MANIPULACIÓN DEL ROBOT UR3 MEDIANTE ROS Y URSIM

MANIPULATION OF THE UR3 ROBOT BY ROS AND URSIM

Hermes Fabián Vargas Rosero
Universidad del Cauca, Colombia
fvargas@unicauca.edu.co

Oscar Andrés Vivas Albán
Universidad del Cauca, Colombia
avivas@unicauca.edu.co

Víctor Fernando Muñoz Martínez
Universidad de Málaga, España
vfmm@uma.es

Recepción: 28/noviembre/2022 **Aceptación:** 16/mayo/2023

Resumen

Este artículo presenta la implementación de un sistema para manipular el robot UR3 de Universal Robots mediante un gamepad y el framework ROS. Se muestra el modelo del robot y las herramientas necesarias para simular paso a paso el robot con el software URSim del fabricante, como paso previo a la manipulación real. Se configura un gamepad para que envíe señales cartesianas deseadas al robot, ya sea simulado o real, a través de ROS. Los resultados muestran una correspondencia entre los movimientos obtenidos del robot en URSim con los movimientos reales del robot UR3.

Palabras Clave: Manipulación de robots, robot UR3, robots colaborativos, ROS, URSim.

Abstract

This article shows the implementation of a system for the manipulation of the Universal Robots UR3 robot, by means of a gamepad and using the ROS framework. The robot model and the necessary tools for the step-by-step simulation of the robot

are shown, using the manufacturer's URSim software, as a previous step to the real manipulation. A gamepad is configured so that it can send the desired Cartesian signals to the robot, simulated or real, via ROS. The results show a correspondence between the movements obtained from the robot in URSim with the real movements of the UR3 robot.

Keywords**:** *Robot manipulation, UR3 Robot, collaborative robots, ROS, URSim.*

1. Introducción

Los robots colaborativos han sido ampliamente aceptados en diversos campos, no solo en la industria, sino también en la medicina, el ámbito militar y los servicios. Aunque los robots convencionales están programados para realizar tareas repetitivas, existen tareas que requieren teleoperación para llevar a cabo funciones específicas, como la manipulación de sustancias peligrosas, la desactivación de bombas, labores remotas y cirugías asistidas por robots. La implementación de un sistema de manipulación remota es especialmente compleja debido a la necesidad de integrar varios sistemas, como el propio robot, la interfaz de manipulación, el controlador, sistemas de visión y otros componentes. Afortunadamente, el Sistema Operativo de Robótica (ROS) ha simplificado de manera significativa la integración de múltiples sistemas en robótica. Aunque los manuales y tutoriales explican los conceptos básicos de ROS, no proporcionan una guía clara para la implementación de una solución de integración utilizando ROS. En investigaciones relacionadas, se ha demostrado que el uso de interfaces de manipulación, como gamepads, ofrece una alternativa viable para la teleoperación de robots. Ejemplos de ello son el sistema de brazo robótico industrial con plataforma móvil controlado por gamepad presentado por [Rahman, 2019] y el sistema controlado por el gamepad inalámbrico PS2 e implementos de bajo costo presentado por [Khaleda, 2017]. Además, en [Mohammadi, 2019], se presenta un innovador sistema de manipulación de brazo robótico mediante una interfaz de lengua inductiva colocada en el paladar, que adapta los comandos de gamepad convencionales a la interfaz. También se ha demostrado que el control de robots industriales mediante gamepad es una alternativa intuitiva, según los resultados presentados en [Wagner, 2016]. En

[Bonaiuto, 2017], se presenta una comparación entre la manipulación de robots utilizando un gamepad, un dispositivo móvil y seguimiento manual.

En este artículo, se busca realizar una contribución al estado actual de la manipulación de robots colaborativos con el objetivo de reducir la curva de desarrollo de sus proyectos. Se propone la implementación de una infraestructura que permita probar la manipulación del robot UR3 de Universal Robots en la máquina virtual URSim, que es un gemelo digital del robot real, para garantizar la fiabilidad del sistema antes de probarlo con el robot real. Además, se ofrece un ejemplo práctico de cómo utilizar ROS como arquitectura de integración y comunicación de diversos elementos en una solución de ingeniería.

Este documento se organiza de la siguiente manera: tras la introducción, se presentan los métodos, que abordan una introducción al sistema operativo ROS, la descripción del robot UR3 y la arquitectura desarrollada. A continuación, se detallan los resultados, que explican paso a paso la comunicación de las señales provenientes del gamepad con el simulador URSim, así como con el robot real.

2. Métodos

En este apartado se presenta la arquitectura del sistema para manipular un robot UR3 utilizando una interfaz de teleoperación de tipo gamepad. El sistema operativo de robótica utilizado es ROS Kinetic, que facilita la integración de equipos como robots, cámaras, dispositivos de control y sensores. El robot colaborativo UR3 de Universal Robots es un robot ultraligero de seis grados de libertad, utilizado en diversas aplicaciones industriales y médicas. Para simular y programar el robot, se emplea el software URSim que provee gemelos digitales de los robots de Universal Robots.

La arquitectura del proyecto se basa en la comunicación mediante tópicos de ROS, conectando un gamepad para leer las posiciones de las palancas y los botones, y un nodo que detecta y publica la posición actual de las articulaciones del robot. Se utiliza la librería ur3 ikfastpy en Python para calcular las posiciones articulares a partir de las posiciones cartesianas y viceversa. Con esta arquitectura, se pueden realizar pruebas de simulación con URSim y, posteriormente, manipular el robot real

utilizando el mismo canal de comunicación. A continuación, se realiza la descripción en detalle de cada componente.

Sistema operativo ROS

ROS es un sistema operativo de robótica que surge con el objetivo de facilitar la integración de diversos equipos, como robots, cámaras, dispositivos de control, sensores, entre otros [Mishra, 2018]. ROS permite interconectar distintas plataformas, ya sean de hardware o software, mediante canales de comunicación llamados tópicos que contienen información de un tipo específico de mensaje. Esta información fluye de manera continua por el canal y es producida por publicadores encargados de difundir datos que circulan por él; estos publicadores pueden estar construidos en diferentes lenguajes de programación y alojarse en distintos dispositivos de hardware. La información que viaja por el canal es escuchada por suscriptores, que pueden ser uno o varios, según lo requiera la aplicación. Lo que tienen en común es estar en la misma red y compartir el nombre del tópico o canal, así como los tipos de mensaje.

Un ejemplo práctico sería el caso de un vehículo autónomo donde hay un tópico con el estado del sensor de proximidad y otro tópico con el sensor de velocidad. Esta información conjunta permite generar datos para el tópico encargado de la potencia y dirección de las ruedas.

En particular, en el presente proyecto se utiliza la versión de ROS Kinetic sobre el sistema operativo Ubuntu 16. ROS Kinetic se descarga e instala según se describe en [Kinetic, 2022], ejecutando los comandos correspondientes desde una terminal en Ubuntu. A continuación, se crea el directorio de trabajo llamado "catkin_ws", se accede al directorio desde una terminal y se ejecuta el comando "catkin_make", que configura y crea archivos y directorios internos, como el subdirectorio "src", donde se copiarán los paquetes que se deseen crear. Para compilar dichos paquetes, se utiliza nuevamente "catkin_make"; las instrucciones detalladas se encuentran en [ROS-Tutorials, 2022]. En este proyecto, se emplea el paquete ros-industrial/ur_modern_driver [Modern driver, 2022], que se copia y compila en la carpeta catkin_ws/src. Una vez compilado el paquete, se abre una nueva terminal

con las teclas (Ctrl + Alt + T) y se ejecuta el siguiente comando: *roslaunch ur_modern_driver ur3_bringup.launch robot_ip:=192.168.xx.xxx.*

Este comando permite establecer la conexión de ROS con un equipo, ya sea la máquina virtual donde se simulará el robot o el robot real, indicando la dirección IP correspondiente. Este comando también habilita los tópicos que permiten comunicarse con el robot, ya sea para conocer el estado de las articulaciones o enviar valores articulares para que el robot se mueva.

Por ejemplo, una vez realizada la conexión con el robot, si el usuario desea conocer la posición de las articulaciones en ese momento, puede abrir otra terminal de comandos y ejecutar el comando echo sobre el tópico /joint_states, tal como se indica a continuación: *rostopic echo /joint_states*.

En la terminal aparecerá la posición de cada articulación, y si el robot se mueve, la posición mostrada se actualizará. Como en toda terminal, para detener el comando, se utilizan las teclas (Ctrl + C).

Robot colaborativo UR3 de Universal Robots

Los robots colaborativos, a diferencia de los robots industriales convencionales, amplían la gama de aplicaciones de la robótica en la industria, la medicina y la educación. Además, cuentan con sensores y sistemas de seguridad, lo que elimina la necesidad de una jaula de protección, como en el caso de los robots industriales. El robot UR3 (Figura 1), es un robot colaborativo ultraligero de la marca Universal Robots, con seis grados de libertad. Está diseñado para aplicaciones de montaje, agarre y colocación de piezas, así como acciones de atornillado. Posee sensores de fuerza que bloquean el movimiento para evitar posibles riesgos con los operarios cercanos. El robot puede soportar una carga de hasta 3 kilogramos y un espacio de trabajo de 500 mm desde su base. Para programar este tipo de robots, se utiliza una tableta especial llamada Polyscope, en la que el usuario puede desplazar las articulaciones y grabar las posiciones que posteriormente el robot repetirá. Alternativamente, también se pueden usar herramientas con representación 3D para la programación de trayectorias, como Rviz [Kam, 2015], MoveIt [Görner, 2019] y Gazebo [Koenig, 2004].

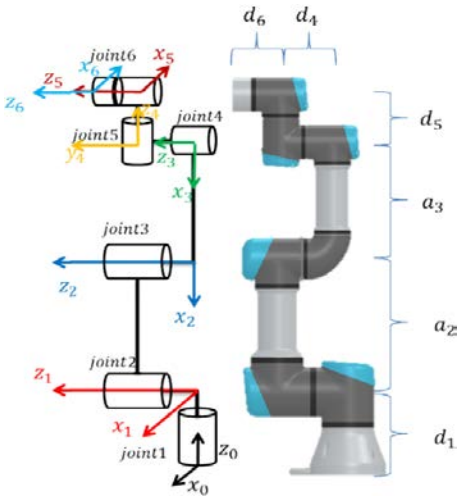

Figura 1 Robot UR3 [Abdelaziz, 2019].

La tabla 1 muestra los parámetros que definen la estructura del robot UR3. El parámetro ai es la distancia desde el eje Zi−1 hasta el eje Zi; la distancia di es la medida de la intersección de la perpendicular común entre Zi−1 y Zi; el ángulo alpha i es el giro que forman los ejes Xi−1 y Xi con respecto al eje Zi−1. La tabla 1 permite obtener la cinemática directa, que encuentra la posición del extremo del robot a partir de las posiciones articulares; de igual manera, permite definir la cinemática inversa, con la cual se obtiene la posición de cada articulación para una determinada posición cartesiana deseada del extremo del robot. Normalmente, para garantizar la integridad del equipo, al calcular la cinemática inversa, se consulta la posición actual del robot y luego se selecciona la configuración más cercana a dicha posición actual, lo que garantiza que el robot no tenga movimientos indeterminados. Para los cálculos de la cinemática del robot, se utiliza la librería ur3 ikfastpy, construida en Python. Esta librería se descarga e instala desde [Cambel, 2019].

Tabla 1 Parámetros DH UR3.

Joint i	theta (rad)	a (m)	d (m)	alpha (rad)
1	0	0	0.1519	π/2
2	0	-0. 243	0	0
3	0	-0.2132	0	0
4	0	0	0.1133	π/2
5	0	0	0.0855	-π/2
6	0	0	0.0819	0

Gemelo digital de Universal Robots URSim

URSim es un software de simulación proporcionado por Universal Robots para la programación fuera de línea y la simulación del comportamiento de sus robots. Específicamente, URSim ofrece gemelos digitales de los robots de la marca, permitiendo la prueba de algoritmos, trayectorias y controladores. Después de simular la fiabilidad de una tarea, solo se requiere cambiar la dirección IP utilizada en la máquina virtual donde se ejecutó URSim por la IP del robot real, donde se debería obtener el mismo comportamiento que en la simulación.

La máquina cuenta con el software que simula el robot preinstalado en Ubuntu 14. Para utilizarlo, se debe descargar el disco duro virtualizado que contiene la máquina desde [UR-Offline-Sim, 2022], luego instalar Virtualbox, crear una nueva máquina y asociar el disco duro mencionado anteriormente. En la máquina, se encontrará el icono asociado que ejecutará el simulador del robot. No requiere configuraciones adicionales, excepto la dirección de red que se describe más adelante.

En el entorno de URSim, se puede desplazar el robot de manera articular o cartesiana, como se observa en la figura 2. Esta herramienta permite representar gran parte del comportamiento de un robot real; sin embargo, las condiciones de fuerza y colisiones con el entorno no se pueden considerar.

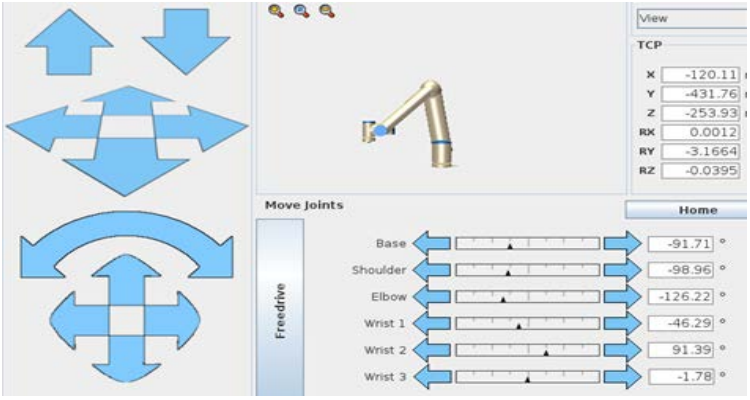

Figura 2 Interfaz de usuario de URSim.

Para utilizar URSim mediante ROS, es necesario instalar el complemento URCaps correspondiente al controlador instalado en ROS. En el caso del ur_modern_driver, se utiliza el archivo externalcontrol-1.0.5.urcap, que se puede descargar de

[UR_driver, 2022]. Para instalar este complemento, se debe abrir la interfaz del robot en URSim, ingresar en la opción de configuración del robot, en la sección de URCaps, donde se añade el archivo mencionado. En el caso del robot real, el proceso es similar, pero se realiza en la tableta de control.

Continuando con la configuración del sistema, en la ventana inicial de URSim o en la interfaz del robot real, se ingresa en la opción “Programar robot”. Posteriormente, se accede a la ventana de “External Control”, como se muestra en la figura 3. Se observa el campo “Host IP”, donde se debe colocar la dirección IP del equipo utilizado para manipular el robot, mientras que en el campo “Custom port” se coloca en 50001.

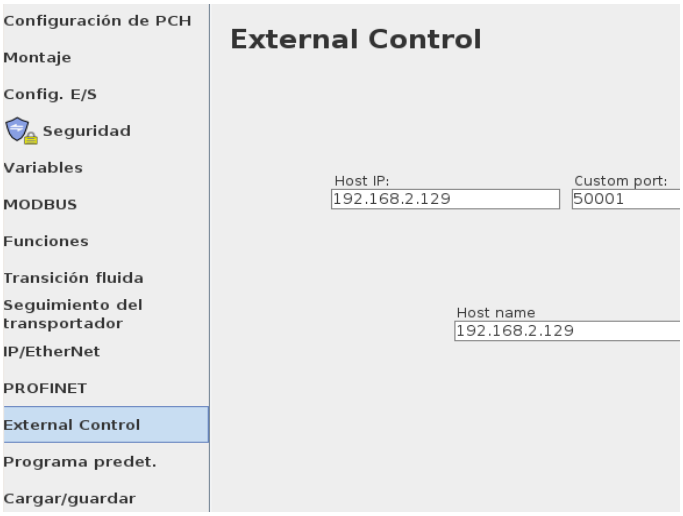

Figura 3 Configuración de la IP del equipo.

Para finalizar la configuración de URSim/robot, es necesario garantizar que se encuentre en la red de trabajo. En el caso de la máquina URSim, se pueden modificar los parámetros de red abriendo una terminal y ejecutando los siguientes comandos: *sudo ifconfig eth0 192.168.xxx.xxx netmask 255.255.255.0.*

La dirección IP 192.168.xxx.xxx corresponde a la dirección IP que tendrá la máquina virtual. Posteriormente, se ejecuta en la misma terminal el siguiente comando: *sudo route add default gw 192.168.xxx.xxx eth0.*

La dirección 192.168.xxx.xxx eth0 corresponde al host donde se aloja la máquina URSim. Si todo está configurado correctamente, se puede realizar una prueba para

verificar la conexión. Para esto, se abre una terminal en el equipo y se ejecuta el siguiente comando. En respuesta a él, el robot deberá moverse a través de unas trayectorias predefinidas. */catkin_ws/src/ur_modern_driver-master$ python test_move.py.*

En resumen, URSim es una herramienta valiosa para simular el comportamiento de los robots de Universal Robots, permitiendo la programación fuera de línea y la prueba de algoritmos, trayectorias y controladores. Al seguir estos pasos de configuración, se puede establecer una conexión entre URSim y ROS, lo que facilita la transición entre la simulación y el control del robot real en aplicaciones prácticas.

Arquitectura desarrollada

La figura 4 muestra la arquitectura del proyecto que permite manipular el robot UR3 mediante una interfaz de teleoperación tipo gamepad. Inicialmente, se probará la simulación de los movimientos del robot con el software URSim y, posteriormente, se aprovecharán las capacidades de ROS para utilizar el mismo canal en la manipulación del robot real. Para determinar la posición de cada articulación, se utiliza la librería de Python ur3 ikfastpy, en la que se ejecutan los algoritmos de cinemática inversa para conocer los valores articulares que permiten dirigir el robot a una posición específica.

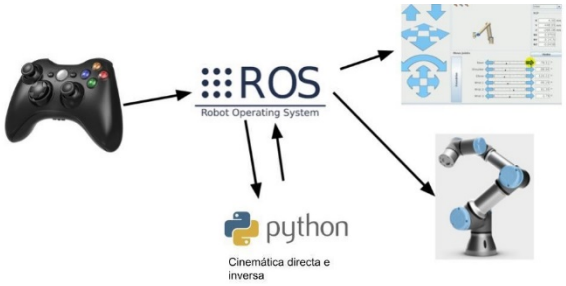

Figura 4 Elementos para la teleoperación del robot UR3.

La figura 5 presenta el diagrama de flujo general para la manipulación del robot. Gracias a la arquitectura de comunicación ROS, el gamepad se conecta mediante un nodo que publica las posiciones leídas de las palancas y los botones. Dado que las palancas solo tienen dos ejes, se utiliza un botón para discriminar los

movimientos de traslación y de rotación. Mientras tanto, se conecta un nodo al robot para que detecte la posición actual de las articulaciones y las publique en un tópico ROS. Las posiciones articulares son escuchadas desde un suscriptor en Python, cuya función realiza la transformación de posiciones articulares a posiciones cartesianas mediante un modelo geométrico directo asociado. El resultado es una matriz de 4x4 que contiene la matriz de orientación y el vector de posiciones x, y, z.

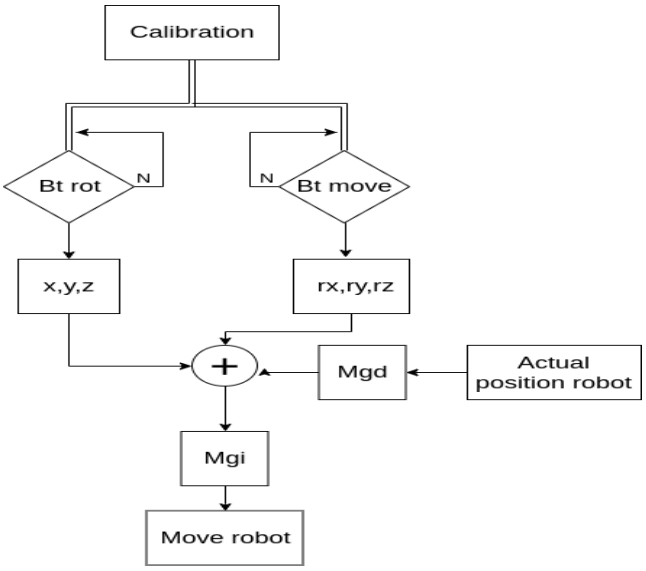

Figura 5 Diagrama de flujo del sistema.

El sistema suma la posición actual del robot con las variaciones de las palancas del gamepad, obteniendo así el desplazamiento de la posición cartesiana. Posteriormente, se transforman estas posiciones a variables articulares mediante el modelo geométrico inverso, lo que permite obtener las posiciones de cada articulación. Dichas posiciones articulares son enviadas a un tópico que es escuchado por el nodo del robot, donde se mueve cada articulación hasta la posición deseada por el usuario.

Como se puede inferir a partir del diagrama de flujo, se considera la posición cartesiana actual del robot y, con las posiciones en los ejes x, y, z, se suman las variaciones en las palancas del gamepad a la posición en cada eje. De esta manera, el movimiento inicia desde la posición original del robot, evitando movimientos diferentes al deseado.

Arquitectura funcional desde el enfoque ROS

ROS es una plataforma que permite la integración de múltiples equipos gracias a la comunicación mediante tópicos. La figura 6 muestra la arquitectura funcional del sistema de manipulación del robot representada desde el enfoque de ROS. Las líneas paralelas con extremos punteados representan los tópicos, que siempre se escriben anteponiendo una barra inclinada.

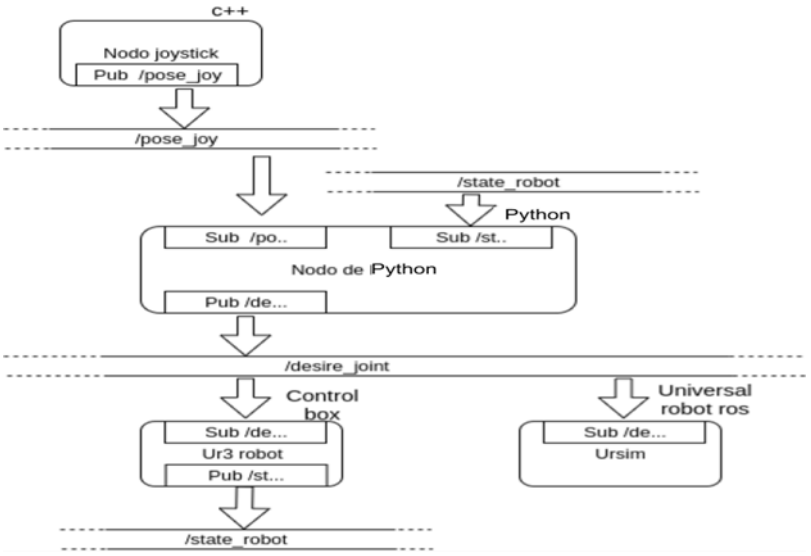

Figura 6 Arquitectura funcional desde el enfoque ROS.

En la figura 6, se observan cuatro tipos de tópicos distintos. Además, se presenta un nodo escrito en C++ encargado de la conexión del gamepad y de publicar el estado de las palancas y botones. También se observa un nodo en Python que posee dos suscriptores para escuchar el estado actual del robot y la posición del gamepad. Dentro de las funciones internas del nodo, se utiliza la cinemática directa y la cinemática inversa para establecer la posición de las articulaciones del robot, y posteriormente se publica la posición articular deseada. En la máquina virtual URSim, también se ejecuta un nodo para escuchar la posición articular deseada y así desplazar el robot virtual. Cabe destacar que, dado que el robot real puede suscribirse al mismo tópico de posiciones, después de calibrar los movimientos y garantizar la seguridad en la máquina virtual, se procede a habilitar los movimientos en el robot real.

3. Resultados

En los apartados anteriores se detalló cómo instalar y configurar ROS en Ubuntu. En este proyecto, se han utilizado paquetes adicionales externos y el paquete propio para la tarea de manipulación del robot. En el enlace [Vargas, 2022] se pueden descargar dichos paquetes, los cuales deben ser copiados en la carpeta "catkin_ws" y compilados con "catkin_make" como se describió anteriormente. Los archivos descargados son los siguientes:

- Paquete de manejo del joystick o gamepad.
- Paquete de drivers ROS para Universal Robots.
- Plugin de conexión externa con la máquina URSim.
- Paquete con archivos de control en Python.

Para ejecutar el sistema, se requiere iniciar los paquetes respectivos en diferentes terminales, así como ejecutar el programa de control que está en Python. En las siguientes líneas se describen los comandos que se deben ejecutar en diferentes terminales, paso a paso:

- Terminal T1: Conexión con el robot. *roslaunch ur_modern_driver ur3_bringup.launch robot_ip:=192.168.x.xxx.*
- Terminal T2: Detección del gamepad. *cd catkin_ws/; source devel/setup.bash; roslaunch smat_surg_s2_v1 smat_surg_.launch.*
- Terminal T3: Administrador que ejecuta la lógica de control. *cd ~/catkin_ws/src/scripts; python smart_actionlib_v4.py.*
- Una vez habilitado el gamepad, se procede a mover las palancas y verificar el movimiento. En primer lugar, se realiza la prueba con la máquina virtual, donde se confirma el desplazamiento deseado.

En la figura 7 se representa el robot en la máquina virtual URSim, después de haber seguido la trayectoria descrita por el trazo de color verde. El punto rojo indica el punto de inicio del recorrido y la línea verde discontinua indica la trayectoria.
Luego de probar que el movimiento se realiza de manera adecuada en la máquina virtual, se procede a corroborar el movimiento en el robot real, como se muestra en

la figura 8. Aunque aún no se ha realizado un estudio detallado de la correspondencia entre un movimiento y otro, se aprecia inicialmente que los movimientos son iguales.

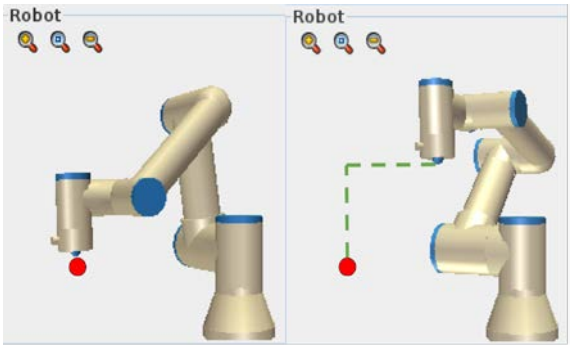

Figura 7 Desplazamiento del robot en URSim.

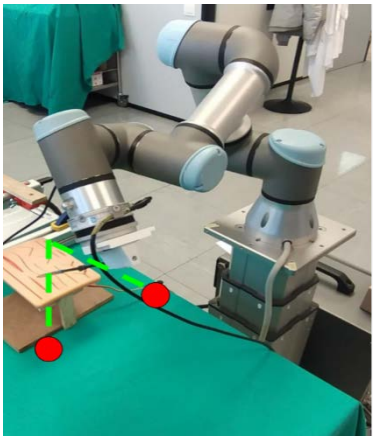

Figura 8 Desplazamiento del robot real.

El estudio del desplazamiento del robot es importante para definir su eficiencia. El paquete "ur_modern_driver" dispone del tópico "/tf", en el cual se publica la posición del órgano terminal del robot. Si se desea graficar el recorrido que ha realizado el robot, es necesario escuchar la información de posición del tópico "/tf" y luego procesarla para su posterior graficación. Esta tarea se puede realizar con el siguiente comando, el cual guarda las posiciones del efector final en un archivo:

rostopic echo /tf/transforms[0]/transform/translation | tee archivo.yaml.

Los datos guardados en archivo.yaml se pueden abrir en cualquier editor de texto, obteniéndose lo siguiente: *x: 0.315, y: 0.281, z: 0.289, x: 0.315, y: 0.281.*

Una vez realizada la configuración correspondiente, se procede a manejar el robot a través del joystick del gamepad. En la figura 9 se puede apreciar la trayectoria recorrida por el robot desde una perspectiva superior. Se puede observar que la línea azul indica el recorrido que ha seguido el robot, el cual incluye secciones con recorridos rectos y otras secciones con movimientos curvos. Cada uno de estos movimientos fueron ejecutados mediante el desplazamiento del joystick según la voluntad del usuario.

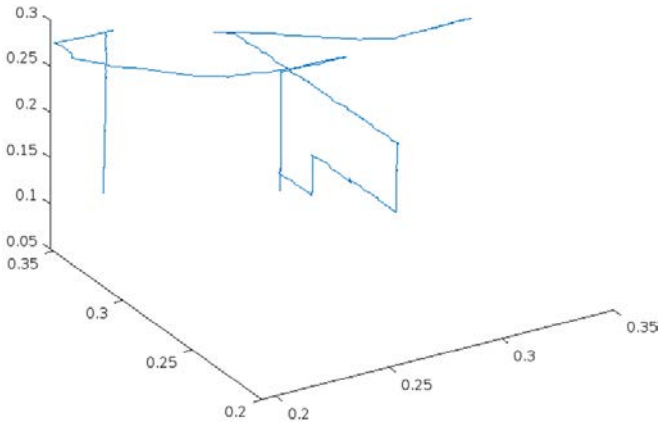

Figura 9 Trayectoria recorrida por el robot, perspectiva.

4. Discusión

La implementación propuesta demuestra la viabilidad de utilizar el framework ROS y un gamepad para controlar y manipular el robot UR3 de manera efectiva. La simulación en URSim permite probar y garantizar la fiabilidad del sistema antes de realizar pruebas con el robot real. El uso de interfaces de manipulación como gamepads facilita la teleoperación de robots y se ha demostrado en trabajos relacionados como una alternativa viable e intuitiva. La propuesta contribuye al estado de la manipulación de robots colaborativos y ayuda a reducir la curva de desarrollo de proyectos similares.

5. Conclusiones

En este trabajo se presenta una descripción detallada del proceso de manipulación del robot UR3 de Universal Robots. Para enviar las señales

cartesianas deseadas al robot, se utiliza un gamepad. Todo el sistema está soportado en el framework ROS, lo que permite la interconexión tanto del gamepad como del robot real. Como paso previo antes de la ejecución del robot real, se utiliza el software URSim para simular el comportamiento del robot ante las consignas recibidas por el gamepad. Los resultados obtenidos muestran un comportamiento similar entre la simulación en URSim y el robot real. En futuros trabajos se implementarán consignas más complejas para ser probadas inicialmente en el simulador y luego ser ejecutadas por el robot real. Además, se pretende verificar la exactitud de las respuestas comparando la simulación con el robot real.

6. Bibliografía y Referencias

[1] Abdelaziz, O., Luo, M., Jiang, G., & Chen, S., (2019). Multiple configurations for puncturing robot positioning. https://doi.org/10.48550/arXiv.1903.02281.

[2] Bonaiuto, S., Cannavò, A., Piumatti, G., Paravati G. and Lamberti, F., (2017). Tele-operation of Robot Teams: A Comparison of Gamepad-, Mobile Device and Hand Tracking-Based User Interfaces. 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), 2017, pp. 555-560, doi: 10.1109/COMPSAC.2017.278.

[3] Cambel A. Kinematic of Universal Robots on Python, (2022). ur_ikfast. https://github.com/cambel/ur_ikfast (Original work published 2019).

[4] Görner, M., Haschke, R., Ritter, H., & Zhang, J., (2019). MoveIt! Task Constructor for Task-Level Motion Planning. 2019 International Conference on Robotics and Automation (ICRA), 190–196. https://doi.org/10.1109/ICRA.2019.8793898.

[5] Kam, H. R., Lee, S.-H., Park, T., & Kim, C.-H., (2015). RViz: a toolkit for real domain data visualization. Telecommunication Systems, 60(2), 337–345. https://doi.org/10.1007/s11235-015-0034-5.

[6] Koenig, N., & Howard, A., (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), 3, 2149–2154 vol.3. https://doi.org/10.1109/IROS.2004.1389727.

[7] Kinetic/Installation/Ubuntu - ROS Wiki. (2022). Retrieved June 21, 2022, from http://wiki.ros.org/kinetic/Installation/Ubuntu.

[8] Khaleda, Sh., Wassan, R., (2017) Wireless Mobile Robotic Arm Controlled by PS2 Joystick Based on Microcontroller. Diyala Journal of Engineering Sciences. 10. 44-53. 10.24237/djes.2017.10304.

[9] Knudsen, M., & Kaivo-oja, J., (2020). Collaborative Robots: Frontiers of Current Literature. Journal of Intelligent Systems: Theory and Applications, 3(2), 13–20. https://doi.org/10.38016/jista.682479.

[10] Kragic, D., Gustafson, J., Karaoguz, H., Jensfelt, P., & Krug, R., (2018). Interactive, Collaborative Robots: Challenges and Opportunities. 18–25. https://www.ijcai.org/proceedings/2018/3.

[11] Mishra, R., & Javed, A., (2018). ROS based service robot platform. 2018 4th International Conference on Control, Automation and Robotics (ICCAR), 55–59. https://doi.org/10.1109/ICCAR.2018.8384644.

[12] Mohammadi, M., Knoche, H., Gaihede, M., Bentsen B. and Andreasen L., (2019). A high-resolution tongue-based joystick to enable robot control for individuals with severe disabilities. 2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR), 2019, pp. 1043-1048, doi: 10.1109/ICORR.2019.8779434.

[13] Rahman, R., Rahman, M., and Bhuiyan, J. R., (2019). Joystick controlled industrial robotic system with robotic arm. 2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON), 2019, pp. 31-34, doi: 10.1109/RAAICON48939.2019.18.

[14] ROS/Tutorials - ROS Wiki., (2022). Retrieved June 16, 2022, from http://wiki.ros.org/ROS/Tutorials.

[15] Sherwani, F., Asad, M. M., & Ibrahim, B. S. K. K., (2020). Collaborative Robots and Industrial Revolution 4.0 (IR 4.0). 2020 International Conference on Emerging Trends in Smart Technologies (ICETST), 1–5. https://doi.org/10.1109/ICETST49965.2020.9080724.

[16] Universal Robots - Offline Simulator - CB-Series - Non Linux - URSim 3.14.3., (2022). Retrieved June 21, 2022, from https://www.universal-robots.com

/download/software-cb-series/simulator-non-linux/offline-simulator-cb-series-non-linux-ursim-3143/.

[17] Universal_Robots_ROS_Driver, (2022). Universal Robots A/S. https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/696cbb334a0b76a2d18e26994adc158768ddba0b/ur_robot_driver/doc/install_urcap_e_series.md (Original work published 2019).

[18] UR modern driver, (2022). ROS-Industrial. https://github.com/ros-industrial/ur_modern_driver (Original work published 2015).

[19] Vargas F. Control de robot UR3 mediante un gamepad, (2022). https://github.com/fabiavargasr/gamepad2robotur3 (Original work published 2022).

[20] Wagner, M., et al., (2016). Gamepad Control for Industrial Robots - New Ideas for the Improvement of Existing Control Devices." ICINCO (2016).

[21] Whitney, D., Rosen, E., Ullman, D., Phillips, E., & Tellex, S., (2018). ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1–9. https://doi.org/10.1109/IROS.2018.8593513.

[22] Zhi, L., & Xuesong, M., (2018). Navigation and Control System of Mobile Robot Based on ROS. 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 368–372. https://doi.org/10.1109/IAEAC.2018.8577901.