Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0340

CONTROL DE LA MANO ALLEGRO USANDO SMARTPHONES

Lucía Mas Lillo
Dpto. Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, lml44@alu.ua.es

Santiago T. Puente
Dpto. Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, santiago.puente@ua.es

Fernando Torres
Dpto. Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, fernando.torres@ua.es

Francisco A. Candelas
Dpto. Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, francisco.candelas@ua.es

Resumen

Este artículo presenta un entorno multiplataforma basado en unity3D para la virtualización de una mano robótica. La razón por la que se usa Unity3D es que, en general los entornos virtuales para las manos robóticas están limitados al uso de ordenadores y de software propietario. Además, la arquitectura usada en el sistema permite la conexión con un servidor con acceso a los nodos de ROS, que puede ser configurado para el uso de una mano robótica real o simulada. El sistema presentado permite realizar el control en tiempo real y recibir el feedback de los movimientos de la mano, permitiendo a varios usuarios realizar la supervisión de los movimientos de ésta. El sistema ha sido probado usando la mano robótica de cuatro dedos Allegro y ha sido controlada usado diferentes dispositivos Android y iOS.

Palabras clave: Realidad virtual, Allegro, ROS, Unity3D

1 INTRODUCCIÓN

Hace algunos años, la mejor manera de simular y controlar un robot era usando ordenadores. Sin embargo, los smartphones pueden actualmente realizar estas dos operaciones. Estos dispositivos han mejorado mucho durante la última década: Tienen una mayor memoria RAM, mayor espacio de almacenamiento, mejores procesadores, mayor duración de las baterías y es más fácil desarrollar software para ellos. Dependiendo del caso puede ser mejor usar dispositivos telefónicos para el control u ordenadores. Aunque los ordenadores continúan teniendo una mayor capacidad de cómputo, los Smartphones tienen algunos puntos fuertes: son más ligeros, no necesitan estar conectados a la red eléctrica, es más sencillo poder usarlos en diferentes entornos. Existen algunos artículos en los que se expone el uso de dispositivos telefónicos para realizar la teleoperación de un sistema robótico, como es el caso de [4] donde se expone el uso de un dispositivo iOS para la interacción hombre-robot .

Actualmente, existe una gran cantidad de dispositivos telefónicos en el mercado, con diferentes sistemas operativos, almacenamiento, resoluciones de pantalla [3]. Aunque sea sencillo desarrollar software para ellos es complicado realizar un buen diseño de pantalla que funcione correctamente en todos los dispositivos actuales. Los motores gráficos como Unity3D simplifican esta tarea al usar canvas escalables para todos los tamaños posibles de dispositivos. Unity3D es un motor de videojuegos. Los motores de videojuegos permiten la realización de simulaciones de manera más sencilla ya que incluyen físicas, carga de modelos 3D, cálculos de iluminación, aplicación de texturas sobre modelos etc. Todo ello de manera optimizada y a tiempo real, algo imprescindible para poder realizar aplicaciones de control de sistemas. La aplicación de motores gráficos de juegos para crear interfaces de control es algo que ya se ha realizado anteriormente [14].

En este artículo se presenta un controlador y simulador para dispositivos telefónicos de la mano robótica Allegro. Realizado mediante Unity3D y ROS (Robotic Operating System). ROS es un framework de código abierto que simplifica el desarrollo de software de los robots.

El sistema realizado tiene una arquitectura cliente/servidor. El cliente se ejecuta en los smartphones y es multiplataforma, funciona en iOS, Android y Windows phone, esta parte utiliza Unity3D. El servidor se basa en ROS y es el encargado de comunicarse con el controlador real de la mano enviando las posiciones de la mano Allegro.

Para la comunicación entre el cliente y el servidor se ha usado sockets TCP asíncronos.

En la sección 2 se va a tratar brevemente el estado del arte, en la sección 3 se realiza una descripción del sistema. En la sección 4 se presenta la experimentación realizada. Finalmente, en la sección 5 se presentan las conclusiones y las líneas de trabajo futuras.

2 ESTADO DEL ARTE

En esta sección se presenta el estado del arte de este trabajo en la sección 2.1 se realiza una descripción de diferentes aplicaciones realizadas en Unity3D. En la sección 2.2 se presentan diferentes entornos de simulación de robots.

2.1 SIMULACIÓN EN UNITY3D

Hoy en día no es necesario realizar un simulador desde el principio, existen diferentes herramientas que simplifican esa tarea y además optimizan significativamente el proceso, como es por ejemplo el uso de motores gráficos. Uno de los motores más recomendados es Unity3D [8].

Es posible usar de múltiples maneras los algoritmos de control de robots, como es por ejemplo combinar Unity3D y Matlab para hacer un seguimiento de rutas [2], el cual utiliza memoria compartida entre ambos.

Es posible conectar la aplicación desarrollada mediante Unity3D con el robot a partir de diferentes métodos. Utilizando por ejemplo el protocolo TCP/IP [14], rosbridge [5] el cual proporciona una API JSON para conectar un sistema ROS con otro tipo de sistemas. Existen en la actualidad otros métodos para conectarse con sistemas ROS como es por ejemplo en el caso de usar tecnología móvil ROS Android, el cual permite conectar dispositivos Android con los nodos de ROS directamente, actualmente es tecnología experimental o ROSPod, permite lo mismo para sistemas iOS aunque actualmente se encuentra en desarrollo. En el caso de este artículo se ha decidido utilizar sockets TCP asíncronos para la comunicación, ya que estos son soportados por todos los dispositivos telefónicos.

Algunos usos de Unity3D para simular el entorno y controlar sistemas robóticos son por ejemplo para el control de múltiples UAV simulando una ciudad [11, 14], crear un entorno para la búsqueda y el salvamento simulando además los sensores de diferentes tipos de robots: aéreos, acuáticos y terrestres en sus respectivos entornos [6, 7] o crear un entorno virtual para una interfaz cerebro máquina en el que se permite realizar el control de un robot a través de las señales eléctricas del cerebro [17].

2.2 SIMULACIÓN DE MANOS ROBÓTICAS

Cuando se habla de manos robóticas un punto importante es la simulación. Es necesario analizar sus características versus las de las manos humanas con la finalidad de transformarlas en un modelo 3D digital [10, 20]. Estos modelos pueden ser usados para diseñar una mano robótica [16]. Existen diferentes posibilidades para simular el control de manos robóticas. Por ejemplo, el framework SynGrasp [13] que es una toolbox para el control de manos robóticas en Matlab Hands.dvi [18] es un framework de programación independiente de dispositivos para mano robóticas. Gazebo [1] es un software de simulación que puede ser usado en ROS (Robotic Operating System) con intención de simular cualquier tipo de dispositivo. Uno de sus usos es la simulación de manos robóticas juntándolo con GraspIt [15], para simular algoritmos de agarre. En ROS otra herramienta de simulación es Rviz la cual proporciona funcionalidades para la simulación.

Para controlar sistemas robóticos el uso de sistemas de realidad aumentada es muy útil [9] para mezclar la información real con la virtual. Además, el uso de un guante o de visión artificial [12] es una técnica interesante para controlar una mano robótica, debido a que sólo tiene que reproducir el movimiento de la mano humana.

Otra manera de controlar un robot es usar las señales eléctricas del cuerpo humano como por ejemplo la electromiografía (EMG) [17]. Con estas señales es posible mover un robot de acuerdo con las referencias proporcionadas con el movimiento de la mano de un usuario.

Teniendo en cuenta los artículos previos, no existen herramientas multiplataforma diseñadas para simular y controlar una mano robótica, esto es lo que se plantea en este artículo. Una aplicación multiplataforma para simular y controlar una mano robótica.

3 DESCRIPCIÓN DEL SISTEMA

En esta sección se describe la arquitectura del sistema, además del hardware usado y las aplicaciones realizadas.

3.1 SIMULACIÓN DE MANOS ROBÓTICAS

La arquitectura del sistema viene descrita en la Figura 1. En esta sección se describe la arquitectura

del sistema, además del hardware usado y las aplicaciones realizadas.

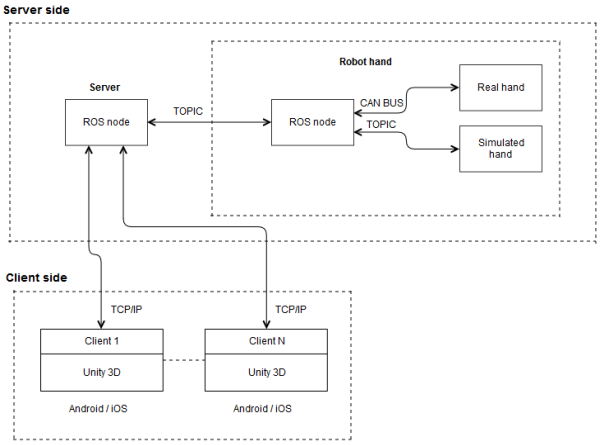


Figura 1: Arquitectura del sistema

Como podemos comprobar en la Figura 1 es posible ejecutar la aplicación en uno o más clientes al mismo tiempo. La arquitectura cliente/servidor ha sido creada usando sockets TCP/IP, ya que esto permite el uso de un sistema multiplataforma. El cliente envía al servidor la posición de cada articulación cuando cambia su posición en el dispositivo móvil. Cuando el servidor recibe la nueva posición desde el cliente la publica en el topic de ROS y envía a todos los clientes conectados la posición actual del sistema. Los topics contienen diferentes tipos de datos, son usados por ROS para comunicar los diferentes nodos mediante la publicación y la suscripción a topics para enviar y recibir información respectivamente. Los nodos son la parte de ROS que contiene las funcionalidades del sistema. En el esquema de la aplicación la interconexión viene representada de la siguiente manera, el servidor contacta con el nodo de ROS que contiene el controlador de la mano ya sea real o simulada mediante la publicación de las posiciones en el topic correspondiente, en el cual está suscrito la mano. Desde ese nodo se envían las ordenes a la mano real (Usando el bus CAN) o simulada (directamente con la información en los topics). El nodo del servidor también está suscrito al topic que contiene las posiciones actuales de la mano, cuando se recibe una nueva posición el servidor actualiza el resto de los clientes con la información publicada en este topic.

El servidor está conectado con los nodos de ROS creados en GNU/Linux por la aplicación del laboratorio simLab. El servidor contiene un nodo de ROS para publicar y suscribirse a los topics de SimLab.

3.2 ESPECIFICACIONES DE ALLEGRO HAND

La mano Allegro del laboratorio SimLab es una mano antropomórfica de bajo coste. Está compuesto de cuatro dedos; uno de ellos pulgar. Cada uno de los dedos tiene cuatro articulaciones, haciendo un total de dieciséis. Esta mano utiliza la tecnología para manos robóticas desarrollada por el Instituto de tecnología industrial de corea (KITECH [19]). Las especificaciones técnicas vienen especificadas en la tabla 1.

Tabla 1: Especificaciones técnicas de Allegro

Característica	**Especificación**	
N de dedos	4, 1 pulgar	
Grado de libertad	4 dedos x 4 =16 (Activos)	
Activación	Tipo	DC Motor
	Ratio eng.	1:369
	Max. Giro	0.7 Nm
	Max. Vel.	0.11 (s/60º)
Peso	Dedos	0.17 (kg)
	Pulgar	0.19 (kg)
	Total	1.08 (kg)
Resolución art.	0.002 (deg)	
Comunicación	CAN 333 (Hz)	
Peso de carga	5 (kg)	
Requisitos energ.	7.4VDC(7.0V– 8.1V), 5A	

La figura 2 muestra la especificación de la rotación de las articulaciones de la mano Allegro, tanto para la mano derecha como para la izquierda. Las direcciones hacia las que se pueden realizar las rotaciones son importantes para poder realizar una correcta simulación en ambas manos y permite realizar la transformación automática de los movimientos de una mano a la otra.

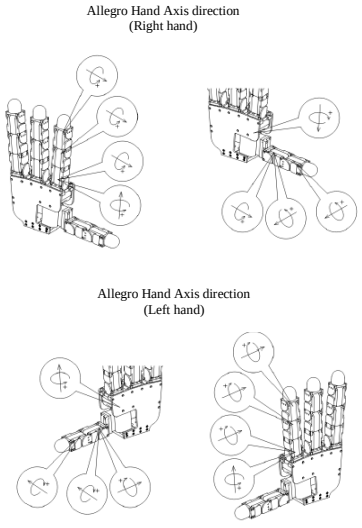


Figura 2: Rotación de las articulaciones de la mano Allegro

Además, es posible teleoprar la mano Allegro en diferentes sistemas operativos, en GNU/Linux usando Rviz o en Windows mediante la aplicación Allegro Studio. En el sistema desarrollado se utiliza GNU/Linux para realizar la comunicación con la mano desde el servidor. Para la comunicación con el sistema real se utiliza el protocolo CAN con una frecuencia de 333 Hz

3.3 APLICACIÓN EN EL CLIENTE

La aplicación desarrollada en el cliente se realizó usando el motor de juegos Unity3D, se programó en C#. Unty3D permite el desarrollo de aplicaciones multiplataforma. La aplicación fue testeada en dispositivos Android y iOS. En esta sección se explican todas las opciones desarrolladas en el cliente.

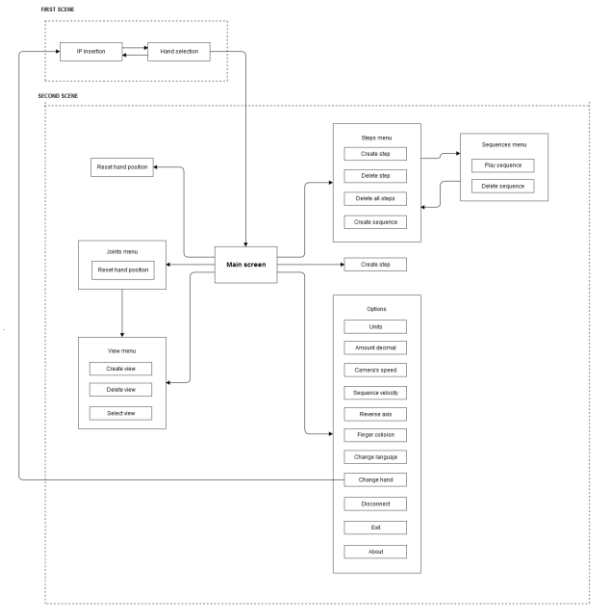
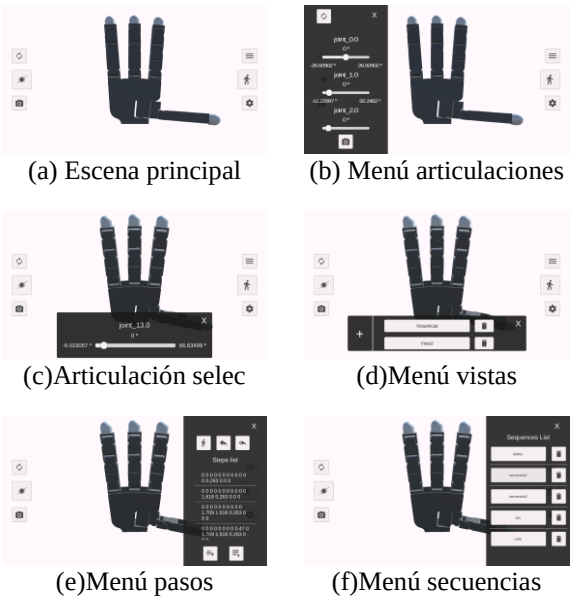


Figura 3: Esquema de opciones del cliente

Unity3D divide la aplicación en escenas. En este caso se han realizado dos escenas. Estas escenas y sus funcionalidades son mostradas en la figura 3. Además, se muestra de manera específica donde están situadas cada escena y cómo se comunican entre si.

En la primera escena de la aplicación el usuario pude decidir introducir una dirección IP para conectarse con el sistema real en GNU/Linux o trabajar de manera local. A continuación, el usuario debe seleccionar entre usar la mano derecha o la mano izquierda. Una vez seleccionada, se carga la mano a partir de la información que está almacenada en un fichero URDF. Los ficheros URDF son ficheros escritos en formato XML que contienen la información de carga de los modelos de los robots. Este formato es muy utilizado en ROS

La segunda escena comienza con la imagen mostrada en la figura 4a, en ella se aprecia la carga del modelo 3D de la mano seleccionada y los botones que permiten acceder a las diferentes opciones implementadas: Movimiento de articulación, movimiento de cámara, acceso a vistas, secuencias, opciones etc. Es posible cambiar entre las opciones de acuerdo con el esquema de funcionalidades mostrado en la figura 3.



(a) Escena principal (b) Menú articulaciones

(c)Articulación selec (d)Menú vistas

(e)Menú pasos (f)Menú secuencias

Figura 4 Segunda escena de la aplicación (a) Escena principal de la aplicación (b) aplicación con la selección de articulaciones (c) movimiento de una articulación cuando esta se selecciona (d) menú de vistas (e) menú de pasos almacenados (f) menú con las secuencias almacenadas.

Para poder mover las articulaciones es posible utilizar el menú de articulaciones que aparece en la figura 4b. Este menú aparece cuando se pulsa el segundo botón de la parte izquierda de la escena principal (figura 4a). En él se ve una lista con todas las articulaciones disponibles y un deslizados que hace referencia a la posición de estas. También es posible hacer el movimiento de una articulación pulsando sobre la articulación que se desea mover en el modelo 3D (figura 4c). Es posible poder reestablecer las posiciones a los valores 0 o mínimos cuando se pulsa el botón representado con el dibujo que se encuentra en la primera posición a la izquierda en la figura 4a

Se han desarrollado unos controles de cámara dentro del sistema de representación 3D. Es posible poder guardar, eliminar o cargar una posición de cámara usando el menú de vistas figura 4d. Para guardar una nueva vista es imprescindible ponerle un nombre. Las vistas son almacenadas en el dispositivo cuando se guardan.

Los menús de secuencias y de pasos se han desarrollado para guardar y mostrar las posiciones de la mano del robot para realizar una tarea. Cuando un paso se crea la posición actual de las articulaciones es guardada dentro de un array. Una secuencia es un conjunto de pasos y cuando se crea una se guarda en el dispositivo con el nombre que se le haya dado por parte del usuario. Para crear un nuevo paso se puede seleccionar el segundo botón de la parte derecha en la figura 4a o dentro del menú de pasos (figura 4e) una vez seleccionado el primer botón de la derecha en la figura 4a, donde pulsando los botones de la parte superior de izquierda a derecha se permite crear un nuevo paso, borrar el paso anterior o eliminar todos los pasos. En la parte inferior al lado izquierdo encontramos el botón para guardar una nueva secuencia y en el lado derecho el botón de acceso al menú de secuencias figura 4f donde es posible reproducir o eliminar una secuencia guardada.

Antes de guardar una secuencia o una nueva vista se preguntará el nombre de estos, este menú es muy similar en ambos casos. En el caso de las secuencias será necesario tener al menos un paso creado.

Por último, se explica el botón de opciones. Este menú aparece cuando se selecciona el tercer botón de la derecha en la figura 4a. Las opciones son guardadas el en dispositivo. Se han desarrollado las siguientes opciones en este menú:

- Los usuarios pueden elegir las unidades de medida en grados o en radianes y elegir la cantidad de decimales que se van a mostrar en los sliders.
- Se puede cambiar la velocidad de la cámara e intercambiar los ejes.
- La velocidad a la que se mueven las secuencias se puede cambiar.
- Activar o desactivar la colisión entre los dedos
- Cambiar de idioma. Actualmente entre español e inglés.
- Cambiar de mano sin cerrar la aplicación.
- Desconectarse del servidor
- Cerrar la aplicación

La interfaz de la segunda escena se diseñó teniendo en cuenta las opciones más usadas por los usuarios, los botones más grandes representan las acciones más usadas por los usuarios, de esta manera se ahorran tiempo al no tener que estar abriendo y cerrando menús.

3.4 APLICACIÓN EN EL SERVIDOR

El servidor permite la posibilidad de conectar al cliente con una mano real o simulada. Para ello primero se tendrá que ejecutar la aplicación de control de la mano desarrollada por SimLab en GNU/Linux. Después de haber ejecutado la aplicación se lanzará el servidor en ROS. Si se utiliza la mano real será necesario indicarlo al servidor usando la palabra real.

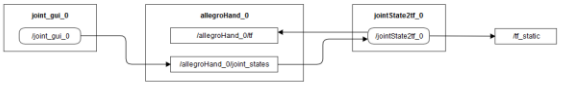
Para mover las diferentes articulaciones de las manos será necesario publicar en determinados topics */allegroHand_0/joint_cmd* o */allegroHand_0/joint_states* dependiendo si se trata de la mano real o simulada. Estos nodos pertenecen a la aplicación del laboratorio SimLab. La aplicación desarrollada sustituye al nodo */joint_gui_0* (Figure 5), el cual se encarga de enviar las posiciones de las manos. Para hacerlo correctamente se mata a este nodo para así evitar interferencias.

Se publica un mensaje de tipo *ensor joint_State type.* Este mensaje contiene las dieciséis posiciones de las articulaciones y sus nombres entre otras cosas. Los valores enviados desde el cliente al servidor serán publicados en el topic modificando las posiciones de las articulaciones. Además el servidor enviará a todas los clientes existentes salvo al que está modificando la posición actual de la mano. Por eso está suscrito el servidor al jointStates.

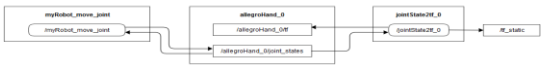
La aplicación del servidor no está preparada para poder trabajar con más de una mano al mismo tiempo. Esta aplicación funciona únicamente con mano identificada como 0. Además, se debe mantener la mano seleccionada en el servidor o relanzarlo para cambiar a una mano diferente.

En la figura 5 se presenta un esquema con los nodos y los topics de ROS. Los topics están representados por rectángulos con las esquinas redondeadas y los nodos por rectángulos. En la figura 5a quedan representados los nodos y los topics antes de ejecutar el servidor. El nodo */joint_gui_0* publica en el topic */allegroHand_0/joint_states.* El nodo */JointState2tf_0* está suscrito al topic */allegroHand_0/joint_states* así pues, este nodo recibe toda la información publicada por el nodo */joint_gui_0.* Este nodo envía las posiciones a las que se quiere llegar en el sistema simulado. En la figura 5b se ejecuta el servidor en una mano simulada, como podemos comprobar se ha reemplazado el nodo */joint_gui_0* por el nodo *myRobot_move_joint.* Lo mismo sucede entre las figuras 5c y 5d en las que se ejecutan los nodos con una mano real sin y con servidor respectivamente, se

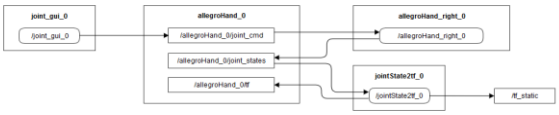
sustituye el nodo */joint_gui_0* con el del servidor. Para recibir correctamente la información el nodo se suscribe a */allegroHand_0/joint_states* donde se almacenan las posiciones en las que se encuentra la articulación del robot. En el caso de enviar ordenes al robot real es necesario realizar la publicación en el topic */allegroHand_0/joint_cmd.*



(a)Simulada sin servidor



(b)Simulada con servidor



(c)Real sin servidor

(d)Real sin servidor

Figura 5 Esquema de nodos y topics antes y después de lanzar la aplicación del servidor en la mano real y la simulada

4 EXPERIMENTOS

Se han comprobado varios experimentos. Inicialmente se realizaron experimentos usando el entorno simulado para asegurar el correcto funcionamiento del sistema. Después de probar el sistema simulado se ha testeado con el fin de realizar el control de la mano real.

La arquitectura del sistema permite el control de una mano simulada o real con uno o más dispositivos simultáneamente, independientemente del sistema operativo

4.1 EXPERIMENTOS EN LA SIMULACIÓN

La simulación se ha probado con sistemas IOS y Android. Todos los dispositivos eran capaces de actualizar los valores de la mano y de teleoperar al mismo tiempo. En el lado del servidor Rviz fue usado para poder vigilar la posición actual de los joints.

En la figura 6 se muestra una secuencia de control de la mano derecha en un dispositivo Android. Esta secuencia de movimientos se está ejecutando usando el menú de joints. En esta figura podemos observar el dispositivo y el movimiento del robot en simulación mediante RVIZ.

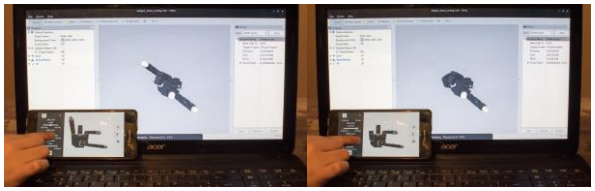


Figura 6: Simulación con un dispositivo Android.

En la figura 7 se muestra una secuencia de movimientos de la mano en simulación. Los dispositivos implicados en este experimento son un iPad Air, un Huawei P9 Lite y un Motorola G3. Al principio el movimiento es controlado usando el menú de articulaciones, pero después se selecciona una articulación y se modifican el valor.

Figura 7: Simulación con tres dispositivos

En simulación se testearon todas las posibles opciones para mover una mano y el sistema funcionó como se esperaba, inclusive en dispositivos de gama media baja (Tabla 2).

Tabla 2: Dispositivos sobre los que se han realizado pruebas del sistema

SO	**Dispositivo**
Android	Huawei P9 Lite
Android	Sony Xperia XA
Android	Motorola G3
Android	Huawei P8 Lite
Android	Samsung Galaxy S5
Android	Huawei Y635
Android	Sony Xperia E
Android	Sony Xperia typo
iOS	iPad Air

4.1 EXPERIMENTOS EN LA MANO REAL

Se ha testeado además la aplicación sobre la mano real. En este primer apartado se verán los resultados con un sólo dispositivo.

En la figura 8 se muestra una secuencia de movimientos realizada mediante la selección de un joint sobre el sistema real. El movimiento realizado se puede visualizar también en el ordenador mediante RVIZ. La sincronización entre el cliente, el servidor y la mano real funcionó adecuadamente y realizó los movimientos que se pidieron

Figura 8: Mano real y un dispositivo

Se testeo el sistema real también con varios dispositivos, tal y como se puede observar en la figura 9. En esta figura se puede observar tres dispositivos android conectados simultáneamente a la mano robot

Figura 9 Mano real y tres dispositivos

5 CONCLUSIONES Y TRABAJOS FUTUROS

Se ha desarrollado una aplicación multiplataforma que permite la simulación y el control de una mano robótica usando Unity3D. El sistema implementado en Unity3D se conecta con los nodos de ROS usando sockets TCP/IP, permitiendo el control tanto de manos simuladas como de manos reales.

Se ha testeado el sistema probando el envío de paquetes tanto a la mano real como a la mano simultánea de manera satisfactoria. La aplicación se ha probado con una gran variedad de dispositivos telefónicos y de sistemas operativos.

Un punto por relucir de esta aplicación es que permite el control y la supervisión de una mano robótica por parte de múltiples usuarios al mismo tiempo usando en algunos casos dispositivos telefónicos de gama baja. Teniendo en cuenta que el control del sistema se ha realizado es posible utilizar cualquier algoritmo existente en ROS con el fin de realizar, por ejemplo, operaciones de agarre, obteniendo en tiempo real el feedback del agarre en su propio dispositivo. Utilizar algoritmos de agarre conectados con la aplicación es actualmente un trabajo en desarrollo.

Otro de sus futuros desarrollos es la conexión de la interfaz para controlar más variedad de manos robóticas. También se puede mejorar la interfaz para incluir objetos en la simulación con los que se pueda interaccionar.

Otra futura mejora es la posibilidad de poder ejecutar simultáneamente más de una mano.

Agradecimientos
Research supported by Spanish Ministry of Economy and University of Alicante, through projects DPI2015-68087-R, GRE 16-20

English summary

SUPERVISION AND CONTROL OF ALLGRO-HAND WITH MOBILE DEVICES

Abstract

This paper presents a multiplatform environment based in Unity 3D, in order to virtualize a robotic hand. The reason to uses Unity 3D is that, in general, virtual environments for robotics hand are limited to computers and proprietary software of the manufacturers. Furthermore, the architecture used in the systems allows to connect to a ROS server, which can be configured to use simulation of the robotic hand or to use real robot hand. The system presented perform a real-time control and feedback of the hand movements, allowing several users to perform a supervision of the hand movements. The system has been tested using the four fingers SimbLab's Allegro robot hand and controlled using several devices with Android and iOS.

Keywords: Virtual reality, robots, Unity 3D, ROS, Allegro-hand

Referencias

[1] Aguero C, Koenig N, Chen I, Boyer H, Peters S, Hsu J, Gerkey B, Paepcke S, Rivero J,

Manzo J, Krotkov E, Pratt G (2015), Inside the virtual robotics challenge: Simulating real- time robotic disaster response. IEEE Transactions on Automation Science and Engineering 12:494–506. doi: 10.1109/TASE.2014.2368997.

[2] Andaluz V.H., Chicaiza F.A. , Gallardo C., Quevedo W. X. ,Varela J., Sánchez J.S., Arteaga O. (2016) Unity3D-MatLab Simulator in Real Time for Robotics Applications. In: De Paolis L., Mongelli A. (eds) Augmented Reality, Virtual Reality, and Computer Graphics. AVR 2016. Lecture Notes in Computer Science, 9768. doi: 10.1007/978-3-319-40621-3_19

[3] Berg L, Vance J (2017) Industry use of virtual reality in product design and manufacturing: a survey. Virtual Reality, 21:1-17. doi:10.1007/s10055-016-0293-9

[4] Codd-Downey R, Jenkin M (2015), RCON: Dynamic Mobile Interfaces for Command and Control of ROS-enabled Robots. 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO),

[5] Codd-Downey R, Mojiri Forooshani P, Speers A, Wang H, Jenkin M (2014), From ROS to Unity: leveraging robot and virtual environment middleware for immersive teleoperation. IEEE International Conference on Information and Automation (ICIA), 932-936. doi: 10.1109/ICInfA.2014.6932785.

[6] Craighead J, Burke J, Murphy R (2008) Using the Unity Game Engine to Develop SARGE: A Case Study. Proceedings of the 2008 Simulation Workshop at the International Conference on Intelligent Robots and Systems (IROS 2008).

[7] Craighead J, Gutierrez R, Burke J, Murphy R.(2008) Validating the Search and Rescue Game Environment as a robot simulator by performing a simulated anomaly detection task. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2289-2295. doi: 10.1109/IROS.2008.4650746

[8] Craighead J, Murphy R, Burke J, Goldiez B. (2007) A Survey of Commercial & Open Source Unmanned Vehicle Simulators. IEEE International Conference on Robotics and Automation 852-857. doi: 10.1109/ROBOT.2007.363092

[9] Green SA, Billinghurst M, Chen X, Chase JG (2008) Human-Robot Collaboration: A Literature Review and Augmented Reality Approach in Design. International Journal of Advanced Robotic Systems, 5:1-18. doi:10.5772/5664.

[10] Hirt J, Berns K, Mianowski K (2012) Designing Arms and Hands for the Humanoid Robot ROMAN, Advanced Materials Research, 463:1233–1237. doi:10.4028/www.scientific.net/AMR.463-464.1233

[11] Hu Y, Meng W (2016) ROSUnitySim: Development and Experimentation of a Real-time Simulator for Multi-UAV Local Planning, Simulation 92:931–944. doi: 10.1177/0037549716666683

[12] Iliukhin VN, Mitkovskii KB, Bizyanova DA, Akopyan AA (2017) The Development of Motion Capture System Based on Kinect Sensor and Bluetooth-Gloves. Procedia Engineering, 176:506-513. doi:10.1016/j.proeng.2017.02.350.

[13] Malvezzi M, Gioioso G, Salvietti G, Prattichizzo D (2015) SynGrasp: A MATLAB Toolbox for Underactuated and Compliant Hands. IEEE Robotics & Automation magazine 52-68. doi:10.1109/MRA.2015.2408772

[14] Meng W, Hu Y, Lin J, Lin F, Teo R (2015) ROS+Unity: An Efficient High-fidelity 3D Multi-UAV Navigation and Control Simulator in GPS-Denied Environments. Industrial Electronics Society, IECON 2015 - 41st Annual Conference of the IEEE. doi**:** 10.1109/IECON.2015.7392488

[15] Miller A, Allen PK (2004) Graspit!: A Versatile Simulator for Robotic Grasping. IEEE Robotics and Automation Magazine, 11:110-122. doi: 10.1109/MRA.2004.1371616.

[16] Ramaswamy CVV, Deborah SA (2015) A Survey of Robotic Hand-Arm Systems. International Journal of Computer Applications, 109:26-31. doi:10.5120/19209-0965.

[17] Santello M, Bianchi M, Gabiccini M, Ricciardi E, Salvietti G, Prattichizzo D, Ernst M, Moscatelli A, Jörntell H, Kappers AML, Kyriakopoulos K, Albu-Schäffer A, Castellini C, Bicchi A (2016) Hand synergies: Integration of robotics and neuroscience for understanding the control of biological and artificial hands. Physics of Life Reviews, 17:1-23. doi:10.1016/j.plrev.2016.02.001.

[18] Sarakoglou I, Tsagarakis N, Caldwell D (2014) HANDS.DVI: A DeVice-Independent Programming and Control Framework for Robotic HANDS. Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe: Technology transfer experiments from the ECHORD project. 197-215. doi:10.1007/978-3-319-02934-4_10.

[19] SimLab (2017) Allegro Hand Overview. (Online). http://wiki.wonikrobotics.com/AllegroHandWiki/index.php/Allegro_Hand_Overview. Accessed on 23 July 2017.

[20] Virgala I, Kelemen M, Varga M, Kurylo P(2014) Analyzing, Modeling and Simulation of Humanoid Robot Hand Motion, Procedia Engineering, 96:489-499. doi:10.1016/j.proeng.2014.12.121