Soluciones a problemas de percepción y acción en el dominio de un equipo de fútbol de robots

Fernando A. Martin fam@cs.uns.edu.ar
Mariano Tucat mt@cs.uns.edu.ar
Alejandro J. García ajg@cs.uns.edu.ar

Grupo de Robótica Cognitiva
Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)*
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Avda. Alem 1253 - (B8000CPB) Bahía Blanca
Tel: ++54 291 4595135 - Fax: ++54 291 4595136

Resumen

Este trabajo tiene como objetivo mostrar algunas soluciones a los problemas encontrados en el desarrollo de un equipo de fútbol de robots que compitió en el campeonato mundial Robocup 2004. Los robots desarrollados toman sus decisiones basándose en un modelo abstracto del entorno donde se encuentran, el cual debe ser lo más preciso posible. Por lo tanto, es imprescindible tratar de minimizar los errores que ocurren en la percepción del entorno realizada por el sistema de visión y los errores que pueden ocurrir al ejecutar las acciones por parte del robot. Estas acciones podrían tener un resultado diferente al esperado en el modelo, debido a la carga de las baterías, la superficie del campo de juego, pequeñas diferencias entre dos motores, etc. El trabajo incluye además una evaluación de las soluciones y los resultados obtenidos.

Palabras clave: Inteligencia Artificial - Robótica Cognitiva - Percepción y Acción

1. Introducción

Este trabajo surge del desarrollo de un equipo de fútbol de robots [6] que compitió en el campeonato mundial Robocup 2004 [5]. Como se explicará en detalle más adelante, los robots disponían de un sistema de visión global y de dos motores para su desplazamiento. Este trabajo tiene como objetivo mostrar algunas soluciones a los problemas comúnmente encontrados en la percepción de los objetos por parte del sistema de visión y también a problemas que pueden ocurrir al ejecutar las acciones básicas utilizando los motores.

Financiado parcialmente por SeCyT Universidad Nacional del Sur (Subsidio: 24/ZN09) y por la Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096)

*Miembro del Instituto de Investigación en Ciencia y Tecnología Informática (IICyTI)

Como los robots desarrollados no son completamente reactivos, éstos toman sus decisiones basándose en un modelo abstracto del entorno donde se encuentran, el cual debe ser lo más preciso posible. En busca de este objetivo, es imprescindible tratar de minimizar los errores que ocurren tanto en la etapa de sensado, así como también al momento de ejecutarse las acciones que modifican el entorno en el que se sitúan.

Por ejemplo, el sistema de visión suele presentar ciertas imperfecciones. Un objeto estático dentro del campo de juego ilustra esta situación, ya que la información provista por el sistema de visión no es siempre la misma, en algunos casos variando considerablemente. Esto podría llevar a la construcción de un modelo abstracto donde las posiciones de los objetos sean diferentes a las reales, lo cual produciría que se tomen decisiones incorrectas.

Con respecto a las acciones llevadas a cabo por los robots, es importante destacar que no puede asumirse de antemano que dichas acciones tendrán el resultado esperado en el modelo. En este trabajo consideramos los siguientes posibles problemas: la carga de las baterías que accionan los motores de los robots afectan el rendimiento de éstos, el tipo de superficie del campo de juego afecta la respuesta de los motores, la inercia del robot tiene efectos sobre la acción realizada, y por último, pequeñas diferencias entre dos motores que deberían ser iguales dificultan algunas acciones básicas.

2. El entorno de desarrollo

Los problemas y soluciones que se presentan en este trabajo surgen del desarrollo de un equipo de robots para competir en la liga E-League [3] del campeonato mundial de fútbol de robots Robocup 2004. Robocup [4, 5] es un evento internacional cuyo objetivo es el desarrollo de tecnología en el área de robótica e inteligencia artificial. En particular, E-league tiene como objetivo principal priorizar el desarrollo de la inteligencia de los robots por sobre el desarrollo de costosos y complejos componentes electrónicos.

Un partido de la E-league se juega en una cancha de 1.52 x 2.74 metros y cada equipo tiene como máximo 4 jugadores que incluye al arquero. Los robots pueden tener diseños diferentes mientras satisfagan las siguientes restricciones:

- Un máximo de 22 cm de diámetro.
- Un máximo de 15 cm de alto.
- A lo sumo 3 actuadores (motores).

Aunque no existen restricciones sobre el diseño de la inteligencia provista a los robots, su comportamiento debería respetar las reglas tradicionales del fútbol.

Es importante destacar que aunque los robots sean programados para respetar el reglamento de un partido de fútbol, esta reglas pueden violarse por errores cometidos por el robot. Por ejemplo, un error en el mecanismo de visión puede producir que un robot choque a otro cometiendo una infracción.

En E-league, la información del mundo real es provista por un sistema de visión global común a ambos equipos. Este sistema incluye una cámara de video conectada a un servidor

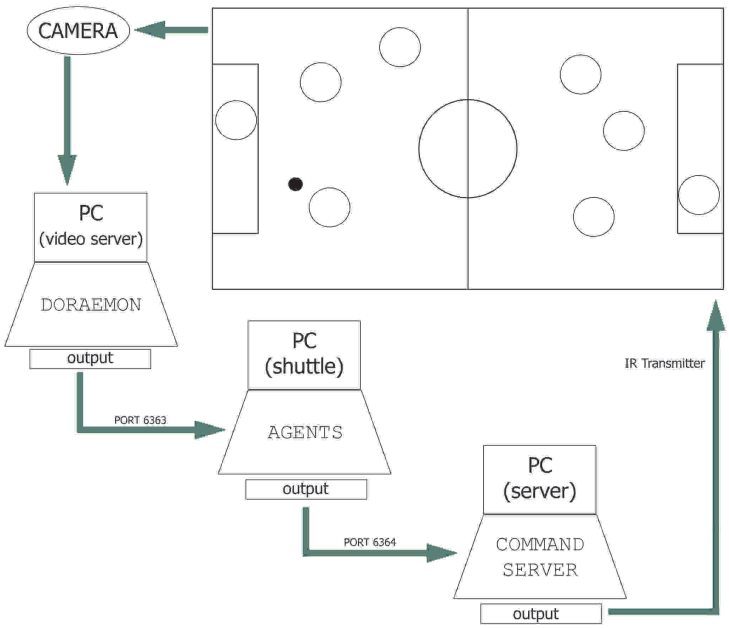

Figura 1: Esquema del hardware involucrado en la E-league [9]

(ver Figura 1) y un software de reconocimiento de imágenes llamado Doraemon [10, 8] capaz de distinguir los distintos objetos (jugadores y pelota) dentro del campo de juego.

La información de los objetos reconocidos por el Doraemon es enviada desde el servidor de visión a las computadoras de los equipos participantes en un paquete con el formato mostrado en la Figura 2

0	No. obj.	Frame no.	Time diff.						
1	CamX	CamY	CamZ						
2	Type	Name	Found?	PosX	PosY	PosZ	Orient.	VelX	VelY
⋮	...	...	...	...	...	...	...	...	...
10	Type	Name	Found?	PosX	PosY	PosZ	Orient.	VelX	VelY

Figura 2: Formato de los datos provistos por el doraemon

En nuestro caso, del paquete de información anterior, para cada objeto se extraen únicamente algunos datos: si fue encontrado o no por el mecanismo de visión (found o nofnd), su posición en la coordenadas X e Y, su orientación, y su velocidad en coordenadas X e Y. Por ejemplo, para el robot 1 podría obtenerse la siguiente información “found 100 250 120 0 0”. La coordenada X corresponde al ancho de la cancha y la coordenada Y al largo de la misma. La orientación es un valor entre 0 y 359 grados, y la velocidad puede tomar cualquier valor entero.

3. Diseño de los robots y las primitivas de movimiento

Debido a que la E-league tiene como objetivo el desarrollo de la inteligencia de los robots por sobre el desarrollo de hardware, nuestros robots fueron construidos con Lego Mindstorms [2, 1].

Las principales ventajas de usar Lego son su bajo costo y la facilidad y flexibilidad en el armado de cada robot. Sin embargo, una desventaja la presentan sus dispositivos de movimiento los cuales introducen errores al realizar acciones muy precisas. Por ejemplo, el error al intentar rotar un determinado ángulo puede ser de hasta 5 grados.

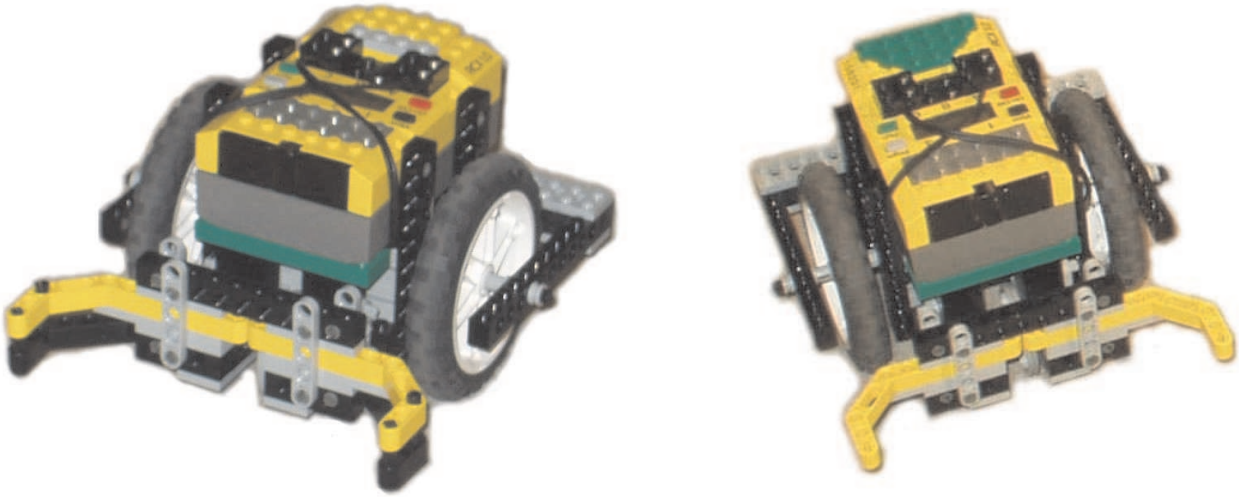

Figura 3: Dos de los robots del equipo

En el diseño físico de los robots nuestro objetivo fue que cuenten con los siguientes movimientos básicos: ir hacia delante, ir hacia atrás, rotar sobre sí mismo en ambas direcciones y realizar arcos. Estas acciones básicas serán denominadas de ahora en más primitivas de movimiento.

El diseño elegido consta de sólo 2 motores, con transmisión directa a cada rueda. Los motores fueron ubicados en el centro del robot para permitir la rotación sobre su eje. Se decidió emplear este diseño debido a que permite realizar todas las primitivas de movimiento sin la necesidad de un tercer motor para darle dirección al robot, simplificando además la programación.

La primitivas de movimiento empleadas están caracterizadas por tres elementos: la dirección, la velocidad de cada motor, y el tiempo durante el cual se ejecuta dicha primitiva (en el orden de los milisegundos). Para la ejecución de una primitiva, cada motor recibe durante un cierto intervalo de tiempo dos elementos: la dirección (adelante o atrás) y la velocidad representada por un valor entero entre 0 y 255.

La decisión de utilizar primitivas de corta ejecución en el tiempo se debe a que esto favorece la reactividad en un entorno dinámico, como lo es un partido de fútbol, donde todos los objetos se encuentran en movimiento simultaneo. Con lo cual los objetivos de cada robot pueden cambiar muy rápidamente.

4. Posibles errores en el sensado y en las acciones

Como muestra el esquema de la Figura 1, para cada robot un agente de software utilizará la información provista por el sistema de visión global, y tomará decisiones que se transformarán en primitivas de movimiento las cuales serán trasmitidas a los robots a través de señales infrarrojas.

En este esquema existen dos puntos críticos en los cuales pueden introducirse errores: el sistema de visión y la ejecución por parte de los robots de las primitivas de movimiento.

4.1. Errores en la visión

Si bien el sistema de visión permite aislarse de la captura de la información y concentrarse solamente en la toma de decisiones, este sistema presenta ciertas imperfecciones. Esto podría llevar a la construcción de un modelo abstracto para los agentes de software donde los posibles errores en la visión produzcan que se tomen decisiones incorrectas. Un objeto estático dentro del campo de juego ilustra esta situación, ya que la información provista por el sistema de visión no es siempre la misma, en algunos casos variando considerablemente. Por ejemplo, estando la pelota detenida es común que el sistema de visión entregue la siguiente información en el tiempo:

"ball found 100 250 0 0 0"
"ball found 100 255 0 0 0"
"ball found 105 240 0 0 0"
"ball found 102 252 0 0 0"

Los errores del sistema de visión se pueden categorizar de la siguiente forma:

- El objeto no es encontrado.
- El objeto es encontrado aunque su ubicación no es la correcta.
- La información de un objeto es correcta pero desactualizada.

El primer caso ocurre cuando el sistema de visión no logra reconocer el objeto. Por consiguiente el sistema de visión global informa de esta situación a través del parámetro encontrado indicando además que la información provista no es certera. Por ejemplo, en la siguiente secuencia el cuarto elemento corresponde a un error en el cual el sistema no logró reconocer al robot 1.

"robot1 found 100 250 120 5 0"
"robot1 found 105 250 120 5 0"
"robot1 found 110 250 120 5 0"
"robot1 nofnd 150 350 190 0 0"
"robot1 found 120 250 120 5 0"

El segundo problema ocurre cuando un objeto es encontrado durante un período de tiempo en una determinada ubicación y de repente aparece en otra posición, muy distinta a la anterior, sin que el objeto se halla podido desplazar hacia esa ubicación tan rápidamente. Por ejemplo, en la siguiente secuencia el quinto elemento corresponde a un error en el cual el sistema reconoció al robot 1 en una ubicación incorrecta.

"robot1 found 100 250 120 5 0"
"robot1 found 105 250 120 5 0"
"robot1 found 110 250 120 5 0"
"robot1 found 115 250 120 5 0"
"robot1 found 800 550 120 5 0"
"robot1 found 125 250 120 5 0"

La situación en que la información es correcta pero desactualizada se da generalmente con los objetos en movimiento, donde si bien la información es coherente no es correcta debido a que pertenece a un instante de tiempo anterior. Un ejemplo de esto se ve claramente en el caso en el cual el objetivo es avanzar hasta una determinada posición donde el robot debe detenerse. Debido al retraso inherente de la información el robot se detiene varios centímetros después.

4.2. Errores en la ejecución de las primitivas de movimiento

Una vez que la información proveniente del sistema de visión es procesada, y las decisiones en base a esta información fueron tomadas por un agente de software, una acción debe ser ejecutada por el robot. El problema aquí, es que no puede asumirse de antemano que la acción que realizará el robot tendrá el resultado esperado. Pueden existir diferentes fuentes de error al ejecutar las primitivas y en consecuencia las acciones realizadas podrán discernir de lo esperado en el modelo abstracto.

Hemos categorizado a los errores al ejecutar acciones de la siguiente forma:

- La carga de las baterías que accionan los motores de los robots afectan el rendimiento de éstos.
- El tipo de superficie del campo de juego afecta la respuesta de los motores.
- La inercia del robot, al momento de realizar la acción, tiene efectos sobre la acción misma.
- La diferencia entre motores similares dificulta algunas acciones básicas.

La fuente de alimentación de los robots son 6 baterías AA, las cuales, con el transcurso del tiempo se van descargando, influyendo directamente en la respuesta de los motores a las primitivas de movimiento. Por ejemplo, al ejecutar la primitiva de movimiento rotar con la máxima fuerza con la baterías totalmente cargadas, se obtenía una rotación de aproximadamente 100 grados. Mientras que esta misma primitiva ejecutada con las baterías a la mitad de su carga, se obtenía una rotación mucho menor, donde la diferencia era mayor a 30 grados.

El segundo problema ocurre debido a que la superficie sobre la cual se juegan los partidos de la liga no es única, pudiendo variar entre parquet, alfombra, cerámicos, cemento, etc. El rozamiento característico de cada terreno puede generar que los efectos de una misma acción tenga consecuencias dispares.

Debido a la inercia del robot al momento de realizar la acción, el resultado de ésta puede variar considerablemente. Por lo tanto, la acción depende no solo de sí misma, sino también del estado de movimiento del robot al momento de realizarla. Una prueba que se puede realizar para ver este problema, es la de ejecutar dos "dash" (movimiento hacia adelante) consecutivos

primero con un intervalo de tiempo entre los “dash” y luego sin el intervalo. En el primer caso la distancia recorrida es menor.

Por último, si bien todos los motores de Lego son de las mismas características, existen pequeñas diferencias entre ellos, lo cual puede influir en la ejecución de ciertas acciones básicas. Como por ejemplo avanzar en línea recta. Si a un robot le indicamos que avance hacia adelante, utilizando ambos motores a la misma velocidad, como puede ocurrir que los motores no giren exactamente a igual velocidad, la trayectoria recorrida no siempre será una línea recta.

5. Soluciones Propuestas

A continuación se detallan algunas soluciones propuestas para los problemas tanto de visión como de acción anteriormente descritos. Dichas soluciones buscan privilegiar simplicidad en su implementación, con el objetivo de minimizar el tiempo de respuesta, e intentan obtener un resultado coherente.

5.1. Soluciones a problemas de visión

En el caso del problema de los objetos no encontrados, una solución trivial sería descartar los nuevos valores y reemplazarlo por el último valor considerado como certero. Esta implementación es viable en un entorno en el cual esta situación ocurre muy esporádicamente, haciendo innecesaria una solución más compleja.

Sin embargo, cuando este problema se presenta con la suficiente continuidad como para afectar la toma de decisiones, es necesario disponer de una mejor solución. Para esto, una alternativa es predecir la posición futura del objeto de acuerdo a posiciones y las velocidades previas del mismo, considerando el tiempo transcurrido desde ese instante.

Consideremos la siguiente secuencia:

“robot1 found 100 250 120 5 0”
“robot1 found 105 250 120 5 0”
“robot1 found 110 250 120 3 -2”
“robot1 nofnd 150 350 190 0 0”

La solución consistiría en reemplazar la ultima lectura “robot1 nofnd 150 350 190 0 0” por la predicción “robot1 found 113 248 120 3 -2”.

En el caso que el objeto es encontrado pero su ubicación no es la correcta, si bien pueden utilizarse las soluciones explicadas antes, previamente se debe determinar cuando una ubicación es incorrecta. Para esto, una primera aproximación es determinar si la distancia a la última ubicación es mayor que un determinado valor. Una mejor solución, pero que a su vez requiere de mayor cómputo, sería predecir la posición y calcular la diferencia con la observada, si esta diferencia es menor que un determinado valor se acepta dicho valor y sino se utiliza la posición predicha.

Por último, el caso cuya solución requiere de mayor elaboración, es cuando la información es correcta pero está desactualizada. Una solución posible es predecir la posición del objeto

utilizando el vector velocidad. Cuando se conoce las acciones ejecutadas, es decir, cuando se trata de la posición propia, es posible realizar una predicción mas precisa, teniendo en cuenta, justamente las acciones realizadas y sus consecuencias.

5.2. Soluciones a problemas de Acciones

Con respecto a la diferencia que existe entre las acciones esperadas y las realmente obtenidas, estas pueden clasificarse en dos grandes ramas:

- Los errores estáticos.
- Los errores dinámicos.

5.2.1. Errores estáticos

Los errores considerados estáticos son de naturaleza determinista, es decir, al ejecutar la misma acción repetidas veces, en general el error encontrado es siempre el mismo. Las causas de estos errores pueden ser, por ejemplo, la superficie sobre el cual se desplazan los robots o las diferencias entre motores similares.

Un caso típico en donde la diferencia de terrenos modifica el comportamiento de los robots es al girar una determinada cantidad de grados, por ejemplo 90. Esto se debe a que el giro dependerá del tiempo en que permanezcan encendidos los motores y de la resistencia aplicada sobre las ruedas. Al variar las superficies, varía dicha resistencia, por lo tanto, para mantener el giro constante se debe aumentar o disminuir el tiempo en que los motores permanecen prendidos. Para lograr esto hay dos alternativas, la primera sería buscar los distintos valores que se adecúen a las instrucciones de rotación deseadas. Es decir, ir probando manualmente con distintos valores de tiempo hasta determinar cual es el que logra el giro esperado. Una solución mejor, aunque más elaborada consiste en que el agente mismo sea el encargado de realizar estas pruebas automáticamente. A este proceso lo hemos denominado "*autocalibración estática*", y consiste en programar al agente de manera que antes de jugar un partido ejecute diferentes pruebas con el objetivo de ir comparando la acción ejecutada con el resultado obtenido y automáticamente modificar sus parámetros para lograr que el resultado de la acción ejecutada sea lo mas cercano posible a lo esperado. Este proceso de autocalibración estática puede considerarse como un entrenamiento previo o reconocimiento del terreno y hardware a usar.

Para las diferencias entre motores similares existen diversas soluciones, una posibilidad sería, en caso de poseer varios kits de Legos, intercambiar entre los distintos motores para obtener pares con similares respuestas. Una mejor aproximación es, dado dos motores arbitrarios, uno ligeramente mas rápido que el otro, disminuir la potencia del más veloz hasta que ambos giren a la misma velocidad. Esto también puede solucionarse incorporando a la velocidad del motor como un parámetro y realizando una autocalibración estática.

Un ejemplo de esta autocalibración sería, para el caso de querer avanzar en linea recta, que el agente almacene su posición actual, se desplace hacia delante y con su nueva posición determinar si realmente su trayectoria fue en linea recta. Si este no ha sido el caso, determinar en que sentido sufrió la desviación, realizar una pequeña corrección sobre el motor correspondiente e intentarlo nuevamente. Luego, se repite este procedimiento hasta que el error obtenido sea menor al error tolerado.

5.2.2. Errores dinámicos

Los errores dinámicos son aquellos que dependen de una circunstancia particular de juego y varían a lo largo del partido. Esta característica hace que dichos errores sean casi imposibles de predecir y muy difíciles de corregir. La causa de estos errores pueden ser el drenaje de la batería, la inercia al momento de realizar la acción, entre otras.

La carga de las baterías influye directamente en la velocidad de los motores y, en consecuencia, a medida que transcurre el tiempo amplía la brecha entre el comportamiento esperado y el realizado. Debido a las reglas del juego no es posible reemplazar las baterías ni mucho menos modificar código alguno durante un partido, luego toda solución a este problema debe implicar decisiones tomadas por el agente de manera autónoma. Una forma de lograr esto sería que el robot tenga una base de conocimiento sobre las acciones que puede ejecutar y el resultado de las mismas. Dicha base de conocimiento podría ser actualizada con el correr del partido, observando para cada acción ejecutada el resultado obtenido. Esto no sólo es difícil de lograr, sino que además, implica una carga extra en la toma de decisiones del agente, aumentando el tiempo de respuesta lo cual en ciertos casos lo torna inviable.

Luego, una solución intermedia a este problema sería actualizar la base de conocimiento solamente en aquellas situaciones donde no influya tanto una toma de decisión tardía. Por ejemplo, que el arquero realice estas actualizaciones sólo cuando la pelota esté muy lejos del arco propio. Se debe tener en cuenta, que en ciertas ocasiones la acción ejecutada no es la esperada debido a circunstancias de juego, como por ejemplo el hecho de impactar contra otro robot, en cuyo caso no se debería actualizar la base de conocimiento.

En ciertos casos la acción ejecutada no es la esperada debido a la inercia del robot al momento de realizarla. Una de las alternativas para solucionar este problema es mantener la información para cada acción, con la acción previa correspondiente y el resultado obtenido. Esto genera una tabla de $N \times N$ donde N es la cantidad de acciones que puede realizar el robot. Cuando N es muy grande, esta solución se torna inviable. Un posible simplificación es solamente diferenciar para cada acción, si los motores se estaban moviendo previamente y en que sentido.

6. Evaluación de las soluciones y resultados obtenidos

6.1. Visión

Para el caso de los objetos no encontrados, la solución que nuestro equipo utilizó fue la más simple, es decir, mantener la última información certera. No se implementó una solución más elaborada debido a que en nuestro laboratorio no era necesaria, ya que éstos errores ocurrían esporádicamente. A diferencia de lo esperado, en la competencia las condiciones no eran las óptimas, generando estos errores más frecuentemente.

La otra alternativa propuesta, que consistía en predecir la posición actual en base a la historia previa, fue implementada por uno de nuestros competidores. Esto les permitió un mejor desempeño en ocasiones donde el período de tiempo en que los objetos no eran reconocidos era mayor al que nosotros esperábamos. Sin embargo, esta solución también fallaba cuando este intervalo de tiempo se extendía demasiado.

Para el caso en que el objeto era encontrado, pero su ubicación no era la correcta, el método que utilizamos para reconocer esta situación fue la de determinar la distancia entre las dos últimas ubicaciones. Si ésta era menor que un máximo desplazamiento posible, la nueva ubicación era tomada como válida. En caso contrario, se utilizaba una predicción.

Cuando la información obtenida era correcta aunque desactualizada, los resultados obtenidos por la utilización de predicciones simples no fueron lo suficientemente satisfactorios como para poder ser tomados en cuenta. Las soluciones para lograr una predicción capaz de permitir una mejor toma de decisiones requerían de una amplia base de conocimientos y mayor tiempo de procesamiento. El costo adicional de mantener esta base de conocimiento sumando al incremento en tiempo de ejecución en cada toma de decisión puede hacer que esta solución no sea viable.

6.2. Acción

Lo solución implementada para contemplar el problema de los distintos resultados de las primitivas de movimiento debida a la diferencia entre las superficies del terreno, así como el nivel de carga de las baterías fue la de autocalibración estática.

Antes de cada partido, cada primitiva se ejecutaba repetidas veces, descartando el valor mínimo y el máximo, y promediando los restantes valores. De ésta forma, se podía predecir el resultado obtenido de cada primitiva en función de esa cancha en particular, y del nivel de carga de las baterías. Los valores obtenidos eran almacenados en una base de conocimiento, la cual era utilizada para la toma de decisiones. Los resultados obtenidos a través de esta implementación fueron satisfactorios, pero su precisión dependía en gran parte de la exactitud de la información provista por el sistema de visión.

Para las diferencias entre motores similares la solución implementada fue, primero buscar las pares de motores con menor diferencia de velocidad. Una vez hecho esto, se bajó la potencia suministrada al motor más rápido, logrando que cada robot puede desplazarse en línea recta. Esta fue la implementación utilizada en un principio, y debido a que los resultados obtenidos fueron satisfactorios a lo largo de todo el proyecto, no fue necesario una implementación más compleja como hubiera sido una calibración automática previo a cada partido.

Como los resultados obtenidos por la autocalibración estática fueron satisfactorios y además considerando que en cada partido utilizábamos baterías con máxima carga inicial, se decidió no incluir una autocalibración dinámica en el código de los agentes.

7. Conclusiones y Trabajo Futuro

En general, cualquier robot que genere un modelo abstracto del entorno donde se encuentra para tomar decisiones, necesita que este modelo sea los más preciso posible. En busca de este objetivo, es imprescindible tratar de minimizar los errores que ocurren tanto en la etapa de sensado, así como también al momento de ejecutarse las acciones que modifican el entorno en que está situado.

Si bien con la intención de minimizar la brecha existente entre la realidad y el modelo, es posible utilizar hardware más preciso, las distintas propuestas presentadas en este trabajo permiten una solución más económica buscando a la vez que la eficiencia sea comparable a la provista por la alternativa anterior.

Como trabajo futuro se planea perfeccionar las técnicas de corrección de errores un módulo de "filtrado"que reciba el paquete de información del servidor de visión y para aquellos valores que no son confiables realice las correcciones correspondientes. Además se experimentará la utilización de un mecanismo de autocalibración dinámica.

Aunque el desarrollo de este trabajo está basado principalmente en la E-League, las soluciones propuestas en este trabajo podrían extrapolarse a otros dominios de aplicación y en particular a otras ligas de Robocup. En E-League se provee una visión global donde cada robot conoce la ubicación de todos los objetos dentro del campo de juego. Sin embargo, el espíritu de la Robocup es que las condiciones de juego sean las más cercanas a la de un partido real, para ello la visión de los robots debería ser parcial, dependiendo de la ubicación y orientación del robot (como es el caso de las ligas simuladas [7]).

Con un visión parcial surgen nuevas complicaciones en la determinación de la ubicación de los demás jugadores dentro del campo. Con estas nuevas restricciones la predicción de la ubicación de los jugadores cambia radicalmente debido a que éstos pueden no ser visibles por un largo período de tiempo, generando la necesidad de nuevas técnicas de corrección y predicción. Como complemento, es posible emplear técnicas de comunicación entre agentes de un mismo equipo para compartir la visión de cada uno.

Agradecimientos

Agradecemos a la Association of Computing Machinery (ACM) por su apoyo para cubrir la mayor parte de los costos de la participación la competencia RoboCup 2004 llevada a cabo en Lisboa y a la Dra. Elizabeth Sklar y al Dr. Simon Parsons de la Universidad de Columbia (New York) por hacer que esto haya sido posible. También estamos muy agradecidos con el Departamento de Ingeniería y Ciencias de la Computación de la Universidad Nacional del Sur por permitirnos disponer del espacio y los recursos necesarios en todo momento; y con muchas otras personas y empresas que colaboraron con este proyecto (ver [6]). Los autores de este trabajo agradecen especialmente a Sebastián Gottifredi, Nicolás D. Rotstein, Diego García, Telma Delladio y Gerardo Simari, quienes forman parte del grupo que desarrolló este equipo de robots y que colaboraron de diversas maneras en la elaboración de lo descripto en este trabajo.

Referencias

[1] First open-source operating system for the lego mindstorms rcx controller. http://brickos.sourceforge.net/.

[2] Lego mindstorms robots and rcx controllers. http://www.legomindstorms.com.

[3] Official e-league webpage. http://agents.cs.columbia.edu/eleague/.

[4] Official webpage of the robocup 2004 competition, held in lisbon, portugal. http://www.robocup2004.pt.

[5] Official webpage of the robocup federation. http://www.robocup.org.

[6] Página oficial del equipo matebots. http://cs.uns.edu.ar/∼ajg/matebots.

[7] The robocup soccer simulator. http://sserver.sourceforge.net/downloads.html.

[8] John Anderson and Jacky Baltes. Doraemon user's manual. http://sourceforge.net/projects/robocup-video.

[9] Alejandro J. García, Gerardo I. Simari, Telma Delladio, Diego R. García, Mariano Tucat, Nicolás D. Rotstein, Fernando A. Martín, and Sebastián Gottifredi. Cognitive robotics in a soccer game domain: A proposal for the e-league competition. In *6to Workshop de Investigadores en Ciencias de la Computación (WICC), Universidad Nacional del Comahue*, pages 289–293, 2004.

[10] Benn Vosseteig Jacky Baltes and John Anderson. Robocup e-league video server. http://sourceforge.net/projects/robocup-video.