Revista Iberoamericana de Automática e Informática industrial
CEA comité español de automática
RIAI
ISSN: 1697-7912. Vol. 7, Núm. 4, Octubre 2010, pp. 17-27
www. revista-riai.org

Desarrollo de una Interfaz de Usuario para el Sistema Robótico Multiagente SMART

Cecilia García Cena, Roque Saltarén, Javier López Blázquez, Rafael Aracil

Universidad Politécnica de Madrid
Centro de Automática y Robótica (CAR) UPM-CSIC
José Gutiérrez Abascal, 2 (28006) Madrid, España
(e-mail: cecilia.garcia,rafael.aracil,roque.saltaren@upm.es,
jlb_yuyi@hotmail.com)

Resumen: En este artículo se presenta al sistema robótico multi-agente SMART. Este sistema está compuesto por varios tipos de agentes software y/o hardware, por lo que puede clasificárselo como heterogéneo. Además, se presenta el desarrollo de una interfaz de realidad virtual a través de la cual el usuario puede intervenir en el sistema si lo considera necesario, en otro caso el sistema opera autónomamente. Asimismo, esta interfaz implementa un gran número de funcionalidades tendientes a lograr el buen desempeño del sistema, una correcta gestión de los recursos robóticos disponibles en el entorno y provee al usuario información en tiempo real de todo lo que sucede en la realidad. Como consecuencia de la heterogeneidad del sistema, la comunicación entre los diferentes agentes se realiza utilizando diversas tecnologías de comunicación (TCP/IP, WiFi y Bluetooth); por lo tanto, se ha diseñado un protocolo de comunicaciones específico para este sistema.

Palabras Clave: robots, agentes, inteligencia artificial distribuida, realidad virtual, cooperación.

1. INTRODUCCIÓN

En las últimas tres décadas el interés científico por los sistemas robóticos multi-agentes (MARS) se ha incrementado considerablemente. Dicho interés se basa en la diversidad de disciplinas del ámbito de la robótica que intervienen para abordar los problemas propios de los MARS: percepción, cognición, comportamiento, coordinación y configuración entre otros. Conceptos propios de etología animal, de la teoría de organización corporativa, de aprendizaje y de la inteligencia artificial distribuida han servido de sustento teórico-práctico en esta área (Veloso M. y Nardi D., 2006; Frankiln D. et. al, 1995; Franklin D. y Gresser T., 1996; Collinot A. et. al, 1996; Deloach S. et. al, 2002).

Las características dinámicas y cinemáticas inherentes a los agentes robóticos los diferencia sustancialmente de los sistemas multi-agente computacionales (MAS) y por lo tanto las técnicas de coordinación y cooperación aplicadas a los MAS no son las más adecuadas para el tratamiento de incertidumbre y falta de información que hay comúnmente en la robótica (Veloso M. y Nardi D., 2006).

Los requerimientos que se exigen a un sistema multi-agente trabajando en entornos reales son los siguientes:

- Adquirir conductas de acuerdo con las circunstancias.
- Tomar decisiones adecuadas ante la ocurrencia de sucesos no previstos en el entorno.
- Desempeñar tareas con eficiencia y en tiempo real.
- Tener "conciencia" de la existencia de otros agentes en el entorno.

Con la inteligencia artificial distribuida se pretende dar solución a problemas como la cooperación entre agentes para cumplir exitosamente un objetivo, la generación de subtareas y la asignación de las mismas a un grupo de agentes que sea capaz de ejecutarlas y el desarrollo de lenguajes y algoritmos paralelos para sistemas concurrentes.

La unidad funcional de un sistema multi-agente es el agente (entidad física o abstracta), que es capaz de percibir su entorno a través de sensores y puede evaluar tales percepciones para generar decisiones a través de algún mecanismo de razonamiento. Además, un agente debe ser capaz de comunicarse con otros agentes, aunque sean de diferente especie, para adquirir/enviar información y actuar sobre el medio en el que se desenvuelven a través de sus actuadores.

Para los agentes sociales, es decir, aquellos que viven en comunidad con otros agentes, es imperativo contar con una estructura orgánica que garantice la correcta cooperación, la adecuada planificación y asignación de subtareas y la coordinación de las actividades individuales y colectivas (Mark S. et. al, 2008; Corkill D. y Lander, S. E., 1998).

La Inteligencia Artificial Distribuida (DAI, Distributed Artificial Intelligence) integra dos campos de conocimiento: la inteligencia artificial y los sistemas distribuidos. A partir de esto se concibe la DAI como el campo del conocimiento que intenta construir conjuntos de entidades autónomas e inteligentes que cooperan para desarrollar un trabajo y se comunican por medio de mecanismos basados en el envío y recepción de mensajes (Zhiwu Li y Shuwen Xu, 2006).

La coordinación y la cooperación entre agentes no se puede lograr si no se dispone de un protocolo de comunicaciones eficiente, sencillo y lo suficientemente rápido en el envío y recepción de mensajes.

Publicado electrónicamente: 08/10/2010
DOI:10.4995/RIAI.2010.04.04

El sistema robótico multi-agente SMART es un sistema heterogéneo compuesto por diferentes tipos de agentes: robots de tres y cuatro patas, cámara IP, software de gestión, software de control y planificación y software de procesamiento de datos. La heterogeneidad del sistema conlleva necesariamente a la utilización de diferentes tecnologías de comunicación, por lo que es necesario establecer un único protocolo de comunicación para todos los agentes del sistema. Las tareas de cada agente y la cooperación entre ellos se modelan a través de Redes de Petri (RdP), herramienta idónea para el modelado de sistemas concurrentes.

Este artículo está organizado de la siguiente manera. En la Sección 2 se presenta una breve descripción de todo el sistema: diseño mecánico, componentes, tecnología de comunicación, etc. En la Sección 3 se detalla el protocolo de comunicaciones desarrollado para la comunicación entre el entorno virtual y la aplicación software que gestiona el funcionamiento del sistema. La interfaz desarrollada se muestra en detalle en la Sección 4 y se hace una especial descripción en la Sección 5 de cómo se gestionan las tareas, modeladas con RdP, a través de la interfaz visual. Las conclusiones y posibles trabajos futuros se detallan en la Sección 6.

2. DESCRIPCIÓN DEL SISTEMA SMART

El Sistema Muli-Agente Robótico Teleoperado (SMART) está formado por agentes heterogéneos, pero que son capaces de organizarse eficientemente para realizar tareas cooperativas. El sistema cuenta con dos tipos de robots: uno de tres patas y otro de cuatro patas, tal como se puede observar en las Figuras 1 y 2.

El agente de tres patas (Figura 1) cuenta con 8 grados de libertad, lineal y rotacional en la pata delantera y uno lineal y dos rotacionales en cada pata trasera. El agente con cuatro patas (Figura 2) tiene 13 grados de libertad, 3 por cada pata más uno rotacional en el centro del cuerpo que permite hacer giros relativos entre ambas partes.

Los servomotores HS-475 de Hitec® se utilizan para la actuación rotacional. La actuación longitudinal se consigue con los servomotores lineales L12 de Firgelli®. Todos los servomotores se controlan con una placa controladora Micro Serial Servo Controller de Pololu Robotics&Electronics®, capaz de controlar hasta 8 servos. Las órdenes de control llegan al robot vía Bluetooth mediante un receptor OEMSPA312i de ConnectBlue® integrado en cada módulo con un alcance de emisión de 50m. La alimentación de los módulos se consigue gracias a unas baterías de litio-polímero EON28 fabricadas por FlightPower® que suministran 7.4V y 1800mAh.

La cooperación entre agentes de un sistema robótico supone ampliar las posibilidades de movimiento y alcance del sistema en general y, por lo tanto, será posible realizar tareas complejas que un único agente sería incapaz de realizar. La Figura 3 es un claro ejemplo de lo anterior. En ella se puede observar cómo los robots SMART, configurados convenientemente, podrían realizar tareas de sujeción, reparación, etc. en una viga o tubería.

Los agentes SMART desarrollan sus tareas en un entorno no estructurado de dimensiones acotadas. Este entorno cuenta con una cámara IP en la parte superior que envía imágenes vía sockets (TCP/IP) a un ordenador de control y al PC remoto donde se sitúa la interfaz de realidad virtual. A través del procesamiento de las imágenes se obtiene la posición y orientación de cada agente y la posición de los obstáculos que pudieran encontrarse en el entorno. Por lo tanto, la cámara IP es quien cierra el lazo de control. En la Figura 4 se muestra una fotografía de los agentes situados en el entorno real de trabajo realizando una tarea cooperativa para evitar obstáculos.

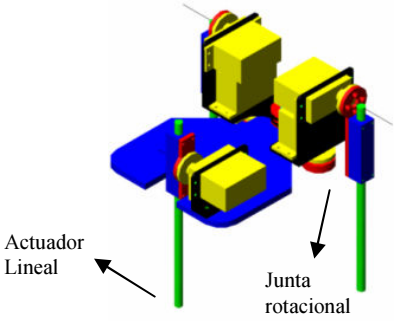


Figura 1: Robot SMART de tres patas.

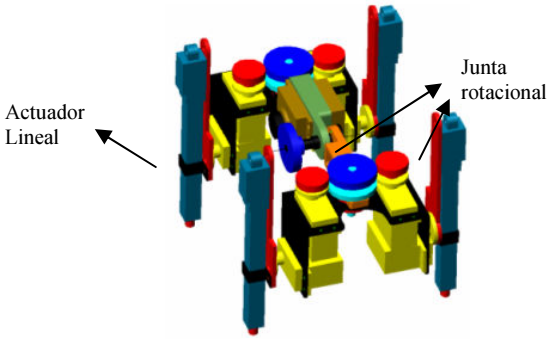


Figura 2: Robot SMART de cuatro patas.

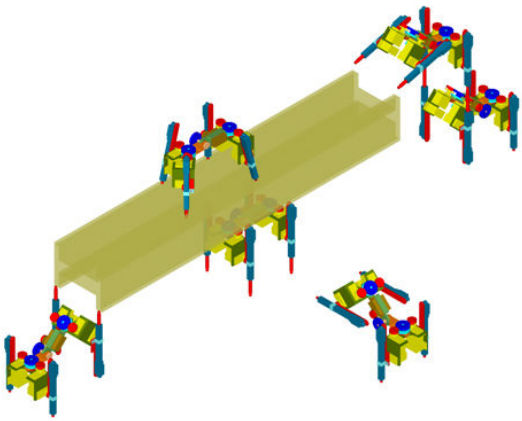


Figura 3: Cooperación entre agentes SMART para la realización de una tarea.

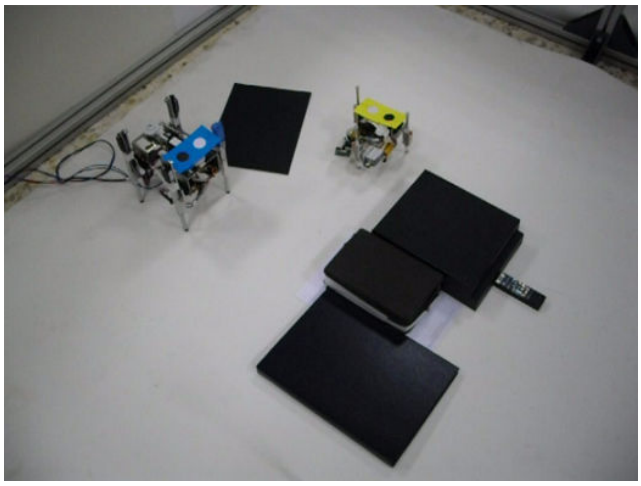


Figura 4: Entorno real de pruebas con obstáculos.

Se dispone además de un ordenador de control, conectado al ordenador remoto vía TCP/IP, en el que se realiza el control de la tarea: planificación de la trayectoria, procesamiento de la imagen, generación de las RdP, entre otras. La Figura 5 muestra un diagrama de bloques en el que se puede apreciar la arquitectura de control de un agente robótico n.

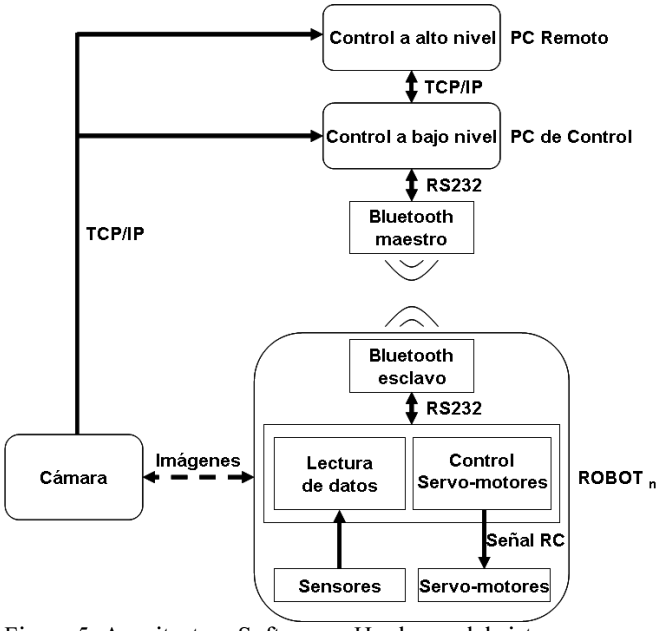


Figura 5: Arquitectura Software y Hardware del sistema.

Las tareas del sistema SMART se modelan mediante RdP y la implementación práctica de las mismas se realiza en un entorno bajo C++.

2.1 Protocolo de comunicación

La Interfaz de Realidad Virtual, que se describe en detalle en la Sección 3, no es un proyecto independiente y aislado, sino que forma parte de una estructura más compleja compuesta por diferentes sistemas. Se trata, por lo tanto, de un módulo dentro de un sistema global. Este sistema global está formado por varios subsistemas, cada uno de los cuales realizan una tarea específica, aunque comparten información a través de un protocolo de comunicaciones. En la Figura 6 se observa un esquema de la arquitectura general del sistema.

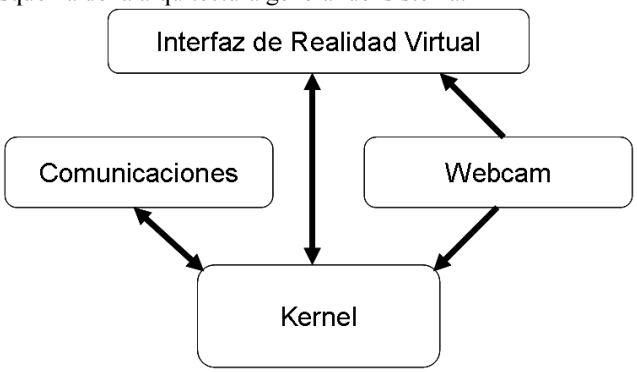


Figura 6: Arquitectura general del sistema.

El componente principal de la arquitectura software es el kernel (de Clercq S., 2009). El diseño de esta aplicación está basado en RdP y su esencia es una estructura multihilos en C++. Ente las funciones del kernel se pueden mencionar:

1- Gestionar las comunicaciones entre los robots.
2- Tratar las imágenes que recibe de la cámara IP mediante herramientas de visión artificial, para conocer la posición y orientación de los robots y de los obstáculos dentro del espacio de trabajo.
3- Enviar datos a la interfaz y gestionar los datos que ésta le envíe.

La comunicación entre la Interfaz de Realidad Virtual y el kernel se realiza con una arquitectura típica cliente-servidor conectados vía sockets bajo Windows®.

2.2 Procesamiento de imágenes

El espacio de trabajo consiste en una estructura con base rectangular de $5m^2$ aprox. donde se sitúan los agentes y los obstáculos. La reflectancia del suelo no es suficientemente alta para poder crear reflejos confusos para el sistema de captación. La porosidad de la superficie no crea problemas a la hora de determinar los píxeles asociadas a ella, etiquetados como espacio libre de circulación. En la parte superior de este cuadrilátero se sitúa la cámara de visión.

Cada agente robótico tiene en la parte superior una marca de color que no sólo permite identificarles y determinar su posición y orientación respecto a un origen de coordenadas, sino que además facilita distinguirles de otros objetos presentes en la escena que está iluminada artificial y uniformemente.

Para la captura de las imágenes se emplea una cámara IP Zaapa, color (RGB) modelo ZA-CIPRW con resolución 640 x 480 pixels. Este espacio de coordenadas es altamente sensible a condiciones variables de luminosidad, los valores RGB para un mismo color son muy diferentes si cambia la intensidad luminosa. Para solventar este inconveniente, se realiza una transformación de las imágenes a HSV. Las imágenes son transmitidas al software de procesamiento a través de una conexión ethernet. Este software, implementado en un hilo bajo C++, se comunica vía socket con el kernel de la aplicación. La Figura 7 muestra una imagen captada por la cámara de visión.

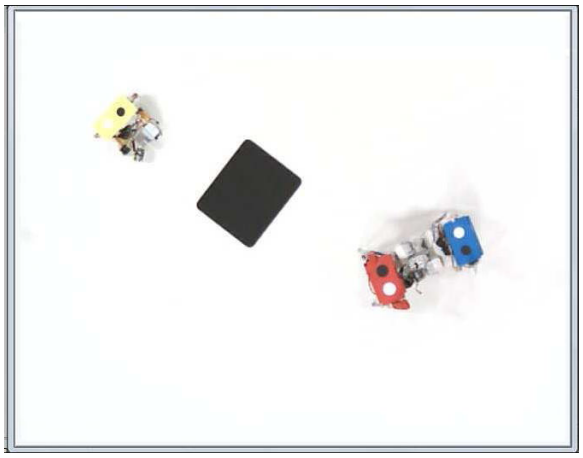


Figura 7: Imagen captada por la cámara de visión.

Las imágenes adquiridas se procesan como matrices de dimensión 680 x 460 píxeles. El origen de coordenadas de la imagen se encuentra en la esquina superior izquierda. El procesamiento de la imagen consta de las siguientes fases:

1) Transformación no-lineal de la imágen RGB a HSV5 (Palaus, 1998; Plataniotis y Venetsanopoulos, 2000): Se define el centro y ángulo del tono de cada color HSV para los cuatro SMART posibles (rojo, amarillo, verde y azul). Como el rango de saturación (S) y brillo (V) identifica cada tonalidad, cada píxel

tendrá un valor en función de estos tres atributos, quedando la imagen segmentada por colores (Figura 8):

- 0 (negro): suelo.
- 1 (blanco): obstáculo.
- 2, 3, 4, 5: SMARTS.

2) Filtrado: Se somete la imagen a un filtrado no-lineal mediante un filtro de moda (Plataniotis y Venetsanopoulos, 2000) en una región alrededor de cada píxel (vecindad 8), eliminando píxeles en blanco aislados en una nueva imagen.

3) Obtención de contornos. (Figura 9). Para cada píxel se analiza la región circundante (vecindad 8) y se observan los posibles cambios respecto los valores de los píxeles: un píxel cuyo valor sea diferente del de un vecino se define como contorno. Una vez obtenidos los contornos (Figura 9a) se operara con ellos en diferentes fases:

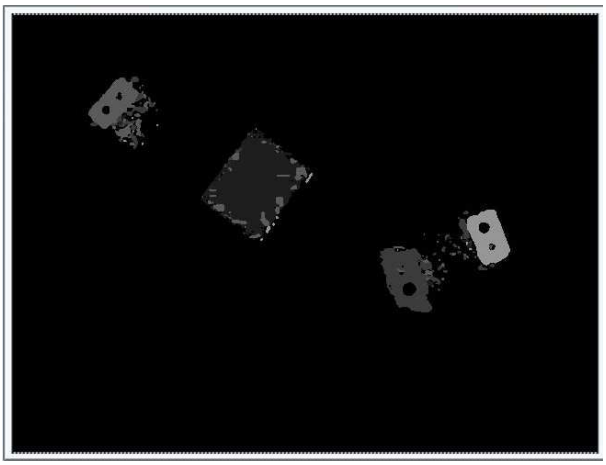
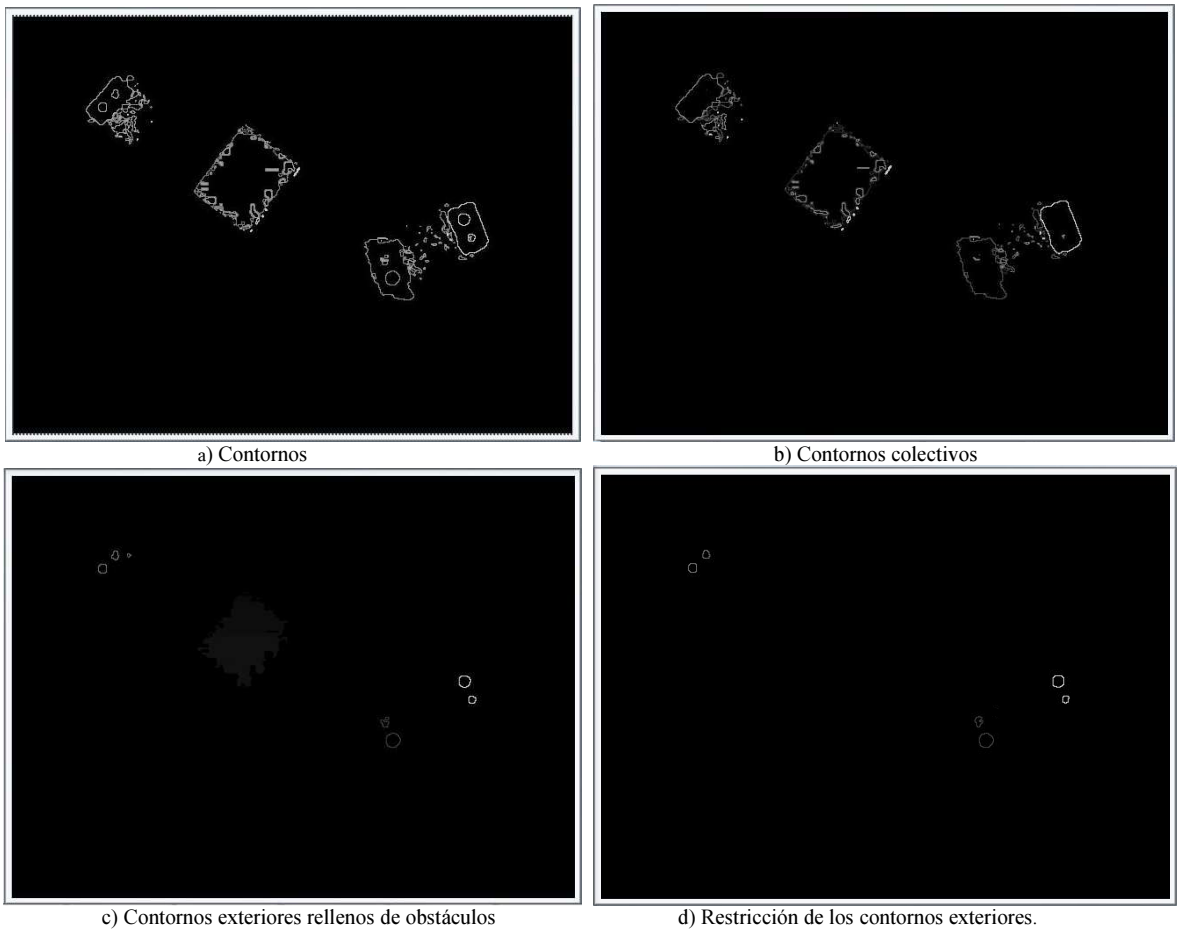


Figura 8: Segmentación de la escena.



a) Contornos b) Contornos colectivos

c) Contornos exteriores rellenos de obstáculos d) Restricción de los contornos exteriores.

Figura 9: Extracción de Contornos.

- Determinar los contornos exteriores mediante barridos horizontales y segmentación por crecimiento de regiones. Estos barridos se realizan con intervalos de varias líneas de píxeles, por lo que se procede a una post-propagación de las etiquetas para delimitar los contornos. Se crea una imagen con valor nulo en los píxeles que no son contornos exteriores. (Figura 9b).

- Determinar los contornos interiores y obstáculos, mediante combinación de la imagen total de contornos y la anterior. Los píxeles pertenecientes al interior de los contornos identificados con el valor correspondiente a los obstáculos toman el valor de obstáculo (Figura 9c).

- Identificar los robots. Se buscan aquellos contornos exteriores que poseen conjuntos interiores (se identifican por etiquetas), y se eligen entre estos últimos los dos de mayor tamaño, que cumplan ser blanco y negro. (Figura 9d).

Una vez identificados los contornos predeterminados, se define en cada robot por un punto central y una inclinación. Para ello se

toma el punto medio de la línea que une los dos contornos interiores (l_1) y se suma un vector perpendicular a dicha línea y longitud $l_1/2$, obteniendo el centro de gravedad del robot (C_R) y el ángulo respecto a la horizontal (Figura 10).

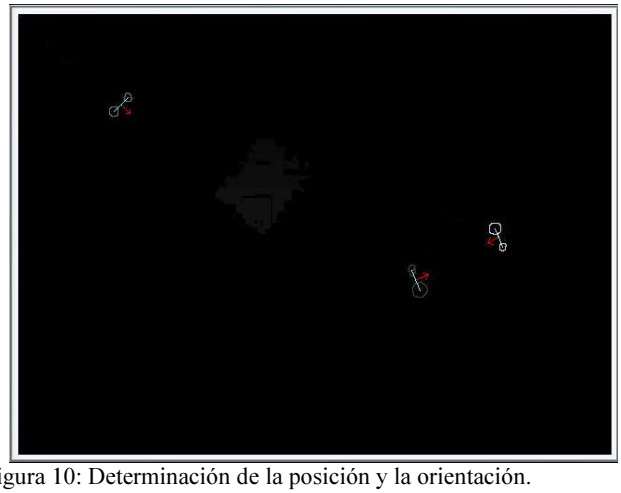


Figura 10: Determinación de la posición y la orientación.

Se representa a cada robot SMART como un cuadrado definido por el punto central (centro de gravedad del robot) y una inclinación tal como se muestra en la Figura 11.

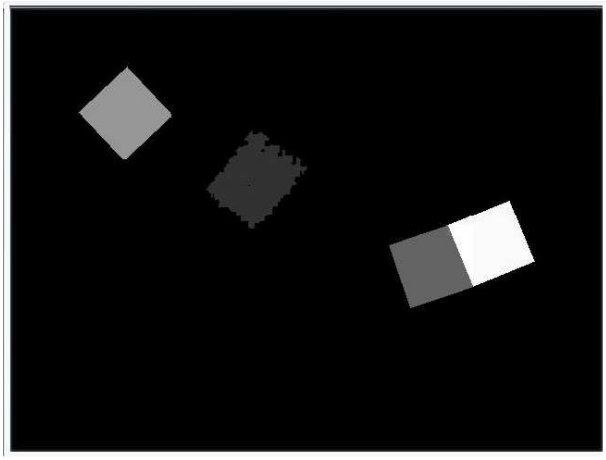


Figura 11: Representación de los robots.

Conocidas las dimensiones del espacio de trabajo se etiquetan los píxeles del borde como obstáculos y se transforman los valores de color dados a cada grupo segmentado por su equivalente en escala de grises. Esto se realiza mediante una equivalencia de los índices correspondientes a cada etiqueta con un valor de entre 0 a 255.

3. ESTRUCTURA SOFTWARE DE LA INTERFAZ

La interfaz de usuario desarrollada fue concebida bajo las siguientes premisas:

- Permitir la simulación de tareas cooperativas entre agentes, modeladas a través de RdP fuera de línea.
- Reproducir lo que sucede en el entorno real a través de la realimentación en tiempo real de la posición y orientación de los agentes, sus trayectorias y cualquier otra información relevante para un potencial usuario.
- Permitir el envío de órdenes de alto nivel a los agentes.

La implementación de la interfaz visual sigue los conceptos fundamentales de la Programación Orientada a Objetos (POO) en C++ bajo Windows. En primer lugar se realiza el Análisis Orientado a Objetos (AOO). El documento principal del AOO es el Modelo del Dominio. En él se muestran las clases conceptuales significativas en un dominio del problema. Las clases conceptuales son categorías reales del universo del problema, esto es, son la esencia del problema. En la Figura 12 se muestra el modelo del dominio propuesto para la interfaz virtual del SMART, donde las clases conceptuales se definen de la siguiente manera:

1- SMART: representa al agente. En este caso el universo del problema está formado por 5 agentes, por lo tanto, hay 5 objetos de esta clase.
2- Estructura: representa la estructura del entorno real. Existe un solo objeto de esta clase.
3- Ladrillo: representa un obstáculo de material duro. Existe un solo objeto de esta clase.
4- Madero: representa un obstáculo de material semi blando. Existe un solo objeto de esta clase.
5- Tubería: esta clase representa un obstáculo cilíndrico metálico. Existe un solo objeto de esta clase.
6- Novint Falcon: esta clase representa el joystick empleado para la tele-operación.
7- Cámara: esta clase representa la cámara IP usada para la captura de imágenes del entorno real.

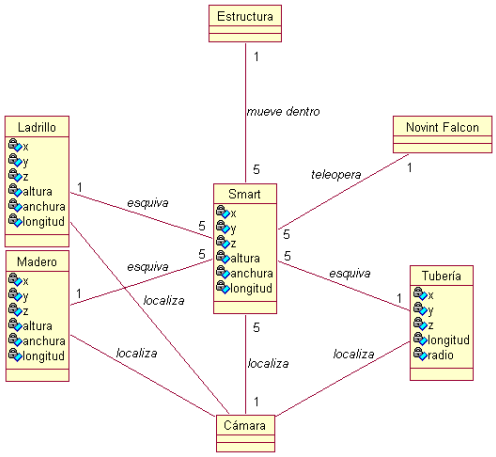


Figura 12: Modelo del Dominio.

En la Figura 13 se muestra la ventana principal de la Interfaz de Realidad Virtual. Aquí se puede apreciar que la misma tiene cuatro zonas bien diferenciadas: barra de título, barra de menús, barra de herramientas y ventana central.

La Figura 14 muestra el Diagrama de Clases de Diseño (DCD). A diferencia de las clases conceptuales del AOO, las clases de diseño de los DCD muestran las futuras clases implementadas o de software. Aquí se muestran los datos y los procedimientos de cada una de las clases. Proporciona toda la información necesaria para la implementación.

En el entorno virtual es posible observar la imagen del entorno real. Esta imagen es transferida vía socket y actualizada 11 veces por segundo, del mismo modo que los datos de posición y orientación de cada agente, trayectoria, etc.

La ventana central está dividida en dos zonas (ver Figura 13). Una de ellas es la que muestra la trayectoria y la posición de los agentes, y en la que es posible variar la posición y orientación de la cámara de la ventana gráfica. En la otra se puede configurar el entorno virtual o indicar la posición final a la que quiere que se dirija un agente.

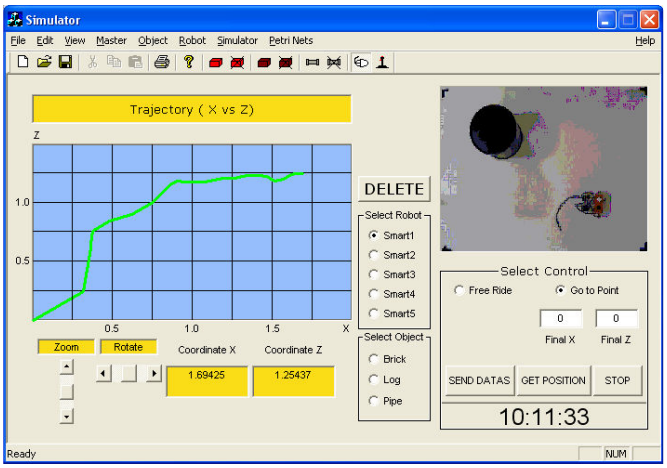


Figura 13: Ventana principal de la Interfaz de Realidad Virtual.

4. LAS REDES DE PETRI EN EL ENTORNO VIRTUAL

Las RdP constituyen una teoría para el modelado de sistemas de flujo en los que los eventos pueden sucederse de manera secuencial o concurrente. Por tal motivo, las RdP se utilizan ampliamente en el modelado de sistemas multi-agentes (Hiraishi ,2002; Hiraishi, 2008; Hsieh, 2009; Lund et al., 2003; Kotb 2007; Moldt y Wienberg, 1997; Zhiwu Li y Shuwen Xu, 2006; Fiorino y Tessier, 1998; Lee 2008). Si bien los agentes suelen entenderse como sistemas software, el concepto de modelado empleado en las referencias anteriormente citadas se puede extrapolar a aquellos sistemas en los que los agentes son robots (Corkill y Lander, 2008; Montano et. al, 2000). Por otro lado, la teoría formal de RdP permite evaluar propiedades de comportamiento de los sistemas por lo que su uso es generalizado.

Una RdP está compuesta por un conjunto de lugares (representados por círculos), transiciones (representados por rectángulos) y un conjunto de arcos dirigidos (representados mediante flechas) que vinculan lugares con transiciones y viceversa.

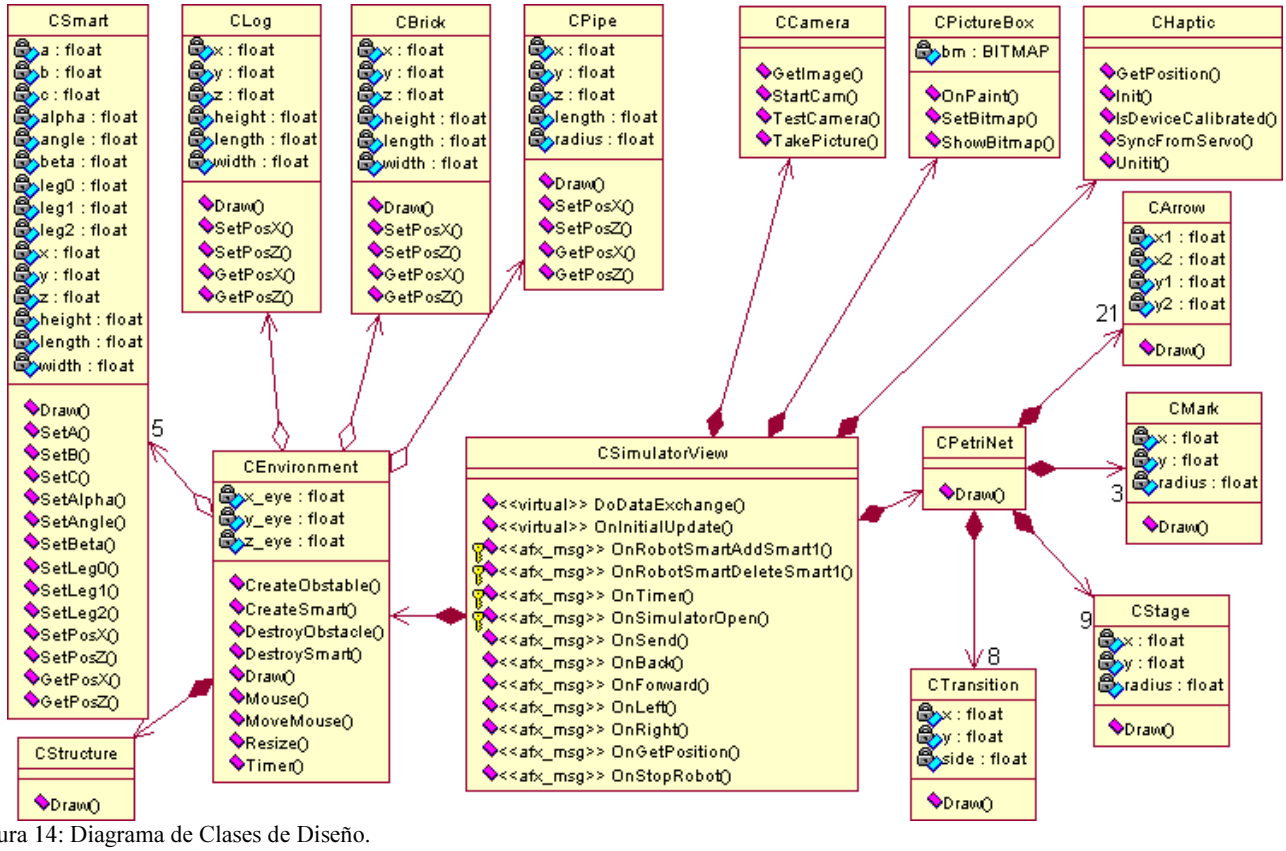


Figura 14: Diagrama de Clases de Diseño.

Una marca (representada mediante un círculo dentro del lugar) puede interpretarse como la disponibilidad de un recurso. Éstas van de un estado a otro según se disparan las transiciones y son las que permiten evaluar el estado dinámico del sistema en un instante determinado.

Formalmente, se define una RdP como una tupla $N=(P,T,F,W)$ donde P es un conjunto finito de estados y T es un conjunto finito de transiciones. Ambos conjuntos cumplen que $P \cap T = \varnothing$ y $P \cup T \neq \varnothing$. F es el conjunto de arcos tal que $F \subseteq (P \times T) \cup (T \times P)$ y el peso del arco f_{ik} que une al lugar p_i con la transición t_k se define como $w \in W : F \rightarrow Na+$ donde $Na+ = \{1,2,.....\}$ es el conjunto de enteros positivos. Un marcaje M sobre una red $N=(P,T,F,W)$ es un mapeo de $P \rightarrow Na$. $M(p_i)$ y denota el número de marcas en el lugar $p_i \in P$. Una RdP marcada, $N_M = (N,M)$, tiene un marcaje inicial M^0.

En el sistema SMART, las Redes de Petri se utilizan para el modelado de los comportamientos de cada agente y/o de un grupo de agentes trabajando en pos de un mismo objetivo. En este sistema se tienen diferentes clases de agentes: robots,

cámara IP, interfaz de usuario, bloque de control y planificador de trayectorias, procesamiento de imagen, etc. Como consecuencia, en este sistema se puede establecer la siguiente definición de agente.

Definición: un agente del sistema SMART es una colección de elementos software y/o hardware capaces de cooperar o competir para alcanzar un objetivo.

4.1 Modelado de las comunicaciones

En esta subsección se presenta como ejemplo una RdP que evita la colisión entre agentes móviles y/o obstáculos. (Figura 15).

Si se define $N_M^i = (N^i, M) \subseteq N_M$ como una sub-RdP para modelar el comportamiento de un agente, en este ejemplo se pueden distinguen 4 sub-redes:

- N_M^{IP-C} modela el software de captura de la imagen por la cámara situada en la escena.
- N_M^{ID-S} modela el software de procesamiento de imágenes para obtener la localización de los agentes y obstáculos en la escena. La captura y procesamiento de la imagen se realiza con una frecuencia de 11 veces por segundo.
- $N_M^{r_i}$ modela el moviendo de un agente robótico r_i cualquiera. Cuando esta subred actúa, implica que el/los agentes se mueven según una referencia dada y libres de obstáculos tanto fijos como móviles.
- N_M^{NCol} modela el algoritmo para evitar colisiones. Si dos agentes están demasiado próximos, esta red se encarga de detenerles y reprogramar las trayectorias.

Las RdP se implementaron en una arquitectura multi-hilos en C++ y la sincronización de las mismas la realiza el kernel (García et. al, 2010). La subred que modela la cámara está dada en (1):

$$N_M^{IP-C} = (P, T, F, W, M) \quad (1)$$

Donde P_1: Captura de la imagen, P_2: Cámara en espera, T_1: Envío de la imagen bmp a través del socket, $T_{2:}$ Nueva captura. En el estado inicial se tiene $M^0 = [1 \quad 0]^T$ y $W \in \Re^{5\times 1} = I$. La transición $T_{2:}$ ocurre solo en caso que la cámara no esté ocupada y que el software de control (modelado en P_6) lo indique.

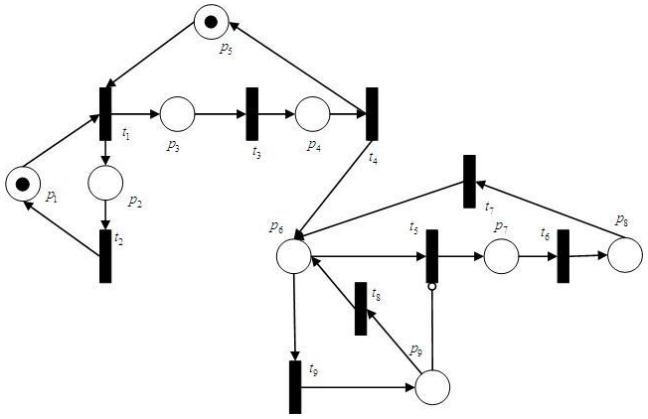


Figura 15: RdP para evitar obstáculos fijos o móviles.

Se puede modelar el comportamiento del software que procesa la imagen identificando entre agentes robots y obstáculos y, además, extraer la posición y orientación de cada entidad presente en la escena mediante (2).

$$N_M^{ID-S} = (P, T, F, W, M) \quad (2)$$

Donde P_3: Recepción de la imagen bmp a través del socket, P_4: Identificación de agentes y obstáculos. Extracción de coordenadas de posición y orientación (tal como se explicara en la sección 2.2), P_5: Espera de la siguiente imagen, T_3: Inicio del procesamiento de la imagen, T_4: Envío de la información al bloque de control de los agentes. Esta comunicación se realiza vía TCP/IP. En el estado inicial la marcación es: $M^0 = [0 \quad 0 \quad 1]^T$ y $W \in \Re^{5\times 1} = I$.

La interfaz de comunicación para enviar comandos a los agentes robots r_i, con $i = 1,2,....n$, y al robot en sí mismo se modelan a través de la sub-red dada en (3).

$$N_M^{r_i} = (P, T, F, W, M) \quad (3)$$

Donde P_6: Toma de decisión basada en la información recibida, P_9: Agente r_i ejecuta la orden de movimiento enviada por el bloque de control, T_9: Envío de comandos al agente r_i vía bluetooth, T_8: Envío de datos al bloque de control. En este caso, $M^0 = [0]$ y $W \in \Re^{3\times 1} = I$.

Finalmente, el bloque de control junto con el planificador de trayectorias se integra en una única unidad de software y por tal motivo se modelan en una misma RdP (N_M^{NCol}). El bloque de control se encarga de tomar decisiones dependiendo de la información que recibe (cantidad de agentes, ID de los agentes, posición y orientación de los mismos, cantidad de obstáculos y posición de los obstáculos).

$$N_M^{NCol} = (P, T, F, W, M) \quad (4)$$

Donde $T_{5:}$ Orden de detención del agente (envío de la posición anterior), P_7: Agente/s r_i detenido, T_6: orden de cálculo de las nuevas posiciones, P_8: calculo de las nuevas posiciones, T_7 envío de las nuevas posiciones para el software de control. $M^0 = [0]^T$ y $W \in \Re^{3\times 1} = I$.

La RdP de la Figura 15 es una red acotada, conservativa y consistente, con tres P-invariantes y T-invariantes. La matriz de incidencia **A** de la red está dad en (5).

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \quad (5)$$

4.2 Las Redes de Petri en la Interfaz Visual

En la Figura 16 se puede ver la parte del DCD relativa a las RdP. El usuario de la interfaz dispone de un banco de redes, es decir de comportamientos preprogramados y debe seleccionar la RdP

que modele la tarea que desea ejecutar. Este entorno virtual le permitirá al usuario visualizar la red y el estado actual del sistema mediante la evolución de las marcas por la red. En la Figura 17 se muestra a modo de ejemplo una de las ventanas desplegadas con la RdP que ha seleccionado el usuario.

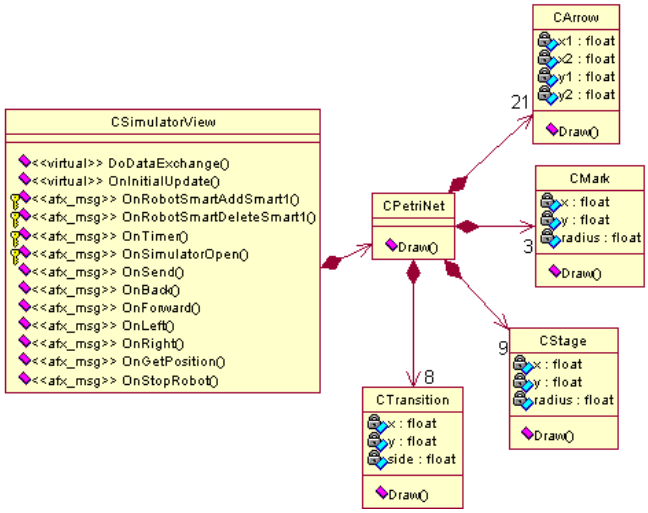


Figura 16: DCD con la RdP.

Uno de los principales objetivos de este proyecto es mover de manera coordinada un conjunto de robots SMART para realizar una tarea predefinida. Tal como se mencionara, los agentes SMART se sitúan dentro de una estructura metálica que tiene una cámara IP color, situada a 2 metros de altura (Figura 18).

La cámara toma la imagen actual del entorno real y la envía al ordenador principal donde está corriendo el software principal (kernel) que gestiona el conjunto del software y realiza el reconocimiento y la toma de decisiones de acuerdo a los resultados de visión artificial. Un diagrama UML se presenta en la Figura 19 que perfectamente resume la arquitectura software. Para más detalles remitimos al lector a De Clerq (2009).

5. IMPLEMENTACIÓN

Cada 150 mseg el kernel ordena la captura de imagen y su procesamiento. Se toman las decisiones de control de alto nivel, y se envían los comandos a los servos a través de una tarjeta Bluetooth.

Una de las posibles tareas que puede realizar el actual sistema SMART es evitar un obstáculo (fijo o móvil). La RdP asociada se presentó en la Figura 15. En la Figura 20 se muestra la interfaz de realidad virtual mientras el robot evita un obstáculo. Aquí también se representa la trayectoria generada por el robot. En la Figura 21 se muestra una secuencia de fotografías que muestran al robot evitando un obstáculo fijo.

En la Figura 22 se puede observar una RdP asociada a una tarea cooperativa que se describe a continuación. En la escena existen diversos obstáculos que impiden el paso del agente de tres patas. Un agente de cuatro patas es enviado en su ayuda para mover, con sus accionamientos lineales, los obstáculos que impiden el paso del robot de 3 patas. Una vez cumplida la misión el robot de 4 patas se retira y el de tres patas puede continuar su camino.

De la misma manera que la red presentada en la Figura 15, aquí también se modela el flujo de información gestionada por el kernel y que será quien organice las tareas. Tal como se observa de la figura, se tienen cuatro subredes:

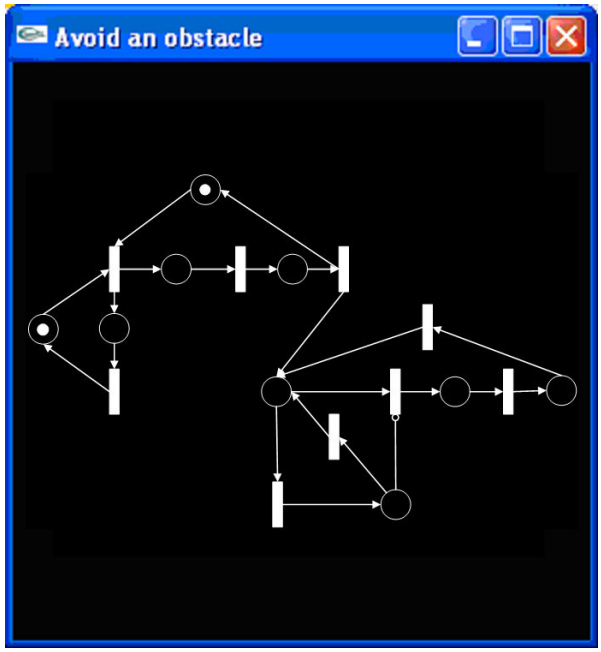


Figura 17: Ventana con RdP del Entorno Virtual.

Figura 18: Entorno real de trabajo.

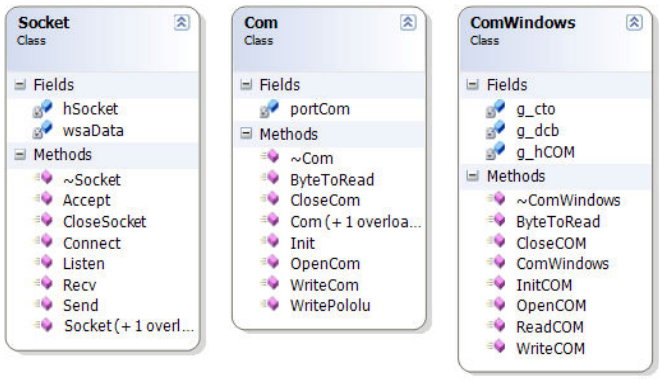


Figura 19: UML de la arquitectura de comunicación.

- Sub-RdP que modela el funcionamiento de la cámara y la captura de la imagen y su transmisión vía TCP/IP al software de procesamiento.

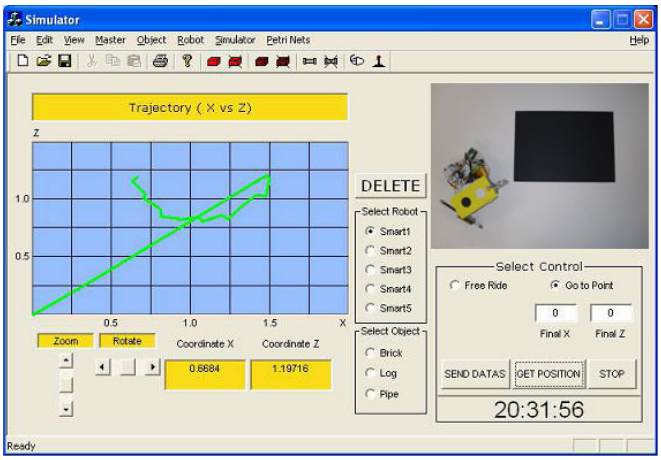


Figura 20: Trayectoria generada en la Interfaz de Realidad Virtual.

- Sub-RdP que modela la función que extrae las características de la imagen para concluir sobre posición y orientación de los agentes robots y los objetos presentes en la escena.
- Sub-RdP que modela la cooperación entre agentes. Esta subred está formada por el estado P_6 y la transición t_5. Esta transición es de tipo XOR-Split (Van der Aalst, 1996a, 1996b, 1997), y su misión es colocar una marca en uno y sólo uno de los estados vinculados. Claramente el mensaje enviado por el kernel en P_6 tiene un claro destinatario: robot ayudante o robot ayudado.
- Sub-RdP del agente ayudado que es idéntica al caso presentado en la Figura 15 cuando el agente se mueve libre de obstáculos.
- Sub-RdP del agente enviado en misión de ayuda. Aquí P_8 agente inicializado para la cooperación. T_7: envío de ordenes al agente (referencia de posición y orientación). P_9: ejecución de la orden. T_8: confirmación de orden ejecutada.

En la Figura 23 se muestra el aspecto que presenta la interfaz de realidad virtual cuando se realiza la tarea de cooperación antes descrita mientras que en la Figura 24 se muestra una secuencia de fotografías de los robots realizando la tarea cooperativa mencionada.

6. CONCLUSIONES

En este artículo se ha presentado el sistema robótica multi-agente SMART. Este sistema heterogéneo está formado por diferentes tipos de agentes software y/o hardware como los robots, la cámara IP, el software de procesamiento de la imagen, el bloque de control y el entorno virtual. Existen dos agentes robóticos diferentes: módulos de tres patas y módulos de cuatro patas con una articulación central en su cuerpo que aporta mayor variabilidad de movimiento.

La interfaz desarrollada ofrece amplias posibilidades de interacción con el entorno real a través de diferentes dispositivos hardware (teclado, ratón, joysticks, etc.). Por otro lado, el usuario recibe toda la información del entorno de manera gráfica (imagen real, trayectoria de cada agente) y en tiempo real para poder evaluar el desempeño de la tarea y actuar si fuera necesario.

El comportamiento de cada agente del sistema, sus interacciones y la tarea a desarrollar se han modelado con RdP. El usuario es quien decide la tarea a ejecutarse en el sistema a través de la selección de una RdP concreta prediseñada. El flujo de información es transmitido a los agentes involucrados en el proceso ya sea por comunicación TCP/IP o Bluethoot.

Los entornos virtuales son una herramienta de gran ayuda en la supervisión y control de sistemas robotizados. Sin embargo en un sistema multiagentes se busca que la intervención del operador humano sea sólo a alto nivel dejando la auto-organización a los agentes. Sin embargo sería muy favorable que el ser humano se integre "dentro" de la escena y sea un agente más utilizando para ello elementos hápticos.

Como trabajos futuros los autores están considerando la inclusión de un simulador de RdP dentro de la interfaz para que el usuario pueda realizar los diseños en el mismo entorno, validarlos con el simulador para luego ejecutarlos en el entorno virtual. El sistema SMART presentado en este articulo trabaja con una estructura de control centralizado, actualmente los autores están desarrollando una arquitectura de control distribuida lo que implica incluir inteligencia a bordo de cada robot.

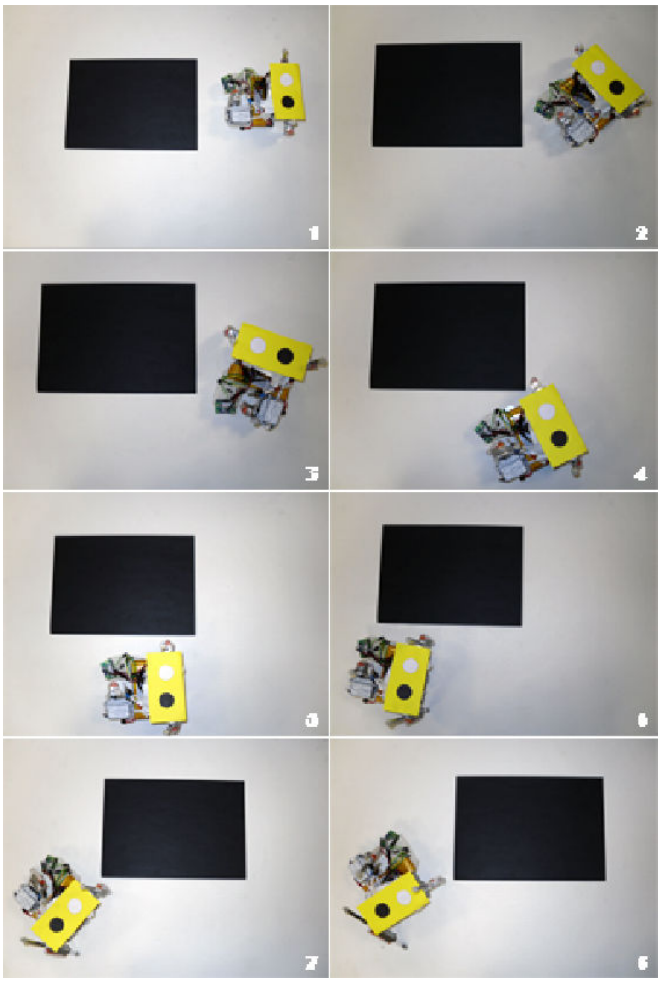


Figura 21: SMART de 3 patas evitando un obstáculo.

AGRADECIMIENTOS

Los autores agradecena la Comunidad de Madrid por la financiación concedida para el proyecto de investigación SMART (ROBOCITY2030 Ref. S-0505/DPI/0176) y al

Ministerio de Ciencia e Innovación (DPI2009-08778).

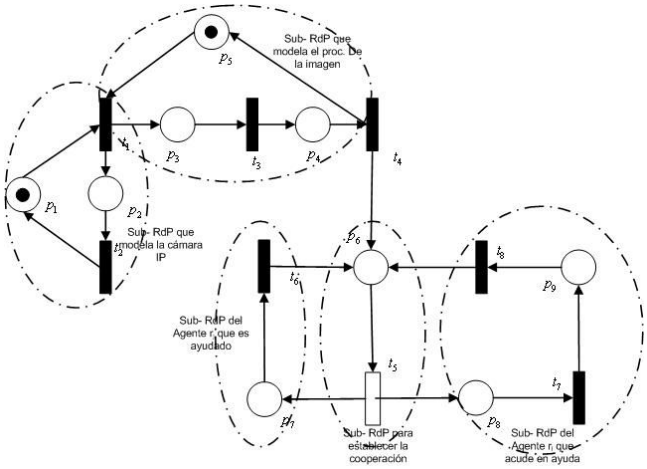


Figura 22: Ejemplo de RdP para modelar una tarea cooperativa con una transición XOR-Split [Kotb et al., 2007].

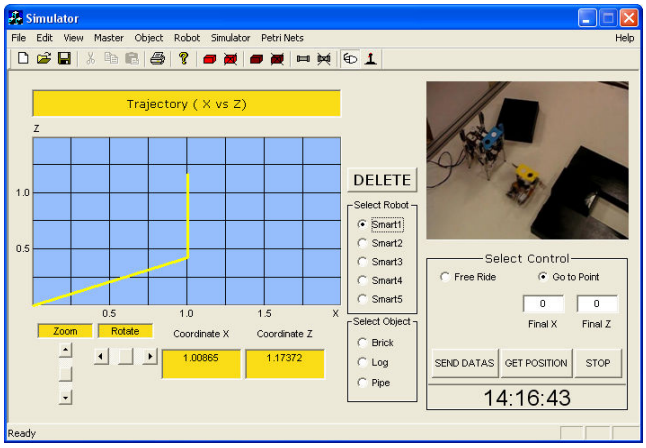


Figura 23: Aspecto de la Interfaz visual durante la tarea de cooperación.

REFERENCIAS

Collinot A. Drogoul A. y Benhamou P. (1996). Agent Oriented Design of a Soccer Robot Team. Proceeding of International Conference on Multi Agents Systems.

Corkill, D. y Lander, S. E. (2008). Modelling, Analysis and Execution of Multi-Robot Tasks using Petri Nets Agent Organizations. *Object Magazine*, **Vol. 8**, Nº 4, pp 41–47.

De Clerq S. Development of Communication Architecture for Multi-Agent Robotic Systems based on Petri-Net (2009). Tesis de Maestría. Universidad Politécnica de Madrid.

Deloach S., Matson E., Li Y.; (2002) Applying Agent Oriented Software Engineering to Cooperative Robotics. Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference.

Fiorino H y Tessier C. (1998) Agent Cooperation: a Petri Net Based Model. Procceding of International Conference on Multi Agent Systems . **Vol 3** pp. 425-426.

Frankiln D, Kahng A. y Lewis A. (1995) Distributed Sensing and Proving with Multiple Search Agents: Towards System-level landmine Detection Solutions. Proceedings of Detection Technologies for Mines and Minelike Targets, **Vol** 2496.

Franklin D. y Gresser T. Is it an Agent, or just a Program?: (1996) A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag.

Garcia C, De Clercq S., Saltaren R., Aracil R. Werner Verschelde W. (2010) Development of communication architecture for the multi-agent robotic system SMART. European Conference on the Use of Modern Information and Communication Technologies, pp 105-112

Hiraishi K. (2002). PN2: An Elementary Model for Design and Analysis of Multi-agent Systems. Proc. on Coordination Models and Languages, 5th International. N. Y. pp 220-235.

Hiraishi K. (2008) Performance Evaluation of Workflows Using Continuous Petri Nets with Interval Firing Speeds. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences archive. **Vol. E91-A**, Nº11. pp 3219-3228.

Hsieha F. (2009). Developing cooperation mechanism for multi-agent systems with Petri nets. *Engineering Applications of Artificial Intelligence* ***Vol. 2****, Nº 4-5. pp 616-627*

Kotb Y.T, Beauchemin S.S. and Barron J.L.. (2007). Petri Net-Based Cooperation In Multi-Agent Systems. 4th Canadian Conference on Computer and Robot Vision. **Vol**. **0**. pp 123-130.

Lund H. H. Rasmus Lock Larsen, and Esben Hallundbæk Østergaard. (2003) Distributed Control in Self-reconfigurable Robots. ISBN 978-3-540-00730-2. Springer. pp. 296-307.

Lee, Jin-Shyan. (2008) A Petri Net Design of Command Filters for Semiautonomous Mobile Sensor Networks. IEEE Transactions on Industrial Electronics, **Vol 55**, Nº 4.

Moldt D. and Wienberg F. (1997) Multi-Agent-Systems Based on Coloured Petri Nets. Proc. of the 18th International Conference on Application and Theory of Petri Nets table of contents. Publisher Springer-Verlag, pp 82 – 101.

Montano L., García F., Villaroel J. (2000). Using the Time Petri Net Formalism for Specification, Validation, and Code Generation in Robot-Control Applications, *The International Journal of Robotics Research,* **Vol. 19**, Nº 1, pp 59-76.

Palus, H. (1998). Representations of colour images in different colour spaces. In: Sangwine, S., and Horne, R. (eds.): The Colour Image Processing Handbook, Chapman and Hall, pp 67-90.

Plataniotis, K.N., Venetsanopoulos, A.N. (2000): Color Image Processing and Applications, Springer-Verlag.

Sims M., Corkill D., Lesser V. (2008). Automated organization design for multi-agent system . *Auton Agent Multi-Agent System*. **Vol. 16**, pp. 151–185. Springer.

Veloso M. y Nardi D. (2006) Special Issue on Multirobot Systems. Proceedings of the IEEE, **Vol. 94**, Nº. 7.

Van der Aalst W.M.P. (1996a). Structural Characterizations of Sound Workflow Nets. Computing Science Reports 96/23 Eindhoven University of Technology.

Van der Aalst W.M.P. (1996b). Three Good reasons for Using a Petri-net-based Workflow Management System. In Proc. of the International Working Conference on Information and Process Integration in Enterprises, pp. 179–201.

Van der Aalst W.M.P. (1997). Verification of Workflow Nets. Lecture notes on Application and Theory of Petri Nets. Springer Verlag (**1248)** 407–426.

Zhiwu Li and Shuwen Xu. (2006). On Modeling a Soccer Robot System Using Petri Nets. Proc. of the IEEE International Conference on Automation Science and Engineering.

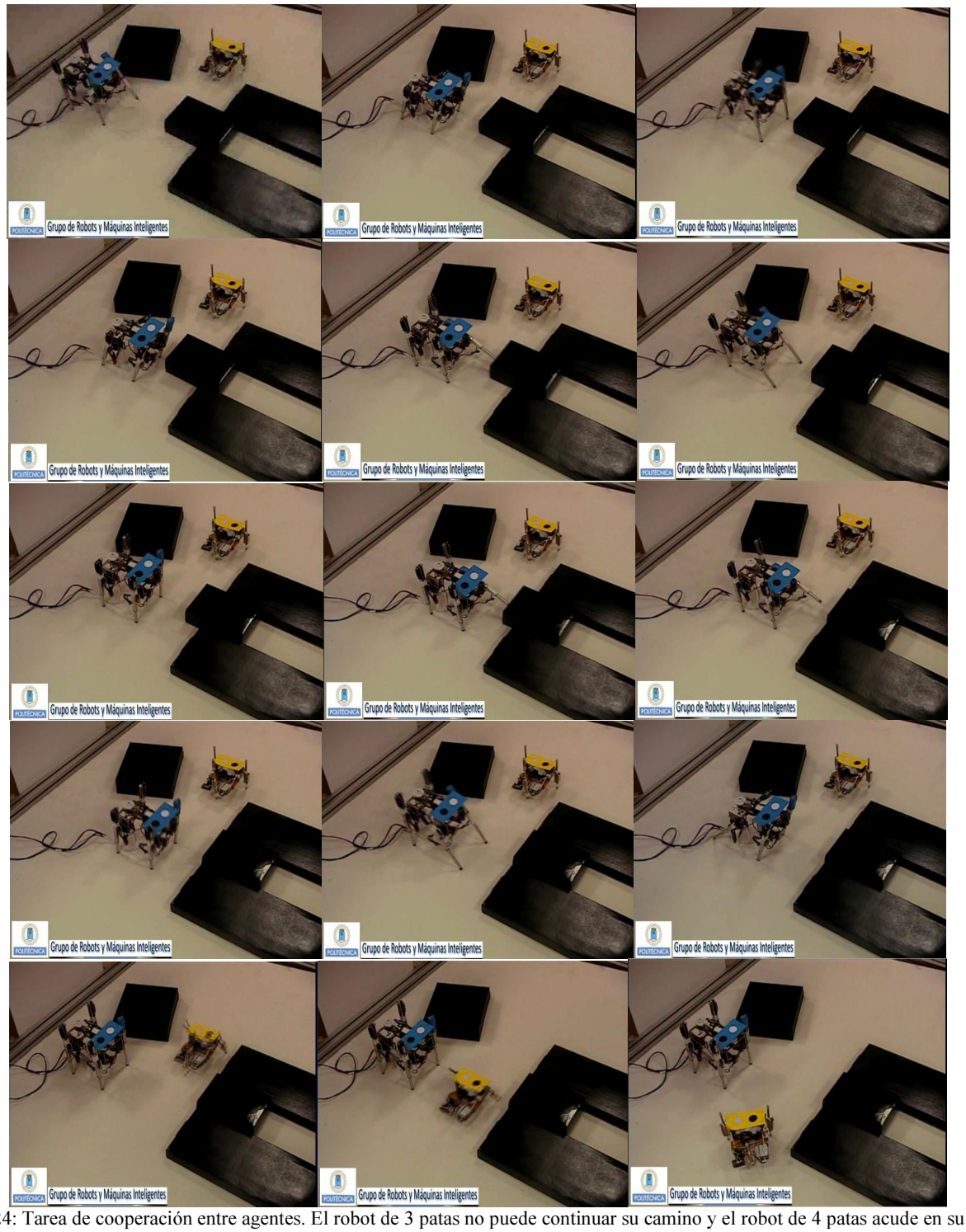


Figura 24: Tarea de cooperación entre agentes. El robot de 3 patas no puede continuar su camino y el robot de 4 patas acude en su ayuda.