Desarrollo e Integración de Comportamientos en el Humanoide Nao: Un portero para la RoboCup

Juan F. García,Francisco J. Rodríguez, Vicente Matellán
Depto. Ingenierías Mecánica, Informática y Aeroespacial
Escuela de Ingenierías Industrial e Informática
Universidad de León
{jfgars, fjrodl, vicente.matellan}@unileon.es

Francisco Martín, Carlos Agüero, José M. Cañas
Depto. Sistemas Telemáticos y Computación
E.T.S. Ingenieros de Telecomunicación
Universidad Rey Juan Carlos
{fmartin, caguero, jmplaza}@urjc.es

Resumen

El trabajo descrito en esta comunicación ha consistido en la implementación de una serie de comportamientos en un robot humanoide que le capacitan para jugar al fútbol de forma autónoma como portero según las reglas de la Standard League de la RoboCup. Se describe igualmente el procesamiento básico de imágenes que se realiza y la arquitectura modular utilizada para integrarlo con las herramientas de depuración del equipo TeamChaos y los jugadores de campo desarrollados por otros miembros del equipo TeamChaos del que los autores forman parte.

Palabras clave: Humanoide, Comportamiento Reactivo, Robótica, RoboCup.

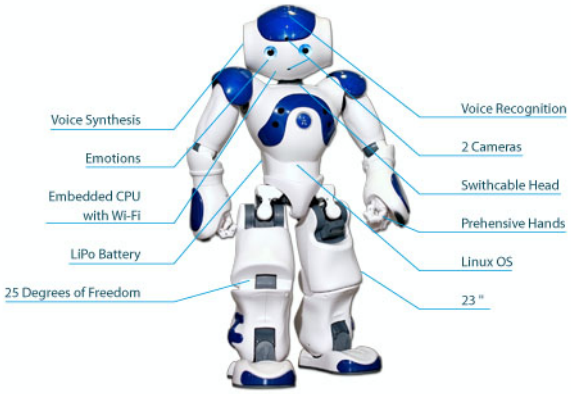


Figura 1: Nao robot (Copyright Aldebaran Robotics)

1. INTRODUCCIÓN

El trabajo descrito en este artículo consiste, resumidamente, en la programación de un robot humanoide para que sea capaz de realizar el papel de portero en un partido de la liga estándar de la *Robotic soccer WorldCup*, conocida habitualmente como RoboCup.

Brevemente, la RoboCup es una iniciativa internacional que promueve la investigación en inteligencia artificial y robótica inteligente proponiendo un problema estándar, el fútbol robótico, y organizando conferencias científicas y competiciones desde el año 1997. La información detallada sobre esta iniciativa, las diferentes ligas (categorías) y su relevancia científica, puede consultarse en el sitio web oficial: http://www.robocup.org

Este trabajo, como se ha indicado, se centra en la liga estándar. Se trata de una categoría en la que todos los equipos utilizan exactamente el mismo robot (*hardware* estándar) a diferencia de todas las demás en las que cada equipo construye o adapta su robot para competir. El robot estándar fue durante varios años el cuadrúpedo Aibo fabricado por Sony. El año 2008 se decidió su cambio por el bípedo Nao fabricado por Aldebaran Robotics. De hecho en la RoboCup2008 celebrada en Shuzou (China) hubo dos competiciones diferentes. En la RoboCup 2009 a celebrar en Graz (Austria) únicamente se mantiene la competición con los Nao.

El robot Nao es un humanoide con 21 grados de libertad, con una altura de 57 cm. y peso aproximado de 4.5Kg. Está equipado con dos cámaras con una velocidad de 30 imágenes por segundo (30 fps) y una resolución de 640x480 pixels aunque no está diseñado para la visión estéreo ya que las cámaras no están colocadas en los ojos, como podría intuirse viendo la imagen de la figura 1, sino en la frente y la boca. Únicamente se puede usar una de ellas a la vez (la conmutación de una a otra es muy lenta) y además las imágenes no se superponen.

El motivo de esa colocación de las cámaras es "histórico". En la versión comercial inicial (conocida como V2) del Nao únicamente existía una cámara, colocada a modo de cíclope en la frente del robot. La competición en la RoboCup 2008 mostró que era muy complicado ver la bola cuando se encontraba cerca (a los pies) del robot, por lo que la nueva versión (V3) se amplió con una segunda cámara situada a la altura de boca que permita al robot verse los pies sin tener que realizar movimientos extraños del cuerpo.

En cuanto a la potencia computacional, el Nao está equipado con un procesador x86 AMD Geode

a 500 MHz, con 256 Mb de memoria SDRAM y 1 Gb de meomoria flash, que puede ser ampliado si se considera necesario. Dispone también de capacidades de comunicación WiFi (802.11g) y Ethernet, aunque durante la competición todo el procesamiento debe hacerse *on-board*, y sólo e usan las comunicaciones para recibir información de la mesa, que transmite las indicaciones del árbitro (faltas, exclusiones, goles, penaltis, fin de tiempo, etc.).

Además de las cámaras, el Nao dispone de otros sensores, en concreto tiene 2 giróscopos y 3 acelerómetros, que le permiten cierta introspección de su posición (detección de caídas, por ejemplo); 2 sensores de ultrasonidos, útiles para detectar de manera sencilla a los oponentes y 2 sensores de presión en la punta de los pies para detectar si se ha golpeado algo con el pié (por ejemplo la bola).

En cuanto al software, el robot Nao se distribuye con una versión de Linux empotrado al que el fabricante ha incluido una serie de *drivers* y funcionalidades propietarias que se pueden acceder a través de un API denominado NaoQi, que tiene *bindings* para C, C++, Ruby y Urbi. También existe una versión de NaoQi para el simulador Webots de Cyberbotics[1].

El resto del artículo se organiza de la siguiente forma. En la segunda sección se describen los requisitos que se han tenido en cuenta a la hora de desarrollar el jugador. La tercera se dedica a describir la arquitectura que hemos empleado basada en esquemas. En las siguientes dos secciones abordamos las cuestiones perceptivas, en la cuarta, y motoras, en la quinta. Finalmente, en la sección sexta presentamos los experimentos realizados para comprobar la validez de la aproximación, los resultados obtenidos y los trabajos futuros que nos planteamos a la vista de los resultados.

2. REQUISITOS

Antes de especificar formalmente los requisitos con los que hemos diseñado el sistema tenemos que hacer explícita una condición de contorno que ha marcado dichos requisitos: el equipo tenía que jugar partidos en Abril de 2009 para el German Open (el equivalente a la Eurocopa del fútbol real) y una versión refinada para la RoboCup (campeonato mundial) en Junio de 2009, esta nueva versión es la que describimos en este artículo. Esta condición, conjugada con que el grupo de robótica de la Universidad de León se incorporó efectivamente al equipo TeamChaos en noviembre de 2008, encargándose de implementar el rol del portero, es la que ha marcado las decisiones.

Obviamente, en medio año escaso no hubiera sido posible realizarlo si no fuese porque se disponía de mucho código previo en el equipo. Sin embargo, esta herencia presentaba dos problemas, en primer lugar el coste de comprender código ajeno no es despreciable; segundo y más importante, dicho código había sido escrito en su mayoría para un robot diferente (el Aibo) y no había sido migrado al Nao.

Teniendo en cuenta esta condición, la misión del portero se puede resumir en buscar la pelota y calcular la mejor acción para evitar que entre en nuestra portería. La estrategia será si la bola se considera "lejana", el portero se moverá lateralmente delante de su portería para colocarse en el centro de la trayectoria teórica de la bola al centro del interior de la portería. Si la bola se está acercando, el portero deberá maximizar el espacio de la portería cubierto con su cuerpo a nivel del suelo, es decir, deberá "estirarse" en la dirección hacia la que se dirige la bola. Finalmente, si la bola está muy cerca, el robot debe tratar de chutar la bola, con cuidado de no hacerlo hacia la propia portería.

Atendiendo a nuestra experiencia previa con los robots cuadrúpedos [10], hay otra restricción crítica en el comportamiento descrito en el párrafo anterior: el portero nunca debe abandonar su área (por ejemplo para tratar de despejar la bola). Si lo hace, las oportunidades del equipo contrario de marcar un gol son muy altas. Esta restricción, combinado con la falta de tiempo para migrar el código previo de localización ([12]), nos permite resumir los requisitos de diseño en:

1. El repertorio de acciones del portero será limitado: desplazamiento lateral a derecha e izquierda (una especie de jugador de futbolín), movimientos reactivos de parada, y patadas en lazo abierto.
2. Sólo utilizaremos información indirecta de odometría para la auto-localización, es decir, se utilizará una memoria de los movimientos realizados.
3. El robot será independiente, es decir, no se coordina con los jugadores de campo
4. El comportamiento será reactivo con respecto a la posición de la pelota, no existiendo ningún otro tipo de plan concurrente.
5. El tiempo de procesamiento de la visión (reconocimiento de la bola y cálculo de la distancia y orientación) debe estar acotado y ser en media inferior a 20ms.

[1]http://www.cyberbotics.com/products/webots

Además de nuestros propios requisitos, hemos intentado realizar una búsqueda bibliográfica sobre este rol como portero, pero debido a que la competición de los Nao es muy reciente, no hay referencias que se refieran concretamente a este papel. Si existen descripciones de equipos completos, como el de la universidad de Texas-Austin [4], que fue finalista en la edición del 2008, o bien, descripciones generales de las capacidades del Nao, como por ejemplo las relativas a la locomoción descritas en [5]).

En concreto, la jerarquía de comportamientos descrita en este artículo se parece bastante a la desarrollada en el equipo chileno [3] y el griego [2]. Esta arquitectura es la que describimos en la siguiente sección.

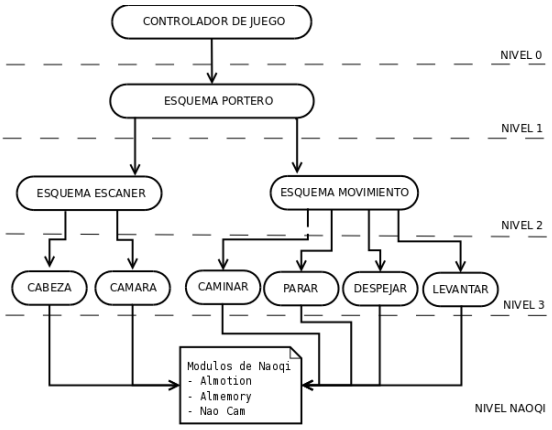


Figura 2: Jerarquìa de esquemas del portero diseñado

3. ARQUITECTURA

Aunque en los requisitos hemos descartado la coordinación con los demás jugadores del equipo, si hemos decidido utilizar la misma arquitectura que se usa en el resto del equipo. Esta arquitectura tiene dos posibles lecturas, la arquitectura como marco teórico de organización de los comportamientos, y la arquitectura como conjunto de herramientas e interfaces, etc.

En cuanto a la parte teórica, la arquitectura del equipo TeamChaos-URJC se basa en la arquitectura JDE [7]. Resumidamente, el comportamiento del robot se organiza como una jerarquía de esquemas (inspirados en la idea etológica de esquema[1]). Con esta arquitectura se pretende obtener reacciones muy rápidas gracias a los esquemas de bajo nivel. Los esquemas de alto nivel pueden seleccionar a qué estímulos tienen que reaccionar los de bajo nivel, así como modularlos para que alinear el comportamiento del robot con los objetivos. Información más completa sobre JDE, así como ejemplos de proyectos y robots que usan JDE se puede encontrar en `htpp://jde.gsyc.es`.

En lo referente a la parte software, no vamos a utilizar las herramientas desarrolladas en el proyecto JDE-robot, en su lugar vamos a implementar una versión más reducida y eficiente, aunque siguiendo los mismos principios de la original.

La figura 2 muestra los esquemas construidos para implementar el comportamiento del portero. Describiendo la jerarquía desde la base, el *Nivel 4* es el encargado de traducir los datos de alto nivel manejados en los niveles altos a los parámetros de bajo nivel necesarios para interactuar con NaoQi. La misión principal de este nivel es precisamente independizar el resto de niveles de las características de la versión concreta de NaoQi y del propio NaoQi, de forma que en el futuro podamos, por ejemplo, utilizar nuestra propia biblioteca de movimiento diferente a la de Aldebaran.

En el *Nivel 3* están los esquemas encargados del procesamiento de información. En concreto, se encargan del procesamiento de las imágenes y de la generación de las secuencias de movimiento apropiadas. Estos esquemas funcionan de forma independiente unos de otros, comunicándose únicamente a través de variables compartidas, es decir, sin dependencias funcionales.

En el *Nivel 2* se encuentra el esquema que coordina la visión y el movimiento, generando el comportamiento de portero deseado. De esta forma, la información procesada en los esquemas del nivel inferior podrían ser usados por otros esquemas que necesitasemos incluir, por ejemplo, de información a los otros compañeros del equipo, de localización, etc.

El *Nivel 1*, el más alto de la jerarquía de esquemas, se encuentra el esquema encargado del control del partido y la interacción con el árbitro, denominado *Game Controller*. Este esquema debe responder a las peticiones recibidas por red o por pulsaciones de botones. Este esquema es común a todos los jugadores del equipo, puesto que debe responder a los mismos eventos.

Finalmente en el nivel 0 contaremos con el controlador de juego que se encarga de hacer las labores activación y desactivación de los jugadores, pero que se trata de un ente ajeno a la estructura de esquemas del Nao.

4. PERCEPCIÓN

El procesado de imágenes con hardware poco potente es una tarea costosa. En concreto, para el comportamiento reactivo del portero, necesitamos

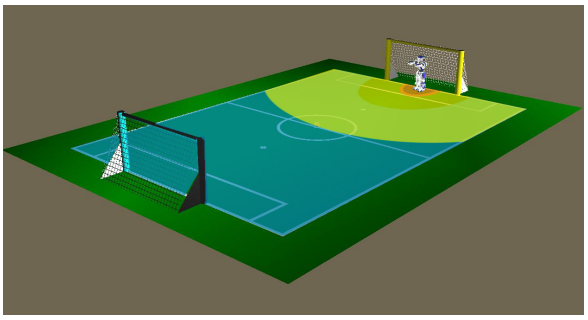


Figura 3: Zona del campo percibida según el modo de escaneo

procesar las imágenes al ritmo que las proporciona la cámara, a 25 imágenes por segundo. Teniendo en cuenta que además de procesar las imágenes el robot tiene que hacer otras operaciones, la restricción temporal es que el tiempo medio de procesamiento esté por debajo de los 20 milisegundos.

Además, necesitamos que el tiempo máximo de procesamiento esté acotado. Esta restricción es necesaria porque alguno de los algoritmos (p.e. el de reconocimiento de la bola) tienen un tiempo de procesamiento dependiente del número de píxeles involucrados que no es fijo. La cota máxima que admitimos es de 30 milisegundos. Como hemos comentado, NaoQi no proporciona un sistema de tiempo real estricto, por lo que la verificación de esta condición se ha realizado a priori, analizando el peor caso posible, el mayor número de píxeles involucrados en el caso del reconocimiento de la bola (que es cuando la bola está más próxima).

A continuación analizamos por separado las dos partes involucradas en la percepción, el control de la mirada, es decir, hacia donde apuntamos la cámara, y por otra parte el procesamiento propiamente dicho de las imágenes.

4.1. Control de la cabeza

El control de la mirada lo hemos realizado dividiendo el problema en dos partes: búsqueda de la bola y seguimiento de la misma. Para la primera parte hemos desarrollado cuatro modos diferentes de escaneo: alto, medio, bajo y muy bajo. Los tres primeros utilizan la cámara de la frente, mientras que el último utiliza la colocada en la barbilla. El uso del scaner alto queda reservado para aquellos casos en los que debido a la postura adquirida por el portero es necesario ampliar el rango de búsqueda, en caso contrario (la posición estándar) este tipo de escaneo produce una busqueda por encima del terreno de juego. El resto de tipos se distinguen por el ángulo diferente de *pitch*, que se muestran en el cuadro 1.

El esquema se encarga de ir variando el *yaw* para realizar un movimiento continuo de izquierda a derecha y de derecha a izquierda hasta los topes del actuador. Dados estos ángulos, el cuadro 1 resume el espacio percibido, que se muestra de forma gráfica en la figura 3. Es importante hacer explícito que suponemos que los objetos relevantes están en el suelo (la bola no salta).

De acuerdo con los principios de la arquitectura, es el esquema del *Nivel 2* (Esquema Escaner) el que se encarga de activar el esquema de percepción de *Nivel 3* (cámara) con los parámetros de modulación adecuados. Igualmente, es el encargado de cambiar de modo si la bola no se encuentra. La figura 4 muestra el autómata que implementa este cambio de modo de escaneo y el paso al modo de seguimiento.

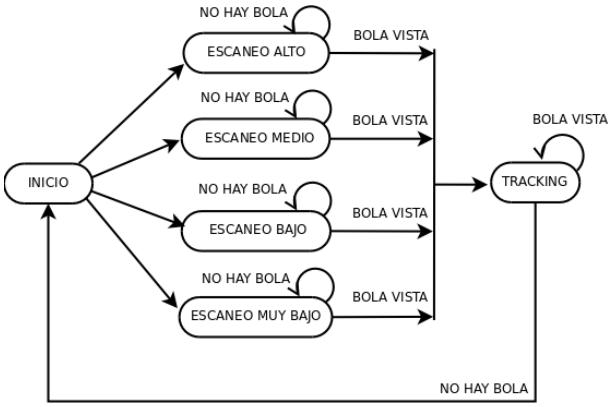


Figura 4: Autómata del control de la cabeza

4.2. Detección de la bola

La percepción de la bola se realiza en dos fases. En primer lugar realizamos una segmentación por color basada en el trabajo de James Bruce et al. [9] y en una calibración realizada *off-line*. La suposición de que la segmentación obtendría como únicos puntos naranjas los de la bola es demasiado optimista. Aunque se haya calibrado la bola en las codiciones de luz, y el entorno sea semi-controlado (el campo está regulado, pero los alrededores no), pueden aparecer múltiples falsos positivos, por lo que se necesita un procesado más cuidadoso.

El primer paso que aplicamos en el algoritmo de eliminación de puntos aislados, para lo que simplemente analizamos cuantos pixeles del mismo color existen en el vecindario del pixel. Ambos parámetros, el umbral (U) para aceptar el punto, y el tamaño del vecindario nxm centrado en (i, j), pueden ser configurables.

Tendremos entonces el algoritmo siguiente que define el procesado:

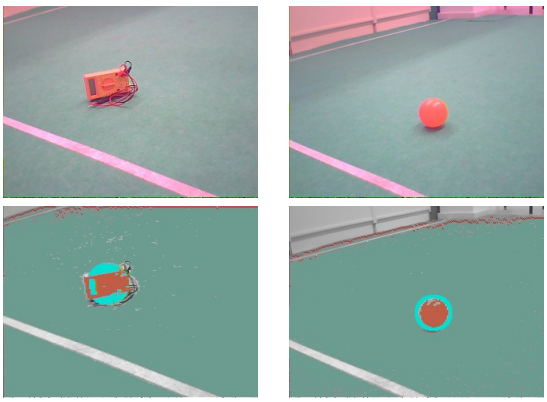
MODO	CÁMARA	ÁNGULO (rad)	$DISTANCIA_{min}$ (m)	$DISTANCIA_{max}(m)$
Alto	frente	$-\frac{\pi}{7}$	1.63	∞
Medio	frente	0	1.63	∞
Bajo	frente	$\frac{\pi}{7}$	0.51	3.00
Muy Bajo	barbilla	$\frac{\pi}{7}$	0.03	0.58

Cuadro 1: Resumen de los modos de escaneo

```
for i,j en [ancho,alto] do
    if image[i, j] ∈ color then
        for a,b in [n, m] do
            if image[a, b] ∈ color then cont++
        end for;
        if cont > U then píxel aceptado
    end if;
end for;
```

Nótese que, a la vez que segmentamos la imagen, eliminamos los puntos aislados y calculamos el centroide del *blob* naranja para la bola. Este centroide lo utilizamos en el algoritmo de reconocimiento de bola. El algoritmo trazará circunferencias concéntricas de radio cada vez mayor hasta encontrar una en la que ninguno de sus puntos pertenezca al blob naranja. En ese momento comprobará si los pixels que encierra dicha circunferencia forman una figura redondeada, pudiendo así identificar la pelota.

La figura 5 muestra el resultado de utilizar el algoritmo sobre un objeto cuya forma no se corresponde a una figura esférica buscada y el caso donde se aplica sobre la pelota oficial de la robocup.



(a) (b)

Figura 5: (a) Ejemplo de rechazo y (b) reconocimiento de bola

El mismo algoritmo de segmentación basada en el color se aplicará para filtrar los diferentes colores que pertenecen a los básicos de los elementos de la Robocup: amarillo y azul cielo para las porterías, rojo y azul para los jugadores así como verde y blanco para la indentificación del cesped y las líneas del campo.

Una vez que hemos eliminado los puntos aislados y discriminado todos aquellos elementos que no pertenecen a la zona del terreno de juego nos encontraremos en la fase de escanear y centrar nuestra posición con la bola. En este estadio, el robot se encuentra permanentemente buscando la bola hasta que es detectada, en ese momento, se activarán los esquemas motores adecuados que generarán las acciones para centrarse frontalmente con la pelota observada.

4.3. Control de Movimiento

El control de las posibles acciones cinemáticas que puede desarrollar el humanoide estarán controladas por este controlador de nivel 2. Los esquemas motores a activar en cada caso son:

Caminar: donde se definen los desplazamientos laterales que el portero podrá dar a izquierda y derecha dependiendo de la situación de la pelota.

Parar: esta acción vendrá provocada por un acercamiento de la bola, accionando los mecanismos que permiten que el robot implemente una postura de parada. estirando el brazo al lateral correspondiente en la que se puede producir peligro de gol.

Despejar: en aquellas situaciones en las que la bola se encuentre a muy corta distancia del robot, este despejará con sus brazos la pelota intentando no marcar en propia puerta o provocar situaciones de peligro de gol (aquellos casos en los que otro robot se encuentra frente a la bola a muy poca distancia y despejar puede provocar un rebote de la bola hacia nuestra portería).

Levantar: Esta opción implementada para aquellos casos en los que nuestro guardameta caiga, se encontrará desactivada para no provocar problemas de orientación cuando se levante al carecer todavía de un sistema de localización sobre el campo.

	Humboldt	Dortmund	**TC-U**	HTWK	L3M	GA	GR	GD	Pts	Rank
Nao Team Humboldt	X	0:0	**0:0**	1:2	0:0	1	2	-1	3	3
Nao Devils Dortmund	0:0	X	**1:0**	0:0	1:0	2	0	2	8	2
TeamChaos-URJC	**0:0**	**0:1**	X	**0:1**	**0:0**	**0**	**2**	**-2**	**2**	**4**
Nao-Team HTWK	2:1	0:0	**1:0**	X	3:0	6	1	5	10	1
Les 3 Mousquetaires	0:0	0:1	**0:0**	0:3	X	0	4	-4	2	5

Cuadro 2: Resultados del Grupo A. German Open 2009

ACCIÓN	TIEMPO 1 (ms)	TIEMPO 2 (ms)
Cubrir todos los ángulos (1 escaneo)	2340	2340
Captura de imagen	25	5
Tratamiento de la imagen	80	15
Centrar bola vista	2032	845
Paso (izquierda o derecha)	5670	5670

Cuadro 3: Resultados experimentales

5. EXPERIMENTOS, RESULTADOS Y TRABAJO FUTURO

Nuestra primera version del desarrollo fue probado durante el German Open 2009 celebrada en Hannover del 20 al 24 de Abril de 2009 formando parte del equipo TeamChaos en la escisión de la URJC, esta división llevada a cabo en este campeonato se realizó para que diferentes test fuesen realizados sobre todo el trabajo implementado por el equipo, pudiendo comprobar de esta forma la situación de los desarrollos en entornos reales.

En aquella prueba los resultado no fueron del todo satisfactorios (ver resultado en el cuadro 2) aunque hay que remarcar que la situación del robot en uno de los goles encajados se produjo estando inactivo debido a un problema con el sistema del "Game Controller" que es el encargado de gestionar la situación de cada uno de los jugadores en el terreno de juego.

Tras analizar la arquitectura y los datos obtenidos mostrados en el cuadro 3 *TIEMPO 1*, se emprende la refactorización de código descrita en este paper, tras la cual mejoramos la actuación del portero en varias de sus facetas (ver cuadro 3 *TIEMPO 2*):

1. Decremento de los tiempos de localización de la bola
2. Separación de los esquemas de busqueda y movimiento
3. Intencionalidad ante diferenctes situaciones (centrado, paradas...)

Se continua trabajando para conseguir que durante la Robocup 2009 que se celebrará en Graz, Austria, se alcancen los siguientes hitos:

1. Mejorar el desplazamiento lateral ya que las funciones utilizadas son aquellas que ofrece Aldebaran dentro de su conjunto de primitivas, lo que hace que se haga de una forma muy lenta.
2. Reducir más los tiempos utilizados para escaneo y segmentación de la bola.

Y sobre estos resolver el problema de la localización, de tal manera que nos permita aumentar los movimientos del portero a lo largo del area pequeña perteneciente a la portería

Agradecimientos

Los autores reconocen y agradecen el apoyo del Ministerio de Ciencia e Innovación a través del proyecto COCOGROM (DPI2007-66556-C03-01).

Referencias

[1] Lorenz, Konrad, *Foundations of ethology.* Springer Verlag, New York, 1981.

[2] Andreas Panakos et al *Kouretes 2008, Nao Team Report* Robocup 2008

[3] J. Ruiz-del-Solar, P. Guerrero, R. Palma-Amestoy, M. Arenas,R. Dodds, R. Marchant, L. A. Herrera *UChile Kiltros 2008 Team Description Paper*. Robocup 2008

[4] Todd Hester, Michael Quinlan and Peter Stone *UT Austin Villa 2008: Standing On Two Legs.* Technical Report UT-AI-TR-08-8

[5] Aaron Tay, *Exploration of Nao and its Locomotive Abilities* Exploration of Nao and its Locomotive Abilities,2008

[6] José María Cañas y Vicente Matellán. *From Bio-inspired vs. Psycho-inspired to Etho-inspired robots*. Robotics and Autonomous Systems. Vol.55, Num. 12, pp. 841-850. DOI:10.1016/j.robot.2007.07.010

[7] J.M.Cañas, J. Ruíz-Ayúcar, C. Agüero, F. Martín. *JDE-neoc: component oriented software architecture for robotics*. . Journal of Physical Agents, Volume 1, Number 1, pp 1-6, 2007.

[8] Scott Lenser and Manuela Veloso. *Visual sonar: Fast obstacle avoidance using monocular vision*. In Proceedings of IROS, pp. 391-406, October 2003.

[9] James Bruce, Tucker Balch, and Manuela Veloso. *Fast and inexpensive color image segmentation for interactive robots*. In Proceedings of IROS, Japan, October 2000.

[10] H. Martínez, V. Matellan, M. Cazorla, A. Saffiotti, D. Herrero, F. Martin, B. Bonev, K. LeBlanc. *Robotics soccer with Aibos*. In Proceedings of WAF 2005 (Workshop de Agentes Físicos) Granada (Spain). ppp 69 - 71

[11] David Herrero Pérez, Humberto Martínez Barberá. *Robust and Efficient Field Features Detection for Localization*. RoboCup 2006

[12] Francisco Martín, Vicente Matellán, José María Cañas y Pablo Barrera. *Localization of legged robots based on fuzzy logic and a population of extended kalman filters*. Robotics and Autonomous Systems. Vol.55, Num. 12, pp. 870-880.doi:10.1016/j.robot.2007.09.006