XLIV Jornadas de Automática 2023

https://jautomatica.es

Simulación y control de robots en aplicaciones de servicio en órbita mediante OnOrbitROS

Ramón, J. L.[a], Pomares, J.[a, *], Felicetti, L.[b]

[a] *Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, Ctra San Vicente s/n, 03690 Alicante, España.*
[b] *Cranfield University, Cranfield, MK43 0AL, Reino Unido.*

To cite this article: Ramón, J. L., Pomares, J., Felicetti, L. 2023. Simulation and control of on orbit servicing robots using OnOrbitROS. XLIV Jornadas de Automática, 465-470.
https://doi.org/10.17979/spudc.9788497498609.465

Resumen

Actualmente, la utilización de herramientas de simulación como ROS/Gazebo es una práctica habitual para la simulación y desarrollo de algoritmos de control para sistemas robóticos terrestres típicos. Sin embargo, en la actualidad aún no está comúnmente aceptado en la comunidad espacial. Numerosos estudios en el campo de la robótica espacial utilizan herramientas construidas ad-hoc, pero que no están estandarizadas, que no son de código abierto y que, además, no han sido verificadas, lo que complica el desarrollo y la simulación de sistemas robóticos versátiles y algoritmos para el control de sistemas robóticos espaciales. Este artículo propone una solución de código abierto para la simulación de robots espaciales en órbita denominada OnOrbitROS. Este trabajo presenta una descripción de la arquitectura, los diferentes módulos software y las posibilidades de simulación de OnOrbitROS. Se describen las características clave de la herramienta desarrollada, con especial atención a la personalización de las simulaciones y a las posibilidades de ampliación de la herramienta. Con el fin de mostrar estas capacidades, se describen los resultados obtenidos al aplicar OnOrbitROS para la simulación del robot ETS-VII.

Palabras clave: robótica espacial, simulación, control de robots, robótica.

Simulation and control of on orbit servicing robots using OnOrbitROS

Abstract

Currently, the use of simulation tools such as ROS/Gazebo is a common practice for the simulation and development of control algorithms for typical terrestrial robotic systems. Numerous studies in the field of space robotics use ad-hoc built tools, but they are not standardized, and are unverified, which complicates the development and simulation of versatile robotic systems. This paper proposes an open-source solution for the simulation of orbiting space robots called OnOrbitROS. This work presents a description of the architecture, the different software modules and the simulation possibilities of OnOrbitROS. The key features of the developed tool are described, with special attention to the customization of the simulations and the possibilities of extension of the tool. In order to show these capabilities, the results obtained in the simulation of the robot ETS-VII by using OnOrbitROS are described.

Keywords: space robotics, simulation, robot control, robotics.

1. Introducción

Las misiones espaciales futuras, incluso las que se encuentran en desarrollo, requerirán el uso de robots en tareas tales como la eliminación de basura espacial, y tareas de servicio como ensamblaje o fabricación en órbita. El uso de ROS en el espacio comenzó con Robonaut 2, desarrollado por la NASA y General Motors. Éste fue el primer robot humanoide en la ISS (Diftler *et al.*, 2011). La NASA también utiliza ROS en otros robots espaciales, como Astrobee (Nolet,

*Autor para correspondencia: autor1@ceautomatica.es

2007; Bualat *et al*., 2018). Ninguno de estos robots se probó en un escenario de actividad extra vehicular, pero las futuras misiones robóticas, como Valkyrie de la NASA, tendrán una autonomía mejorada y se probarán fuera de la ISS (Radford *et al.*, 2015). Cabe mencionar otros desarrollos basados en Gazebo que permiten simular las condiciones del espacio y los planetas, como el terreno lunar (Allan *et al.*, 2018). Además, en la actualidad existen varias herramientas y librerías de código abierto para resolver diferentes problemas en el espacio. Por ejemplo, en (Wang et al., 2022) se presenta un algoritmo de optimización de desacoplamiento jerárquico libre de modelos para realizar la planificación de trayectorias multiobjetivo en 6D para el robot espacial de flotación libre.

El desarrollo y test de estos sistemas a menudo requieren una aproximación iterativa que, junto con las dificultades técnicas de reproducir las condiciones en el espacio utilizando los recursos existentes en la tierra, y los altos costes asociados con estos test, desaconsejan la utilización de aproximaciones basadas en hardware en la primera fase de diseño de estas misiones. Herramientas de simulación como las que ofrece ROS/Gazebo son actualmente una aproximación muy común para el desarrollo y test de algoritmos de control y guiado para robots típicos terrestres, pero aún en la actualidad esta opción no es aceptada como válida por la comunidad investigadora en tecnología espacial. Una de las razones de esto es que este tipo de entornos de desarrollo no permiten una simulación completa y realista de las condiciones en el espacio como pueden ser las condiciones de microgravedad y la ausencia de fricción. Existen numerosos estudios (Seddaoui, 2021) en el campo de la robótica espacial que utilizan soluciones ad-hoc para la simulación y validación de los sistemas de control, pero son soluciones no estándar. No utilizan código abierto y, a veces, se utilizan herramientas no verificadas que complican, en lugar de promover, el desarrollo y la realización de algoritmos y sistemas robóticos versátiles para la robótica espacial.

Teniendo en cuenta todos estos aspectos que definen la robótica espacial, la solución propuesta en este artículo ha sido la de desarrollar una herramienta de código abierto para la simulación de sistemas robóticos en órbita. Esta herramienta, denominada OnOrbitROS se ha publicado en (Ramón, 2023). El framework utilizado se basa en ROS, un meta sistema operativo de código abierto desarrollado específicamente para el desarrollo de aplicaciones de robótica. Habitualmente se combina ROS con Gazebo como herramienta para simular robots en este tipo de entornos. Sin embargo, estas herramientas han sido modificadas para incluir y reproducir las principales condiciones que los robots y manipuladores pueden experimentar en un escenario de servicio en órbita. Esta solución permite la simulación de complejos sistemas robóticos y, al mismo tiempo, aprovechar el gran número de paquetes software ya desarrollados en ROS para control, visión, teleoperación y modelado. De esta manera, es posible realizar la simulación de robots llevando a cabo tareas de servicio en órbita sin la necesidad de realizar código ad-hoc para la simulación y utilizando una herramienta validada ampliamente, siguiendo, de esta manera, el principio de ROS de "no reinventar la rueda".

Este artículo presenta una descripción de la arquitectura y de los diferentes módulos software que la componen. Se muestran las principales características de la herramienta desarrollada prestando especial atención a las posibles extensiones de la herramienta para incluir librerías de terceros.

El resto del artículo se estructura de la siguiente manera: en el Apartado 2 se describe la arquitectura de OnOrbitROS. Tras presentar esta arquitectura se describen en detalle algunos de sus módulos más importantes. Así, en el Apartado 4 se describe el paquete que realiza la simulación de la trayectoria orbital del robot simulado (Orbit), y en el Apartado 5 se describe el paquete OORplugin que simula fundamentalmente las condiciones de microgravedad y gradiente orbital. Previamente, en el Apartado 3 se habrán descrito los sistemas de referencia que se emplean en la definición de OnOrbitROS. El artículo finaliza describiendo los resultados obtenidos al simular el robot ETS-VII utilizando OnOrbitROS (Apartado 6) y las principales conclusiones que se pueden extraer del desarrollo e implementación de la herramienta.

2. Arquitectura de OnOrbitROS

OnOrbitROS pretende ser el punto de partida del estudio y desarrollo de aplicaciones robóticas de servicio en órbita. El diseño de su arquitectura (ver Figura 1) se ha centrado en explotar las enormes posibilidades que ofrece como plataforma de pruebas y simulación hiperrealistas la combinación de ROS y Gazebo. Cabe destacar que ROS y Gazebo, son de código abierto y que son una de las plataformas más utilizadas en el desarrollo y simulación de aplicaciones robóticas, tanto en investigación por parte de la comunidad científica, como en proyectos de robótica industrial y de servicios. Sin embargo, se ha de tener en cuenta que todas estas aplicaciones modelan robots que trabajan en la superficie terrestre. Por este motivo se debe, por un lado, modificar la configuración de las propiedades que así lo permitan, al tiempo que se implementan las necesarias para simulación de aplicaciones de robótica de servicios en órbita. De este modo se han desarrollado los paquetes necesarios que cubren los huecos de la dinámica propia de aplicaciones robóticas espaciales.

La Figura 1 muestra la estructura principal de la plataforma de simulación OnOrbitROS. En ella se pueden apreciar los dos paquetes principales que, junto con ROS y Gazebo, forman la plataforma. El primero de ellos, que aparece en azul, es el paquete Orbit. La función principal de este paquete será la de generar las distintas trayectorias en función del tiempo de las naves espaciales, satélites, robots y objetos presentes en la simulación.

El segundo paquete, OORplugin, que aparece en naranja en el esquema, es el encargado de modificar y extender la funcionalidad de los motores de física del simulador para que se ajusten a las condiciones de una aplicación de OOS. También será el encargado de aplicar los pares y fuerzas propios de este tipo de aplicaciones, tales como el gradiente de gravedad, fuerzas debidas al desplazamiento relativo al sistema de referencia de la órbita descrita, etc.

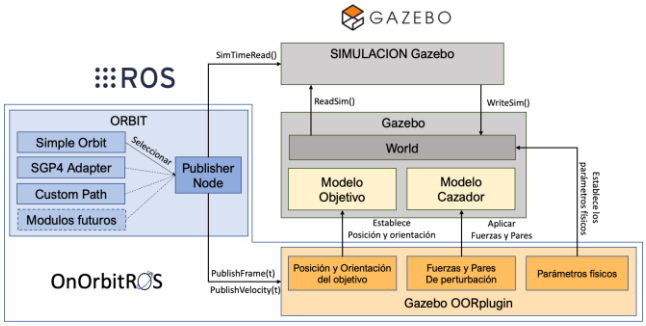

Figura 1: Paquetes principales de OnOrbitROS.

3. Principales sistemas de referencia

Para la descripción de los distintos módulos es necesario definir qué sistemas de referencia se van a emplear y las transformaciones entre ellos. Una vez más se va a emplear una aplicación típica de robótica de servicios en órbita para definir qué sistemas serán necesarios en su simulación. En este tipo de aplicaciones se dispone de dos aeronaves: una que será la que recibe el servicio que es conocida como objetivo, y la que realiza ese servicio que se denominará cazador. En esta última será donde está instalado el manipulador o será un robot en sí misma. El sistema de referencia que representa la posición y orientación de la nave objetivo es $\boldsymbol{F}_{\mathrm{t}}$ y el que representa la posición y orientación de la base de la nave espacial cazador, es $\boldsymbol{F}_{\mathrm{c}}$. En el caso en el que el cazador, sea un robot humanoide, normalmente el sistema de referencia estará situado en el torso de éste. Como sistema de referencia inercial se utilizará el ECI que recibe su nombre de las siglas en ingles de "Earth Center of Inertia". El ECI tiene su origen en el centro de la Tierra, con su eje $\boldsymbol{x}$ que va desde el centro de la Tierra pasando por el equinoccio vernal en la línea del ecuador. Su eje $\boldsymbol{z}$ coincide con el eje de rotación de la Tierra apuntando hacia el polo norte. Finalmente, el eje $\boldsymbol{y}$ simplemente completa la terna ortogonal. El sistema de referencia ECI estará representado por $\boldsymbol{F}_{\mathrm{i}}$. El sistema de referencia que describe la posición teórica que describe la órbita que sigue la nave objetivo también conocida como Coordenadas de Navegación Local será $\boldsymbol{F}_{\mathrm{l}}$. Este sistema de referencia es comúnmente llamado LVLH de las siglas en ingles de "Local-Vertical-Local-Horizontal". En la definición de los ejes del sistema de referencia existen varias convenciones. En este artículo, por simplicidad, se ha adoptado la que define que el eje $\boldsymbol{x}$ corresponde con la dirección del vector radial que va desde el centro de la tierra hasta la nave o satélite. El eje $\boldsymbol{y}$ junto con el eje $\boldsymbol{x}$ forman el plano orbital que contiene la trayectoria descrita por el satélite, y apunta en la dirección en la que se desplaza el satélite. Por último, el eje $\boldsymbol{z}$ completa el sistema ortogonal y su dirección es normal al plano orbital. Como se puede observar en la Figura 2, $\boldsymbol{R}_{\mathrm{l}}$ corresponde con la matriz de rotación y $\boldsymbol{t}_{\mathrm{l}}$ la de traslación de $\boldsymbol{F}_{\mathrm{l}}$ respecto de sistema de referencia inercial, $\boldsymbol{F}_{\mathrm{i}}$. Ambas matrices son calculadas por el módulo Orbit Publisher Node que aparece en azul en el esquema de la Figura 1.

El entorno de simulación Gazebo tiene su propio sistema de referencia que se denominará $\boldsymbol{F}_{\mathrm{g}}$. Para las simulaciones inicialmente se podría considerar que la opción más sencilla sería hacer coincidir los sistemas de referencia $\boldsymbol{F}_{\mathrm{g}}$ y $\boldsymbol{F}_{\mathrm{i}}$, pero teniendo en cuenta que para robótica de servicios en órbita se consideran distancias de cientos de kilómetros entre $\boldsymbol{F}_{\mathrm{l}}$ y $\boldsymbol{F}_{\mathrm{i}}$ no resulta la mejor opción. Por lo que se adopta que $\boldsymbol{F}_{\mathrm{g}}$ coincida en orientación con $\boldsymbol{F}_{\mathrm{i}}$ pero se entiende que su origen es igual al de $\boldsymbol{F}_{\mathrm{l}}$. De modo que se puede decir que $\boldsymbol{F}_{\mathrm{g}}$ esta desplazado $\boldsymbol{t}_{l}$ respecto a $\boldsymbol{F}_{\mathrm{i}}$. De este mismo modo se puede afirmar que la orientación de los sistemas de referencia $\boldsymbol{F}_{\mathrm{l}}$ y $\boldsymbol{F}_{\mathrm{g}}$ está relacionada por la matriz de rotación $\boldsymbol{R}_{\mathrm{l}}$.

Para concluir, queda definir el sistema de referencia de la nave cazadora, que se denominará $\boldsymbol{F}_{\mathrm{c}}$. La posición y orientación de $\boldsymbol{F}_{\mathrm{c}}$ respecto a $\boldsymbol{F}_{\mathrm{g}}$ puede obtenerse en base a las matrices de rotación y traslación ${}^{\mathrm{g}}\boldsymbol{R}_{\mathrm{c}}$ y ${}^{\mathrm{g}}\boldsymbol{t}_{\mathrm{c}}$ tal y como puede observarse en la Figura 2.

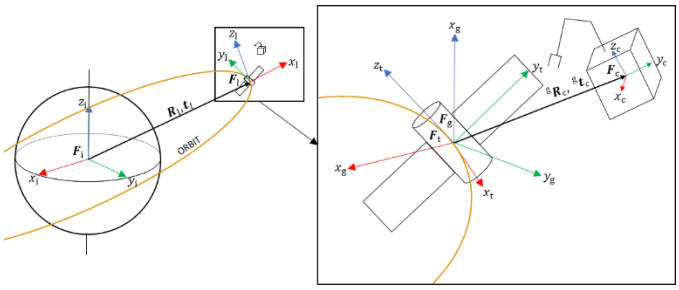

Figura 2: Sistemas de referencia empleados OnOrbitROS

4. El paquete Orbit

En este apartado se definirán los módulos que componen el paquete Orbit, que está representado en azul en la Figura 1 y que puede verse con más en detalle en la Figura 3. Está implementado como un nodo de ROS, y su función principal es la de publicar las trayectorias descritas por los cuerpos que componen la simulación. Los cuerpos pueden describir órbitas simples generadas con Simple Orbit, órbitas más complejas en las que se tiene en cuenta mayor número de perturbaciones y hacen uso de la biblioteca externa SGP4 Adapter, o trayectorias que no corresponden a una órbita, implementadas en Custom Path. Será el usuario el que seleccione el tipo de órbita o trayectoria que más se ajuste para el elemento que desea modelar como un parámetro de entrada al nodo. Una vez seleccionada, Publisher Node instancia el objeto correspondiente de entre Simple Orbit, SGP4 Adapter o Custom Path. De este modo, según la frecuencia de actualización fijada como parámetro, Publisher Node pasa el tiempo actual extraído del reloj central de la simulación al objeto correspondiente, y éste retorna la posición $\boldsymbol{t}_{\mathrm{l}}$, orientación $\boldsymbol{R}_{\mathrm{l}}$ y velocidad $\boldsymbol{v}_{l}$ de la órbita o trayectoria generada.

Como se puede observar en la Figura 3 se necesitarán distintos juegos de parámetros en función del tipo de órbita o trayectoria descrita. Si, por ejemplo, se selecciona una órbita simple, se necesitarán parámetros como el semieje mayor a, la excentricidad e, el argumento de periapsis ω, la longitud del nodo ascendente Ω, la inclinación i o el instante de paso por el perigeo t_{p}. Si, por el contrario, se utiliza la biblioteca SGP4, ésta utiliza como parámetros de entrada el Two Line Element

TLE, además de informar qué modelos de perturbaciones se desean incluir en los cálculos de la órbita. Finalmente, para los elementos que describen una trayectoria que no se corresponde con una órbita se empleará Custom Path y ésta ha de recibir cuál es el punto inicial, el final y los puntos intermedios, así como los tiempos y velocidades con que se pasa por cada uno de ellos. Esta información se suministra al nodo mediante el ROS Parameter Server una vez se cargan en éste los parámetros almacenados en archivos YAML. El ROS Parameter Server es común a todos los nodos de la simulación, por lo que cualquiera puede conocer los valores de un determinado parámetro sin necesidad de compartirlo explícitamente.

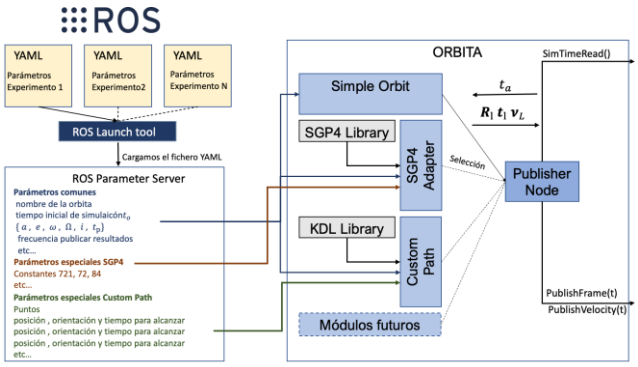

Figura 3: Detalle del Publisher Orbit

5. El paquete OORplugin

En este apartado se definirá el paquete OORplugin, que está representado en naranja en la Figura 1. Este paquete tiene dos objetivos principales. El primero es modificar las propiedades de los motores de físicas de Gazebo para eliminar gravedad, viento, campo magnético, etc., es decir, todas las características del motor de física que suponen una simulación en la superficie terrestre. El segundo es, en base a los datos obtenidos mediante la suscripción al nodo Orbit Publisher Node (OPN), calcular y aplicar las fuerzas y pares correspondientes a una aplicación de servicio en órbita. De modo que se ha de indicar en el archivo SDF del modelo a qué nodo se quiere suscribir del lado de OPN y de qué tipo es, órbita o trayectoria. En una misma simulación se pueden tener varios nodos OPN y uno o ningún plugin del tipo OORplugin por cada modelo de la simulación.

En OORplugin se dispondrá de una serie de acciones que se ejecutan una única vez en la carga del plugin y otras que se realizan en cada iteración del simulador, según se haya configurado el motor de físicas.

5.1 Operaciones iniciales

La primera de las operaciones iniciales es modificar las propiedades del motor de física como se ha comentado anteriormente. Los archivos SDF son una extensión de los archivos de descripción de robots URDF de ROS. Desde un plugin de Gazebo se puede acceder al archivo SDF para leer la información que contiene. Para el correcto funcionamiento del modulo OORplugin ha de conocer a qué OPN ha de suscribirse, este parámetro se obtiene del archivo SDF del modelo, creando en ROS una suscripción al nodo OPN correspondiente. También se leerá de qué tipo está definido el modelo. En caso de estar definida como nave objetivo, una vez se ha suscrito al nodo OPN, y en base a la posición relativa entre el sistema de referencia $\boldsymbol{F}_t$ respecto a $\boldsymbol{F}_l$, se actualiza la posición inicial del eslabón etiquetado como base.

En caso de ser una nave definida como cazadora, se busca en el modelo el eslabón etiquetado como base y, partiendo de éste, se genera un vector de objetos OORobot por cada uno de los eslabones que forman el robot. Con sus parámetros correspondientes, masa, matriz de inercias, etc. Una aeronave que no disponga de robot ha de definirse como un robot de un único eslabón correspondiente al de la base.

5.2 Bucle de simulación

Una vez realizadas las operaciones iniciales pasa a ejecutarse de forma cíclica el bucle de simulación. La frecuencia con la que se ejecuta el bucle de simulación se define dentro del motor de física seleccionado.

Las naves definidas como objetivo mantendrán la posición y orientación relativas inicialmente respecto a $\boldsymbol{F}_l$, siguiendo en cada iteración los cambios relativos a orientación de este sistema de referencia. Esto se debe a que una nave definida como objetivo se entiende que su control queda fuera de nuestro estudio y que dispondrá de los sistemas de control necesarios para el seguimiento de la órbita a la que esta suscrita.

Por el contrario, una nave definida como cazadora es la que realiza la acción dentro de una aplicación de robótica de servicios en órbita. En este caso sí es responsabilidad nuestra tanto la aproximación como el seguimiento de una trayectoria determinada. Además, se aplicarán las fuerzas y pares correspondientes a las perturbaciones modeladas en cada uno de los eslabones que forma el robot empezando por la base, que en estos casos suele ser el satélite o nave espacial donde se ha instalado el robot. Estas fuerzas y pares son calculadas por el objeto OORobot correspondiente a cada eslabón. Antes de realizar los cálculos se actualizan sus valores de posición, orientación y velocidad leyéndolos directamente de Gazebo. Una vez actualizados estos valores se procede al cálculo de fuerzas y pares de perturbación aplicándose posteriormente los resultados a cada eslabón. Es importante reseñar que Gazebo incluye la posibilidad de que las fuerzas y pares se apliquen a un determinado cuerpo como el total de fuerzas y pares que actúan sobre él o se sumen al resto de las fuerzas que se le aplican. Esto significa que posteriormente se podrán aplicar fuerzas provenientes de controladores, producidas por una colisión, etc. Esta última opción hace que se pueda utilizar el OORplugin sin necesidad de modificar su código en conjunto con otros módulos existentes o que se desarrollen en un futuro.

Por último, los objetos que siguen una trayectoria marcada por Custom Path, no se verán afectados por ninguna fuerza de perturbación. Estos seguirán la trayectoria marcada desde su nodo OPN correspondiente.

5.3 Perturbaciones del gradiente de gravedad

En este apartado se describen los principales pares de perturbación originados por el gradiente de gravedad. Los pares debidos al gradiente de gravedad se originan debido a

que el campo gravitatorio no es uniforme. De hecho, la fuerza de la gravedad para un cuerpo que se encuentra orbitando es inversamente proporcional al cubo de la distancia respecto al centro de la Tierra. Esto provoca que se generen unos pares que tienden a orientar el cuerpo rígido en una posición de equilibrio en función de cómo esté distribuida la masa en el cuerpo.

Para aplicar estos pares, primero se ha de conocer cuál es la posición y orientación de cada uno de ellos con respecto al sistema de referencia $\boldsymbol{F}_{\mathrm{g}}$. Esta información puede obtenerse leyéndola directamente del simulador. Para cada cuerpo rígido, ${}^{g}\boldsymbol{t}_{c,k}$ (con $k = 0$ para la base de la nave espacial hasta $k = numero\ de\ eslabones - 1$), es la matriz de traslación desde $\boldsymbol{F}_{\mathrm{g}}$ hasta el centro de masas del eslabón *k*. De modo que la distancia entre cada centro de masas y el centro de la Tierra que corresponde con nuestro sistema de referencia inercial $\boldsymbol{F}_{\mathrm{i}}$. Esta será la suma de las distancias entre $\boldsymbol{F}_{\mathrm{i}}$ y $\boldsymbol{F}_{\mathrm{g}}$ que no es otra que $\boldsymbol{t}_l$ y la distancia obtenida del simulador ${}^{g}\boldsymbol{t}_{c,k}$.

$$\boldsymbol{t}_{c,k} = {}^{g}\boldsymbol{t}_{c,k} + \boldsymbol{t}_l \tag{1}$$

Una vez obtenida la distancia, para calcular los pares del gradiente de gravedad se necesita conocer la orientación del vector que va desde el centro de la Tierra al centro de masas de cada uno de los eslabones. Tal y como se mencionó anteriormente cuando se definieron los sistemas de referencia, $\boldsymbol{F}_{\mathrm{g}}$ y $\boldsymbol{F}_{\mathrm{i}}$ tienen la misma orientación, de modo que $\boldsymbol{R}_{\mathrm{l}}$ es la matriz de rotación de $\boldsymbol{F}_{\mathrm{l}}$ a $\boldsymbol{F}_{\mathrm{i}}$ o $\boldsymbol{F}_{\mathrm{g}}$. Del mismo modo que se obtuvieron la posición de cada uno de los eslabones, se puede leer su orientación relativa al sistema de referencia $\boldsymbol{F}_{\mathrm{g}}$ por lo que ${}^{g}\boldsymbol{R}_{c,k}$ (con $k = 0$ para la base de la nave espacial hasta $k = numero\ de\ eslabones - 1$) es la matriz de rotación desde $\boldsymbol{F}_{\mathrm{g}}$ hasta el centro de masas del eslabón *k*. En la Figura 2 se puede observar un detalle de los sistemas de referencia respecto a una nave espacial con un manipulador. Para el gradiente de gravedad no se necesita todo el sistema de referencia, únicamente se requiere el vector que va desde el centro de la Tierra hasta el centro de masas del eslabón. Por cómo se definió el sistema de referencia, $\boldsymbol{F}_{\mathrm{l}}$ corresponde a su eje $\boldsymbol{x}$, que es la primera columna de la matriz de rotación. Por lo tanto:

$$\boldsymbol{r}_{(x)c,k} = [\ {}^{g}\boldsymbol{R}_{c,k}\boldsymbol{R}_{\mathrm{l}}{}^{\mathrm{T}}] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

La matriz de inercia de cada eslabón $\boldsymbol{I}_{c,k}$ se obtiene del simulador en las operaciones iniciales y $\mu_{\oplus}$ corresponde a la constante gravitacional de la Tierra. El par correspondiente al gradiente de gravedad de cada uno de los eslabones es igual a:

$$\boldsymbol{\tau}_{grav,k} = \frac{3\mu_{\oplus}}{\boldsymbol{t}_{c,k}{}^{3}} sk\left(\boldsymbol{r}_{(x)c,k}\right) \boldsymbol{I}_{c,k}\, \boldsymbol{r}_{(x)c,k} \tag{3}$$

Una vez calculados los pares, estos son aplicados a su eslabón correspondiente por medio de OORplugin en cada iteración.

6. Resultados

Este apartado describe los resultados obtenidos al aplicar OnOrbitROS para la simulación del robot ETS-VII cuyos principales parámetros aparecen en la Tabla 1 (se pueden consultar más detalles de este robot en (Yoshida, 2003)). Esta tabla muestra los momentos de inercia y masas del robot y del brazo.

Tabla 1: Parámetros del robot ETS-VII

Base	Inercia (kg·m^2)					
Masa (Kg)	I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
2552	6206	3541	7087	48.16	78.52	-29.22
Brazo	Masa (Kg) Inercia, I_{zz} (kg·m^2)					
	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
Masa	35.01	22.45	21.89	16.54	26.00	18.49
Inercia	1.69	3.75	2.53	0.072	0.129	0.259

En este apartado se realiza un modelado del escenario presentado en (Yoshida, 2003) (Abiko and Yoshida, 2001) para validar el sistema de simulación propuesto basado en ROS. Para ello, se compararán los datos disponibles del vuelo en órbita del ETS-VII con los obtenidos usando OnOrbitROS para esas mismas trayectorias. Más concretamente, en (Abiko and Yoshida, 2001) se presenta una trayectoria del robot ETS-VII con un perfil de evolución articular como el que se muestra en la Figura 4. La Figura 5.a representa la orientación del satélite base del robot (roll, pitch, yaw) cuando se aplica la trayectoria articular indicada en la Figura 4. Esta última figura representa la orientación de la base sin tener en cuenta el efecto del gradiente gravitacional. Como el robot no recibe ninguna fuerza o momento externo, la orientación al principio y al final de la trayectoria es la misma. OnOrbitROS permite la simulación de trayectorias considerando el efecto del gradiente gravitacional. En este caso, la Figura 5.b representa la orientación del satélite base del robot teniendo en cuenta el efecto del gradiente gravitacional. En esta última figura, se puede apreciar una disminución o incremento en la evolución de los ángulos de roll y pitch respectivamente (este efecto se observa también en los datos reales obtenidos del robot ETS-VII y presentados en (Abiko and Yoshida, 2001)). En los resultados obtenidos, se aprecia que los datos y evolución se corresponden muy fielmente con los datos reales de vuelo obtenidos y presentados en (Abiko and Yoshida, 2001).

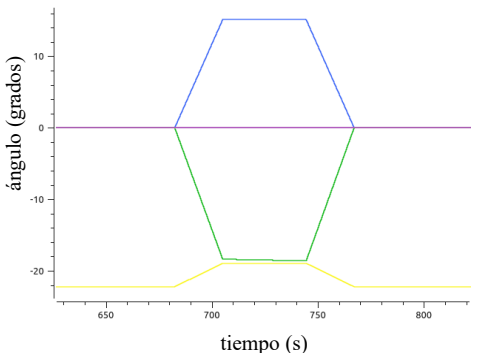

Figura 4: Trayectoria articular del robot ETS-VII

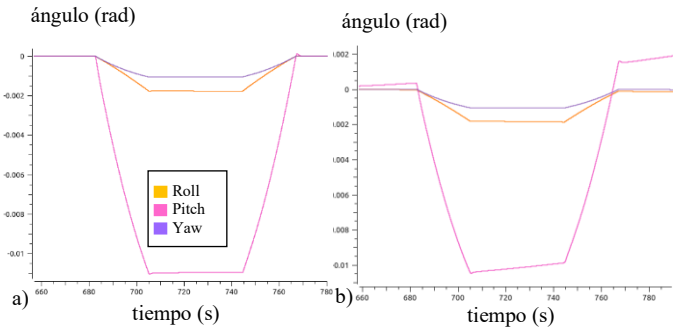

Figura 5: a) Orientación de la base sin tener en cuenta el efecto del gradiente gravitacional. b) Orientación de la base considerando el gradiente gravitacional

7. Conclusiones

En este artículo se ha mostrado la arquitectura, principales objetivos, implementación y opciones disponibles en OnOrbitROS. El uso de ROS permite no sólo definir rápidamente las características de los robots, sino también complejos entornos de trabajo dotados de sensores de distintas características. Así, OnOrbitROS permite extender estos desarrollos a las propiedades específicas de la robótica espacial en órbita. Para conseguirlo, este entorno permite conocer e incluir en la simulación las condiciones específicas de gravedad y perturbaciones orbitales.

En este artículo, los resultados de simulación se han contrastado con los datos reales obtenidos del robot ETS-VII, comprobando que la simulación reproduce fielmente los datos reales de robots en órbita. Actualmente se está extendiendo la herramienta para incluir otras perturbaciones como las debidas al denominado “differential drag”.

Referencias

Abiko, S., & Yoshida, K. 2001. Post Flight Analysis of ETS-VII Space Robotic Experiments. International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1, 1-8.

Allan, M., Chen, I. 2018. Gazebo Renders the Moon, RosCon 2018, Workshop on Lunar Mapping for Precision Landing.

Bualat, M.G., Smith, T., Smith, E.E., Fong, T. Wheeler, D.W. 2018. Astrobee: A New Tool for ISS Operations, AIAA 2018-2517. 2018 SpaceOps Conference. DOI: https://doi.org/10.2514/6.2018-2517

Diftler, M. A. et al.,. 2011. Robonaut 2 - The first humanoid robot in space, 2011 IEEE International Conference on Robotics and Automation, 2178-2183.

Nolet, S. 2007. The SPHERES Navigation System: from Early Development to On-Orbit Testing, AIAA 2007-6354. AIAA Guidance, Navigation and Control Conference and Exhibit. https://doi.org/10.2514/6.2007-6354

Radford, N., et al. 2015. Valkyrie: NASA’s First bipedal humanoid robot, Journal of Field Robotics, 32(3), 397-419. DOI: https://doi.org/10.1002/rob.21560.

Ramon, J.L., Pomares, J., Felicetti, L. 2023. OnOrbitROS Github, https://github.com/OnOrbitROS/, (accessed 29.05.23).

Seddaoui, A., Saaj C. M., Nair, M. H. 2021. Modeling a Controlled-Floating Space Robot for In-Space Services: A Beginner's Tutorial. Front Robot AI. Vol. 8:725333. DOI: 10.3389/frobt.2021.725333

Wang, S., Cao, Y., Zheng, X., Zhang, T. 2022. Collision-Free Trajectory Planning for a 6-DoF Free-Floating Space Robot via Hierarchical Decoupling Optimization, IEEE Robotics and Automation Letters, 7(2) 4953-4960.

Yoshida, K. 2003. Engineering Test Satellite VII flight experiments for space robot dynamics and control: Theories on laboratory test beds ten years ago, now in orbit. The International Journal of Robotics Research, 22(5), 321-335. DOI: https://doi.org/10.1177/0278364903022005003.