XIII Workshop en Agentes Físicos

WAF 2012

Santiago de Compostela

3rd - 4th of September, 2012

Centro Singular de Investigación

en Tecnoloxías da Información (CITIUS)

University of Santiago de Compostela

Coordinators of the edition

Roberto Iglesias Rodríguez

Carlos Vázquez Regueiro

Víctor Álvarez Santos

Adrián Canedo Rodríguez

Coordinators of the edition

Roberto Iglesias Rodríguez

Centro Singular de Investigación en Tecnoloxías da Información (CITIUS).
Universidade de Santiago de Compostela (Spain).
roberto.iglesias.rodriguez@usc.es

Carlos Vázquez Regueiro

Departamento de Electrónica e Sistemas.
Universidade de A Coruña (Spain).
cvazquez@udc.es

Víctor Álvarez Santos

Centro Singular de Investigación en Tecnoloxías da Información (CITIUS).
Universidade de Santiago de Compostela (Spain).
victor.alvarez@usc.es

Adrián Canedo Rodríguez

Centro Singular de Investigación en Tecnoloxías da Información (CITIUS).
Universidade de Santiago de Compostela (Spain).
adrian.canedo@usc.es

ISBN: 978-84-940469-0-2
Legal deposit: C-1566-2012
Prints: Unidixital, s.l.

Soporte del robot humanoide Nao en el simulador 3D Gazebo

Jorge Bermejo, Jose María Cañas

***Abstract*—Los simuladores son una herramienta muy útil en robótica. Emulan el comportamiento de sensores y actuadores en mundos ficticios y permiten depurar algoritmos antes de llevarlos al robot real. Por otro lado, en los últimos años la investigación e interés en robots humanoides están creciendo significativamente. Este trabajo presenta el desarrollo para dar soporte al robot humanoide Nao dentro del simulador 3D Gazebo, incluyendo todas sus articulaciones, cuello y cámara. Este soporte se ha validado experimentalmente programando varias aplicaciones robóticas en el humanoide simulado utilizando la plataforma Jderobot.**

***Index Terms*—Simuladores, humanoide, middleware**

I. Introducción

UNA herramienta muy útil en robótica son los *simuladores*. Ofrecen un entorno virtual en el que emulan las observaciones de los sensores y los efectos de las órdenes a los actuadores. Sirven como evaluación, depuración y ajuste de los programas de control antes de llevarlos al robot real. Por ejemplo permiten con ello probar sin riesgo los algoritmos de navegación.

También son muy interesantes en la investigación con grupos numerosos de robots porque abaratan la inversión necesaria [2]. Otra ventaja es la facilidad de obtener ciertos datos, como la verdad absoluta sobre la posición del robot, más complejos o caros de obtener en el mundo real. Los simuladores permiten la repetitividad de los experimentos o prácticas que se realicen, lo que resulta muy difícil con los robots reales, ya que las condiciones siempre cambian en un entorno real, por ejemplo la iluminación, el estado de las baterías, etc.. Por último, en docencia son una opción económica para los centros que no cuentan con la posibilidad de adquirir y mantener hardware real.

A pesar de estas ventajas, el uso de simuladores no ha sido históricamente muy popular. Los primeros simuladores eran poco realistas y almacenaban mundos planos donde había obstáculos estáticos bidimensionales. Con los años han ido ganando en realismo, incorporando ruido en los sensores y actuadores, objetos dinámicos, y más detalles en los entornos. Hoy día se tienen simuladores tridimensionales que simulan robots con ruedas, con patas, brazos robotizados y sensores tan potentes como las cámaras, cada vez más frecuentes en el equipamiento de los robots. También se ha incluido en los simuladores más potentes la capacidad de representar un *conjunto* de robots operando en el mismo escenario, simultáneamente.

Universidad Rey Juan Carlos.

E-mails: jorge_bermejo0817@hotmail.com, jmplaza@gsyc.es

Muchos fabricantes de robots incluyen un simulador para sus robots, por ejemplo el `EyeSim` para el robot EyeBot o `MobileSim` para los robots de MobileRobotics como el Pioneer. Igualmente hay simuladores generalistas que incluyen soporte para los robots de varios fabricantes. Por ejemplo el simulador `Webots` desarrollado por la empresa Cyberbotics incluye soporte para los robots LEGO, Kephera y Koala entre otros. También Microsoft Robotics Developer Studio 4[1] [5] incluye soporte para varias plataformas.

Son relevantes también varios simuladores creados dentro de la comunidad investigadora en robótica. Uno de los pioneros fue el `SRIsim`[2] de Kurt Konolige (Standford Research Institute), que permitía simular un único robot en un mundo bidimensional estático. Un ejemplo más reciente es el simulador `SimRobot` [7], que se ha usado en la liga SPL dentro de la competición RoboCup-soccer. También es destacable el simulador `USARSim` (Unified System for Automation and Robot Simulation) [1], que se emplea en la competición RoboCup-Rescue. Otro simulador interesante y reciente es MORSE[3] [3] dirigido a entornos educativos. Tal vez los más difundidos actualmente sean dos herramientas de software libre: `Stage` [4], simulador bidimensional multirrobot, y `Gazebo` [6], simulador tridimensional.

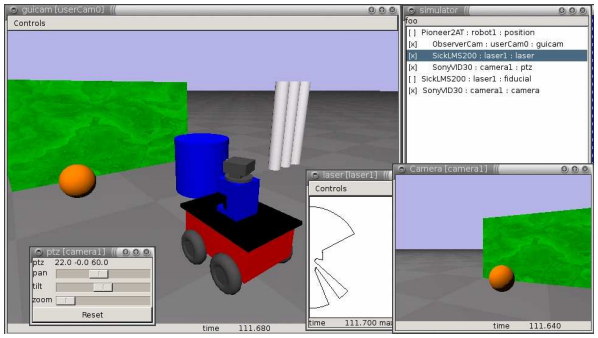

Fig. 1. Escena 3D simulada en Gazebo

En particular Gazebo[4] es un simulador 3D que ofrece un entorno rico para desarrollar y probar de forma rápida sistemas multirrobot y que simula incluso cámaras de manera realista (Figura 1). Nació dentro del proyecto Player/Stage/Gazebo [10] y recientemente WillowGarage ha centralizado su

[1] `http://www.microsoft.com/robotics`
[2] `http://www.ai.sri.com/~konolige/saphira`
[3] `http://www.openrobots.org/wiki/morse`
[4] `http://gazebosim.org`

desarrollo, como herramienta auxiliar para ROS [5].

Una campo de interés creciente en robótica son los humanoides. Prototipos como el Asimo de Honda o el HOAP-3 de Fujitsu sirven de base para numerosas investigaciones, muchas de ellas orientadas a replicar la inteligencia y maniobrabilidad humana. La apariencia similar a las personas facilita su aceptación e interacción natural con humanos como asistente personal, en el ámbito de la robótica de servicios. La funcionalidad conseguida en los humanoides Asimo progresa significativamente en los últimos años, permitiéndole correr, subir escaleras, empujar carritos, servir bebidas, etc.

Dentro de este campo el robot humanoide Nao, fabricado por
Aldebaran Robotics, ha irrumpido con fuerza recientemente. Desde 2008 es la plataforma oficial de la liga estándard SPL en la competición robótica internacional RoboCup. Debido a su reducido precio (comparativamente con otros humanoides), su vistosidad y sus buenas características el número de usuarios de este robot está creciendo en la comunidad robótica, tanto para su uso en investigación como en docencia. El simulador de referencia para él es Webots, de la empresa CyberBotics, que es software de código cerrado y comercial.

El objetivo central de este trabajo es desarrollar el soporte para este robot dentro del simulador 3D de software libre Gazebo.

En la segunda sección repasamos la infraestructura de partida de este trabajo: el simulador Gazebo, el robot Nao y el entorno Jderobot. En la tercera describimos la construcción y ensamblado del humanoide simulado desde sus piezas constituyentes. En la cuarta explicamos el componente software que ofrece acceso estandarizado a los sensores y actuadores del humanoide simulado. Seguidamente describimos varias aplicaciones realizadas para validar experimentalmente el sistema y finalizamos con las conclusiones alcanzadas.

II. Infraestructura

Una vez dibujado el marco general del trabajo damos ahora más contexto técnico de los tres componentes empleados y que sirven como punto de partida: el simulador Gazebo, el robot Nao y el entorno de aplicaciones robóticas JdeRobot.

A. Simulador Gazebo

Gazebo está diseñado para reproducir con precisión los entornos dinámicos que un robot puede encontrar. Simula sensores y actuadores integrados en un robot y en un determinado escenario. Todos los objetos de ese escenario tienen masa, velocidad, fricción, y otros numerosos atributos que les permiten comportarse de forma realista.

Gazebo permite crear nuestros propios robots simulados fácilmente. Para ello el concepto central en el software de Gazebo es el *modelo*. Un *modelo* es cualquier objeto que mantiene una representación física. Esto abarca desde bloques de geometría simple hasta robots complejos. Los modelos están formados por al menos un cuerpo rígido, cero o más articulaciones y sensores. Para cada uno de ellos Gazebo permite asociar un interfaz básico de programación para acceder a él, configurarle parámetros, enviarle órdenes o leer sus medidas. La Figura 2 ilustra la arquitectura software del simulador, con los ingredientes del modelo.

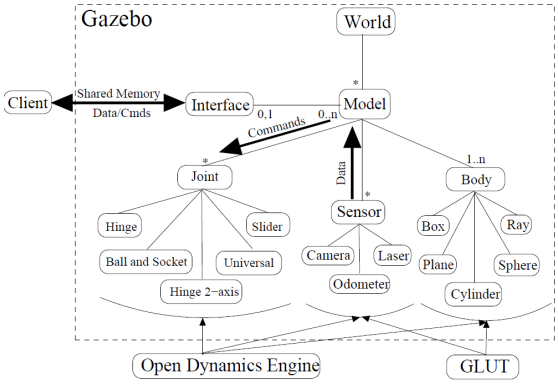

Fig. 2. Arquitectura software de Gazebo

Primero, los cuerpos representan los bloques básicos de construcción de un modelo. Su representación física es en forma de figuras geométricas básicas como cajas, esferas, cilindros, planos y líneas. A cada cuerpo se le asigna una masa, la fricción, el factor de rebote, y renderizado en propiedades como el color, textura, transparencia, etc.

Segundo, las articulaciones proporcionan el mecanismo para conectar cuerpos entre sí con el fin de formar relaciones cinemáticas y dinámicas. Además de la conexión de dos cuerpos, estas articulaciones pueden actuar como motores. Para ello, a la superficie de estas articulaciones se le pueden aplicar fuerzas, tanto angulares como lineales, y la fricción entre los cuerpos conectados produce el movimiento, la locomoción e interacción con el entorno. Con ellas se pueden simular motores asociados a las ruedas, motores en brazos robotizados, cuellos mecánicos, etc.

Y tercero, un sensor en Gazebo es un dispositivo abstracto que carece de una representación física. Sólo sirven cuando se incorporan a un modelo. Esta característica permite la reutilización de los sensores en numerosos modelos reduciendo así el código y la confusión. Los sensores permanecen separados de la simulación dinámica, ya que sólo recogen datos, o los emiten si se trata de un sensor activo. Se pueden construir sensores nuevos, pero existe ya un conjunto bastante rico de sensores soportados: láseres (laser Sick), sensores de ultrasonido, de contacto, etc. Uno de sensores más interesantes que simula Gazebo son las cámaras, por ejemplo la SonyVID30.

El terreno y los edificios representan modelos inmóviles mientras que los robots y otros objetos son dinámicos. Los robots son estructuras dinámicas compuestas por cuerpos rígidos conectados a través de articulaciones y algún sensor. En Gazebo hay varios tipos de robots ya creados. Por ejemplo el Pioneer2DX, Pioneer2AT, SegwayRMP, etc. Todos los sensores, articulaciones y modelos existentes se pueden utilizar para la composición de nuevos modelos de robots.

[5]Recientemente han publicado la versión 1.0 de Gazebo

En cuanto a la simulación de la física utiliza ODE (Open Dynamics Engine)[6], diseñado para simular la dinámica y la cinemática asociada a los cuerpos rígidos articulados. Este motor físico incluye muchas características tales como numerosas articulaciones, detección de colisiones, la masa y las funciones de rotación, y muchas geometrías. Suele requerir cierta potencia de cómputo.

Gazebo utiliza estas características proporcionando una capa de abstracción situada entre ODE y los modelos. Dicha capa permite la fácil creación de normales y objetos abstractos tales como los rayos laser, mientras se mantiene toda la funcionalidad proporcionada por ODE.

Un buen simulador suele proporcionar algún tipo de interfaz gráfico que permita al usuario humano saber lo que está pasando en el mundo simulado, cambiar la posición de determinados objetos, etc. Para realizar esta labor Gazebo ofrece un visor con una cámara virtual desde la cual el usuario puede ver la escena simulada, observar el comportamiento de los objetos, configurarlos, visualizar el valor de determinados sensores, etc. En la Figura 1 se muestra este visor. Esa cámara virtual se puede también mover y orientar a voluntad con el ratón.

Para agilizar el coste computacional de este interfaz gráfico Gazebo usa OpenGL y GLUT (OpenGL Utility Toolkit) como herramientas de visualización. OpenGL es una biblioteca estándar para la creación de aplicaciones 2D y 3D interactivas. Es independiente de la plataforma, altamente escalable, estable y está en constante evolución. Más importante aún, muchas características en OpenGL se han implementado en hardware de la tarjeta gráfica lo que libera a la CPU para otras tareas. Por su parte GLUT es un sistema de ventanas simple con un conjunto de herramientas independientes para las aplicaciones OpenGL. Escenas renderizadas utilizando OpenGL se muestran en las ventanas creadas por GLUT. Esta herramienta también proporciona mecanismos para la interacción del usuario con Gazebo a través de teclados y ratones.

B. Robot Nao

El robot Nao (Figura 3) es un robot humanoide que mide 58 centímetros de alto y pesa unos 4,3 kilogramos (versión 3,4). En su cabeza incorpora un microprocesador AMD Geode a 500 Mhz con sistema operativo Linux. En su parte sensorial dispone de varios detectores de presión en los pies (FSR), un sensor inercial, 4 sónares en el pecho que permiten estimar la distancia a los obstáculos que hay en su entorno en rango de detección de 15 a 70 centímetros. Además incorpora 2 cámaras con distintas zonas de visión en su cabeza, que proporcionan imágenes de una resolución de 640x480 pixeles a 30 fps. y permiten al robot Nao capturar fotos, secuencias de video, reconocer objetos de colores, detectar y reconocer caras, etc.

Incluye también de 2 altavoces y varios micrófonos e integra un sistema de reconocimiento de voz que localiza de dónde viene el sonido de modo que pueda girar su cabeza hacia el origen de ese sonido y un sistema de reproducción de texto a voz para que pueda interactuar con su entorno mediante la pronunciación verbal de cualquier texto que le haya sido enviado.

[6]`http://www.ode.org`

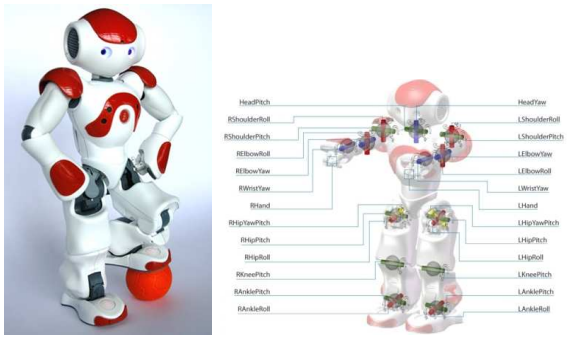

Fig. 3. Robot humanoide Nao

El robot Nao es totalmente autónomo y capaz de realizar una gran variedad de movimientos como caminar, sentarse, levantarse, bailar, evitar obstáculos, coger objetos, etc. Para realizar todos estos movimientos dispone de 25 grados de libertad, con sendos motores en cada una de sus articulaciones (Figura 3).

En cuanto a su software, el fabricante proporciona un entorno para acceder de modo sencillo al hardware y a varias funcionalidades de alto nivel que resultan útiles a la hora de programar aplicaciones con el Nao. Este entorno de programación se llama `NaoQi` [8] y permite el desarrollo de aplicaciones en C++ y Python. Naoqi es una arquitectura software distribuida en la que varios módulos software pueden comunicarse entre sí. La funcionalidad se encapsula en módulos y hay que comunicarse con ciertos módulos para acceder a los sensores y actuadores. Por ejemplo el módulo `ALMotion` proporciona varios métodos para mover los motores del robot, tanto articulación por articulación como conjuntos de actuadores en cada extremidad (cabeza, piernas, brazos) y de manera coordinada, con varios modos de movimiento. El módulo `ALVideoDevice` permite adquirir las imágenes de la cámara.

Naoqi está integrado por tres componentes básicos: los *brokers*, los módulos y los *proxies*. Cada *broker* se ejecuta independientemente y se asocia a una IP y un puerto. Una aplicación robótica completa puede estar compuesta de varios *brokers* que se comunican entre sí. Pueden funcionar a bordo del robot (en ese caso se genera con un compilador cruzado) o en un ordenador externo, lo cual resulta útil si se quiere repartir el coste computacional de la aplicación considerando las limitaciones del computador a bordo. La funcionalidad de cada *broker* se materializa con uno o varios módulos que también se comunican entre sí, o con otros módulos remotos a través de *proxies*. Esta arquitectura software es utilizada tanto para la comunicación con el robot nao real como con el simulado en Webots indistintamente.

C. Entorno de aplicaciones robóticas Jderobot

La plataforma software Jderobot[7] ha sido creada por el Grupo de Robótica de la URJC para la programación de aplicaciones con robots y visión artificial. Se utiliza tanto en investigación, como docencia y aplicaciones comerciales. Jderobot (versión 4.3) está desarrollada en C y C++ y proporciona un entorno de programación donde la aplicación se compone de distintos hilos de ejecución asíncronos llamados *esquemas*. Cada esquema es un *plugin* que se carga automáticamente en la aplicación y que realiza una funcionalidad específica que puede ser reutilizada.

Se utilizan *interfaces estándard* para la comunicación entre los diferentes esquemas. Los esquemas que proporcionan acceso software a los sensores y actuadores se denominan *drivers* y se encargan de dialogar directamente con los dispositivos hardware concretos. Jderobot simplifica el acceso desde programa a estos dispositivos. El modo de leer la medida de un sensor es simplemente leer una variable local del interfaz estándard que ofrece un *driver*. La manera de lanzar órdenes a los motores es tan simple como escribir en otra variable local del interfaz estándard que ofrece otro *driver*.

La aplicación robótica en Jderobot se organiza como un conjunto de esquemas concurrentes. Por un lado esquemas perceptivos, que realizan algún tipo de procesamiento de datos para proporcionar información sobre el robot y el mundo en el que opera. Por otro los esquemas de actuación, que toman decisiones para alcanzar una meta, como lanzar órdenes de control a los motores o lanzar nuevos esquemas, ya que pueden combinarse formando jerarquías.

Jderobot ofrece una plantilla de esquema para materializar algoritmos de control, con ejecución iterativa del código de control a cierto ritmo configurable. Además, está diseñado para que el interfaz gráfico de cada esquema corra en paralelo en otra hebra separada de la de procesamiento iterativo.

Varias razones por las que se ha elegido Jderobot son que ofrece un entorno sencillo para la programación de aplicaciones de control y que resuelve problemas relacionados con la sincronización de los procesos y la adquisición de datos sensoriales. Otro motivo es la facilidad con la que se pueden reutilizar drivers y aplicaciones ya creadas para esta plataforma de desarrollo. Además, está preparada para aprovechar la potencia de los procesadores multinúcleo, evitando por ejemplo que la interfaz de usuario ralentice el procesamiento de la información.

III. Construcción del humanoide en Gazebo

El primer paso para dar soporte al humanoide Nao en Gazebo es construir su modelo dentro del simulador a partir de los objetos básicos: cuerpos, articulaciones y sensores. Además de ensamblar todas las piezas, el simulador nos ofrece un interfaz básico de programación para el acceso a cada uno de sus objetos constituyentes.

Gazebo ofrece mecanismos para dar soporte a nuevos robots, construyéndolos a partir de objetos compuestos por bloques básicos ya definidos, como un hexaedro, una esfera, un cilindro, etc. (Figura 4). A estos bloques geométricos se les puede dar el tamaño y la masa necesarios para el modelo concreto. Estos cuerpos se unen entre sí mediante articulaciones, con las cuales podemos darle una o varias direcciones de giro. Con ello se otorga a nuestro modelo simulado diferentes grados de libertad. De todas las partes que componen nuestro modelo de robot Nao detallaremos a continuación el brazo y la cabeza con cámara como muestra representativa.

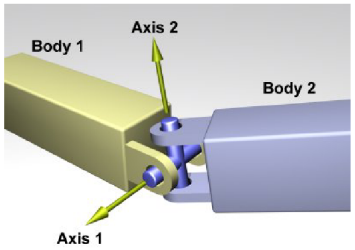

Fig. 4. Dos bloques conectados por dos articulaciones

En primer lugar, el brazo del humanoide en Gazebo se compone de la parte del húmero, la parte del cúbito y radio y la mano. La parte del húmero la hemos creado construyendo un hexaedro rectangular con unas dimesiones de 90x50x50 mm. y con una masa de 163 gramos, la parte del cúbito y radio también la creamos construyendo un hexaedro rectangular, pero esta vez con unas dimensiones de 135x50x50 mm. y una masa de 87 gramos (Figura 5, izquierda).

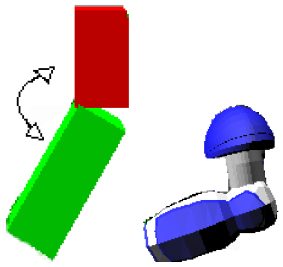

Fig. 5. Brazo como bloques conectados (izda.) y con piel (dcha.)

Estos dos cuerpos se unen mediante la articulación del codo. Esta articulación proporciona el grado de libertad normal del codo, que permite doblar y estirar el brazo. Además, debe proporcionar otro grado de libertad adicional disponible en el Nao real, el que permite girar el antebrazo con respecto del húmero. Para conseguir estos dos grados de libertad en el codo utilizamos en el modelo dos articulaciones de tipo bisagra. Gazebo no permite unir dos cuerpos con más de una articulación al mismo tiempo, con lo que introducimos entre nuestro brazo y nuestro antebrazo otro cuerpo. Es la única forma de conseguir que al girar el antebrazo gire con él la otra articulación del codo que nos permite estirar y doblar el brazo. Este nuevo cuerpo creado tiene un tamaño lo suficientemente pequeño para que no se aprecie de forma visual y una masa

[7] http://jderobot.org

también lo suficientemente pequeña para que sea despreciable en el cómputo global de la masa de nuestro robot simulado.

En código esto se hace invocando a funciones del API de programación para Gazebo que permiten registrar un modelo (`GZ_REGISTER_PLUGIN`), crear objetos dentro de ese modelo y configurarlos. Por ejemplo en la Figura 6 se pueden ver las instrucciones para definir el húmero del brazo izquierdo, su posición de anclaje en el bloque del torso, su masa, tamaño, color, etc.

```
// Create the left  arm
 this->leftArm = new Body(this->world);
 anchor = this->torso->GetPosition();
 anchor.x += torsoSize.x*0.5-ArmSize.x*0.5;
 anchor.y += torsoSize.y*0.5+ArmSize.y*0.5 + 0.01;
 this->leftArm->SetPosition(anchor);
 this->leftArm->SetAngularVel(GzVectorSet(0,0,0));
 this->leftArm->SetLinearVel(GzVectorSet(0,0,0));
 this->leftArm->SetGravityMode(true);
 this->AddBody(this->leftArm, false);

 geom = new BoxGeom(this->leftArm,this->modelSpaceId,
                    ArmSize.x,ArmSize.y,ArmSize.z);
 geom->SetColor( GzColor(1, 0, 0) );
 geom->SetMass(0.163);
 geom->SetRelativePosition(GzVectorSet(0.0,0.0,
                           ArmSize.z * 0.5) );
```

Fig. 6. Instrucciones para la creación del brazo izquierdo

Una vez que tenemos el brazo construido en Gazebo mediante figuras geométricas básicas el siguiente paso es diseñar la piel del brazo (Figura 5, derecha), de forma que su apariencia sea más realista, visualmente más parecida al robot Nao real, manteniendo la funcionalidad. Para ello hemos utilizado Blender. Blender es una aplicación multiplataforma, libre y gratuita de gráficos 3D que permite modelar, dar textura, animar, renderizar, simular efectos de partículas, etc.. Además, también nos ofrece herramientas de simulación avanzada tales como dinámicas de cuerpos rígidos, de cuerpos suaves y de fluidos, herramientas de animación de personajes, un sistema de simulación física avanzado, y un sistema de composición de materiales basado en nodos.

En código la adición de pieles se materializa cargando el fichero con la textura concreta con la función `SearchFilename`, definiendo su escala con `SetSkinScale`, posicionándola y orientándola en el modelo con `SetSkinPose`. Los argumentos para estas funciones se recogen del fichero de configuración del mundo a simular por Gazebo, que incluye secciones para cada uno de los modelos que se usan. Por ejemplo en la Figura 7 se aprecian las líneas relativas a la configuración de la piel del húmero del brazo izquierdo.

```
<skinFileLeftArm>left_Arm.3ds</skinFileLeftArm>
<skinScaleLeftArm>0.15 0.15 0.11</skinScaleLeftArm>
<skinXyzLeftArm>0.015 0.015 0</skinXyzLeftArm>
<skinRpyLeftArm>90 0 270</skinRpyLeftArm>
```

Fig. 7. Configuración de la piel del brazo izquierdo

A la vez que recreamos la parte mecánica del brazo, Gazebo permite añadir un interfaz básico de programación a las articulaciones con las funciones `gz_joint_alloc` y `gz_joint_create`. Gracias a este interfaz se puede leer su posición cuando se pidan desde una aplicación externa y dar valores desde una aplicación externa para que las articulaciones se muevan hasta un determinado ángulo.

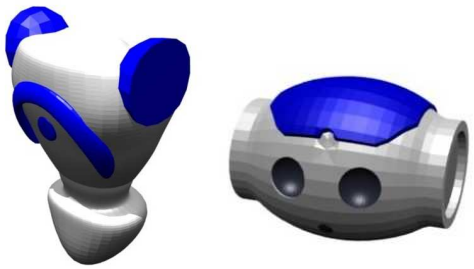

Fig. 8. Tronco y cabeza del humanoide

En segundo lugar, la cabeza del humanoide está construida mediante una esfera, con su piel, e incorpora un sensor cámara. Para ello añadimos un sensor a nuestro modelo. Este sensor lo hemos creado basándonos en la cámara SonyVID30, sensor que ya está soportado y disponible en Gazebo. Esta cámara por defecto proporciona imágenes de 320x240 píxeles, con un campo de visión horizontal (HFOV) de 60 grados y un refresco de 10 fotogramas por segundo. Una vez incorporada mecánicamente, se crea el interfaz básico de programación que permite acceder a los datos de la cámara, y poder ver así lo que el robot Nao en Gazebo está observando en cada momento.

Mediante estos pasos vamos diseñando y construyendo todas las partes del humanoide Nao simulado: los brazos, las piernas, el tronco y la cabeza (Figura 8). En la Figura 9, a la izquierda se puede apreciar al humanoide completo sin texturas, con los bloques mecánicos y a la derecha la apariencia del mismo robot cuando se le han incorporado pieles, con un aspecto muy parecido al del Nao real. Además del ensamblado mecánico de los bloques, este modelo incorpora todo un conjunto de interfaces básicos de programación para cada uno de sus bloques: los actuadores en sus articulaciones y los sensores, incluyendo la cámara.

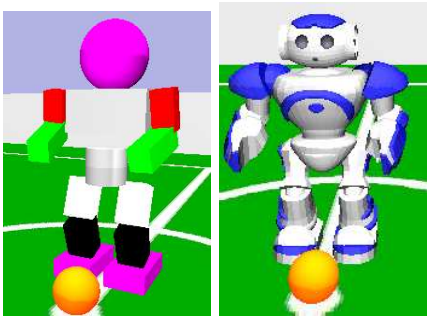

Fig. 9. Nao con los bloques básicos sin piel (izda.) y con piel (dcha.)

Interfaces libgazebo	Interfaces Jderobot
R_Shoulder	Right_arm
R_Elbow	Right_arm
R_Elbow_Yaw	Right_arm
L_Shoulder	Left_arm
L_Elbow	Left_arm
L_Elbow_Yaw	Left_arm
Waist	Left_leg & Right_leg
R_Hip	Right_leg
R_Knee	Right_leg
R_Ankle	Right_leg
L_Hip	Left_leg
L_Knee	Left_leg
L_Ankle	Left_leg
Camera	VarcolorA
Head	Ptmotors & Ptencoders

Fig. 10. Interfaces básicos de Gazebo (izda.) y estándard de Jderobot (dcha.)

IV. Driver software para Nao en Jderobot

Una vez construido el modelo de humanoide en Gazebo como hemos señalado en la sección anterior, ya se pueden programar aplicaciones que lo usen, que empleen sus sensores y ordenen comandos a sus actuadores. Para ello el simulador ofrece los interfaces básicos que hemos mencionado en forma de funciones de biblioteca, de la biblioteca *libgazebo*, accediendo por nombre de cada uno de los elementos del robot. Son los que aparecen en la columna izquierda de la Figura IV.

Como hemos mencionado, dentro de la plataforma Jderobot las aplicaciones robóticas utilizan los interfaces estándard que ofrecen los *drivers* para acceder a los sensores y actuadores de cada robot. Uno de los drivers existentes es `NaoBody` [9] que ofrece ya ciertos interfaces estándard para el Nao real. Son los que aparecen en la columna derecha de la figura IV.

Para poder desarrollar aplicaciones en Jderobot que utilicen el humanoide simulado en Gazebo hemos creado el *driver* NaoGazebo. Este *driver* ofrece exactamente las mismas interfaces estándard que el *driver* para el Nao real, `NaoBody`, envolviendo las interfaces básicas que da *libgazebo* para acceder al humanoide simulado. De este modo las aplicaciones para el humanoide funcionan con independencia de si se conectan al robot real o al simulado, sin necesidad de realizar cambios en su código, simplemente configurándolas para que la fuente de sus interfaces sea uno u otro *driver*. Otra ventaja de ofrecer interfaces estandard es la alta compatibilidad con otras aplicaciones y herramientas ya creadas para Jderobot.

El diseño de este *driver* se muestra en la Figura 11. Por un lado establece una comunicación con el simulador usando los interfaces básicos de `libgazebo`, y por otro se comunica con la aplicación Jderobot usando los interfaces estándard.

La comunicación entre la aplicación en Jderobot y el *driver* NaoGazebo se realiza mediante una serie de variables compartidas que conforman los interfaces estándard. En nuestro caso son la variable `VarcolorA` para los datos de la cámara y los arrays *left_arm*, *right_arm*, *left_leg* y *right_leg* para las articulaciones del brazo izquierdo, derecho, pierna

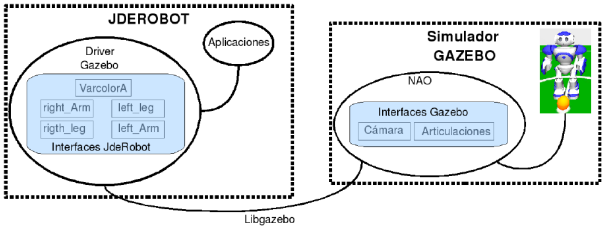

Fig. 11. Diseño del driver NaoGazebo, entre el simulador y la aplicación

izquierda y derecha respectivamente. Siguiendo el mismo convenio que el utilizado en el *driver* `NaoBody` hemos divido en esas cuatro partes la interfaz de las articulaciones.

La comunicación entre el *driver* y el simulador se realiza gracias a la librería `libgazebo`. Para cada interfaz estándard de Jderobot ofrecido por `NaoGazebo`, este *driver* busca la correspondencia con el interfaz básico correspondiente en el simulador, y realiza las conversiones oportunas, tanto en el sentido de datos sensoriales como de órdenes a los actuadores. Por ejemplo en la Figura 12 se puede ver cómo las órdenes desde Jderobot a las articulaciones del hombro (`left_shoulder`) y codo (`left_elbow`) (que forman parte del interfaz *left_arm*) se traducen en órdenes al array de articulaciones que mantiene Gazebo (`joints`). En concreto a las articulaciones 0,1 y 14, que corresponden con los interfaces Gazebo de `L_shoulder` y `L_elbow`, haciendo la adaptación entre grados y radianes y cambios de signo para respetar convenios.

```
joints[0].cmd_angle =-(((left_shoulder.pitch-90)*PI)/180);
joints[0].cmd_angle2 = (left_shoulder.roll*PI)/180;
joints[1].cmd_angle = (left_elbow.roll*PI)/180;
joints[14].cmd_angle = -(left_elbow.yaw*PI)/180;
```

Fig. 12. Correspondencia entre interfaces Jderobot y Gazebo para el brazo izquierdo

Se ha implementado como un único hilo que ejecuta iteraciones periódicas. En cada iteración recorre todas las interfaces Gazebo preguntando en qué posición se encuentran las articulaciones en el simulador, actualizando su copia local de las variables sensoriales para que puedan ser leídas por la aplicación en Jderobot. Además, en ese mismo hilo manda órdenes a los actuadores de las articulaciones del modelo simulado dándole los datos que la aplicación ha escrito en las variables de actuación. De este modo hemos materializado el soporte a las aplicaciones de Jderobot para que puedan manejar y controlar cualquier movimiento del robot Humaoide Nao simulado en Gazebo.

V. Experimentos

Para validar la infraestructura creada hemos probado el teleoperador existente para manejar el Nao real, llamado `NaoOperator`, y probado con él que la recepción de los datos de la cámara y de la odometría de las articulaciones del Nao simulado funciona correctamente. En la Figura 13 se

aprecia cómo la imagen de la cámara se recibe correctamente y la odometría de los motores de las articulaciones se refleja en los valores de los diales verticales. También hemos probado a comandar órdenes de movimiento al cuello del humanoide y cada una de sus articulaciones simuladas.

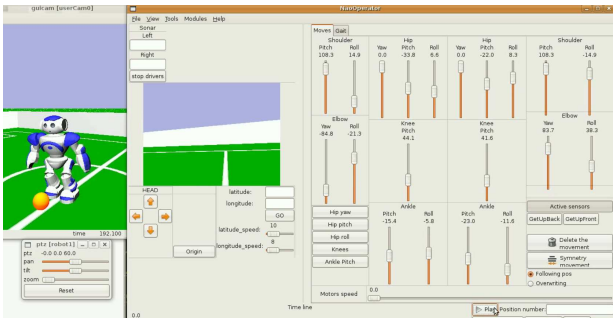

Fig. 13. Teleoperador del Nao conectado al humanoide simulado

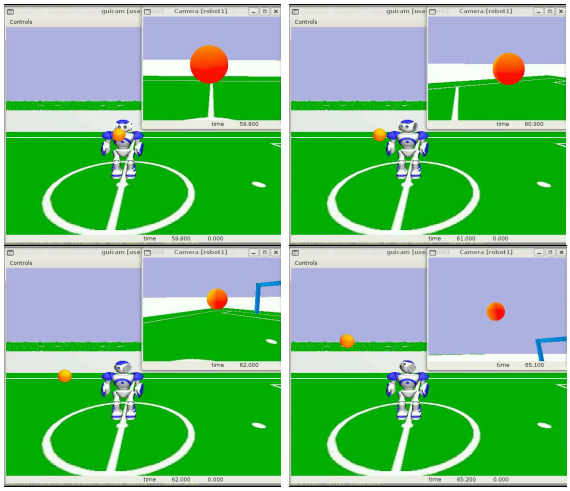

Fig. 14. Secuencia del comportamiento sigue-pelota

Además, hemos desarrollado varios comportamientos autónomos con el humanoide Nao en Gazebo que prueban el soporte de los distintos actuadores y sensores. En particular una aplicación que hace al humanoide bailar, otra que le hace caminar y que de una patada a una pelota y otra que le hace seguir con la mirada el movimiento de esa misma pelota. Con el baile y la caminata, que incluyen piernas y brazos, se muestran dos ejemplos de control en lazo abierto donde se van ordenando distintas posiciones en secuencia. En esta página web[8] se puede apreciar un video con la caminata que termina chutando la pelota.

Con el seguimiento visual de la mirada se ilustra un control en lazo cerrado que coordina cámara y cuello del humanoide para un comportamiento autónomo concreto. En la Figura 14 se muestran cuatro instantes del robot moviendo su cabeza para perseguir con la mirada el movimiento de la pelota naranja.

En todos los experimentos, tanto teleoperados como autónomos, hemos utilizado la versión 0.7 de Gazebo y la versión 4.3 de Jderobot.

VI. Conclusiones

El robot humanoide Nao es la plataforma oficial de la liga estandard SPL de la RoboCup desde 2008 y tiene un número creciente de centros que lo usan en su investigación y/o su docencia. El simulador de referencia para el humanoide Nao es Webots, un producto comercial de la empresa Cyberbotics. Los simuladores son una herramienta muy útil en robótica que permiten depurar y madurar algoritmos antes de llevarlos al robot real.

Uno de los simuladores más usados en la comunidad robótica es Gazebo, que se ha incorporado como herramienta auxiliar para ROS, de Willow Garage. En este trabajo hemos desarrollado una extensión del simulador Gazebo para que simule el robot Nao y se puedan programar aplicaciones robóticas con él, usando la plataforma Jderobot. La extensión consiste en dos partes. Primero, la creación de un modelo en Gazebo del humanoide, con sus piezas mecánicas de tamaño y peso adecuados, sus sensores, articulaciones y apariencia realista. Segundo, se ha creado un *driver*, llamado `NaoGazebo`, que establece las correspondencias adecuadas entre los interfaces de bajo nivel que ofrece el simulador y los esperados por las aplicaciones que manejan el robot Nao real en Jderobot. Las aplicaciones en Jderobot usan este *driver* para acceder al humanoide simulado igual que accederían al humanoide real.

La funcionalidad ha sido validada experimentalmente desarrollando varios comportamientos en el humanoide simulado: un baile, dar una patada a una pelota y seguir con la mirada el movimiento de esa pelota en el entorno. Se elegieron comportamientos que ponían a prueba todos los actuadores y sensores soportados. Los tres comportamientos se han realizado satisfactoriamente.

Con este trabajo se ha abierto la posibilidad de trabajar con el humanoide Nao en un simulador generalista y de software libre como es Gazebo. De hecho, junto con Jderobot forman un conjunto de herramientas libres que se pueden usar en la docencia de robótica. Emplear el simulador permite por ejemplo plantear prácticas individuales y que cada alumno trabaje en casa sobre ellas, incluso si no se dispone de los costosos robots reales. El hecho de que sea software libre y gratuito permite usarlo incluso si no se dispone de dinero para comprar licencias de simuladores comerciales.

En cuanto a líneas futuras, actualmente estamos trabajando en actualizar el software a versiones más modernas de Gazebo (la 1.0) y de jderobot (la 5.0). También en preparar prácticas docentes que utilicen este humanoide en clases de robótica, por su atractivo para los alumnos y las posibilidades docentes que ofrece. Una tercera línea interesante es envolver al Nao simulado en Gazebo en interfaces NaoQi en lugar de interfaces Jderobot. De esta manera que las aplicaciones desarrolladas para el humanoide real directamente sobre Naoqi funcionarían sin apenas cambios en el simulado en Gazebo.

[8] `http://jderobot.org/index.php/Jbermejo-pfc-itis`

AGRADECIMIENTOS

Este trabajo está financiado en parte por el proyecto S2009/DPI-1559, RoboCity2030-II, de la Comunidad de Madrid y por el proyecto 10/02567 del Ministerio de Ciencia e Innovación.

REFERENCES

[1] S. Carpin, M. Lewis, J. Wang, S. Balarkirsky, and C. Scrapper. Usarsim:a robot simulator for research and education. In *Proceedings of the IEEE 2007 International Conference on Robotics and Automation, pp. 1400-1405*, 2007.

[2] Jeff Craighead, Robin Murphy, Jenny Burke, and Brian Goldiez. A survey of commercial open source unmanned vehicle simulators. In *Proceedings of the IEEE 2007 International Conference on Robotics and Automation, pp. 852-857*, 2007.

[3] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular openrobots simulation engine: Morse. In *Proceedings of the IEEE ICRA*, 2011.

[4] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics ICAR-2003, pp. 317-323*, Coimbra (Portugal), 2003.

[5] J. Jackson. Microsoft robotics studio: a technical introduction. *IEEE Robotics & Automation Magazine*, 14(4):82–87, 2007.

[6] N. Koening and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[7] T. Laue, K. Spiess, and T. Röfer. Simrobot - a general physical robot simulator and its application in robocup. In *In A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Hrsg.), RoboCup 2005: Robot Soccer World Cup IX, Nr. 4020, S. 173–183, Lecture Notes in Artificial Intelligence. Springer*, 2006.

[8] F. Martín, C. Agüero, Cañas J.M., and E. Perdices. Humanoid soccer player design. In Vladan Papic, editor, *Robot Soccer*, pages 67–100. IN-TECH, 2010.

[9] F. Rivas, J. Cañas, and J. González. Aprendizaje automático de modos de caminar para un robot humanoide. In *Proceedings of Robot 2011, III Workshop de Robótica: Robótica experimental*, pages 120–127, Sevilla, November 2011.

[10] Richard T. Vaughan and Brian P. Gerkey. Reusable robot software and the player/stage project. In D. Brugali, editor, *Software Engineering for Experimental Robotics*, pages 267–289. Springer, Berlin / Heidelberg, 2007.