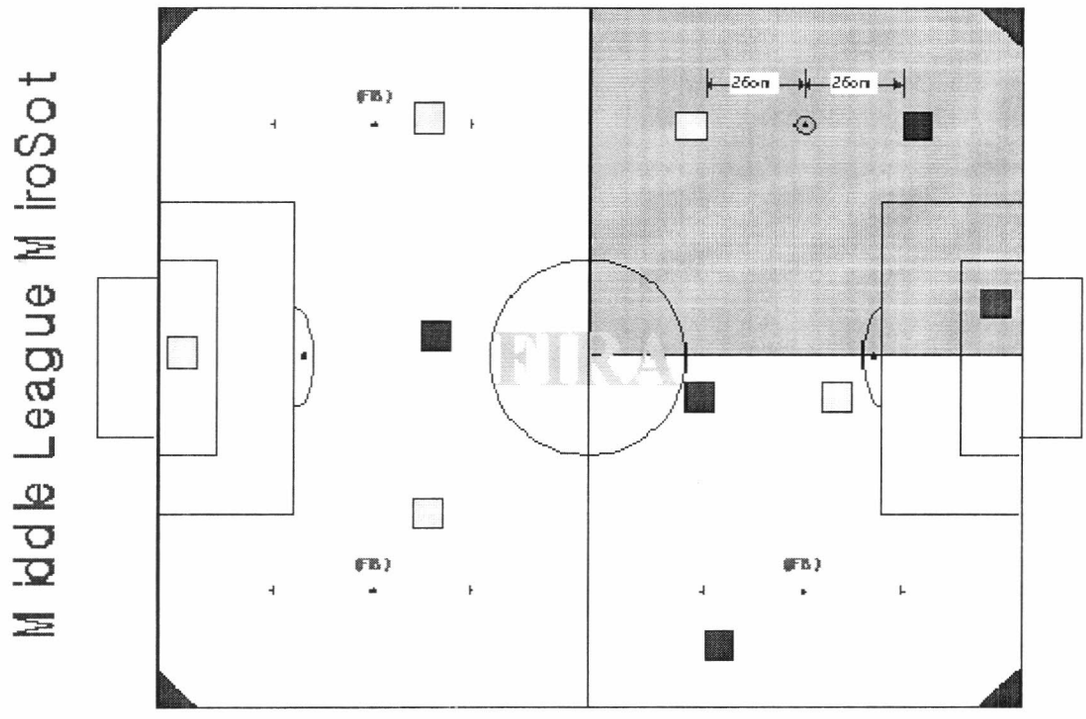


Figura B.6: Pique.