

El equipo Morasot

Claro, Martín, Gazolli, Andrés, Potenza, Aníbal, Viscuso, German, Ierache, Jorge

Facultad de Informática Universidad de Morón
Cabildo 134 Morón (1708) Pcia. Buenos Aires
República Argentina
Teléfono: 5627-2000 Interno 272
informatica@unimoron.edu.ar

Resumen: El equipo Morasot de la Universidad de Morón compite exclusivamente en la categoría Simurosot (FIRA), cuenta con un equipo dinámico que modeliza el ambiente utilizando una arquitectura orientada a objetos. Algunas de las características importantes del equipo son: manejo de roles y formaciones, predicción y retrosección, división y ponderación del campo de juego y un módulo de control de faltas independiente.

I. Introducción.

El equipo Morasot surgió por una convocatoria (a principios del 2003) de la cátedra de Robótica de la Facultad de Informática gracias al advenimiento de CAFR2003. Es por esto que a pesar de que los integrantes tienen conocimientos en otras áreas son novatos en lo referente a fútbol de robots.

En las primeras reuniones se discutió implementar el equipo con una arquitectura similar a la de un sistema experto utilizando una base de hechos y otra de reglas. Era factible porque el procesamiento de la estrategia del equipo podía centralizarse en un módulo de control (dll en el simulador RobotSoccer) que se comunicara con el simulador y calculara todos los movimientos de los jugadores. Por cuestiones de complejidad y para lograr descentralizar la arquitectura en los comienzos, se descartó esta opción pero podría resultar de interés para nuestro equipo una vez hayamos adquirido experiencia en el manejo en bajo nivel del equipo.

Es así como nuestros primeros esfuerzos estuvieron orientados a que los jugadores pudieran realizar las tareas más básicas y a enriquecer el ambiente para proporcionarles

información más útil.

Pronto surgió la necesidad de particionar el campo para determinar el correcto posicionamiento de nuestro equipo de donde rápidamente surgió la idea de definir formaciones de juego.

A partir de este punto se pudo comenzar a pensar en cómo queríamos que juegue nuestro equipo (aunque no todas las primitivas de juego estaban terminadas o refinadas). Comenzamos a evaluar técnicas de swarm-intelligence pero la óptica predominante fue definir una estrategia y no hacer de ésta un emergente. Es así como entonces se decidió ponderar el campo de juego para determinar rápidamente la situación del equipo y responder con estrategias básicas de juego de acuerdo a la situación respetando los roles asignados a nuestros jugadores.

Este es el estado actual de Morasot. Tenemos planeado mejorar las características de proactividad del equipo, de modelización de nuestro equipo y el contrario y sería deseable agregar capacidad de aprendizaje de la estrategia del equipo contrario a medida que se desarrolla la partida.

En el punto II se explica cómo se ha modelizado el equipo, qué abstracciones se

han utilizado en el desarrollo del equipo.

En el punto III se explican las funciones de retrospectión y predicción que utiliza el equipo para la toma de decisiones.

En el punto IV se explica cómo se ha dividido el campo y se a asignado un valor ponderado a estas divisiones para determinar la estrategia y los roles de los jugadores.

En el punto V se explican las diferentes estrategias de juego.

En el punto VII se explican los distintos roles que puede asumir cada robot.

En el punto VIII se detallan la conclusión y las futuras líneas de investigación y desarrollo.

II. Modelización

En Morasot todo elemento del ambiente (jugadores y pelota) está definido por una clase C++. Si alguno de estos elementos tiene una localización en las coordenadas del campo entonces hereda de la clase `MVector3D` (fig. 1) que determina un punto bidimensional en las coordenadas de la cancha. Las responsabilidades básicas de la clase `MVector3D` son: la determinación de distancias, ángulos, sectores, áreas y caminos libres entre puntos.

La pelota está representada por la clase `MBall` (fig. 1) que, como es un elemento con coordenadas, hereda de `MVector3D`. `MRobot` (fig. 1), que también hereda de la misma clase, es una clase abstracta, superclase de `MTeamRobot` (que representa a un robot de nuestro equipo) y de `MOpponentRobot` (un robot oponente)(fig. 1). `MRobot` tiene responsabilidades como: determinar la cercanía de la pelota, la rotación del robot y determinar la visibilidad de otros elementos en el campo.

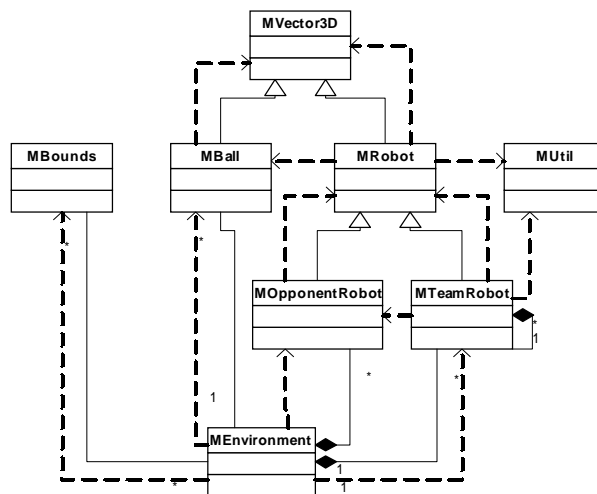


Figura 1 - Diagrama de clases de Morasot

La clase `MOpponentRobot` agrega muy poca funcionalidad a la clase base. Mientras que la clase `MTeamRobot` tiene funciones como: dirigir al robot a un punto, determinar cercanía de elementos del campo, su visibilidad y varias primitivas de acción.

La clase `MEnvironment` (fig. 1) representa al ambiente y cumple varias funciones siendo la mas importante la organización del juego y el agrupamiento de estrategias.

La clase `MBounds` encapsula los límites del campo. Mientras que la clase `MUtil` contiene rutinas matemáticas como conversión y normalización de ángulos (fig. 1).

III. Predicción y retrospectión.

Las funciones de retrospectión se utilizan para que los jugadores puedan tomar decisiones basados en jugadas pasadas. Las de predicción se utilizan para que los jugadores puedan anticiparse a los sucesos del juego y jugar teniendo en cuenta hechos que podrían suceder a corto plazo. Para ofrecer a nuestro equipo más información que la que nos provee el servidor en un momento dado, implementamos dos funcionalidades: la recuperación de estados anteriores de juego (retrospección) y la predicción de estados futuros. En ambos casos la información se provee como ambientes, de

la misma forma en que nuestro equipo obtiene el estado actual, sólo que desfazado en el tiempo (ambientes del pasado o ambientes del futuro). De esta forma obtenemos la capacidad de retrospcción y predicción (manejo temporal).

IV. Ponderación y división del campo de juego.

El campo de juego está delimitado por áreas, sectores y zonas. Un área (fig. 2) está compuesta por dos o más sectores y tiene forma de rectángulo o cuadrado. Las zonas (fig. 3), por el contrario, ofrecen una visión más real del campo de juego. Por ejemplo, definen la zona de ataque, la zona de corner, la zona defensiva, etc.

Para definir las áreas y los sectores se declararon constantes de posición para el eje X y para el eje Y. Cada constante determina una posición en el campo de juego, trazando una línea imaginaria. Las áreas generalizan y los sectores especifican el campo de juego. Estos elementos se utilizan para saber la posición de los robots y de la pelota. También para posicionar a los robots del equipo dentro del campo de juego. Esta delimitación es el primer paso para establecer una estrategia.

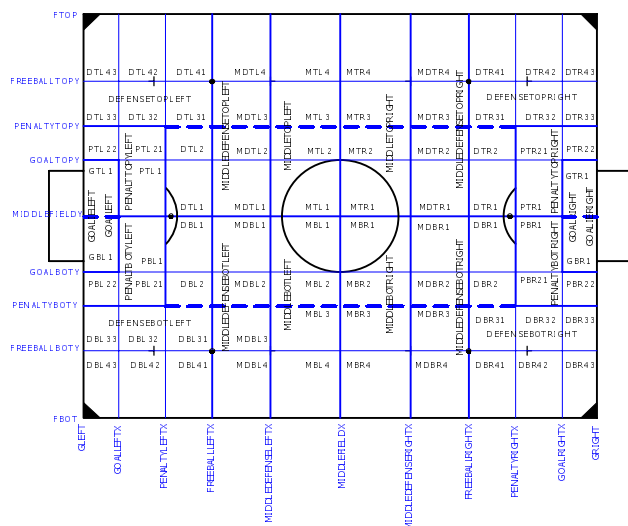


Figura 2 - Sectores del campo

Para definir las zonas se usaron los sectores. Una zona está compuesta por uno o más sectores.

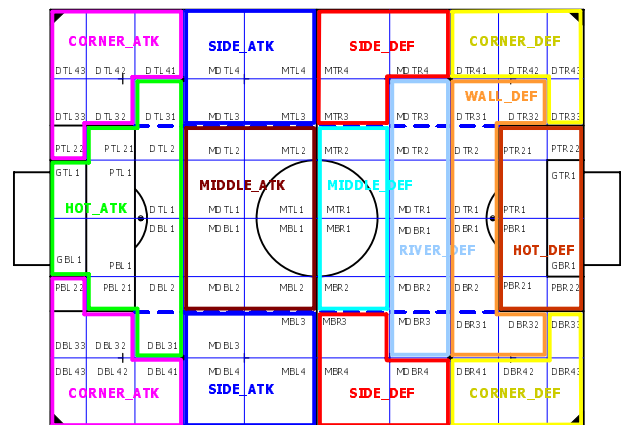


Figura 3 - Zonas del campo

Las zonas facilitan la adopción de posiciones en función de los roles a cada uno de los jugadores en función de la situación del juego. La estrategia de juego se define en función de los resultados obtenidos de la ponderación de la situación del equipo dentro del campo de juego y de la pelota. Cada zona recibe un valor con un peso determinado. La ponderación se obtiene de los valores de las zonas donde se encuentran los robots. También se considera, antes de aplicar las estrategias de juego, la posición de la pelota en el campo de juego.

V. Estrategia de Juego

De las estrategias de juego dependen los roles. Los roles son comportamientos que se le asignan a los robots. Cada estrategia asigna en que sectores, áreas o zonas puede moverse un robot y que rol debe cumplir.

A la fecha de hoy se implementaron cinco estrategias:

Estrategia de defensa fuerte

Ante una situación muy desfavorable en el campo de juego, se aplica una estrategia de defensa activa. Los roles defensivos dominan esta estrategia. pero siempre se destina un rol

de recuperación de la pelota para un posible contraataque.

Estrategia de defensa

La estrategia se aplica ante un ataque moderado. Se destina uno o dos robots para la defensa. Los robots restantes se posicionan en ataque o en recuperación de la pelota. Esta estrategia permite aprovechar un despeje por parte de la defensa.

Estrategia de mediocampo

Si se tiene una situación estable en la mitad del campo de juego, el objetivo de la estrategia es dominar el mediocampo y volcarse al ataque. La estrategia aplica los roles de robot de mediocampo y de ataque.

Estrategia de ataque

Al estar bien posicionado en el campo de juego, el equipo ataca activamente pero defiende los rebotes. Esta estrategia además de atacar intenta aprovechar los centros.

Estrategia de ataque fuerte

Si el equipo contrario se encuentra en una situación muy defensiva, la estrategia utilizará los cuatro robots para atacar. Esta estrategia se debe usar con cuidado, porque el equipo queda expuesto a un contraataque. Otra consideración es que cantidad no significa necesariamente más probabilidad de gol.

VI. Asignación de roles

Los roles definen el comportamiento que el robot tendrá en el campo de juego. Actualmente los robots no tienen implementada la capacidad para estudiar el ambiente y tomar la mejor decisión. Para las próximas versiones de Morasot, se intentará darle una mayor dinámica e independencia al robot. Los roles implementados son:

Rol Atacar área:

Para todos los roles se definen los límites de juego del robot. Si éste pasa los límites debe

volver a los mismos. Dependiendo la posición del robot se establece un ángulo de ataque. El robot intentará dirigir la pelota hacia ese ángulo.

Rol Atacar un punto:

Para este rol se necesita el punto de ataque, los límites de juego y la precisión con la que nuestro robot va a patear la pelota. Cuando hablamos de precisión decimos que un robot en función de su rol selecciona la precisión adecuada. Los roles como atacar contarán con mayor precisión a diferencia por ejemplo de un rol de defensa cuyo objetivo es despejar la pelota.

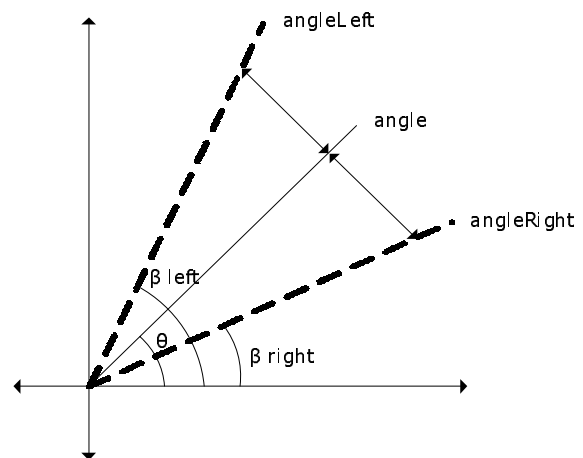


Figura 4 – Precisión de un ángulo

Rol Ataque pescador:

Como su nombre lo indica, el robot intenta aprovechar cualquier pelota delante del mismo. Por ejemplo, si se produce un centro en el arco contrario, el robot 'pescador' intentará patear la pelota hacia el arco contrario.

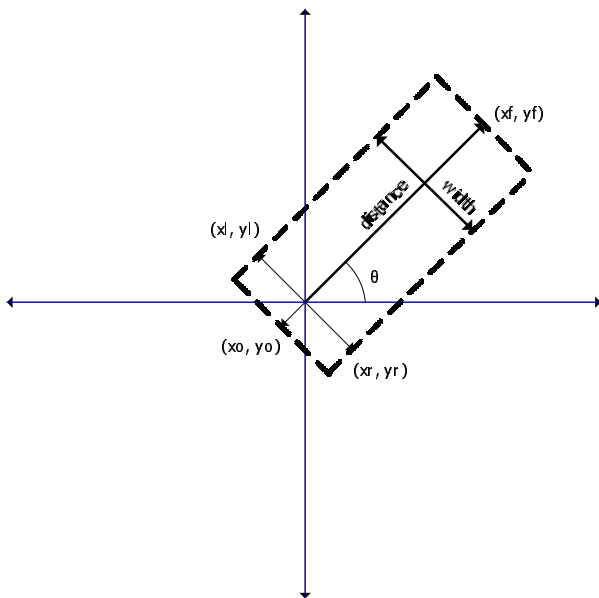


Figura 5 – Visión para interceptar la pelota

Rol Defender en forma lateral:

Para usar este rol se necesita el ángulo de defensa y los límites para el desplazamiento lateral. Se puede dar cualquier ángulo. Por ejemplo, si le indicamos un ángulo de θ grados, el robot se desplazará de forma lateral con su rotación a θ grados.

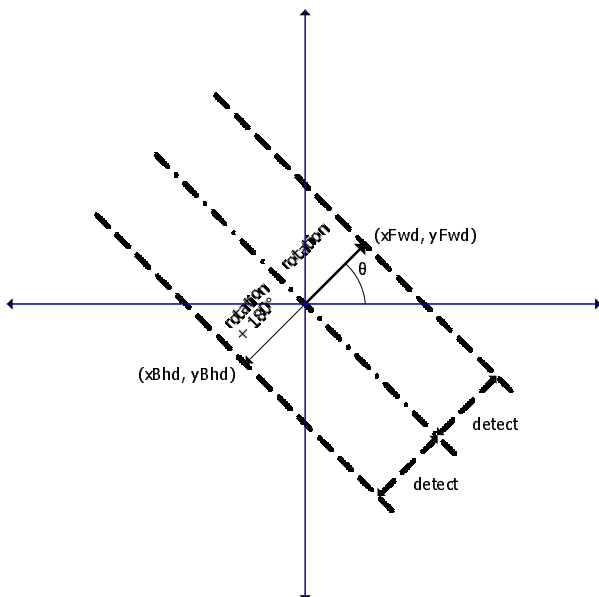


Figura 6 – Defensa lateral

Rol Defender en forma lateral y despejar la pelota:

Este rol es igual al anterior, con el agregado que si detecta la pelota la despeja.

Rol Seguir pelota:

Este rol le indica al robot que debe seguir la pelota en los límites fijados.

Rol Seguir pelota pescador:

Este rol está relacionado con el ataque ‘pescador’. Cómo en los centros la pelota tiene una velocidad considerable, el rol sólo intenta interceptar la misma.

Rol Atajar:

El objetivo del robot arquero es tratar de predecir la posición de la pelota y bloquearla con un movimiento lateral. Otras de las cualidades del mismo es el despeje de la pelota y reaccionar ante el empuje de un robot oponente.

Rol patear a un ángulo:

Este rol intenta patear la pelota hacia un ángulo dado. Si el robot se encuentra en un ángulo desfavorable intentará desde una mejor posición. También se establece la precisión del ángulo para patear.

VII. Módulo de control de faltas

El equipo implementa un módulo de control de faltas que evalúa, mediante procedimientos probabilísticos y de predicción, si los jugadores del equipo van a incurrir en una falta. Este módulo se aplica independientemente de las funciones de comportamiento de juego del equipo, justo antes de enviar las acciones deseadas de cada uno de los jugadores al simulador: el ambiente se analiza previo al envío para determinar posibles faltas. Se verifica si con los nuevos parámetros, es probable que se cometa alguna falta. El conjunto de las probables faltas a cometer determina como se deben variar los parámetros con respecto a los anteriormente calculados, con el fin de evitar cometer la o las faltas.

Algunas faltas se consideran de mayor prioridad que otras, dado que se llegó al supuesto de que evitando una falta determinada en ciertos casos, se pueden llegar a evitar otras faltas (por ejemplo en el choque con oponentes), por lo que en determinadas ocasiones no se toman en cuenta todas las faltas que son probables de ser cometidas sino aquella o aquellas de mayor prioridad.

VIII. Conclusiones

Morasot cuenta con poca experiencia en el campo pero ha disfrutado mucho el contacto inicial con el tema fútbol de robots. Desde nuestra óptica, y luego de cuatro meses de trabajo, contamos con un equipo de fútbol robótico que consideramos nuestro primer prototipo. Sin embargo contamos con muchos planes e ideas para aplicar en el futuro y las ganas de desarrollar el equipo a largo plazo y mejorarlo.

Nuestras líneas de investigación y desarrollo a corto plazo son: enriquecer la información aportada al equipo para la toma de decisiones, desarrollar mejores primitivas de juego y definir el acercamiento que se dará al equipo como sistema multi-agente e investigar las aplicaciones de machine-learning en el desarrollo del equipo

Por otra parte analizar la factibilidad de desarrollar: a) un servidor TCP para el simulador RobotSoccer de forma que podamos trascender la arquitectura de dll y elegir una mejor plataforma donde desarrollar nuestro equipo, b) una vez logrado el punto anterior desarrollar una capa de adaptación amigable para que los jugadores puedan ser manejados por humanos o adaptados a otras arquitecturas multi-agente, y que permita que nuestro equipo pueda ser adaptado a otros simuladores, c) una vez logrado el punto anterior organizar una competencia en nuestra Universidad que permita tanto la intervención de jugadores humanos como artificiales.