9
MODELADO Y SIMULACIÓN DE ROBOTS PARALELOS CON V-REP. APLICACIÓN DE TELEOPERACIÓN Y MONITORIZACIÓN DE UN ROBOT DE 3 GRADOS DE LIBERTAD

Ángel Valera[29], Vicente Mata[30], Álvaro Page[31], Marina Vallés[32] y Ernesto Oliver[33]

El objetivo de este trabajo es mostrar cómo se puede llevar a cabo el modelado y la simulación de robots paralelos. Para ello se propone utilizar la versión educativa de la aplicación *V-REP*. La utilización de esta aplicación es muy importante porque se trata de un simulador 3D muy potente y completamente gratuito que incluye un conjunto muy amplio de módulos para el cálculo de la cinemática y la dinámica de robots, la planificación de trayectorias, la detección de colisiones, etc. Además, presenta otra ventaja adicional, que es la posibilidad de la programación de macros y de funciones, lo que facilita poder personalizar la aplicación.

En el trabajo se muestra primero cómo se puede hacer el modelado de cualquier sistema robotizado a partir de ficheros de CAD de las

[29]*Instituto Universitario de Automática e Informática Industrial (ai2), Universitat Politècnica de València, Valencia, España. {avalera@ai2.upv.es, mvalles@ai2.upv.es, erolchi@etsii.upv.es}*

[30]*Centro de Investigación en Ingeniería Mecánica, Universitat Politècnica de València, Valencia, España. {vmata@mcm.upv.es}*

[31]*CIBER de Bioingeniería, Biomateriales y Nanomedicina (CIBER-BBN), Spain*

Instituto de Biomecánica de Valencia. Universitat Politècnica de València, Valencia, España. {alvaro.page@ibv.upv.es}

[32]*Instituto Universitario de Automática e Informática Industrial (ai2), Universitat Politècnica de València, Valencia, España. {avalera@ai2.upv.es, mvalles@ai2.upv.es, erolchi@etsii.upv.es}*

[33]*Instituto Universitario de Automática e Informática Industrial (ai2), Universitat Politècnica de València, Valencia, España. {avalera@ai2.upv.es, mvalles@ai2.upv.es, erolchi@etsii.upv.es}*

distintas piezas y componentes del robot. A partir de estos ficheros con *V-REP* se debe completar la estructura cinemática del robot indicando las articulaciones y las restricciones de movimiento.

Una vez modelado el robot, ya se puede proceder a la simulación y el análisis. En el artículo se detallan las formas que dispone *V-REP* para hacer la simulación: mediante programación off-line, la programación on-line mediante *sockets* y la programación on-line con ROS.

Por último, el trabajo presenta una aplicación desarrollada con este entorno, con la que se puede establecer la teleoperación de un robot real y la monitorización remota de éste.

Palabras clave: cinemática de robots, robots paralelos, simulación, control de robots, control por computador

INTRODUCCIÓN

Podemos definir un robot paralelo como una plataforma móvil conectada a la base fija por varias cadenas cinemáticas (también conocidas como piernas). Esta configuración hace que la carga del elemento final se comparta entre las piernas del robot. Comparado con los robots de cadena cinemática abierta (robots serie), los PR suelen tener mayor rigidez, velocidad y precisión. Estas características han provocado que los PR se hayan convertido en un tema de investigación muy importante en el área de la robótica en los últimos años.

De esta forma se pueden encontrar numerosas aplicaciones basadas en PR, como simuladores de movimientos o vuelo y máquinas de pruebas y ensayo (Steward, 1965; Gough y Whitehall, 1962; Merlet, 2000, aplicaciones de fabricación asistida por computador (Pierrot et al. 2009: Valles y col., 2013), (Tsai, 1999), (Chablat y Wenger, 2003), aplicaciones médicas como equipos de reanimación cardiopulmonar (Li y Xu, 2007), robots quirúrgicos (Merlet, 2002) o robots de rehabilitación (Vallés y col. 2015).

En muchas de estas aplicaciones, los PR disponen de seis grados de libertad (DOF) con configuraciones que se siguen desarrollando con nuevas arquitecturas (Cao y col., 2015). Sin embargo, habitualmente, las aplicaciones requieren un número menor de DOF, lo que permite reducir

sustancialmente el precio del desarrollo de estas. De hecho, una de las arquitecturas de PR más rápidas se basa en la arquitectura 3T1R (Pierrot y col., 2009). Además, las aplicaciones de *pick-and-place* se pueden hacer con tres DOF de traslación (3T), como en el robot Delta (Clavel, 1988).

Como se ha comentado anteriormente, los PR tienen un conjunto de características que los hacen muy interesantes. Sin embargo, si se compara esta configuración paralela con la configuración serie de los robots manipuladores industriales típicos se puede comprobar que los PR tienen un conjunto de limitaciones y problemas relacionados con la pequeña área de trabajo que suelen proporcionar y la aparición de singularidades. Además, en los PR es más complejo el cálculo del problema dinámico del robot y de la cinemática directa, lo que complica la simulación y el desarrollo de los sistemas de control de dicha clase de robots.

En este trabajo se va a abordar el modelado y la simulación de PR, lo que ayudará a hacer un análisis de su comportamiento y facilitará posteriormente establecer el control de estos robots. Para poder hacer este modelado y la simulación se puede utilizar una gran variedad de paquetes informáticos. En este artículo se propone utilizar el software *V-REP*.

V-REP es un potente simulador 3D del robot disponible para las plataformas Windows, Mac-OS y Linux. Cuenta con varios módulos de cálculo versátiles que incluyen la cinemática y la dinámica de robots, detección de colisiones, cálculos de distancia mínima, planificación de trayectorias, etc., por lo cual es una herramienta muy indicada para la creación rápida de prototipos y su verificación, la simulación de robots, el desarrollo de sistemas de monitorización remotos, etc.

V-REP presenta además dos ventajas sumamente interesantes: la primera es que, aunque se trata de un software propietario, disponemos de una versión educativa gratuita. La segunda ventaja es la facilidad de programación de algoritmos y macros que proporciona esta aplicación.

Este trabajo presenta cómo se pueden modelar, simular y analizar robots paralelos con *V-REP*, mostrando las diferentes formas de trabajo que este permite. Para ello se va a partir de un PR de 3DOF desarrollado previamente.

El trabajo se estructura de la forma siguiente: en la sección 2 se presenta el robot paralelo desarrollado, detallándose la arquitectura

3-PRS y el conjunto de variables generalizadas que definen al robot. La sección 3 presenta el software de simulación *V-REP*, analizándose las características más destacables de este. Posteriormente, en la sección 4 se presenta cómo se puede hacer el modelado y la simulación del robot paralelo. En esta sección de detalla cómo se puede hacer con *V-REP* la programación *off-line*, *on-line* mediante *sockets* y la programación on-line con el middleware *Robot Operating System* (ROS).

Finalmente, en la sección 5 se aborda un ejemplo de aplicación desarrollado con *V-REP* y con el robot paralelo, presentándose la arquitectura programada para la teleoperación y la visualización remota de dicho robot.

Cabe destacar que, aunque en este trabajo se ha optado por V-REP y *middleware* de control de robots, el robot podría ser presentado como un banco de pruebas muy interesante para utilizarlo con otros entornos de programación y/o aplicaciones de simulación.

ROBOT PARALELO 3-PRS

Los autores de este artículo son miembros de equipo del proyecto de investigación: *Metodología de Diseño de Sistemas Biomecatrónicos. Aplicación al desarrollo de un Robot Paralelo híbrido para diagnóstico y rehabilitación* (MEBIOMEC, 2016). Se trata de un proyecto de I+D+i correspondiente al Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad del Ministerio de Economía y Competitividad del Gobierno de España.

Uno de los objetivos de este proyecto fue el desarrollo de un robot que permitiera hacer el análisis funcional y la rehabilitación de lesiones en miembros inferiores de las personas.

La elección de la arquitectura y movimiento del PM vino determinada por la necesidad de desarrollar un robot de bajo costo capaz de generar rotación angular en dos ejes (alabeo γ y cabeceo β) y elevación como movimiento lineal (altura z). Tras analizar las ventajas e inconvenientes de diversas arquitecturas, finalmente se seleccionó la arquitectura 3-PRS. La Fig. 1 muestra el robot desarrollado.

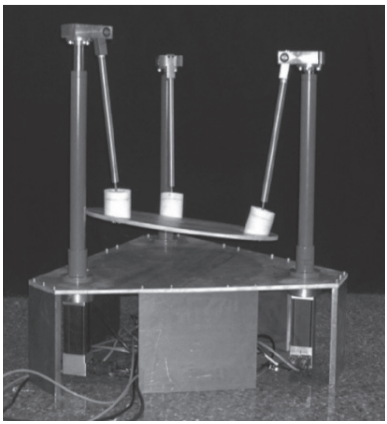

Figura 1. Robot paralelo desarrollado

El robot desarrollado está definido por las 9 coordenadas generalizadas mostradas en la Fig. 2. De estas, q_1, q_6 y q_8 son las articulaciones activas, que corresponden a las articulaciones prismáticas del robot controladas mediante motores de corriente continua *brushless*.

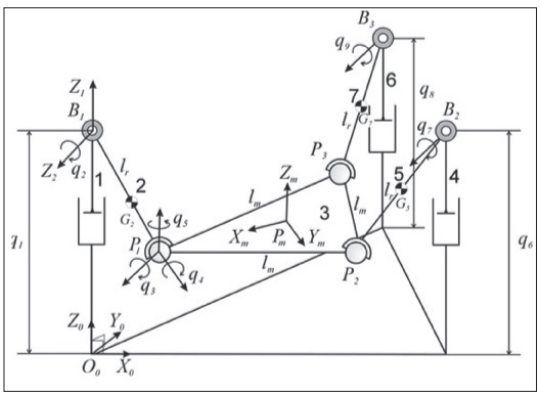

Figura 2. Arquitectura 3-PRS robot paralelo

Para resolver el modelo cinemático directo se plantea un sistema no lineal en el que las coordenadas q_2, q_7 y q_9 son incógnitas desconocidas, utilizándose el método numérico de Newton-Raphon, puesto que es un

método de convergencia rápida (convergencia cuadrática) cuando la estimación inicial está cerca de la solución deseada (Jalón y Bayo, 1994).

La localización de la plataforma móvil se define mediante un sistema de coordenadas locales asociado a ella. Las coordenadas de las articulaciones esféricas de la plataforma se obtienen después de haber encontrado las coordenadas generalizadas de cada pierna del robot. Esas 3 articulaciones comparten el plano de la plataforma, y por tanto se define el eje local X_p como un vector unitario $\vec{u}$ con la dirección dada por $p_1 p_2$. El eje Z_p se define por un vector unitario $\vec{v}$ perpendicular al plano definido el plano definido por los puntos p_1, p_2 y p_3. El eje Y_p (eje $\vec{w}$) se determina por el producto cruzado $\vec{u}$ x $\vec{v}$ Finalmente, las coordenadas generalizadas restantes (q_3, q_4 y q_5) se pueden obtener a partir de la matriz de rotación de la plataforma móvil.

Por otro lado, la resolución del problema cinemático inverso consiste en encontrar los valores de las articulaciones prismáticas (controladas mediante los actuadores eléctricos) a partir de la orientación (alabeo γ y cabeceo β) y la altura (z) de la plataforma. En este problema se debe calcular primero el ángulo de guiñada (α) a partir del alabeo y cabeceo, y a partir de estos tres ángulos se puede calcular la matriz de rotación de la plataforma obteniéndose finalmente las posiciones de los actuadores (Tsai, 1999).

SOFTWARE DE SIMULACIÓN ***V-REP***

Para modelar y simular el PM se ha empleado la aplicación *V-REP* (ver Fig. 3). Se trata de un software de código abierto orientado a la simulación de robots, el cual integra multitud de funciones para la simulación y el control de estos (Coppelia Robotics, 2016).

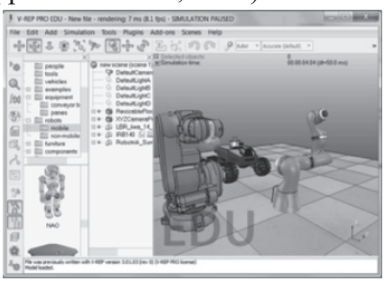

Figura 3. Interfaz V-REP

V-REP se basa en una arquitectura de control distribuido, es decir, que cada objeto o modelo puede ser controlado mediante un *script* embebido, un *plug-in*, un nodo de *ROS*, un cliente API remoto o una solución personalizada (ver Fig. 4). *V-REP* también admite distintos lenguajes de programación (C/C++, Python, Java, Lua, Matlab, Urbi).

Las diferentes formas de controlar la simulación de robots con *V-REP*, son las siguientes:

- *Script embebido*: Mediante el lenguaje de programación de *scripts* Lua, un lenguaje muy intuitivo, se personaliza la simulación a efectuar a través de una secuencia de comandos. Con este método no se puede personalizar el propio el propio simulador.
- *Add-on*: Consiste en la escritura de *scripts* con objeto de personalizar el simulador en sí. Se pueden iniciar automáticamente y se ejecutan en segundo plano o al ser llamados por una función.
- *Plugin*: Adhesión de un plugin para *V-REP* con objeto de personalizar el simulador y/o la simulación.
- *Remote API*: Permite la personalización de la simulación y/o simulador a través de una aplicación externa. Una API básicamente consiste en una biblioteca de instrucciones de un programa para poder ser usadas en otro programa. Existen una gran cantidad de APIs las cuales permiten conectarse a *V-REP* de forma sencilla.
- *ROS node*: ROS es una herramienta basada en la abstracción de *hardware* que permite la interacción entre diversos sistemas. *V-REP* también incluye la posibilidad la posibilidad de conectarse a un nodo de ROS con el objeto de personalizar la simulación.
- *Custom client/server*. Consiste en la personalización del simulador a través de una aplicación externa, la cual actúa como cliente/servidor. Este método es el más complejo, pero a su vez el que permite conseguir un mayor grado de personalización.

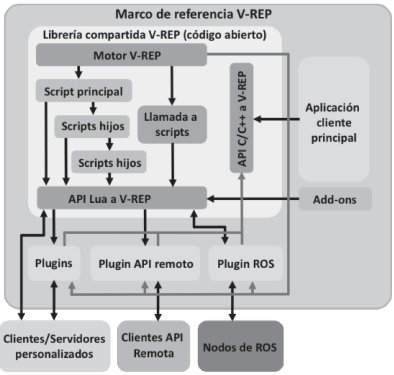

Figura 4. Posibles formas de personalización de *V-REP*

MODELADO Y SIMULACIÓN CON ***V-REP*** DEL ROBOT PARALELO

Modelado del robot paralelo

Para modelar y simular el PM se ha empleado la aplicación *V-REP*. Se trata de un *software* de código abierto orientado a la simulación de robots, el cual integra multitud de funciones para la simulación y el control de éstos.

Para poder trabajar con el modelado y la simulación con *V-REP*, primero se tuvieron que modelar los diferentes elementos del PM: base y vástagos fijos (Fig. 5a), vástagos motrices (Fig. 5b), vástagos guiados (Fig. 5c) y la plataforma (Fig. 5d). Para ello se utilizó el programa de diseño asistido por computador *SolidWorks*, importándose a *V-REP* mediante ficheros formato *".STL"*.

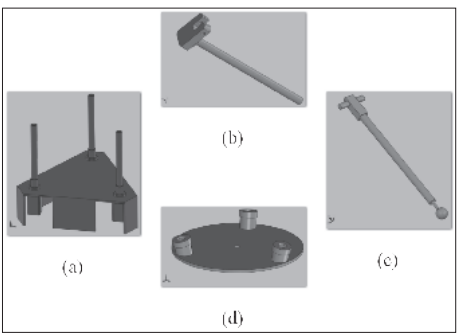

Figura 5. Diseño de los componentes del robot paralelo

Para llevar a cabo la importación de la geometría del robot hay dos opciones. La primera consiste en importar a *V-REP* componente a componente, montando posteriormente el robot mediante las herramientas que tiene *V-REP* para hacer rotaciones y traslaciones de sólidos. La segunda opción consiste en ensamblar el robot en SolidWorks (2016) e importar a *V-REP* el sistema completo. Debido a que en la versión actual la metodología de ensamblaje en *V-REP* no está lo suficientemente desarrollada y presenta algunas limitaciones, es más recomendable la segunda.

El paso siguiente es especificar a cada elemento con qué otros elementos está unido, añadiendo de este modo las restricciones cinemáticas de estos. Para la definición de los elementos hay dos grupos claramente diferenciados: a) elementos estáticos o con un movimiento motriz, y b) elementos con un movimiento guiado. En el primer grupo está la base y el vástago fijo y los vástagos motrices. En el segundo grupo están los vástagos guiados y la base móvil.

A continuación se deben indicar las articulaciones y las restricciones de movimiento. Es necesario tener en cuenta que para cada uno de los vástagos motrices del PM es necesario especificar un par prismático (coordenadas generalizadas q_1, q_6 y q_8), uno rotacional (coordenadas q_2, q_7 y q_9) y un esférico (q_3, q_4 y q_5). Para eso se utilizará la opción *Add joint* y se colocarán los pares prismáticos alineados con los vástagos motrices. Los pares rotacionales se colocarán alineados con el eje del bulón del vástago guiado, y finalmente se ubicarán los pares esféricos en los extremos esféricos de los vástagos guiados.

Por último, para completar la cadena cinemática hay que unir los elementos y restricciones. Para eso hay que pinchar el elemento *slave* que queremos que cuelgue y se arrastra al elemento *master* de tal forma que uno dependa del otro. De este modo, la cadena cinemática debe partir de la base fija y acabar en la base móvil.

Programación y simulación del robot paralelo

Una vez listo el ensamblado del robot se desarrollaron distintas formas de programar el movimiento del PM. En concreto, para este trabajo se ha usado la programación *off-line*, la programación *on-line* con Matlab y la programación *on-line* con ROS.

Con la *programación off-line* se puede simular y analizar el comportamiento del robot para la trayectoria especificada. En este trabajo se ha desarrollado una macro que toma como datos de partida las coordinadas directas del robot: la altura (z) y las orientaciones (θ, ψ) de la plataforma. Estos datos estarán definidos en un archivo de texto. De esta forma, para cada terna de las coordenadas directas se calculará el problema cinemático inverso del robot, obteniéndose un vector con los valores de las coordenadas de traslación de las articulaciones prismáticas q_1, q_7 y q_8.

Para la creación del script de programación off-line se debe seleccionar el botón *Script* de la barra del menú lateral de la parte izquierda de la ventana de *V-REP* y seleccionar la opción *Insert new script: child script (threated)* (Fig. 6).

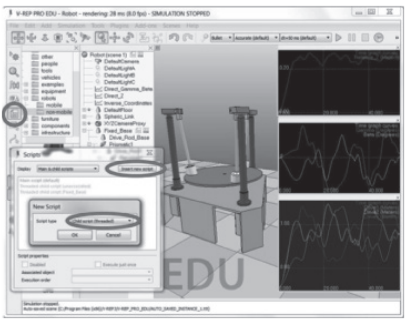

Figura 6. Creación de script de movimiento del robot paralelo

El script estará asociado al primer eslabón de la cadena cinemática (la base fija, en este caso). Posteriormente, solo hay que hacer doble *click* en el icono azul de la base fija (icono del script) y escribir el código que ejecutará el mecanismo.

En los *scripts* de movimiento (Fig. 7) se pueden identificar cuatro partes bien diferenciadas: la declaración de pares motrices, la parametrización de datos de movimiento, los vectores de posición y la instrucción de movimiento *SimRMLMoveToJointPositions*. En esta última parte del *script* se encuentra un bucle *for*, que mediante la instrucción anterior recorrerá el vector de posiciones de las variables prismáticas del robot.

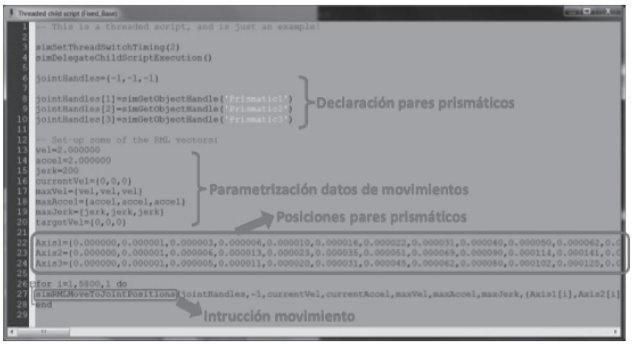

Figura 7. Script de movimiento del robot paralelo para la programación off-line

El segundo método desarrollado para la programación del movimiento del PM es la *programación on-line basada en sockets y Matlab*. El concepto *socket* es una forma muy cómoda y utilizada de comunicación a través de una red entre dos programas alojados en computadoras distintas. Esta forma de comunicación es más conocida por el término API (Interfaz de Programación de Aplicaciones). Para definir un *socket* se necesitan las direcciones IP de ambos computadores y dos puertos que identifiquen cada uno de los programas dentro de cada máquina. Esta forma de comunicación se basa en una arquitectura cliente-servidor.

En este caso, el servidor será la aplicación *V-REP*, mientras que los clientes se pueden desarrollar como una aplicación C/C++, un programa Matlab, una aplicación Java, un script de Python, Lua o Urbi, etc.

En este trabajo se ha establecido la comunicación con un programa de Matlab. Para ello, lo primero que hay que hacer es iniciar el servicio de servidor remoto en *V-REP.* Esto se puede hacer de dos formas diferentes. La primera es modificando el archivo raíz *remoteApiConnections.txt* que está en el directorio de instalación de *V-REP*. Este método ejecuta la simulación al arrancar *V-REP*, comenzando a transmitir sus servicios desde el inicio y permitiendo cargar modelos, objetos, escenarios etc. de forma remota.

La segunda manera es muy parecida a la programación *off-line* comentada anteriormente. Para ello se debe generar un *script* como

el mostrado en la figura 7 sustituyendo la segunda mitad de este (la definición de movimientos de cada punto, los vectores con las posiciones de los pares prismáticos y el bucle *for* con la instrucción de movimiento) por un único comando: *simExtRemoteApiStart*, el cual contiene el puerto asignado para establecer la comunicación.

Para poder iniciar la aplicación cliente remota de Matlab se necesitan las macros *remApi.m, remoteApiProto.m* y la librería *remoteApi.dll*. Todos estos ficheros se encuentran en la carpeta de instalación de *V-REP*.

El programa de Matlab deberá constar principalmente de 3 etapas:

1. *Inicio de la conexión*: con el comando *simxStart* se especifica la dirección *IP* donde se está ejecutando *V-REP* y el puerto de trabajo.
2. *Declaración de los pares prismáticos del robot*. Esto se hace mediante el comando *simxGetObjectHandl* y permite operar con ellos desde el lado del cliente.
3. *Envío de los valores de los pares prismáticos*. Para poder actuar sobre los pares de forma simultánea se utiliza el comando *simx-SetJointPosition.*

El tercer método desarrollado para este trabajo ha sido la *programación on-line mediante ROS*. Este es un metasistema operativo de código abierto desarrollado inicialmente en el *Willow Garage* con colaboración con la *Stanford University* y la *University of California*. Este presenta las funcionalidades de los sistemas operativos adaptados a los sistemas robotizados (ROS, 2016). Para ello utiliza, entre otros conceptos, los nodos y los *topics*. Un nodo es un programa ejecutable capaz de comunicarse con otros nodos emitiendo y leyendo mensajes almacenados en los *topics*.

V-REP permite la integración con *ROS* de una forma muy sencilla mediante *topics* de *ROS*. Para ello se debe generar un *script* parecido a los programados para la comunicación *off-line* y *on-line* con Matlab, utilizándose la instrucción *simExtRos_enableSubscriber*, a la cual se le debe indicar en nombre del *topic* al que se debe suscribir, la acción a ejecutar y el nombre del objeto a manipular. De esta forma, cada vez que se modifique el valor de una articulación prismática en *ROS*, *V-REP*, recibirá dicho cambio modificando la posición y/o la orientación de la plataforma del PM.

Ejemplo de aplicación

A partir del modelado y la simulación del robot paralelo 3-PRS se pudo comprobar que la configuración escogida cumplía satisfactoriamente las especificaciones de movimiento requeridos para la plataforma, por lo cual se procedió posteriormente a su desarrollo práctico. El robot está compuesto por dos unidades: la unidad mecánica y la de control.

La unidad mecánica se efectuó a partir de 3 motores *brushless* BMS465, de la compañía AEROTECH. Se trata de motores muy potentes y lineales equipados con codificadores digitales incrementales (*encoders*) muy precisos.

La unidad de control se ha desarrollado a partir de un PC industrial de altas prestaciones 4U Rackmount equipado con tarjetas de adquisición de datos que permiten la lectura de las posiciones de las articulaciones prismáticas, así como para poder, mediante convertidores digital/analógicos, proporcionar las acciones de control para controlar los 3 motores. En Vallés et. al. (2013) se puede encontrar una descripción más exhaustiva sobre la arquitectura *hardware* del PM.

Para programar las distintas aplicaciones y algoritmos de control del robot se ha instalado en el PC el sistema operativo *Linux Ubuntu* y los *middlewares* de control de robots ROS y Open RObot COntrol Software (Orocos).

Orocos se está consolidando como una de las mejores opciones para llevar a cabo el control de robots, proporcionando la abstracción del sistema operativo, el soporte en tiempo real, la infraestructura de comunicaciones y un modelo basado en componentes (Orocos, 2016).

Mediante Orocos se han implementado diferentes controladores en tiempo real para este robot 3-PRS, como, por ejemplo, controladores por dinámica inversa (Vallés y col. 2012) basados en la pasividad (Díaz-Rodríguez y col. 2013) o controladores adaptativos (Cazalilla y col. 2014).

Para establecer la integración y comunicación entre los componentes de Orocos y la red de ROS se ha utilizado el paquete *rtt_ros_integration*. De esta forma, los componentes pueden publicar y subscribirse a los topics de ROS haciendo completamente compatible los dos *middlewares*. Así, mientras que ROS tiene una gran variedad de herramientas y funcionalidades muy útiles en el desarrollo de aplicaciones robóticas, Orocos proporciona un núcleo sólido para la ejecución de esquemas de control en tiempo real.

Gracias a esta integración entre los dos *middlewares* se ha podido desarrollar una aplicación de teleoperación y visualización del PM cuya arquitectura se muestra en la Fig. 8. Para llevar a cabo la teleoperación se puede utilizar cualquier dispositivo móvil equipado con giróscopos, como los teléfonos inteligentes o las tabletas digitales. Para la visualización remota se ha utilizado el modelo virtual del PM en *V-REP*.

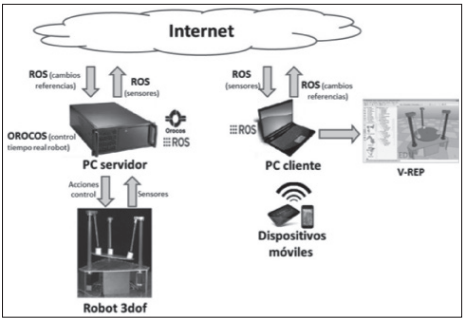

Figura 8. Arquitectura, teleoperación y visualización robot paralelo

La aplicación está dividida en 2 partes: el PC industrial y un PC cliente. En el PC industrial se establece el control de posición en tiempo real con Orocos a una frecuencia de 100Hz. Por ello, cada 10mS Orocos calcula y aplica las acciones de control. Además, se encarga de publicar la posición actual de las articulaciones del robot en *topics* de ROS y de leer los cambios en la referencia suministrados por el dispositivo móvil.

Por otra parte, el PC cliente se encarga de leer los cambios de referencia suministrados por el dispositivo móvil. Además, otro nodo de ROS se encarga de que el robot virtual desarrollado en *V-REP* se mueva a partir de la información suministrada por el robot real.

Utilizando este modelo se hace una visualización en tiempo real de los movimientos del robot real. Se debe destacar que para ello, los valores leídos por el nodo que implementa el modelo virtual *V-REP* provienen directamente de los sensores del robot real. De esta forma, si se diera un error en el control o un fallo inesperado, se podría ver en el robot virtual exactamente la misma posición que la del robot real. Además, puesto que el modelo virtual solo necesita 3 valores (la posición de las 3 variables

prismáticas q_1, q_6 y q_8), se requiere un ancho de banda muy reducido, por lo que no se ha detectado ningún problema con la visualización de forma remota de la evolución de los movimientos del robot.

CONCLUSIONES

En este trabajo se ha presentado una herramienta de simulación de sistemas robotizados: la aplicación *V-REP*. Se trata de una herramienta que permite modelar, simular y analizar cualquier configuración de sistemas robotizados. Las ventajas de *V-REP* son evidentes: es una aplicación muy potente con una versión educativa gratuita que proporciona gran variedad de formas de programación y comunicación.

Este trabajo ha mostrado primero cómo se ha modelado un PM con configuración 3-PRS, y tres formas diferentes de programar *V-REP* para hacer la simulación de este: la programación off-line, la programación *on-line* basada en *sockets* y la programación *on-line* mediante ROS.

Por último, se ha presentado una aplicación que utiliza esta última forma de programar. Gracias al bajo ancho de banda necesario, el robot virtual modelado en *V-REP* reproduce en tiempo real los movimientos que del robot real. Esto permite hacer una teleoperación y una visualización remota del equipo real sobre un modelo virtual.

Las ventajas de esto podrían ser numerosas. Un médico fisioterapeuta, por ejemplo, podría estar supervisando y analizando de forma remota la evolución de un conjunto de PM diseñados para la rehabilitación de pacientes con lesiones en piernas o brazos.

AGRADECIMIENTOS

Los autores de este trabajo desean hacer llegar su agradecimiento al Plan Nacional de I+D de la Comisión Interministerial de Ciencia y Tecnología (FEDER-CICYT) bajo el proyecto de investigación DPI2013-44227-R, así como al Instituto Universitario de Automática e Informática Industrial (ai2) de la Universitat Politécnica de Valencia por la financiación parcial de este estudio.

REFERENCIAS

Cao, R.; Gao, F.; Zhang, Y.; Pan, D.; Chen, W. (2015) *Anewparameter designmethod of a 6-DOF ParallelMotion Simulator for a GivenWorkspace*. Mechanics Based Design of Structures and Machines 43(1):1–18.

Cazalilla, J.I.; Vallés, M.; Mata, V.; Díaz-Rodríguez, M.; Valera, A. (2014) *Adaptive control of a 3-DOF parallel manipulator considering payload handling and relevant parameter models*. Robotics and Computer-Integrated Manufacturing, 30(5), 468–477.

Chablat, D.; Wenger, P. (2003) *Architecture optimization of a 3-dof translational parallel mechanism for machining applications, the orthoglide*. IEEE Transactions on Robotics and Automation 19(3), 403–410.

Clavel, R. (1988) *DELTA, a fast robot with parallel geometry. Proceedings of 18th International Symposium on Industrial Robot*, Lausanne, April, 91–100.

Coppelia Robotics (2016) *V-REP, virtual robot experimentation platform*. Descargado de: http://www.coppeliarobotics.com/index.html

Díaz-Rodríguez, M.; Valera, A.; Mata, V.; Vallés, M. (2013) *Model-Based Control of a 3-DOF Parallel Robot Based on Identified Relevant Parameters*. IEEE/ASME Transaction on Mechatronics, 18(6), 1737-1744.

Gough, V.; Whitehall, S. (1962) Universal tire test machine. In: *Proceedings of 9th International Technical Congress FISITA*.

Jalón, J.G.; Bayo, E. (1994) *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time challenge*. Springer-Verlag, New-York.

Li, Y.; Xu, Q. (2007) *Design and development of a medical parallel robot for cardiopulmonary resuscitation*. IEEE/ASME Transaction on Mechatronics, 12(3), 265–273.

MEBIOMEC (2016) Metodología de *diseño de sistemas biomecatrónicos. Aplicación al desarrollo de un robot paralelo híbrido para diagnóstico y rehabilitación*. Disponible on-line en: https://mebiomec.ai2.upv.es/

Merlet, J. P. (2000) *Parallel Robots*. Kluwer, London, U.K.

Merlet, J.P. (2002) Optimal design for the micro parallel robot MIPS. In *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp.1149-1154.

OROCOS (2016) *The Orocos Project*. Disponible on-line en: http://www.orocos.org/

Pierrot, F.; Nabat, V.; Company, O.; Krut, S.; Poignet, P. (2009) Optimal design of a 4-dof parallel manipulator: From *Academia to Industry*. IEEE Transactions on Robotics 25(2), 213–224.

ROS (2016) *ROS, Powering the world's robots*. Disponible on-line en: www.ros.org/

SolidWorks (2016) Disponible on-line en: http://www.solidworks.es/

Steward, D.A. (1965) A platform with 6 degree of freedom. In: *Proceedings of the Institution of mechanical engineers*, 180(1=, 371-386.

Tsai, L. W. (1999) Robot Analysis: The Mechanics of Serial and Parallel Manipulator. Wiley Interscience.

Vallés, M.; Cazalilla, J.I.; Valera, A.; Mata, V.; Page, A. (2012) *Mechatronic Development and Dynamic Control of a 3-DOF Parallel Manipulator*. Mechanics Based Design of Structures and Machines, 40(4), 434–452.

Vallés, M.; Cazalilla, J.I.; Valera, A.; Mata, V.; Page, A. (2013) Implementación basada en el middleware OROCOS de controladores dinámicos pasivos para un robot paralelo. En: *Revista Iberoamericana Automática e Informática Industrial* 10(1), 96–103.

Vallés, M.; Cazalilla, J.I.; Valera, A.; Mata, V., Page; Díaz-Rodríguez, M. (2015) *A 3-PRS parallel manipulator for ankle rehabilitation: towards a low-cost robotic rehabilitation*. Robotica, 35(10), 1939-1957.