Evasión de obstáculos con bajo costo computacional para un equipo de fútbol de robots

Nicolás D. Rotstein
ndr@cs.uns.edu.ar

Alejandro J. García
ajg@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
Tel: (0291) 459-5135 / Fax: (0291) 459-5136

Resumen

En este trabajo se presenta un método de evasión de obstáculos desarrollado para un equipo de robots que compitió en la E-League dentro del Campeonato Mundial de Fútbol de Robots RoboCup 2004. La solución a este problema es central dentro de este desarrollo ya que una tarea básica que debe realizar un robot es la de trasladarse evadiendo posibles obstáculos, los cuales pueden estar en movimiento. El método propuesto está basado en la teoría de los campos de potencial y fue desarrollado desde su concepción para lograr un muy bajo costo computacional. El objetivo de este trabajo es describir el método desarrollado y su implementación. También se incluye un análisis del mismo indicando su costo computacional y sus limitaciones.

Palabras clave: Inteligencia Artificial – Robótica Cognitiva – Evasión de obstáculos

1. Introducción

Una tarea básica que debe realizar un robot al desenvolverse en un medio físico es la de trasladarse desde su posición actual hacia una ubicación deseada. Para lograrlo, debe tener la capacidad de evadir posibles obstáculos, estáticos o dinámicos en cuanto a su posición.

Para solucionar este problema es común encontrar, en la literatura, la utilización de la teoría de los campos de potencial. Esta teoría describe las fuerzas de atracción y repulsión entre masas y se utiliza como modelo para el método de evasión de obstáculos que se analizará a continuación.

El robot que utiliza este método posee una *ubicación actual* y tiene que alcanzar un *punto destino*. En el mundo en que se mueve hay *obstáculos*, cuya posición y forma son conocidos. El objetivo es lograr que el robot se mueva desde su posición actual hasta la ubicación final evitando colisionar con los obstáculos que se le presenten en su recorrido. Cabe aclarar que estos elementos pueden estar en movimiento, lo cual introduce una dificultad adicional.

De acuerdo al modelo de los campos de potencial, el método de evasión le asocia al *punto destino* una *fuerza atractiva* y, a cada obstáculo, una *fuerza repulsiva*, de forma tal que el agente siga un camino determinado por la sumatoria de estas fuerzas y alcance su objetivo rodeando los obstáculos que se presenten en el *camino directo* hacia él. Como ejemplo puede tomarse el siguiente: en un partido de fútbol, el *punto destino* podría ser la pelota y los obstáculos serían los demás jugadores. Este es un caso en el que todos los elementos en juego están en movimiento.

El objetivo de este trabajo es describir el método desarrollado y su implementación. También se incluye un análisis del mismo indicando su costo computacional y sus limitaciones.

2. Entorno: Partido de Fútbol

Los problemas y soluciones que presentamos en este trabajo surgen del desarrollo de un equipo de robots que compitió en la E-League dentro del Campeonato Mundial de Fútbol de Robots RoboCup 2004 [ROB04], llevado a cabo en Lisboa, Portugal.

En esta liga los partidos transcurren en una cancha rectangular de 274 cm. x 152 cm. y los equipos están conformados por cuatro robots cada uno, cuyas dimensiones máximas están determinadas por el volumen de un cilindro de 15 cm. de altura y base de 22 cm. de diámetro.

En nuestro caso, los robots fueron construidos utilizando kits LEGO Mindstorms [LEG04,BRI04] y, para su locomoción, se los proveyó de dos ruedas con movimiento independiente.

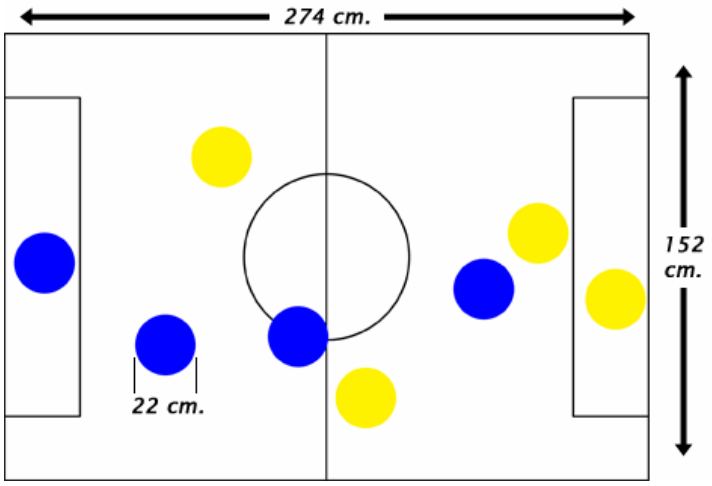

Figura 1: Esquema de la cancha y los jugadores

A pesar de que la captura de datos nos provee información tridimensional [GAR04], en la implementación de este método se consideraron sólo dos dimensiones; es decir, el entorno con el que se trabajó es *bidimensional*: las posiciones de los objetos son puntos en el plano y determinan dónde se encuentra el centro de los mismos. No fue necesario modelar un entorno tridimensional, ya que los jugadores se mueven sobre el campo de juego, sin despegarse de éste ni levantar la pelota intencionalmente. Además, el hecho de que se juegue con una pelota de golf hace que la probabilidad de que ésta se desprenda del piso sea muy baja. Teniendo en cuenta esta bidimensionalidad y debido al tamaño máximo impuesto por la liga, se asumirá que los jugadores son círculos de 22 cm. de diámetro.

Cada uno de los jugadores considerará al resto (compañeros y rivales) como obstáculos. En algunos casos, la pelota también podrá ser considerada un obstáculo, lo cual está contemplado en el algoritmo.

Anteriormente se mencionaba el dinamismo de las situaciones que se pueden presentar cuando un robot se traslada. En el entorno de un partido de fútbol es deseable que un robot futbolista tenga la capacidad de buscar los huecos dentro del campo de juego (ie, lugares libres de obstáculos). Esta puede ser una característica interesante en un delantero, quien debe estar lo más libre posible para ser un buen receptor de pases. Este es el caso de un *punto destino* dinámico, ya que el hueco estará determinado por las posiciones de los otros jugadores en la cancha, los cuales posiblemente estén en movimiento.

3. Propuesta de Implementación

Para implementar una solución utilizando la teoría de los campos de potencial puede construirse una ecuación matemática que represente la fuerza de atracción en cada punto; un ejemplo de ello es la descripta más adelante en esta sección. A partir de esta ecuación puede hallarse un máximo restingido a un área circular con centro en la posición del jugador. En la Figura 2 puede observarse el camino que debe seguir el jugador ***j*** para llegar al *punto destino* ***t***, de acuerdo a la aplicación de las fuerzas atractivas y repulsivas mencionadas anteriormente en cada punto de la cancha.

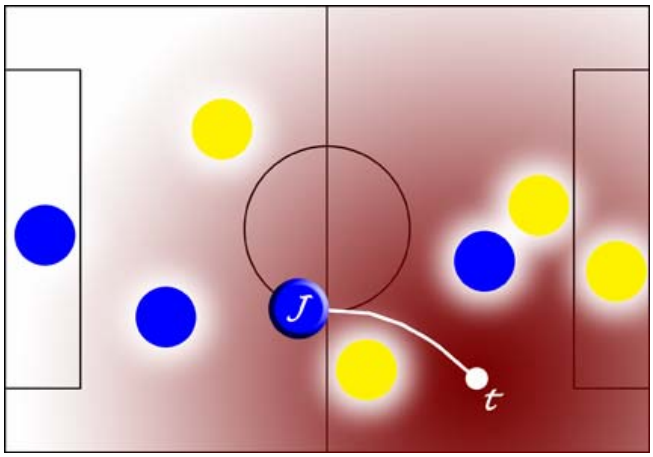

Figura 2: visión global aplicando los campos de potencial

Sin embargo, al intentar obtener una expresión para el punto de máxima atracción dentro de dicho entorno (en función de las posiciones de los objetos), la misma se tornó muy compleja, incluso considerando sólo un obstáculo. Esta complejidad se hizo evidente cuando se intentó derivar la expresión utilizando software matemático, ya que hubo casos en que no se obtuvo resultado alguno; en el resto de los casos, la solución era aproximada por un polinomio de una longitud inmanejable. Por esta razón decidió discretizarse el problema con la técnica que se explicará a continuación, representando el campo de juego mediante una grilla.

Para determinar el próximo punto al cual dirigirse, el jugador evaluará en un entorno (cuyo tamaño será explicado más adelante en esta sección) con centro en su posición la resultante de las fuerzas que lo rodean. Este entorno estará conformado por un conjunto de celdas y el próximo punto al cual dirigirse será el correspondiente al centro de la celda que tenga el mayor valor de atracción, entre todas las consideradas. Una vez obtenido dicho punto, el agente se dirigirá hacia él siguiendo una trayectoria recta. Por razones que se analizarán más adelante, el conjunto de celdas que representa el entorno de evaluación pertenecen al borde de una región romboidal; por esto, de aquí en adelante se la denominará *frontera de evaluación*.

En la Figura 3 se observa al jugador ***j***, quien tiene como *punto destino* a ***t***, y a la *frontera de evaluación*, conformada por el conjunto de celdas marcadas con la letra ***f***. Cada una de estas celdas será llamada *celda de evaluación* y aquella que tenga el mayor valor de atracción se denominará *próximo punto de avance* (identificada como ***p*** en el gráfico). Es importante notar la escala de colores utilizada en todas las figuras que esquematizan la grilla indica el grado de atracción de cada celda: cuanto más oscuro sea el tono, más alto será el valor de la fuerza de atracción resultante en esa celda en particular.

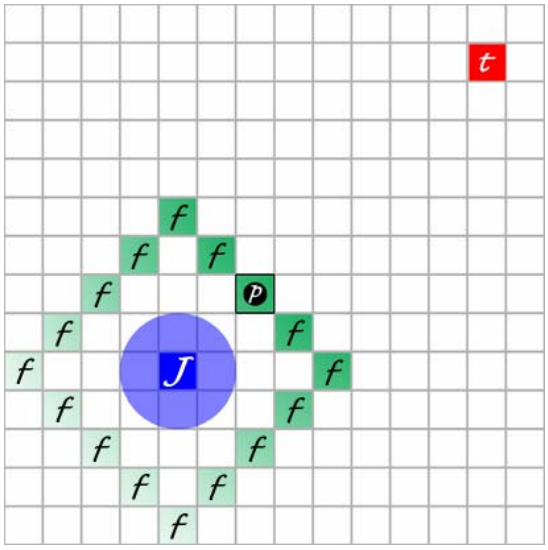

*Figura 3: el jugador **j** se dirige hacia el destino **t** a través del punto **p***

El valor de la fuerza de atracción resultante en cada *celda de evaluación* está determinado por la siguiente ecuación:

$F = F_a - \sum_{i=1}^{n} F_i$, donde:

- *n* representa la cantidad de obstáculos;
- $F_a = \frac{k_1}{D_t}$, representa la fuerza atractiva asociada al punto destino;
- $F_i = \frac{k_2}{D_i}$, representa la fuerza repulsiva asociada a cada obstáculo.
  - La constante k_1 determina la magnitud de atracción.
  - La distancia D_t es la distancia entre la celda considerada y la celda destino.
  - La constante k_2 determina la magnitud de repulsión.
  - Cada distancia D_i es la distancia entre la celda considerada y la ubicación del obstáculo *i*.

Los valores de las constantes utilizadas son obtenidos empíricamente y tienen como objetivo ajustar el rango de valores que tomará la fuerza de atracción resultante en una celda. En nuestro caso, los valores que utilizamos para estas constantes fueron:

- $k_1 = 50000$;
- $k_2 = 10$.

Cabe aclarar que, a los fines de una mayor precisión en los cálculos, siempre se utilizan las *posiciones reales* de los objetos, no aquellas correspondientes al centro de las celdas asociadas a ellos.

El cálculo de estas fuerzas en cada *celda de evaluación* y la posterior elección del *próximo punto de avance* permitirá que el agente pueda navegar por el campo de juego evitando las colisiones con el resto de los robots, como puede verse en la Figura 4. También allí se detallan los

valores de atracción en cada una de las celdas destacadas, siendo el mayor el correspondiente al *próximo punto de avance* ***p***.

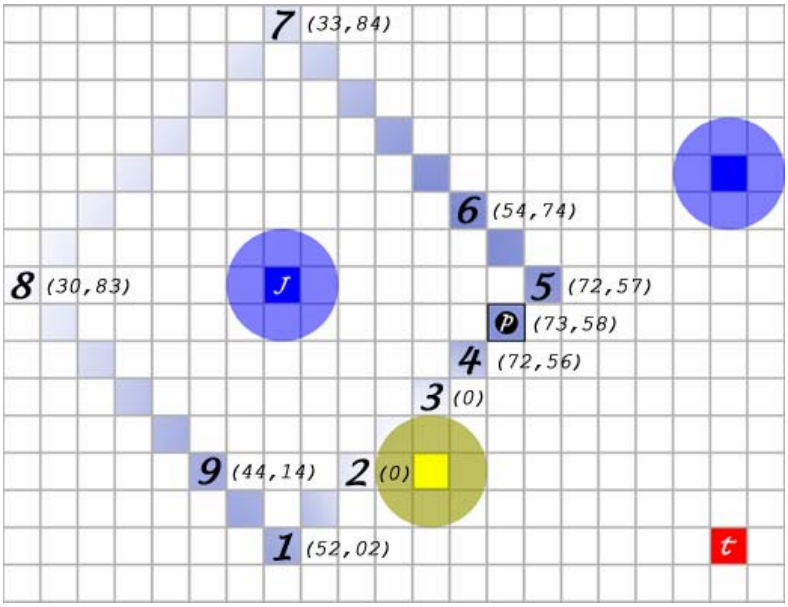

Figura 4: valores de atracción en algunas celdas de la frontera de evaluación

En lo que resta de la sección se detallarán diferentes cuestiones referentes a la *frontera de evaluación*, como el radio de la misma, el porqué de su forma y algunas consideraciones especiales que pueden mejorar la performance de este método. También se introducirá el concepto de *entorno de colisión*, mediante el cual se explicará el valor nulo de las celdas 2 y 3.

3.1 La frontera de evaluación

La decisión de considerar sólo el borde de la región se apoya no sólo en la menor cantidad de celdas que contiene la frontera con respecto al entorno completo, sino también en los siguientes resultados experimentales:

Punto de próximo avance	Cantidad de corridas
En el borde del área	26
Dentro del área	5

Como puede verse, sobre un total de 31 corridas, en el 84% de los casos el algoritmo eligió un *punto de próximo avance* ubicado sobre el borde de la región romboidal.

En general, es posible considerar cualquier tipo de forma para la *frontera de evaluación*, siempre que cumpla con la propiedad de estar contenida en el entorno circular cuyo radio sea la distancia al obstáculo más cercano. Esto significa que el radio impuesto por el círculo antedicho es un máximo que debe respetarse para el correcto funcionamiento del algoritmo.

El radio de la *frontera de evaluación* está determinado por la distancia al obstáculo más cercano (como se observa en la Figura 4); en caso de que esta distancia supere un valor máximo determinado por la implementación, se asignará dicho valor al radio del entorno. La razón de este máximo es acotar el tiempo de ejecución del algoritmo a un valor conocido.

Como la evaluación por parte del agente se realiza hasta el objeto más cercano, nunca se elegirá como *celda destino* una celda ubicada detrás de alguno de los obstáculos. De no ser así, el método podría fallar, ya que una vez obtenido el objetivo, el agente se dirige hacia él siguiendo una trayectoria recta.

La forma óptima de la *frontera de evaluación* es circular; de esta manera, se evalúan celdas equidistantes a la posición del agente. Sin embargo, el costo computacional de generar un conjunto de celdas tal que describa un círculo alrededor del agente involucraría un cálculo de distancia por cada una de ellas (ie, una raíz cuadrada), lo cual atentaría contra la velocidad del algoritmo. Por esto, se definió la forma romboidal mencionada anteriormente.

Se realizará una consideración final para contemplar el caso en el que el robot esté próximo al *punto destino*: a medida que el robot avanza, el tamaño de la *frontera de evaluación* está dado por la distancia al obstáculo más cercano. Este funcionamiento debe modificarse cuando el *punto destino* está encerrado por dicha frontera, ya que la celda asociada al objetivo (de aquí en adelante, *celda destino*) debería ser evaluada para que el agente se traslade directamente hacia ella. Una optimización posible es reconocer esta situación y no realizar evaluación alguna, ya que las propiedades de la *frontera de evaluación* aseguran que existe un camino directo libre de colisiones desde la *ubicación actual* hacia cualquier punto dentro de ella.

3.2 El entorno de colisión

Debido a que los robots tienen un tamaño máximo determinado, puede evitarse el cálculo del valor de atracción en aquellas celdas que caen dentro de cierto entorno con centro en el obstáculo. Debido a que dentro de este entorno se produciría una colisión inevitable, esta región se denominará *entorno de colisión* y en todas las celdas pertenecientes a él se asignará un valor de atracción mínimo. De esta manera, el agente nunca intentará dirigirse hacia un punto que implique una colisión segura con otro jugador. El radio de este entorno será llamado *radio de colisión*.

El mismo tipo de consideración vale para el caso en que la *celda de evaluación* en cuestión sea la *celda destino*. En este caso, se asigna un valor máximo (a menos que se encuentre dentro del *entorno de colisión*).

El valor mínimo elegido estará determinado por la elección de las constantes k_1 y k_2, las cuales determinan las magnitudes de atracción y repulsión, respectivamente. Por ejemplo, si se desea que dicho valor mínimo sea 0 (como puede verse la Figura 4), debe ajustarse el valor de las constantes realizando el siguiente análisis:

$$F_a - \sum_{i=1}^{n} F_i > 0 \equiv F_a > \sum_{i=1}^{n} F_i$$

Considerando el peor caso, en el cual F_a es mínimo y $\sum_{i=1}^{n} F_i$ es máximo:

- el *punto destino* está a la distancia máxima entre dos puntos en la cancha, tomando como referencia a la *ubicación actual*. Considerando las medidas del campo de juego, esta distancia sería:

  $$\sqrt{2740^2 + 1520^2} = 3133{,}37 \text{ mm.};$$

- los 7 jugadores están a 220 mm. de distancia (la mínima), lo cual, si bien es prácticamente imposible, facilita el análisis matemático y no alterará la correctitud del algoritmo;
- la pelota es considerada un obstáculo y está a 130 mm. de distancia (el radio de la pelota es de 20 mm.).

Resolviendo aritméticamente bajo esas consideraciones obtenemos:

$$\frac{k_1}{3133} > k_2 \times \left(\frac{7}{220} + \frac{1}{130} \right) \equiv$$
$$\equiv \frac{k_1}{3133} > k_2 \times \frac{113}{2860} \equiv$$
$$\equiv k_1 > k_2 \times 123,79 .$$

Luego, para obtener valores de fuerza de atracción en un rango lo suficientemente amplio, puede aumentarse el valor de k_1 según lo sugieran los resultados experimentales.

La longitud mínima del radio del *entorno de colisión* está determinada por el análisis del caso que puede verse en la siguiente figura:

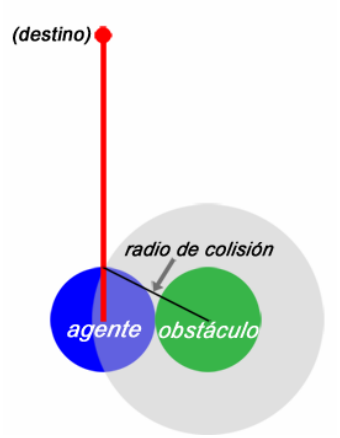

Figura 5: determinación del radio del entorno de colisión

El caso mostrado en la Figura 5 marca un punto de inflexión, ya que el *obstáculo* está en el límite de interferir en la trayectoria del *agente* hacia el punto *destino*. Es decir, si el obstáculo se mueve ínfimamente hacia arriba y a la izquierda, el agente debe esquivarlo.

Sean:

- $r_{col}^{mín}$ el radio mínimo del *entorno de colisión*;
- r_{rob} el radio de cada uno de los robots.

La distancia entre los centros de ambos objetos es, en este caso, $2 \times r_{rob}$. Entonces, $r_{col}^{mín}$ es la hipotenusa de un triángulo rectángulo cuyos catetos miden r_{rob} y $2 \times r_{rob}$, respectivamente, y se calcula de la siguiente manera:

$$r_{col}^{mín} = \sqrt{{r_{rob}}^2 + (2 \times r_{rob})^2} = \sqrt{5} \times r_{rob} .$$

La razón por la cual este radio es mínimo es porque no puede contemplarse la posibilidad de que sea más corto, ya que esto implicaría potenciales colisiones. En cambio, sí puede considerarse un radio más grande, lo cual provocaría que el agente evada obstáculos con un mayor margen de error, es decir, más conservadoramente. Como los obstáculos están generalmente en movimiento, esta actitud conservadora puede resultar muy conveniente.

4. Implementación del Algoritmo de Evasión Propuesto

Asumiendo que los robots constan de un ciclo "sensar-avanzar", si fueran programados sin utilizar un sistema de evasión de obstáculos, luego de sensar la ubicación del objetivo, su locomoción básicamente se resumiría a perfilarse hacia él y luego avanzar. Esto es, se dirigirían hacia el *punto destino* en línea recta.

Contando con un método como el explicado en este trabajo, en lugar de avanzar directamente al objetivo final, el ciclo "sensar-avanzar" se plantearía objetivos parciales que, puestos en sucesión, formarán un camino (no necesariamente recto) hacia el *punto destino*.

Para implementar este algoritmo debe contarse con ciertos datos, los cuales pueden ser de entrada o variables globales:

- posición actual del robot;
- posiciones de los obstáculos;
- posición del objetivo.

También pueden considerarse parámetros especiales:

- si la pelota es obstáculo o no;
- valor de ajuste al radio del *entorno de colisión* (para esquivar con un mayor margen de error).

En nuestro caso, el algoritmo implementado tiene como dato de salida el *próximo punto de avance* y obtiene las posiciones de los obstáculos mediante variables globales. Consta de los siguientes tres pasos:

- obtenerFrontera (CelAct, CelDest, FrontEval): recibe la celda con la *ubicación actual* del robot y la *celda destino* y devuelve el conjunto de celdas que conforma la *frontera de evaluación*.
- aplicarCampo(FrontEval, CelDest, FrontValores): recibe el conjunto de celdas que conforma la *frontera de evaluación* y la *celda destino* y devuelve la *frontera de evaluación* conteniendo los valores de atracción correspondientes a cada una de sus celdas.
- elegirMayor(FrontValores, ProxCelAvance): recibe la *frontera de evaluación* conteniendo los valores de atracción correspondientes a cada una de sus celdas y devuelve la celda cuyo centro es el *próximo punto de avance*.

4.1 Limitaciones del modelo

El algoritmo propuesto presenta algunos problemas inherentes a cualquier algoritmo basado en el modelo de campos de potencial. Estas limitaciones se hacen evidentes cuando el método es puesto en práctica; por ejemplo, le es imposible sobrepasar obstáculos en forma de 'U'. Este tipo de obstáculos induce un máximo local al evaluar el *próximo punto de avance*. Por esto, el robot, al intentar esquivar, mantiene un movimiento oscilatorio enfrente de los obstáculos y nunca alcanzará el *punto destino*, a menos que las posiciones de éstos cambien. Esto puede verse claramente en la Figura 6.

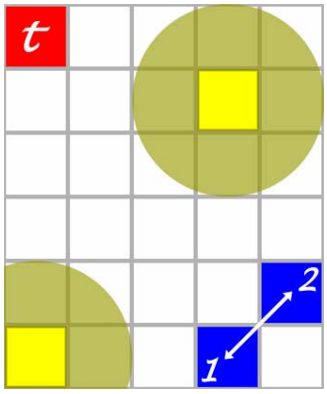

Figura 6: el jugador oscila entre las posiciones 1 y 2.

4.2 Propuesta de solución para las limitaciones del modelo

Para lograr que la resolución del problema de evasión de obstáculos sea completa puede usarse un planificador global cuando se detecten este tipo de situaciones. Este planificador analizaría las posiciones de todos los objetos en el campo de juego y calcularía un camino hacia el objetivo. Se dice que la solución es completa porque siempre que haya un camino desde la posición actual hasta el objetivo, el método lo encontrará. La razón por la cual no es conveniente utilizar un planificador global como primera estrategia, se debe a su altísimo costo computacional como para ser aplicado en cada ciclo en un sistema de tiempo real.

Cabe aclarar que en nuestro caso no implementamos una solución para esta limitación. Este se debe a que estimamos que el dinamismo en las alternativas de un partido de fútbol es suficiente como para que estas situaciones sean resueltas con la rapidez necesaria como para que no se interrumpa la fluidez del juego.

5. Costo Computacional

El análisis que sigue a continuación muestra por qué nuestro algoritmo es de bajo costo computacional. En cada invocación del método se genera el conjunto de celdas que conforma la *frontera de evaluación*; esta etapa tiene un tiempo de ejecución del orden de la cardinalidad de dicho conjunto (lo cual es analizado más adelante). Esto también se cumple en la etapa siguiente, en la cual se aplican los campos de potencial a cada celda de esta frontera. Aunque ambas tareas pueden realizarse en un mismo bucle, esto no alteraría el orden del tiempo de ejecución.

El tiempo que insume la asignación de un valor de atracción a una celda es constante, ya que está dado por una sumatoria cuya cantidad de términos es fija y el cálculo de cada uno de éstos es, simplemente, una inversa de distancia multiplicada por una constante.

La cardinalidad de la *frontera de evaluación* está determinada por dos factores:

- su radio;
- el tamaño de la celda.

Como fue dicho antes, el radio de la frontera de evaluación será la distancia a la que se encuentra el obstáculo más cercano o un valor máximo impuesto por la implementación. El peor caso sería un radio de la mitad del ancho de la cancha (ver Figura 7).

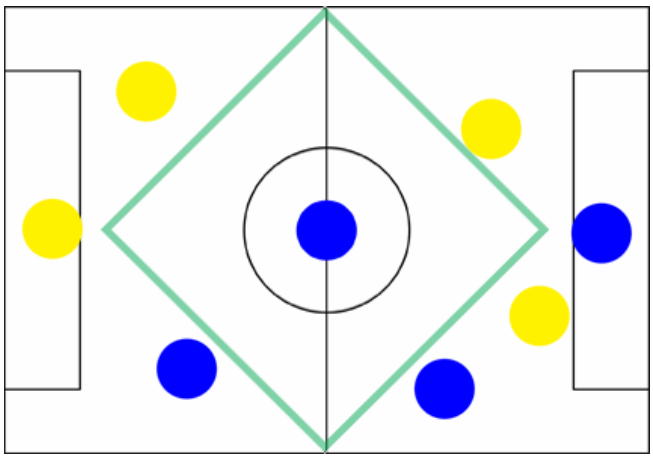

Figura 7: peor caso para la frontera de evaluación

Para calcular la cantidad de celdas que contiene la *frontera de evaluación* debe tenerse en cuenta que cada lado del rombo está formado por una celda por fila; por esto, la cantidad total de celdas es el doble de la cantidad que tiene el ancho de la cancha. Considerando celdas de 4 cm. de lado y sabiendo que el campo de juego tiene 152 cm. de ancho, entonces el total de celdas de la *frontera de evaluación* es:

$$\frac{152 \times 2}{4} = 76 \text{ celdas}$$

El caso en el cual la *frontera de evaluación* está conformada por la menor cantidad de celdas se da cuando el robot está junto a un obstáculo (eg, la Figura 5). Cuando esto ocurre, la distancia hasta el obstáculo más cercano es de 22 cm. (ie, el doble del radio de un robot). Por lo tanto, haciendo el mismo análisis que antes, se desprende que la cantidad de celdas de la frontera es 22 para el mejor caso. El caso promedio está dado por el punto medio entre la cantidad de celdas del mejor y el peor caso, es decir 49 celdas.

Como puede verse, aún en el peor caso el algoritmo evalúa la fuerza de atracción sobre una cantidad muy reducida de celdas, y como fue explicado antes, el cálculo en cada una de ellas insume un tiempo constante.

6. Conclusiones y Trabajo Futuro

El método de evasión de obstáculos propuesto en este trabajo, está basado en la teoría de los campos de potencial y surgió como una solución a una parte del problema de lograr que un equipo de cuatro robots pueda jugar al fútbol. Esto provocó que la ejecución del algoritmo sea evaluada constantemente en un entorno físico, ya que la locomoción es inherente a toda rutina de comportamiento implementada en los jugadores.

Por otra parte, es importante destacar que el método fue desarrollado, desde su concepción, teniendo en mente su desempeño en un sistema de tiempo real, en el cual el objetivo fue que el tiempo entre dos mensajes consecutivos enviados a los robots no exceda los 200 milisegundos. Además, debe tenerse en cuenta que el tiempo destinado a la ejecución de este algoritmo debe ser acotado, ya que el razonamiento que debe llevar a cabo cada uno de los robots es la parte principal del problema.

La velocidad de ejecución del algoritmo propuesto puede ser ajustada modificando el tamaño de las celdas en la grilla. Al aumentar este tamaño se logra que el algoritmo sea más rápido, pero se

pierde precisión. El tamaño final de las celdas deberá estar determinado por una evaluación empírica de la performance del método; en nuestro caso, a cada lado de las celdas le asignamos 4 cm., lo cual arrojó resultados precisos y con una velocidad satisfactoria. Sin embargo, es importante destacar que este tamaño es muy dependiente del problema.

En futuros trabajos, para mejorar la calidad de este método, se considerará lograr que la *frontera de evaluación* se aproxime a una circunferencia, ya que (como se ha analizado anteriormente) esto arrojaría resultados más precisos. También será analizada la posibilidad de utilizar un borde más grueso como *frontera de evaluación*, dependiendo de los resultados experimentales que se obtengan.

Finalmente, es importante notar que este método es incompleto, en el sentido de que la existencia de un camino hasta el *punto destino* no implica que dicho camino será encontrado por el algoritmo. Para cubrir esta falencia, se medirá la factibilidad (así como el beneficio) de implementar la siguiente extensión: detectar las situaciones en las cuales el método no converge a un camino válido y ejecutar un planificador global que destrabe esa situación.

Agradecimientos

Agradecemos a la Association of Computing Machinery (ACM) por brindarnos la beca que posibilitó que pudiéramos cubrir la mayor parte de los costos de la participación en esta edición de RoboCup 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 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 http://cs.uns.edu.ar/~ajg/matebots).

Los autores de este trabajo agradecen especialmente a Sebastián Gottifredi, Fernando Martin, Mariano Tucat, 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.

El trabajo fue financiado parcialmente por SeCyT Universidad Nacional del Sur (Subsidio: 24/ZN09) y por la Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 N° 13096).

Referencias

[BRI04] First open-source operating system for the Lego Mindstorms RCX Controller. URL: http://brickos.sourceforge.net/.

[DOR04] Benn Vosseteig, Jacky Baltes and John Anderson. Robocup e-league video server. URL: http://sourceforge.net/projects/robocup-video.

[DOM04] John Anderson and Jacky Baltes. Doraemon user's manual. URL: http://sourceforge.net/projects/robocup-video.

[ELE04] Official E-League webpage. URL: http://agents.cs.columbia.edu/eleague/

[GAR04] Alejandro J. García, Gerardo I. Simari, Telma Delladio, Diego R. Garcia, Mariano Tucat, Nicolás D. Rotstein, Fernando A. Martín y Sebastián Gottifredi. Cognitive Robotics in a

Soccer Game Domain: A Proposal for the E-League Competition. 6*to Workshop de Investigadores en Ciencias de la Computación (WICC 2004)*, Universidad Nacional del Comahue, Neuquén, 20 de mayo de 2004.

[LEG04] Lego Mindstorms robots and RCX controllers. URL: http://www.legomindstorms.com

[ROB04] Campeonato Mundial de Fútbol de Robots. Robocup. URLs: http://www.robocup2004.pt y http://www.robocup.org