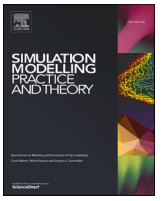
Simulation Modelling Practice and Theory 127 (2023) 102790

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat



Task space control for on-orbit space robotics using a new ROS-based framework

José L. Ramón [a], Jorge Pomares [a,*], Leonard Felicetti [b]

[a] Department of Physics, Systems Engineering and Signal Theory, University of Alicante, San Vicente del Raspeig, Alicante 03690, Spain
[b] School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield MK43 0AL, United Kingdom

ARTICLE INFO

Keywords:
Robot control
Space manipulator
Space robotics

ABSTRACT

This paper proposes several task space control approaches for complex on-orbit high degrees of freedom robots. These approaches include redundancy resolution and take the non-linear dynamic model of the on-orbit robotic systems into account. The suitability of the proposed task space control approaches is explored in several on-orbit servicing operations requiring visual servoing tasks of complex humanoid robots. A unified open-source framework for space-robotics simulations, called OnOrbitROS, is used to evaluate the proposed control systems and compare their behaviour with state-of-the-art existing ones. The adopted framework is based on ROS and includes and reproduces the principal environmental conditions that eventual space robots and manipulators could experience in an on-orbit servicing scenario. The architecture of the different software modules developed and their application on complex space robotic systems is presented. Efficient real-time implementations are achieved using the proposed OnOrbitROS framework. The proposed controllers are applied to perform the guidance of a humanoid robot. The robot dynamics are integrated into the definition of the controllers and an analysis of the results and practical properties are described in the results section.

1. Introduction

The utilisation of robotic systems in space opens up new possibilities and applications for various in-orbit tasks. Space robots play a crucial role in numerous operations conducted in space, including servicing, assembly, and manufacturing. Ongoing missions like Astroscale Elsa-D [1] and Clearspace-1 [2] are already showcasing the necessary technologies for approaching and manipulating cooperative and uncooperative objects in space. Spacecraft such as the Northrop Grumman's Mission Extension Vehicle-1 (MEV-1) [3] have proven the viability of commercial on-orbit servicing and life extension in GEO. Future missions, such as the DARPA's Robotic Servicing of Geosynchronous Satellites (RSGS) [4], NASA's On-orbit Servicing, Assembly and Manufacturing – 1 (OSAM-1) [5] will heavily rely on multiple robotic arms and autonomous systems to perform intricate tasks such as in-orbit satellite repairs, assembly, and manufacturing.

The development and testing of robotic systems for space often necessitate an iterative approach that, together with the technical difficulties of reproducing space conditions in ground-based test facilities and the high costs associated with these tests, discourage the utilisation of hardware-based approaches in the earlier stages of the design [6]. Consequently, software-based simulations offer significant advantages in terms of cost, versatility, and the ability to rapidly model and test space robotic systems. In cases where specific

* Corresponding author.
E-mail address: jpomares@ua.es (J. Pomares).

https://doi.org/10.1016/j.simpat.2023.102790
Received 29 March 2023; Received in revised form 6 June 2023; Accepted 10 June 2023
Available online 11 June 2023

on-orbit phenomena cannot be easily or fully reproduced in ground-based facilities, software simulations become essential. For instance, reproducing the coupling between relative orbit dynamics and gravity gradient effects, atmospheric drag interactions, and sun-radiation pressure effects is generally challenging using flat table-based facilities like [7,8], or robotic systems [9]. In such situations, simulations represent the only option to get a representative and quantitative idea of the overall behavior of the robotic systems in space conditions.

This paper presents a unified open-source framework for space-robotics simulations called OnOrbitROS. The adopted framework is based on Robot Operating System (ROS) and includes and reproduces the principal environmental conditions that eventual space robots and manipulators could experience in an on-orbit servicing scenario.

In order to showcase the capabilities of the new framework, the paper includes a specific test case analysis that focuses on comparing different task space control strategies specifically designed for the purpose. Space robots operate in inherently non-deterministic environments and subject to various disturbances. These robots possess redundant degrees of freedom (DOFs), meaning they have more DOFs than what is strictly necessary for their tasks. This redundancy in DOFs allows the robot to operate in unpredictable workspaces and carry out multiple tasks concurrently or sequentially. Compliant task space control approaches are employed to ensure safety and robust disturbance rejection. These approaches effectively address the redundancy of the robot, allowing for appropriate torque control actions. By resolving the redundancy, the robot determines the optimal joint configuration while performing the task, maximising its effectiveness in the given context.

In this framework, the paper not only introduces various task space control approaches but also examines their suitability for complex high degrees of freedom robots operating in the on-orbit environment. Sensor-based approaches are utilised, which involve utilising data from robot sensors to plan guidance strategies based on workspace information. One of the most promising sensor-based approaches for OOS applications is the visual servoing method [10]. In this case, the robot is guided by leveraging image information captured by a camera. Specifically, the paper focuses on applying the proposed task space control approaches to enable visual servoing of a sophisticated on-orbit humanoid robot within an on-orbit servicing scenario.

The remaining sections of the paper are organised as follows: the state-of-the-art on both simulation tools for in-orbit operations and tasks space controls applied to space robotics operation scenarios is presented in Section 2. Section 3 introduces and describes the proposed unified open-source framework for space-robotic simulations, OnOrbitROS. Section 4 details the suitability of different task space control approaches for complex high DOF robots. This section proposes velocity-based, acceleration-based and force-based approaches for on-orbit manipulators, and it ends with the description of a new direct position-based approach based on the previous task-based control systems. Section 5 presents the simulation results. Specifically, the OnOrbitROS framework simulations are compared against published flight mission data from some experiments carried on by NASDA in the frame of the ETS-VII mission. After this validation, the task-based approaches and the proposed visual servoing system are simulated using a humanoid robot to assess the performance of the proposed control systems in an OnOrbitROS simulated environment. Finally, Section 6 summarises the main concluding remarks associated with this study.

2. Related works

This section describes the related works and synthesises the main paper's contribution to the state-of-the-art. The paper's primary contribution, as outlined in the introduction, is the establishment of a unified open-source framework for space robotics based on ROS. Different teams within the space community have carried on numerous attempts to develop simulation tools. These include ad-hoc build libraries and simulation tools, some of them openly available online, such as SpaceDyn [11], and others that have been commercialised, such as DCAP [12,13]. On the other hand, numerous research papers have presented methodologies and algorithms suitable for developing ad-hoc simulation tools to analyse and design control algorithms for on-orbit robotic operations. These tools are often used to simulate space-based test cases with specific robot configurations and investigate various effects such as structural flexibility [14–16], sloshing effects [17], contact dynamics, and impacts [18]. In most cases, ad-hoc tools are developed to test control strategies for space manipulators: among the vast literature, it is worth mentioning the works related to the reaction null control [19], the virtual manipulator strategies [20], image-based control strategies [21] and impedance control to mitigate contact dynamics effects in space [22]. However, these studies often rely on *ad-hoc* built simulation tools that are not standardised, open-source, or rigorously verified. This lack of standardisation and accessibility hampers the development and implementation of versatile robotic systems and algorithms for space robotics. In contrast, using simulation tools like ROS and Gazebo has become common practice for testing and developing control algorithms for ground-based robotic systems [23–25]. Nevertheless, the use of ROS/Gazebo in the space community is not yet widely accepted due to the inability to fully and realistically simulate space conditions such as microgravity and frictionless environments.

Keeping in mind this panorama, the solution proposed in this paper moves towards developing a unified open-source tool for space-robotic simulations. This tool is called OnOrbitROS and is freely available in [26]. The adopted framework is based on ROS, an open-source meta-operative system to develop robot applications, combined with Gazebo, a tool to simulate populations of robots in customised environments. These tools have been modified to incorporate and replicate the primary environmental conditions that space robots and manipulators may encounter in an OOS scenario. This solution facilitates the simulation of complex space robotic systems while leveraging the extensive range of packages already available in ROS for control, vision, teleoperation, and modeling tools. By utilising ROS and Gazebo in this way, the framework provides a powerful and versatile platform for developing and testing space robotics applications. By adhering to the ROS principle of "Don't reinvent the wheel", the framework avoids the need to create a new platform from scratch and instead builds upon existing ROS infrastructure to simulate OOS applications. This paper presents a description of the architecture of the different software modules. It shows the key features of the developed tool, with a particular focus

on the customisation of the simulations, eventual possibilities of further expansion of the tool, and implementation of task space controllers (and visual servoing tasks). The proposed OnOrbitROS architecture has the potential to be used by open source platforms such as AWS RoboMaker [27]. Gazebo is the default simulation tool for AWS RoboMaker. Consequently, by using Gazebo and OnOrbitROS, AWS RoboMaker users can create complex OOS simulations in the cloud, eliminating the requirement for high-performance local machines and with physical parameters closer to the real orbit condition. To ensure the authenticity of the results in space conditions, the paper presents the validation of the simulator suite. This validation process involves comparing the simulations against published flight mission data from experiments conducted by NASDA during the ETS-VII mission [28,29]. By validating the simulator against actual mission data, the paper demonstrates the accuracy and reliability of the simulation results in replicating space conditions.

The paper introduces a novel direct position-based visual servoing system designed specifically for complex high DOF on-orbit robots. Specifically, this control strategy is applied to different task space control approaches in an OOS scenario. The proposed system includes velocity-based, acceleration-based, and force-based controllers tailored for guiding complex and redundant space robots. These controllers take into account the specific conditions of robots in an OOS scenario, including their dynamics and perturbations. A comparative analysis is conducted to determine the most adequate controller during the tracking of trajectories. Efficient real-time implementations are achieved using the proposed OnOrbitROS framework, which, among the other features, allows for the integration of the robot dynamics in the characterisation of the proposed controllers. From the previous analysis, the more promising task space controller is extended to obtain a new direct position-based visual servoing controller. This approach utilises image information and takes into consideration the system dynamics and environmental perturbations, such as gravity gradient torques, during the robot's guidance. By incorporating these factors, the controller enhances the precision and stability of the visual servoing process in challenging on-orbit conditions.

Visual servoing methods can be categorised as *image-based visual servoing*, where the control law is defined in the image space [30], and in *position-based visual servoing*, where image features are extracted from the captured image, and this information is used to determine the pose of the target with respect to the frame attached to the camera. Vision-based navigation approaches, such as those described in [31], often employ 3D model-based tracking. It is also worth mentioning the work presented in [32], where stereo vision is used to perform the pose estimation for the final phase of the rendezvous and docking of non-cooperative satellites. Similarly, [33] presents a relative navigation method for rendezvous and docking of an unknown tumbling object utilising a monocular camera and Kalman filters. Specifically, two extended Kalman filters with different models are used for relative orbit estimation in far range and relative position and attitude estimation in close range. This approach has been used in OOS applications such as [34,35]. In [34] a monocular camera is used to determine the target pose estimation required to perform the relative navigation for space rendezvous and proximity operations. In [34], the vision-based navigation of SPHERES satellites is presented for the microgravity environment of the International Space Station (ISS). All these previous approaches are considered indirect position-based visual servoing systems, where the control action is specified in terms of velocity applied to the robot without considering the system dynamics.

On the other hand, in direct controllers, the control actions are directly computed in terms of forces and torques applied to the robot joints. Previous studies show the necessity to integrate robot dynamics in the visual servoing systems using a direct approach [36]. Previous works by the authors, such as [37,38], proposed direct image-based visual servoing to guide free-floating robotic manipulators and to perform spacecraft rendezvous manoeuvres, respectively. In contrast with these previous approaches, a new direct position-based control law in this paper has been designed in the task space that includes redundancy resolution. This approach is extended for the guidance of a redundant humanoid robot. Classical indirect position-based visual servoing approaches do not take into account the system dynamics. In order to take into account the non-linear dynamic model of the humanoid, a new direct position-based visual servoing control is proposed. This control is suitable when the robot performs fast and accurate movements. A comparison of the proposed direct position-based visual servoing system with previous position-based visual servoing systems [39] is also included in the results section and evaluated in the on-orbit scenario.

3. OnOrbitROS architecture

ROS [40] is an open-source framework providing a collection of tools to develop complex robotic software systems. It was initially developed at Stanford's AI lab in 2007. It has been widely accepted by research communities and it collects open-source contributions from robotics communities all over the world. ROS follows a publisher-subscriber model using peer-to-peer communication that allows efficient inter-node communication via messages over ROS topics. It acts as a middleware offering a hardware abstraction layer.

The ROS ecosystem comprises ROS packages, each active package representing a module in the software architecture. A ROS package is composed of one or more C++ or Python nodes. Together, these nodes exchange data over ROS topics, allowing for a coherent modular system that is easier to extend and debug. As the ROS community has expanded, many useful tools have been developed in the form of ROS packages to benefit both users and developers. ROS inherently provides visualisation plugins like RQT [41] that allow real-time inspection and monitoring of the system. It also allows utilising various communication interfaces using ROS services and actionlib to perform request-response-based tasks. The advantage of an open-source framework like ROS is that with easy access to the internal functioning of the system, debugging becomes easy across all levels. The ROS code developed for this study acts as a solution for the simulation of complex on-orbit space robotic systems.

In addition to ROS, Gazebo has been chosen as the simulation environment for the project. Gazebo is a simulation tool specifically designed for fast and efficient test development. Originally, Gazebo was conceived as a tool for 3D simulation of multi-robot environments with the ability to recreate complex and customisable environments. Similar to ROS, Gazebo is open-source and freely available. Gazebo is selected as a complement to ROS in the OnOrbitROS framework because it can simulate complex 3D environments

where each element possesses properties such as mass, velocity, and friction. This aligns with the project's objective to create a realistic simulation of on-orbit conditions. By utilising Gazebo along with ROS, the framework can provide a comprehensive simulation environment that accurately represents the dynamics and interactions of space robots in complex 3D settings. There are certain aspects for which Gazebo has been selected as a complement to ROS for OnOrbitROS:

- High integration of Gazebo within ROS. The Gazebo simulations are generated from an XML file based on the SDF description language, which is an extension of the URDF used in ROS for robot description. This allows for quickly creating and configuring new environments without programming modifications.
- Possibility to include additional plugins. Gazebo offers the possibility to extend its functionality through plugins. As it is described throughout the paper, OnOrbitROS includes the required plugins to simulate the on-orbit-specific conditions. Additionally, Gazebo integrates interfaces to use multiple physics engines such as Open Dynamics Engine, Bullet, Simbody, Dynamic Animation and Robotics Toolkit (DART).
- Gazebo and ROS have been widely used in the simulation and development of new robots for Terrestrial applications. This aspect allows the possibility to use a great number of software libraries and tools that help in the building of robot applications.

However, Gazebo lacks a simulation environment for on-orbit robotics applications. OnOrbitROS provides features and packages that allow setting and simulating OOS applications. The proposed library presents a ROS and Gazebo architecture so that all the control, visualisation, sensors, etc. packages already available in the standard ROS/Gazebo suite can be used to develop complex experiments for on-orbit space robotics.

OnOrbitROS serves as a foundational platform for the study and development of OOS applications, leveraging the powerful combination of ROS and Gazebo for hyper-realistic simulations. Fig. 1 illustrates the main structure and packages of OnOrbitROS. This structure includes the two main elements of an OOS application: the target spacecraft (*target*) and the servicing spacecraft (*chaser*). The chaser spacecraft hosts the robot or serves as the robot itself, with both models defined using Simulation Description Format (SDF) definitions. Multi-robot applications can also be simulated by incorporating multiple instances of simulated robots. OnOrbitROS consists of two packages. The first package, depicted in blue in Fig. 1, generates spacecraft trajectories, as will be elaborated in Section 3.2. The second package, represented in orange, extends and modifies the physics engines to align with the conditions of an OOS application, as explained in more detail in Section 3.3. From this basic structure, complex OOS robotics applications can be simulated where elements such as force sensors, cameras, laser sensors, etc. can be integrated. At the same time, robots with numerous DOFs, such as humanoid robots, can be integrated while keeping the physics engine options in the simulation as close as possible to orbital conditions.

The results section explores several of the capabilities of OnOrbitROS in different scenarios. The first scenario simulates the ETS-VII operations, as described in [28,29], and validates them by comparing them with data from past experiments in orbit. Additionally, several other scenarios are defined to assess the ability of OnOrbitROS to simulate new controllers for on-orbit robot manipulators and more complex robotic scenarios, incorporating sensor systems and humanoid robots. Hence, the presented architecture and simulation system can be used for the simulation of future OOS applications such as collaborative multi-robot applications in space debris removal, trajectory control in formation flights close to a reference orbit, on-orbit manufacturing, etc.

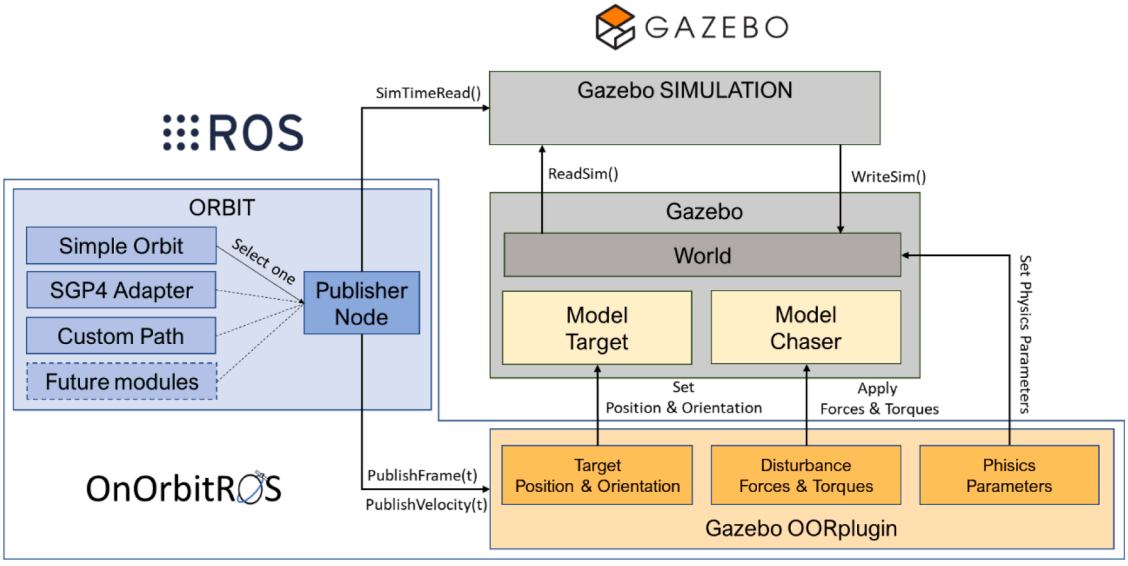


Fig. 1. Main packages and architecture of OnOrbitROS.

3.1. Definition of reference frames

This section defines the main coordinates frames considered throughout the paper. Fig. 2 provides a visual representation of these coordinate frame and their transformations. The spacecraft to be serviced is named as *target* spacecraft, and the corresponding coordinate frame, $\boldsymbol{F}_t$, is located at its centre of mass. The servicing spacecraft is called *chaser* and its coordinate frame, $\boldsymbol{F}_c$, is located at its centre of mass too. The *chaser* may have a robotic manipulator to perform OOS of the target. The Earth-Centered Inertial frame (ECI) is denoted as $\boldsymbol{F}_i$ with its origin located at the centre of the Earth with its x axis going from the centre of the Earth passing along the vernal equinox, the z axis coincides with the axis of rotation of the Earth, and pointing to the north pole, finally the y axis completes the orthogonal triad.

A Local Vertical Local Horizontal (LVLH) reference frame is used to define the position of any object with respect to a specific orbital position. This reference frame is denoted by $\boldsymbol{F}_l$ in Fig. 2. In this frame, the x axis corresponds to the direction of the radial vector that goes from the center of the Earth to the spacecraft, the y axis, which together with the x axis forms the orbital plane pointing this in the direction of the spacecraft movement. Finally, the z axis completes the triad being normal to the orbital plane. As shown in Fig. 2, the matrices $\boldsymbol{R}_l$ and t_l represent the rotation and translation of the LVLH frame with respect the inertial frame. The Orbit module computes both matrices, shown in blue in Fig. 1, which calculates the theoretical orbit of the target spacecraft (see Section 3.2 for details).

The Gazebo simulation environment has its own reference frame, which is denoted by $\boldsymbol{F}_g$. The proposed framework assumes that the orientation of $\boldsymbol{F}_g$ and $\boldsymbol{F}_i$ are coincident, but the origin of $\boldsymbol{F}_g$ is the same as the origin of $\boldsymbol{F}_l$. In this way, we can simulate the proximity dynamics of the *chaser* around the *target*, assuming that the latter is moving along its orbit and the relative orientation between $\boldsymbol{F}_l$ and $\boldsymbol{F}_g$ is given by $\boldsymbol{R}_l$, calculated by the *Orbit* module. Finally, the rotation and translation matrices ${}^g\boldsymbol{R}_c$ and ${}^g\boldsymbol{t}_c$ describe the relative attitude and position of the *chaser* frame $\boldsymbol{F}_c$, with respect to the Gazebo frame $\boldsymbol{F}_g$.

3.2. Orbit package

This section describes the modules that compose the *Orbit* package. Fig. 3 illustrates the main components of the *Orbit* module. The main function of this module, implemented as a ROS publisher node, is to generate the orbital trajectories of the bodies that compose the simulation. At the time of writing this paper, the *Orbit* module has three submodules:

- *Simple Orbit*. This module can be used for the generation of trajectories that describe Keplerian orbits. This module is described in greater detail in Appendix A because it is the default option for OnOrbitROS and it does not require any external library.
- *SGP4 adaptor*. This module can be used for the propagation of other kinds of orbits (this module incorporates the third-party library by Spacetrack [42,43]). This module creates an interface to the library so that it can obtain the parameterisation from the ROS parameter server while publishing the position and orientation of the reference frame $\boldsymbol{F}_l$.
- *Custom Path*. This module can be used to model user-defined trajectories that do not fall in the previous two cases. This last module uses the external library Kinematics and Dynamics Library (KDL) [44] to generate trajectories based on a series of points by interpolating them by splines. In this case, a series of points are introduced by the user in the ROS parameter server (a data structure composed of a set of positions, velocities and accelerations, and the time instant in seconds to achieve each pose), and the Custom Path module generates the interpolated trajectory.

The user defines the orbit or trajectory that best fits the trajectory to be modelled in the launch file, and the Publisher Node instantiates the corresponding object. The Publisher Node provides the timing to the corresponding object, and the object generates the

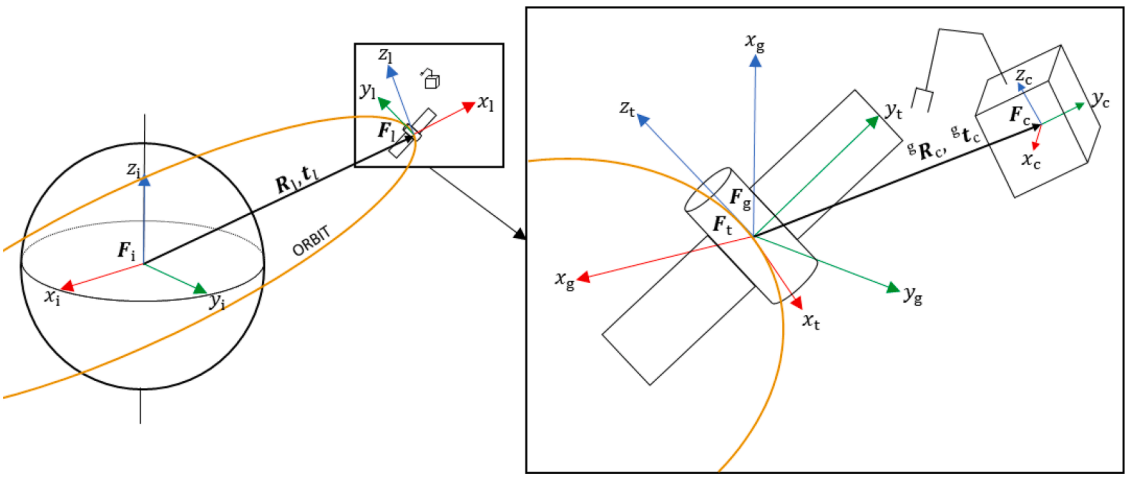


Fig. 2. Coordinate frames in OnOrbitROS.

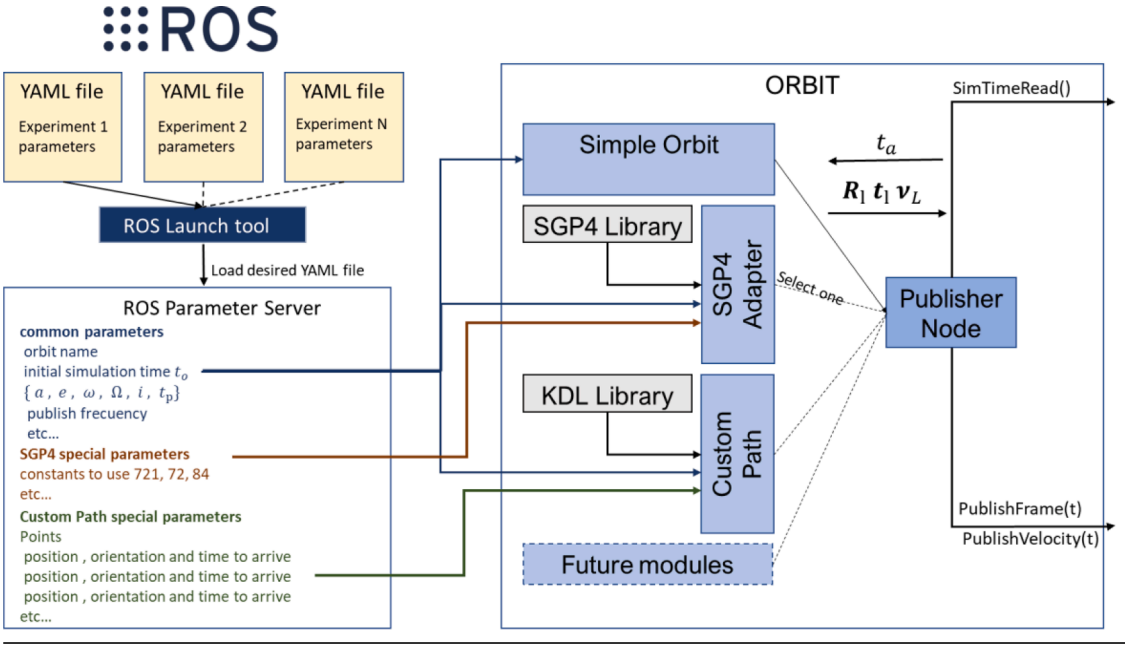


Fig. 3. Orbit Publisher detail.

values $\boldsymbol{R}_l$, $\boldsymbol{t}_l$ and $\boldsymbol{\nu}_l$ of the generated orbit or trajectory. Different parameters are required depending on the orbit or trajectory to be generated. For example, for the *Simple Orbit* module, parameters such as the semi-major axis, a, eccentricity, e, argument of periapsis, ω, right ascension of ascending node, Ω, inclination, i, and instant of time of perigee passage, t_p should be defined. When the *SGP4 adaptor* is used, the Two Lines Elements (TLE) should be given as input. Finally, for defining a custom trajectory, initial, final, and intermediate positions, velocities, accelerations and times should be provided. This information is indicated by the user in the YAML configuration file and stored in the ROS Parameter Server. The ROS Parameter server is common to all nodes and allows sharing of parameters between different nodes.

3.3. OORplugin package

This module in OnOrbitROS is responsible for modifying the parameters of the physics engines used by Gazebo to create a realistic simulation environment for OOS applications. It eliminates effects like gravity, wind, and magnetism, typically applied in simulations assuming they occur on the Earth's surface. Instead, this module applies the relevant torques and forces specific to OOS scenarios.

Gazebo provides two main features that are utilised in the simulation generation. First, simulations are described and parameterised using the Simulation Description Format (SDF), an XML file loaded by the simulator at start-up. The SDF description allows for defining the target spacecraft, chaser spacecraft, their respective orbits, and other relevant details (*Model Target, Model Chaser* and *World* in Fig. 1). The second feature of Gazebo used to simulate OOS applications is the possibility of extending the Gazebo functionality through plugins. These plugins have access to all the simulation elements, being able to consult the state of these and, at the same time, to act with them. So, all the links of the simulated spacecraft will be subject to the forces and torques that we will describe in the next two sections (perturbation from gravity gradient and forces generated by the relative motion to an elliptical orbital).

The current version of OnOrbitROS simulates the relative dynamics and the perturbation due to gravity gradient. There are considered among the main effects characterising the dynamics in the near-Earth space region. Other perturbations, such as aerodynamic drag, solar radiation pressure and effects of the third-body will be included in the next versions of the tool.

3.3.1. Perturbation from gravity gradient

This section describes the main perturbing torques caused by the gravity gradient and simulated in OnOrbitROS. These torques vary in the different parts of each rigid body that form the *chaser* spacecraft. Fig. 4 represents the main transformation matrices used to compute the gravity gradient perturbing torques. The position and orientation of each rigid body with respect to the reference frame $\boldsymbol{F}_g$ can be easily obtained from the robot kinematics. The translation matrix from $\boldsymbol{F}_g$ to the center of mass of each link is represented as ${}^g\boldsymbol{t}_{c,k}$, where $k = 0$ for the base of the spacecraft and $k = 1$ to ne represents the k link of the robot (being ne the DOF of the manipulator). Therefore, the distance between any robot element and the ECI frame is the sum of the distances between $\boldsymbol{F}_i$ and $\boldsymbol{F}_g$ (represented by the vector $\boldsymbol{t}_l$) and ${}^g\boldsymbol{t}_{c,k}$:

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

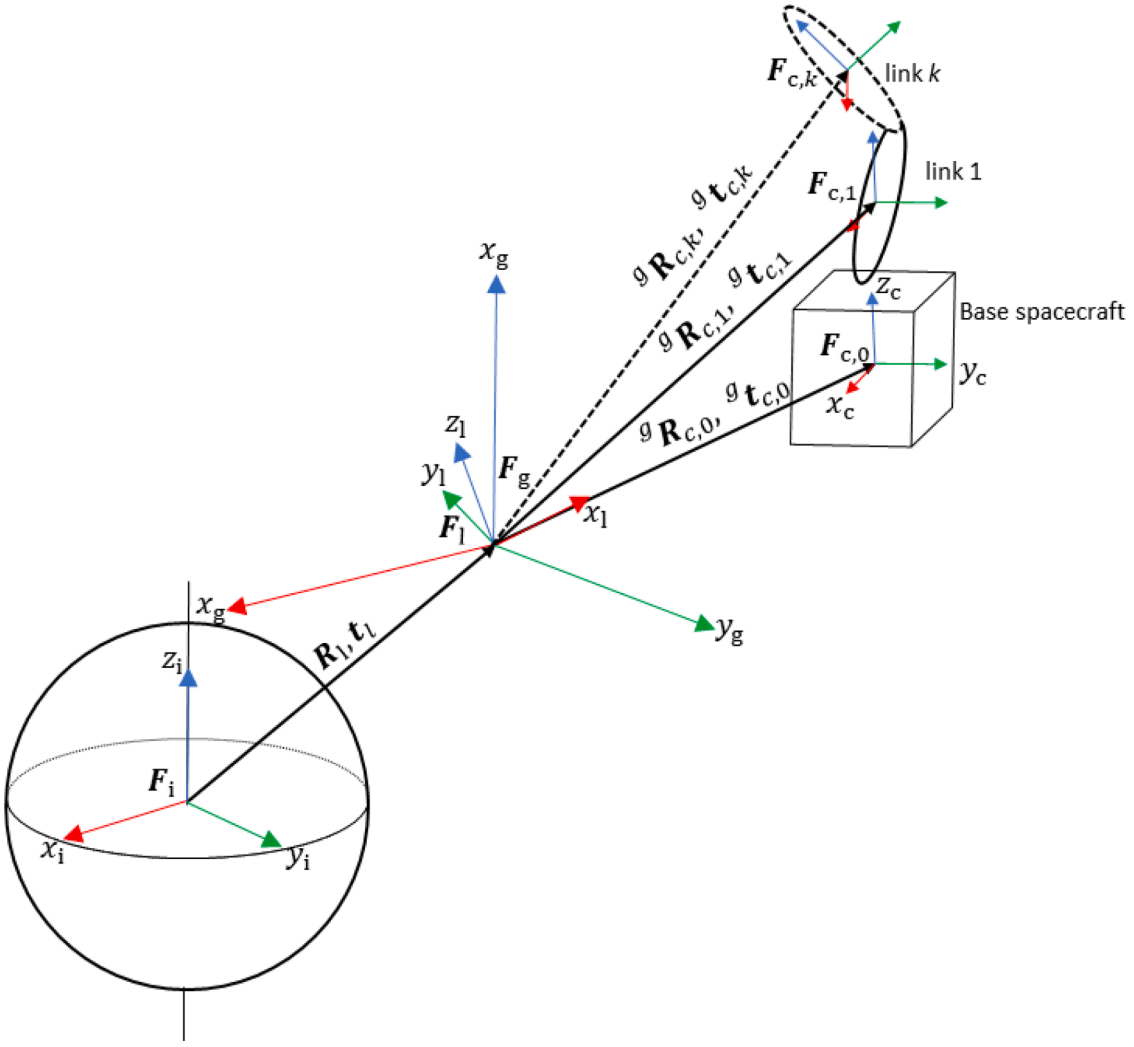


Fig. 4. Main transformations to compute the perturbing torques.

As already commented in Section 3.1, F_g and F_i have the same orientation, being R_l the rotation matrix from F_l to F_i or F_g. As shown in Fig. 4, the relative orientation of each of the robot links with respect to F_g is represented as ${}^{g}R_{c,k}$ where $k = 0$ for the base of the spacecraft and $k = 1$ to *ne* represents the *k* link of the robot. Specifically, to compute the gravity gradient, the vector pointing from the centre of the Earth to the centre of mass of each link must be known. Given the definition of the reference frames in Section 3.1, this last vector corresponds with the *x* axis of the LVLH reference frame defined by the first column of F_l. Therefore, the corresponding unit vector is:

$$r_{(x)c,k} = \left[{}^{g}R_{c,k}R_l^{T}\right]\begin{bmatrix}1\\0\\0\end{bmatrix} \tag{2}$$

The corresponding torques of the gravity gradient of each link read as:

$$\tau_{grav,k} = \frac{3\mu_{\oplus}}{t_{c,k}^{3}} sk\left(r_{(x)c,k}\right) I_{c,k}\, r_{(x)c,k} \tag{3}$$

where $I_{c,k}$ is the inertia matrix of each link, $\mu_{\oplus}$ is the Earth gravitational constant and $sk() \in \Re^{3x3}$ is the skew-symmetric matrix. Once the gravity gradient torques have been calculated, they are applied in each iteration of the simulation by the module Gazebo OOR-plugin.

3.3.2. Relative motion with respect to an elliptical orbit

In this section, we show the method used to compute the apparent forces due to the relative motion with respect to the reference frame describing the orbit, $\boldsymbol{F}_l$, in OnOrbitROS. These forces are of great importance in operations where a *chaser* satellite flies in proximity of a *target*. The first step for computing such kind of forces is the calculation of the angular momentum of the reference orbit. This can be obtained as $\boldsymbol{h} = \boldsymbol{t}_l \times \boldsymbol{v}_l$ (both the position $\boldsymbol{t}_l$ and linear velocity $\boldsymbol{v}_l$ calculated in the *orbit* node of OnOrbitROS, see Section 3.2), and its corresponding absolute value is equal to $h = \| \boldsymbol{t}_l \times \boldsymbol{v}_l \|$. The position of each link of the chaser satellite with respect to the $\boldsymbol{F}_l$ reference frame is denoted as ${}^l\boldsymbol{t}_{c,k}$ $({}^lt_{c,k_x}, {}^lt_{c,k_y}, {}^lt_{c,k_z})$ and its linear velocity as ${}^l\dot{\boldsymbol{t}}_{c,k}$ $({}^l\dot{t}_{c,k_x}, {}^l\dot{t}_{c,k_y}, {}^l\dot{t}_{c,k_z})$, where $k = 0$ for the base of the spacecraft and $k = 1$ to *ne* represents the k link of the manipulator. Both position and velocity can be obtained as ${}^l\boldsymbol{t}_{c,k} = \boldsymbol{R}_l^{Tg}\boldsymbol{t}_{c,k}$ and ${}^l\dot{\boldsymbol{t}}_{c,k} = \boldsymbol{R}_l^{Tg}\dot{\boldsymbol{t}}_{c,k}$, respectively. Using the linearised version of the equations of relative orbital motion as in [45], and by defining the mass of each link as m_k, the three components of the apparent forces due to the relative motion to be applied to each link can be calculated as follows:

$$f_{form,k_x} = m_k\left(\left(\frac{2\mu_\oplus}{t_l^3} + \frac{h^2}{t_l^4}\right){}^lt_{c,k_x} - \frac{2\left(\boldsymbol{t}_l.\dot{\boldsymbol{t}}_l\right)h_l}{t_l^4}{}^lt_{c,k_y} + 2\frac{h_l}{t_l^2}{}^l\dot{t}_{c,k_y}\right) \tag{4}$$

$$f_{form,k_y} = m_k\left(-\left(\frac{\mu_\oplus}{t_l^3} - \frac{h^2}{t_l^4}\right){}^lt_{c,k_y} + \frac{2\left(\boldsymbol{t}_l.\dot{\boldsymbol{t}}_l\right)h_l}{t_l^4}{}^lt_{c,k_x} + 2\frac{h_l}{t_l^2}\dot{t}_{c,k_x}\right) \tag{5}$$

$$f_{form,k_z} = m_k\left(-\frac{\mu_\oplus}{t_l^3}{}^lt_{c,k_z}\right) \tag{6}$$

Such forces are calculated and applied to each body of the OOS scenario, alongside the gravity gradient forces, by the module Gazebo OORplugin at each iteration of the simulation.

4. Cartesian controllers for space manipulators

This section presents different task space control approaches for complex DOF robots and describes their application to a visual servoing task in an OOS scenario. For the controller's description, an anthropomorphic robotic arm with *ne* degrees of freedom located in a base spacecraft is considered. The different coordinate frames represented in Fig. 2 are considered. With respect to the system kinematics, the servicing spacecraft configuration can be defined by the position and attitude of the base spacecraft, $\boldsymbol{t}_b$ and $\boldsymbol{\phi}_b$ (both with respect the inertia frame), and by the joint configuration of the manipulator's arm, $\boldsymbol{q}^T \in \Re^{ne}$. The full kinematics of the spacecraft kinematics is defined by the vector $[\boldsymbol{t}_b^T, \boldsymbol{\phi}_b^T, \boldsymbol{q}^T]^T$.

The system dynamics provide a relationship between the acceleration and forces and torques in both the base satellite and manipulator. Specifically, the system dynamics relate to the linear and angular accelerations of the base spacecraft $\dot{\mathbf{v}}_b = [\ddot{\boldsymbol{t}}_b^T, \dot{\boldsymbol{\omega}}_b^T]^T \in \Re^6$ expressed in the Inertial coordinate frame, the joint accelerations of the manipulator's arm, $\ddot{\boldsymbol{q}}$, with the forces and torques exerted on the base of the servicing spacecraft, $\boldsymbol{h}_b \in \Re^6$, and the torques applied on the robot manipulator joints, $\boldsymbol{\tau} \in \Re^{ne}$. This relationship can be defined as:

$$\begin{bmatrix}\boldsymbol{h}_b \\ \boldsymbol{\tau}\end{bmatrix} = \begin{bmatrix}\boldsymbol{M}_{bb} & \boldsymbol{M}_{bm} \\ \boldsymbol{M}_{bm}^T & \boldsymbol{M}_{mm}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{v}}_b \\ \ddot{\boldsymbol{q}}\end{bmatrix} + \begin{bmatrix}\boldsymbol{c}_b \\ \boldsymbol{c}_m\end{bmatrix} \tag{7}$$

where $\boldsymbol{M}_{bb} \in \Re^{6\times 6}$ is the inertia matrix of the base spacecraft, $\boldsymbol{M}_{bm} \in \Re^{6\times ne}$ is the coupled inertia matrix of the base spacecraft and the manipulator's arm, $\boldsymbol{M}_{mm} \in \Re^{ne\times ne}$ is the inertia matrix of the manipulator's arm, $\boldsymbol{c}_b$, and $\boldsymbol{c}_m$, $\in \Re^6$ are a velocity/displacement-dependant, non-linear terms for the base spacecraft and manipulator arm, respectively.

Eq. (7) can be extended by including the gravity gradient torques and rewritten in the following form:

$$\boldsymbol{M}^*\ddot{\boldsymbol{q}} + \boldsymbol{C}^* = \boldsymbol{\tau} + \boldsymbol{\tau}_{grav} \tag{8}$$

where $\boldsymbol{\tau}_{grav} \in \Re^{ne}$ are the gravity gradient torques, $\boldsymbol{M}^* \in \Re^{ne\times ne}$ is the generalised inertia matrix, $\boldsymbol{C}^* \in \Re^{ne}$ is the generalised Coriolis and centrifugal vector for the manipulator arm, defined explicitly as:

$$\boldsymbol{M}^* = \boldsymbol{M}_{mm} - \boldsymbol{M}_{bm}^T\boldsymbol{M}_{bb}^{-1}\boldsymbol{M}_{bm} \tag{9}$$

$$\boldsymbol{C}^* = \boldsymbol{c}_m - \boldsymbol{M}_{bm}^T\boldsymbol{M}_{bb}^{-1}\boldsymbol{c}_b \tag{10}$$

The linear and angular momenta of the system $[\mathscr{L}^T, \boldsymbol{\Psi}^T]^T \in \Re^6$ are defined as:

$$\begin{bmatrix}\mathscr{L} \\ \boldsymbol{\Psi}\end{bmatrix} = \boldsymbol{M}_{bb}\boldsymbol{v}_b + \boldsymbol{M}_{bm}\dot{\boldsymbol{q}} \tag{11}$$

$\boldsymbol{v}_b = [\dot{\boldsymbol{t}}_b^T, \boldsymbol{\omega}_b^T]^T \in \Re^6$ denotes the linear and angular velocities of the base spacecraft expressed in the inertial coordinate frame, and $\dot{\boldsymbol{q}} \in \Re^{ne}$ represents joint speeds of the arm. The relationship between the joint speeds and the corresponding end-effector's absolute linear and angular velocities can be expressed through differential kinematics:

$$\dot{\boldsymbol{p}} = \boldsymbol{J}_m \dot{\boldsymbol{q}} + \boldsymbol{J}_b \boldsymbol{v}_b \tag{12}$$

where $\dot{\boldsymbol{p}} \in \Re^6$ is the linear and angular velocity of the manipulator end-effector in the inertial frame, $\boldsymbol{J}_m \in \Re^{6 \times ne}$ is the manipulator Jacobian matrix, and $\boldsymbol{J}_b \in \Re^{6 \times 6}$ is the Jacobian matrix of the robot. Combining Eq. (12) with Eq. (11) yields an equation that directly relates the joint speeds and end-effector motion of the robot manipulator:

$$\dot{\boldsymbol{p}} = \boldsymbol{J}_g \dot{\boldsymbol{q}} + \boldsymbol{J}_b \boldsymbol{M}_{bb}^{-1} \begin{bmatrix} \not{\boldsymbol{\ell}} \\ \boldsymbol{\Psi} \end{bmatrix} \tag{13}$$

$$\boldsymbol{J}_g = \boldsymbol{J}_m - \boldsymbol{J}_b \boldsymbol{M}_{bb}^{-1} \boldsymbol{M}_{bm} \tag{14}$$

where $\boldsymbol{J}_g$ is the Generalised Jacobian Matrix for the manipulator. The second derivative of Eq. (13) can be expressed as:

$$\ddot{\boldsymbol{p}} = \boldsymbol{J}_g \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_g \dot{\boldsymbol{q}} + \dot{\boldsymbol{v}}_{gm} \tag{15}$$

where $\boldsymbol{v}_{gm} = \boldsymbol{J}_b \boldsymbol{M}_{bb}^{-1} \begin{bmatrix} \not{\boldsymbol{\ell}} \\ \boldsymbol{\Psi} \end{bmatrix}$.

4.1. Velocity-based control

This section defines a velocity-based controller for the guidance of an OOS manipulator. The velocity-based controller presents a velocity command in the Cartesian space. Therefore, in this case, the Cartesian reference can be expressed as:

$$\dot{\boldsymbol{p}}_r = \dot{\boldsymbol{p}}_d + \boldsymbol{K}_p (\boldsymbol{p}_d - \boldsymbol{p}) \tag{16}$$

where $\boldsymbol{p}_d$, $\dot{\boldsymbol{p}}_d$ represent the desired position and velocity corresponding to the Cartesian trajectory to be tracked, and $\boldsymbol{K}_p$ is a matrix containing the gains for building a proportional feedback. In this case, the reference joint velocities can be obtained from the previous Cartesian reference as:

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_g^+ (\dot{\boldsymbol{p}}_r - \boldsymbol{v}_{gm}) - \alpha \left(\boldsymbol{I} - \boldsymbol{J}_g^+ \boldsymbol{J}_g \right) \boldsymbol{\upsilon}_v \tag{17}$$

where α is a positive constant, and $\boldsymbol{\upsilon}_v$ is considered as a null space cost function. A simple optimisation criterion for redundancy is considered in joint space, which pulls the joins, $\boldsymbol{q}$, toward a given configuration, $\boldsymbol{q}_s$:

$$\boldsymbol{\upsilon}_v = \boldsymbol{K}_{ps} (\boldsymbol{q}_s - \boldsymbol{q}) \tag{18}$$

where $\boldsymbol{K}_{ps}$ is a positive definite matrix containing gains for proportional feedback of the joint variables. Considering the reference Cartesian velocities $\dot{\boldsymbol{p}}_r$ and the reference joint velocities $\dot{\boldsymbol{q}}_r$ obtained in Eqs. (16) and (17), the velocity-based control law can be defined using the dynamics equation given by Eq. (8):

$$\boldsymbol{\tau} = \boldsymbol{M}^* \ddot{\boldsymbol{q}}_r + \boldsymbol{C}^* + \boldsymbol{K}_d (\dot{\boldsymbol{q}}_r - \dot{\boldsymbol{q}}) - \boldsymbol{\tau}_{grav} \tag{19}$$

where $\boldsymbol{K}_d$ is a positive definite matrix containing gains for building the derivative feedback term of the joint variables. When the velocity-based controller is applied, the reference joint accelerations can be obtained by differentiation of the reference joint velocities given in Eq. (17) as:

$$\ddot{\boldsymbol{q}}_r = \frac{d}{dt} \dot{\boldsymbol{q}}_r \cong \frac{\dot{\boldsymbol{q}}_r(t) - \dot{\boldsymbol{q}}_r(t - \Delta t)}{\Delta t} \tag{20}$$

where Δt is the sampling period. By considering the definition of the Cartesian reference in Eq. (16), integrating its derivative in the control action given in Eq. (19) and by noting that $\ddot{\boldsymbol{q}}_r = \ddot{\boldsymbol{q}}_d + \boldsymbol{J}_g^+ \boldsymbol{K}_p (\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \dot{\boldsymbol{J}}_g^+ \boldsymbol{K}_p (\boldsymbol{p}_d - \boldsymbol{p})$, the following expression can be obtained for the velocity-based controller:

$$\boldsymbol{\tau} = \boldsymbol{M}^* \left(\ddot{\boldsymbol{q}}_d + \boldsymbol{J}_g^+ \boldsymbol{K}_p (\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \dot{\boldsymbol{J}}_g^+ \boldsymbol{K}_p (\boldsymbol{p}_d - \boldsymbol{p}) \right) + \boldsymbol{C}^* + \boldsymbol{K}_d (\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{K}_d \boldsymbol{J}_g^+ \boldsymbol{K}_p (\boldsymbol{p}_d - \boldsymbol{p}) - \boldsymbol{\tau}_{grav} \tag{21}$$

The closed-loop behaviour can be obtained as:

$$\boldsymbol{M}^* \ddot{\boldsymbol{q}} = \boldsymbol{M}^* \left(\ddot{\boldsymbol{q}}_d + \boldsymbol{J}_g^+ \boldsymbol{K}_p (\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \dot{\boldsymbol{J}}_g^+ \boldsymbol{K}_p (\boldsymbol{p}_d - \boldsymbol{p}) \right) + \boldsymbol{K}_d (\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{K}_d \boldsymbol{J}_g^+ \boldsymbol{K}_p (\boldsymbol{p}_d - \boldsymbol{p}) \tag{22}$$

By pre-multiplying $\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}$ to both sides of Eq. (22), the closed loop behaviour can be simplified in:

$$\left(\ddot{\boldsymbol{p}}_d - \ddot{\boldsymbol{p}}\right) + \boldsymbol{K}_p(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \left(\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\boldsymbol{K}_d\boldsymbol{J}_g^{+}\boldsymbol{K}_p + \boldsymbol{J}_g\dot{\boldsymbol{J}}_g^{+}\boldsymbol{K}_p\right)(\boldsymbol{p}_d - \boldsymbol{p}) = -\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\boldsymbol{K}_d(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) \tag{23}$$

These controllers are simple to implement, and the results show that a good overall performance is obtained. However, from the error dynamics analysis, several problems should be highlighted. On the one side, the control law ignores information about the target accelerations and it requires numerical differentiation of the joint reference velocities. Additionally, the proportional gain $\boldsymbol{K}_p$ appears as a task space damping gain, affecting the term multiplying the error velocity; therefore, the task space stiffness and damping cannot be controlled independently. Both issues limit the tracking and impedance performance of such control strategy.

4.2. Acceleration-based control

To solve the aforementioned issues, other two approaches based on acceleration are proposed in this section.

4.2.1. Acceleration-based controller with inertia matrix pre-multiplication

The acceleration-based controller presented in this section generates an acceleration command in the Cartesian space. Therefore, in this case, the reference can be expressed as:

$$\ddot{\boldsymbol{p}}_r = \ddot{\boldsymbol{p}}_d + \boldsymbol{K}_d(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p}) \tag{24}$$

where $\boldsymbol{p}_d$, $\dot{\boldsymbol{p}}_d$ represent the desired position and velocity corresponding to the Cartesian trajectory to be tracked, and $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are proportional and derivative positive definite matrices, respectively. The acceleration-based control action can be obtained considering the system dynamics defined in (8):

$$\boldsymbol{\tau} = \boldsymbol{M}^*\ddot{\boldsymbol{q}}_r + \boldsymbol{C}^* - \boldsymbol{\tau}_{grav} \tag{25}$$

where $\ddot{\boldsymbol{q}}_r$ can be obtained from (15):

$$\ddot{\boldsymbol{q}}_r = \boldsymbol{J}_g^{+}\left(\ddot{\boldsymbol{p}}_r - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}} - \dot{\boldsymbol{v}}_{gm}\right) + \left(\boldsymbol{I} - \boldsymbol{J}_g^{+}\boldsymbol{J}_g\right)\boldsymbol{\upsilon}_a \tag{26}$$

and:

$$\boldsymbol{\upsilon}_a = \boldsymbol{K}_{ps}(\boldsymbol{q}_s - \boldsymbol{q}) - \boldsymbol{K}_{ds}\dot{\boldsymbol{q}}\cdot \tag{27}$$

This controller introduces a null space projection with damping term in joint space. This controller achieves asymptotic tracking in operational space since the following behaviour is obtained in closed loop:

$$\ddot{\boldsymbol{p}}_d - \ddot{\boldsymbol{p}} + \boldsymbol{K}_d(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p}) \tag{28}$$

4.2.2. Acceleration-based controller without inertia matrix pre-multiplication

The final control law obtained in Section 4.2.1 can be summarised in:

$$\boldsymbol{\tau} = \boldsymbol{M}^*\boldsymbol{J}_g^{+}\left(\ddot{\boldsymbol{p}}_r - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}} - \dot{\boldsymbol{v}}_{gm}\right) + \boldsymbol{M}^*\left(\boldsymbol{I} - \boldsymbol{J}_g^{+}\boldsymbol{J}_g\right)\boldsymbol{\upsilon}_a + \boldsymbol{C}^* - \boldsymbol{\tau}_{grav} \tag{29}$$

In the simulations, we noted that the pre-multiplication of the null space optimisation term by the inertia matrix M^* can be problematic if the inertia matrix has modeling inaccuracies. To avoid this problem, the following variation of the controller is proposed:

$$\boldsymbol{\tau} = \boldsymbol{M}^*\boldsymbol{J}_g^{+}\left(\ddot{\boldsymbol{p}}_r - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}} - \dot{\boldsymbol{v}}_{gm}\right) + \left(\boldsymbol{I} - \boldsymbol{J}_g^{+}\boldsymbol{J}_g\right)\boldsymbol{\upsilon}_a + \boldsymbol{C}^* - \boldsymbol{\tau}_{grav} \tag{30}$$

so that the closed loop behaviour can be obtained as:

$$\boldsymbol{M}^*\ddot{\boldsymbol{q}} = \boldsymbol{M}^*\boldsymbol{J}_g^{+}\left(\ddot{\boldsymbol{p}}_r - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}} - \dot{\boldsymbol{v}}_{gm}\right) + \left(\boldsymbol{I} - \boldsymbol{J}_g^{+}\boldsymbol{J}_g\right)\boldsymbol{\upsilon}_a \tag{31}$$

By pre-multiplying $\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}$ to both sides of Eq. (31), the closed loop behaviour can be simplified in:

$$\ddot{\boldsymbol{p}}_d - \ddot{\boldsymbol{p}} + \boldsymbol{K}_d(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p}) = \boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\left(\boldsymbol{I} - \boldsymbol{J}_g^{+}\boldsymbol{J}_g\right)\boldsymbol{\upsilon}_a \tag{32}$$

This last equation shows that the null space projection interferes in the asymptotic tracking in the operational space. However, good practical results are obtained in the application of this controller, as indicated in the results section.

4.3. Force-based control

This section proposes a force-based controller where the reference is given as a command. This information is employed directly to generate the manipulator joint torques. Based on the framework proposed by Khatib [46], the desired task dynamics can be expressed as:

$$\widetilde{\boldsymbol{M}}(\boldsymbol{p})\ddot{\boldsymbol{p}} + \widetilde{\boldsymbol{C}}(\boldsymbol{p},\dot{\boldsymbol{p}}) = \boldsymbol{F} \tag{33}$$

where:

$$\widetilde{\boldsymbol{M}} = \left(\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\boldsymbol{J}_g^T\right)^{-1} \tag{34}$$

$$\widetilde{\boldsymbol{C}} = \left(\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\boldsymbol{J}_g^T\right)^{-1}\left(\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\boldsymbol{C}^* - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}}\right) \tag{35}$$

Considering the desired dynamics imposed by Eq. (33), the control action can be formulated as:

$$\boldsymbol{F} = \widetilde{\boldsymbol{M}}\ddot{\boldsymbol{p}}_r + \widetilde{\boldsymbol{C}} \tag{36}$$

where $\ddot{p}_r$is the cartesian reference that can be obtained as in the acceleration-based controllers using Eq. (24). Therefore, the final control law that generates the joint torques is:

$$\boldsymbol{\tau} = \boldsymbol{J}_g^T\boldsymbol{F} + \left(\boldsymbol{I} - \boldsymbol{J}_g^T\widetilde{\boldsymbol{J}}_g^T\right)\boldsymbol{\upsilon}_f - \boldsymbol{\tau}_{grav} \tag{37}$$

where υ_f can be considered as in Eq. (27) equal to $\boldsymbol{K}_{ps}(\boldsymbol{q}_s - \boldsymbol{q}) - \boldsymbol{K}_{ds}\dot{\boldsymbol{q}}$, and $\widetilde{\boldsymbol{J}}_g$ is the inertia-weighted pseudo-inverse:

$$\widetilde{\boldsymbol{J}}_g = (\boldsymbol{M}^*)^{-1}\boldsymbol{J}_g^T\left(\boldsymbol{J}_g(\boldsymbol{M}^*)^{-1}\boldsymbol{J}_g^T\right)^{-1} \tag{38}$$

As it can be seen in Eq. (37), the operational space part of the controller is decoupled with respect the null space dynamics. Additionally, this controller achieves asymptotic tracking in operational space since the following behaviour is obtained in closed loop:

$$\ddot{\boldsymbol{p}}_d - \ddot{\boldsymbol{p}} + \boldsymbol{K}_d(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p}) \tag{39}$$

Table 1 summarises the proposed Cartesian controllers that will be evaluated and compared in the results section using OnOrbitROS.

4.4. Visual servoing

This paper proposes the use of OnOrbitROS to extend the controllers presented in the previous section to perform the guidance of a humanoid robot in a realistic scenario by using visual servoing. Specifically, a robotic extravehicular activity for the maintenance of the International Space Station (ISS) has been modeled and simulated. A visual servoing strategy guide the arms of a humanoid robot. Therefore, the robot arms will be guided by the visual information extracted by the camera located at the robot head. Fig. 5 illustrates the resulting modeled scenario where the ISS and the humanoid robot are shown. The humanoid robot has two arms with seven degrees of freedom each and robot hands at their end-effectors; the joint coordinates of both arms are denoted as $\boldsymbol{q}_1, \boldsymbol{q}_2 \in \Re^{ne}$, where $ne = 7$. The head of the robot hosts a range camera, which is used to determine the positions of the grasping points. Different handrails are located on the exterior of the ISS, as shown in Fig. 5. The main kinematics and dynamics parameters of the humanoid robot are detailed in the results section (Table 3). This section shows how to extend the Cartesian controllers presented in the previous sections to guide the robot arm to perform the grasping of a given handrail.

As it can be seen in Fig. 5, an Aruco marker is located in the workspace. The Aruco codes are markers widely used for camera localisation in augmented reality, virtual reality and even some applications in on-orbit space applications [47]. This code will be used to obtain the pose of the marker with respect to the robot frame, ${}^r\boldsymbol{T}_m$, by using the camera located at the robot head. In order to perform such an operation the algorithm described in [48] is used. In addition, the transformation between the Aruco marker and the grasping

Table 1
Proposed Cartesian controllers.

Controller	Control action
Velocity-based controller (1)	$\boldsymbol{\tau} = \boldsymbol{M}^*(\ddot{\boldsymbol{q}}_d + \boldsymbol{J}_g^+\boldsymbol{K}_p(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \dot{\boldsymbol{J}}_g^+\boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p})) + \boldsymbol{C}^* + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{K}_d\boldsymbol{J}_g^+\boldsymbol{K}_p(\boldsymbol{p}_d - \boldsymbol{p}) - \boldsymbol{\tau}_{grav}$
Acceleration controller with inertia matrix pre-multiplication (2)	$\boldsymbol{\tau} = \boldsymbol{M}^*(\boldsymbol{J}_g^+(\ddot{\boldsymbol{p}}_r - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}} - \dot{\boldsymbol{v}}_{gm}) + (\boldsymbol{I} - \boldsymbol{J}_g^+\boldsymbol{J}_g)\boldsymbol{\upsilon}_a) + \boldsymbol{C}^* - \boldsymbol{\tau}_{grav}$
Acceleration controller without inertia matrix pre-multiplication (3)	$\boldsymbol{\tau} = \boldsymbol{M}^*\boldsymbol{J}_g^+(\ddot{\boldsymbol{p}}_r - \dot{\boldsymbol{J}}_g\dot{\boldsymbol{q}} - \dot{\boldsymbol{v}}_{gm}) + (\boldsymbol{I} - \boldsymbol{J}_g^+\boldsymbol{J}_g)\boldsymbol{\upsilon}_a + \boldsymbol{C}^* - \boldsymbol{\tau}_{grav}$
Force-based controller (4)	$\boldsymbol{\tau} = \boldsymbol{J}_g^T(\widetilde{\boldsymbol{M}}\ddot{\boldsymbol{p}}_r + \widetilde{\boldsymbol{C}}) + (\boldsymbol{I} - \boldsymbol{J}_g^T\widetilde{\boldsymbol{J}}_g^T)\boldsymbol{\upsilon}_f - \boldsymbol{\tau}_{grav}$

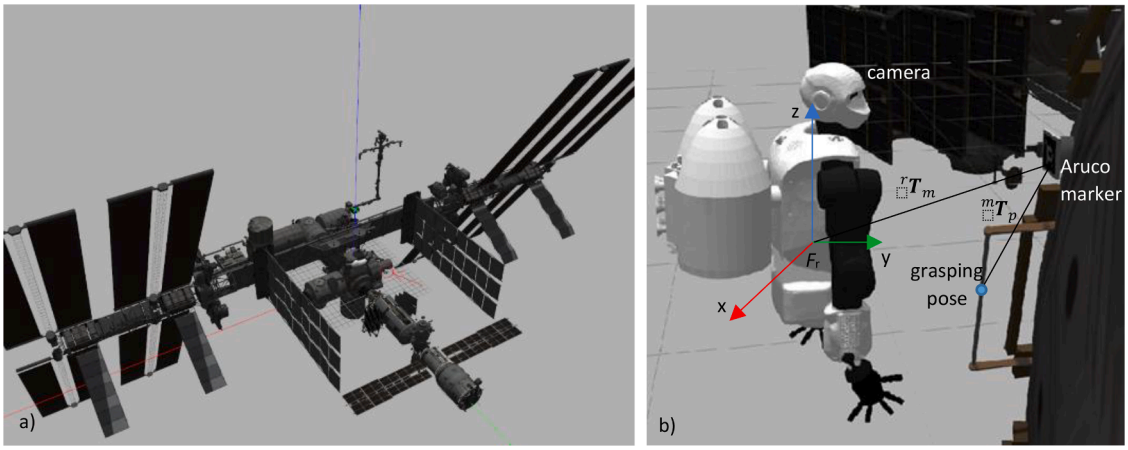


Fig. 5. (a) ISS Gazebo simulation. (b) Simulated humanoid robot with an eye-to-hand camera.

pose, ${}^mT_p$, is fixed and known. Therefore, both transformations can be jointly used to determine the grasping and target pose for the robot hand with respect to the robot frame, ${}^rT_p$. As an example of implementation of a direct position-based visual servoing approach from the task-controllers proposed in Sections 4.1, 4.2 and 4.3, Fig. 6 illustrates the implementation of a position-based visual servoing system considering the acceleration controller without inertia matrix pre-multiplication. The path planning module provides the desired trajectory ($\ddot{\boldsymbol{p}}_d, \dot{\boldsymbol{p}}_d, \boldsymbol{p}_d$). Due the free-floating conditions, the desired Cartesian trajectory to be tracked must be updated depending on the target grasping location for the robot arm (${}^rT_p$, obtained using the camera). The visual feedback computes a quintic spline as the desired Cartesian trajectory between the current pose of the robot manipulator end, and the target position computed by the camera. This last will be the desired Cartesian trajectory to be tracked by the Cartesian controller.

5. Simulation results

This section summarises the main simulation results obtained using OnOrbitROS in different free-floating robotics scenarios.

The first simulation is used for validating the dynamics implemented in OnOrbitROS. Specifically, the dynamics of the ETS-VII robotic experiment is simulated, and results will be compared to the actual flight data obtained from the mission, which can be found in [28,29]. The main kinematic and dynamic parameters of the ETS-VII robotic system are shown in Table 2.

The second scenario is used to show the capabilities of OnOrbitROS as a tool for assessing the performance of the afore-developed task space controllers. In this case, a robotic extravehicular activity is simulated, as shown in Fig. 5. Table 3 summarises the main dynamic parameters of the robot used in this second scenario. This table lists the moment of inertia, the mass, and the dimensions of the main body of the robot and the links of both arms. It is assumed that the two arms are symmetric and have the same dynamic parameters.

Finally, a third simulation is presented to show the OnOrbitROS capabilities in simulating complex tasks, with inputs received by simulated sensors, as in the case of the visual servoing control applied to the humanoid robot as previously described in Section 4.4. The reference orbit is supposed to be circular and with a radius of $r_{orb} = 6878\ km$. In the current version of OnOrbitROS only the relative dynamic and the gravity gradient are included in the simulations. Even if other perturbations, such as the drag and the solar radiation pressure effects are not considered, the simulations can still be considered a reasonably good approximation of the real conditions. Indeed, given the very low area-to-mass ratio that strongly characterises the robots (7,19,569 e-3 m^2Kg^{-1} for the humanoid

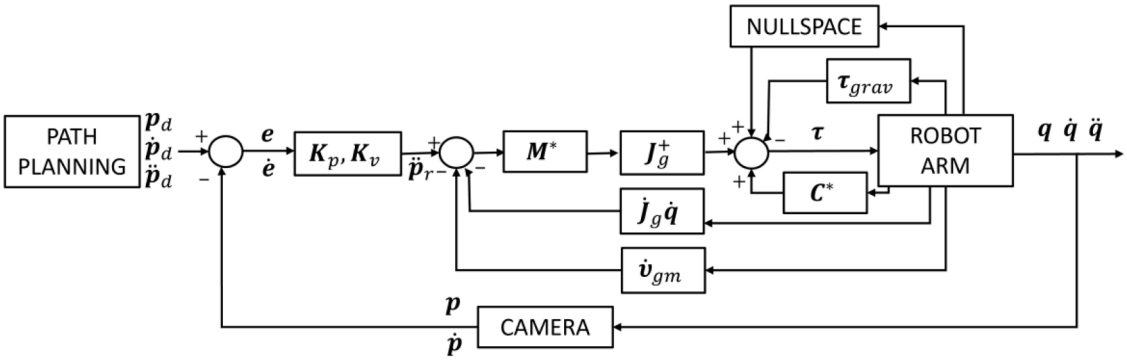


Fig. 6. Control scheme of the position-based visual servoing using the acceleration controller without inertia matrix pre-multiplication.

Table 2
Mass and inertia parameters of the ETS-VII robot.

Base Mass (Kg)	Inertia (kg•m^2) I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
2552	6206	3541	7087	48.16	78.52	−29.22

Arm	Inertia (kg•m^2) Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
Mass (kg)	35.01	22.45	21.89	16.54	26.00	18.49
Inertia I_{zz} (kg•m^2)	1.69	3.75	2.53	0.072	0.129	0.259

Table 3
Mass and inertia parameters of the humanoid robot.

Body	Mass (kg)	Height (m)	Inertia (kg•m^2) I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
	93	0.843	18.6	15.4	4.1	-0.008	-0.027	0.058
Arms	Mass (kg)	Length (m)	Inertia (kg•m^2) I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
Link 1	2.741	0.28	0.0124	0.0042	0.0136	3.6e-05	7.1e-05	-0.0002
Link 2	2.425	0.144	0.013	0.0138	0.0049	1.2e-05	-0.0032	-0.0001
Link 3	2.209	0	0.007	0.0069	0.0039	-0.0001	0.0007	0.0004
Link 4	0.877	0.274	0.0025	0.0027	0.0012	0.0001	-0.0003	0.0004
Link 5	1.878	0.265	0.0035	0.0044	0.0023	1.3e-05	1.03e-05	-9.7e-05
Link 6	0.409	0	0.0001	0.00014	0.00015	-8.9e-08	-4.4e-08	4.2e-07
Link 7	0.308	0	0.0003	0.0002	0.00017	-1.6e-06	1.7e-06	-1.2e-05

robot and 4,898,119 e-4 m^2Kg^{-1} the ETS-VII robotic system), the effects of atmospheric drag are negligible with respect to gravity effects.

5.1. Scenario 1. ETS-VII gravity gradient simulation

This section considers the same scenario presented in [28,29] to validate the proposed simulation system based on Gazebo, ROS and OnOrbitROS. The ETS-VII flight data will be compared with the one obtained by using OnOrbitROS for the same trajectories. Specifically, in [28] a ETS-VII trajectory is presented with the joint evolution represented in Fig. 7. The 3D representation obtained using Gazebo and OnOrbitROS is shown in Fig. 8. Fig. 8a represents the initial robot pose and Fig. 8b the final one. Fig. 9 illustrates the attitude behavior of the robot base, represented by the three Euler angles roll, pitch and yaw, when applying the joint trajectory indicated in Fig. 7. Fig. 9a represents the attitude of the base in the case the gravity gradient effect is not considered. As the robot does not receive any external forces or moments, the attitude at the beginning and the end of the trajectory is the same, as expected from the conservation of the angular momentum. As described in Section 3.3, OnOrbitROS allows for simulating more realistic scenarios that take the gravity gradient effect into consideration. Fig. 9b represents the attitude of the base of the robotic system in the case of a simulation that includes the gravity gradient effect. The effects of such disturbance are evident in this figure, where the roll angle reaches a slightly lower value and the pitch angle a higher value compared to the simulation without gravity gradient. The same effects

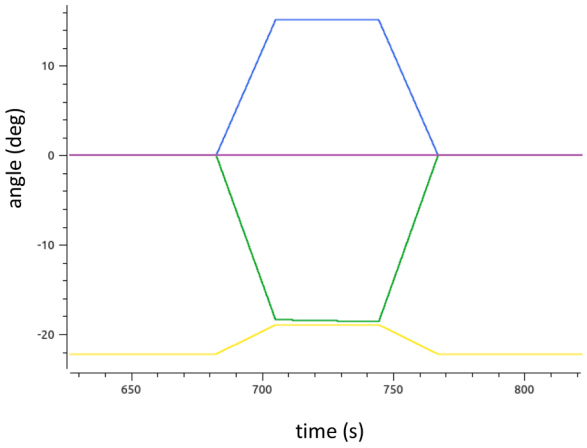


Fig. 7. ETS-VII joint trajectory described in [28].

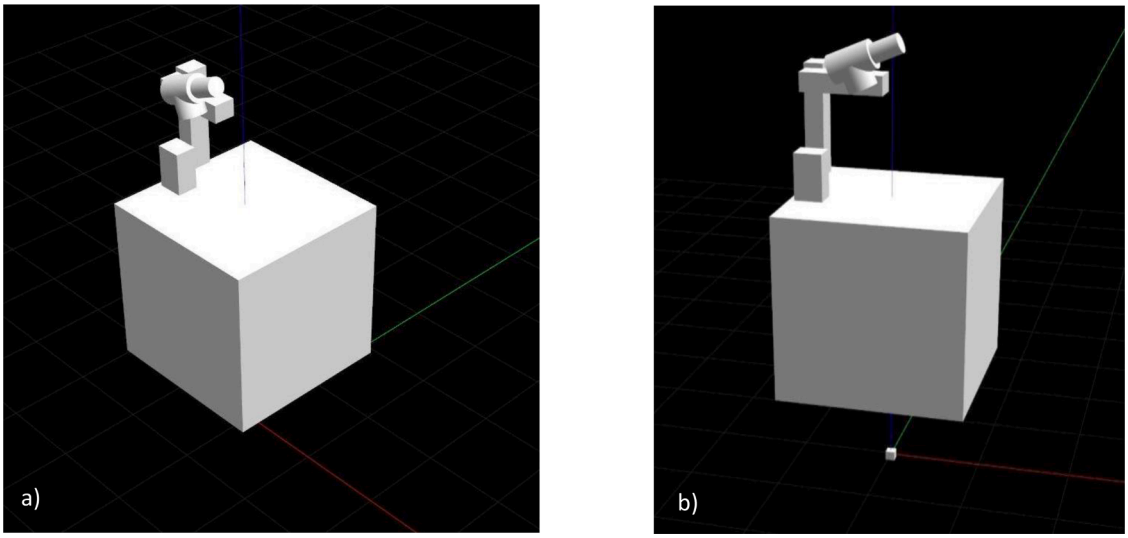


Fig. 8. 3D trajectory of the robot. (a) Initial robot pose. (b) Final robot pose.

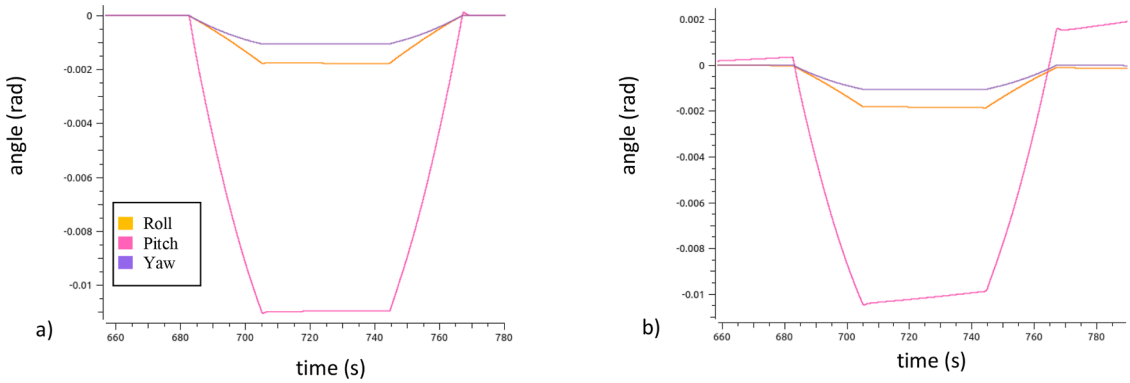


Fig. 9. (a) Base attitude without considering gravity gradient effect. (b) Base attitude considering gravity gradient effect.

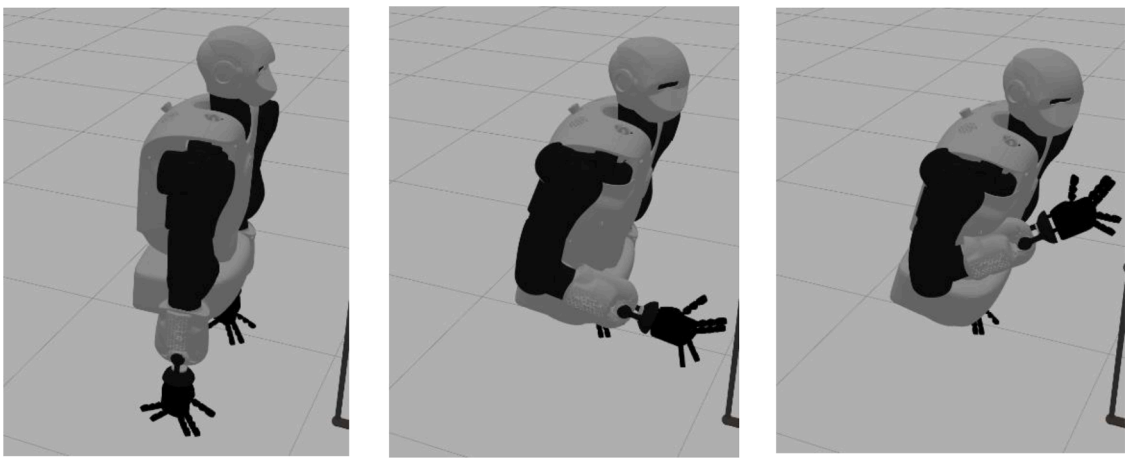


Fig. 10. 3D trajectory of the robot. (a) Initial pose. (c) Intermediate pose. (c) Final pose.

and trends are also observed in the real data of the ETS-VII robot presented in [29]. The comparison of the results obtained by the proposed simulation system shows very good agreement with the real flight data indicated in [29], and this provides a tangible validation of the gravity gradient simulation tools in OnOrbitROS.

5.2. Scenario 2. Control of the robot arm of a humanoid robot

This section evaluates the performance of the four proposed controllers, whose final expression is indicated in Table 1, during the tracking of the trajectory represented in Fig. 10. This last figure represents the robot at the initial, intermediate, and final configurations. This trajectory is performed in 10.2 s. The proportional and derivative matrices used in these simulations are indicated in Table 4, where $diag() \in \Re^{7 \times 7}$ is a matrix with diagonal elements equal to the argument of the function.

Fig. 11 represents, the desired trajectory (black) and the one obtained during the tracking for each controller (red). All four controllers reach the desired final pose by tracking the desired trajectory. To highlight more clearly the differences between each controller, Fig. 12 represents the control error, $\boldsymbol{p}_d(t) - \boldsymbol{p}(t)$, for each controller during the tracking and Fig. 13 represents the control actions (joint torques) during the tracking. Although the torques remain low during the tracking, some differences can be observed in the behaviors of the four proposed controllers. Among all the controllers, the acceleration-based controller without inertia matrix pre-multiplication (controller 3) is the most promising approach in terms of tracking task performance, ease of parameter tuning, and general robustness and compliance. The experimental results demonstrate the effectiveness of this controller in face of inevitable modeling errors. On the other hand, other velocity/acceleration/force-based controllers had several performance problems, as will be described in the next paragraphs.

The proposed velocity-based controller was straightforward to implement and achieved overall good performance. However, as indicated in Section 4.1, it ignores information about the target acceleration. Additionally, there is a practical limitation in the choice of the proportional gain because the task space position gain is also included as a task space damping gain. This effect implies that an increase in the position gain also increases the damping gain. Therefore, the proportional gain cannot be increased without affecting the stability of the system. The damping behaviour is indeed affected by the proportional gain and this behaviour becomes difficult to tune. As it can be seen in Fig. 12, small oscillations appear during the tracking when the velocity-based controller is applied, while acceleration and force-based controllers present smoother behaviors.

Two acceleration-based approaches have been proposed in Section 4.2 (with and without inertia matrix pre-multiplication). As already mentioned, the second approach, the acceleration-based controller without inertia matrix pre-multiplication, presents an excellent and robust tracking performance. From an empirical point of view, the pre-multiplication of the null space optimisation term by the inertia matrix can introduce tracking errors due to the inevitable modeling inaccuracies. This aspect has motivated the implementation of the new acceleration-based controller without inertia matrix pre-multiplication. As indicated in Section 4.2.2, the null space projection interferes with the asymptotic tracking in the operational space. However, a lower tracking error is obtained by using this approach with a smoother behaviour in the joint space.

As described in Section 4.3, the force-based controller uses the inertia-weighted pseudo-inverse and it allows the decoupling between null space dynamics and the operational control. However, a better behaviour is obtained in accelerations-based approaches. This effect should be due to possible inaccuracies in the computations of the inertia matrix and its inversed that is used in many different terms of the proposed control law.

5.3. Scenario 3. Visual servoing using a complex humanoid robot

This third scenario shows different possibilities of OnOrbitROS for on-orbit sensor-based applications. In this case, the direct position-based visual servoing system described in Section 4.4 is used for the guidance of the arms of a humanoid robot in a grasping application. To illustrate the implementation of this control approach, the grasping trajectory shown in Fig. 14 is considered. Fig. 15a represents the pose of the Aruco marker with respect to the robot frame, ${}^{r}\boldsymbol{T}_{m}$, by using the camera located at the robot head. This pose changes during the grasping task due to the free-floating conditions. The end-effector trajectory during the tracking is represented in Fig. 15b. To achieve the desired grasping point while the manipulator-end tracks the desired trajectory, an acceleration controller without inertia matrix pre-multiplication is applied, and the control actions represented in Fig. 15c are obtained. The tracking error is represented in Fig. 15d. As it can be seen in this last figure, the tracking error remains low during the trajectory, allowing accurate tracking with the eye-to-hand camera feedback. Table 5 shows the mean error obtained during the tracking when different direct position-based visual servoing systems are applied based on the proposed task space controllers. This table also shows the mean error obtained by using previous indirect position-based visual servoing systems [39]. It is worth noting that lower tracking errors are obtained when the proposed controller based on the acceleration controller without inertia matrix pre-multiplication is applied.

Table 4
Cartesian controllers' gains.

Controller	$\boldsymbol{K}_p$	$\boldsymbol{K}_d$
Velocity-based controller (1)	$diag(0.1)$	$diag(0.5)$
Acceleration controller with inertia matrix pre-multiplication (2)	$diag(0.5)$	$diag(10)$
Acceleration controller without inertia matrix pre-multiplication (3)	$diag(0.5)$	$diag(10)$
Force-based controller (4)	$diag(\sqrt{0.5})$	$diag(\sqrt{10})$

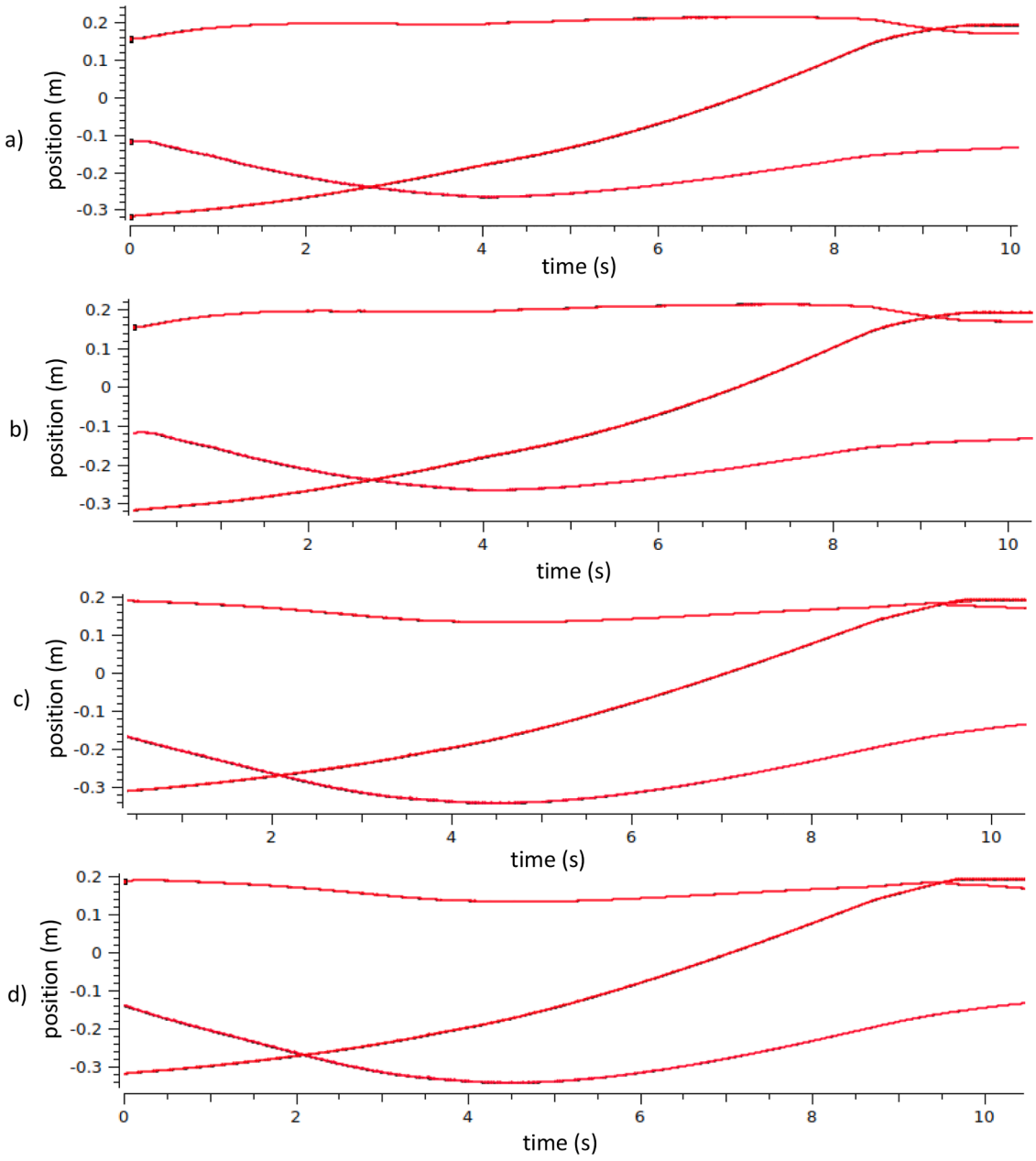


Fig. 11. Desired (black) and obtained 3D trajectory (red) during the tracking. (a) Velocity-based controller. (b) Acceleration controller with inertia matrix pre-multiplication. (c) Acceleration controller without inertia matrix pre-multiplication. (d) Force-based controller.

Finally, Fig. 16 shows a detail of the robot hand in the grasping position, where we can observe that the achieved location allows the correct grasping of the handrail. The OnOrbitROS-Gazebo simulation of this experiment can be seen in [49].

6. Conclusions

The paper presented the architecture of the different software modules of OnOrbitROS, a framework to simulate complex space robotic systems, also taking advantage of the number of packages already developed in ROS for control, vision, teleoperation, and modelling tools. The tool was validated by directly comparing the simulations and the flight data of the ETS-VII robot experiment.

In order to show the different applications of such tools, the paper also offers a case study where several task space control strategies were developed and traded off in an OOS scenario with a humanoid robot performing extravehicular operations around the ISS. Different velocity-based, acceleration-based and force-based approaches were evaluated. In contrast with classical position-based visual servoing systems, a direct position-based visual servoing scheme was proposed to integrate robot dynamics and sensor

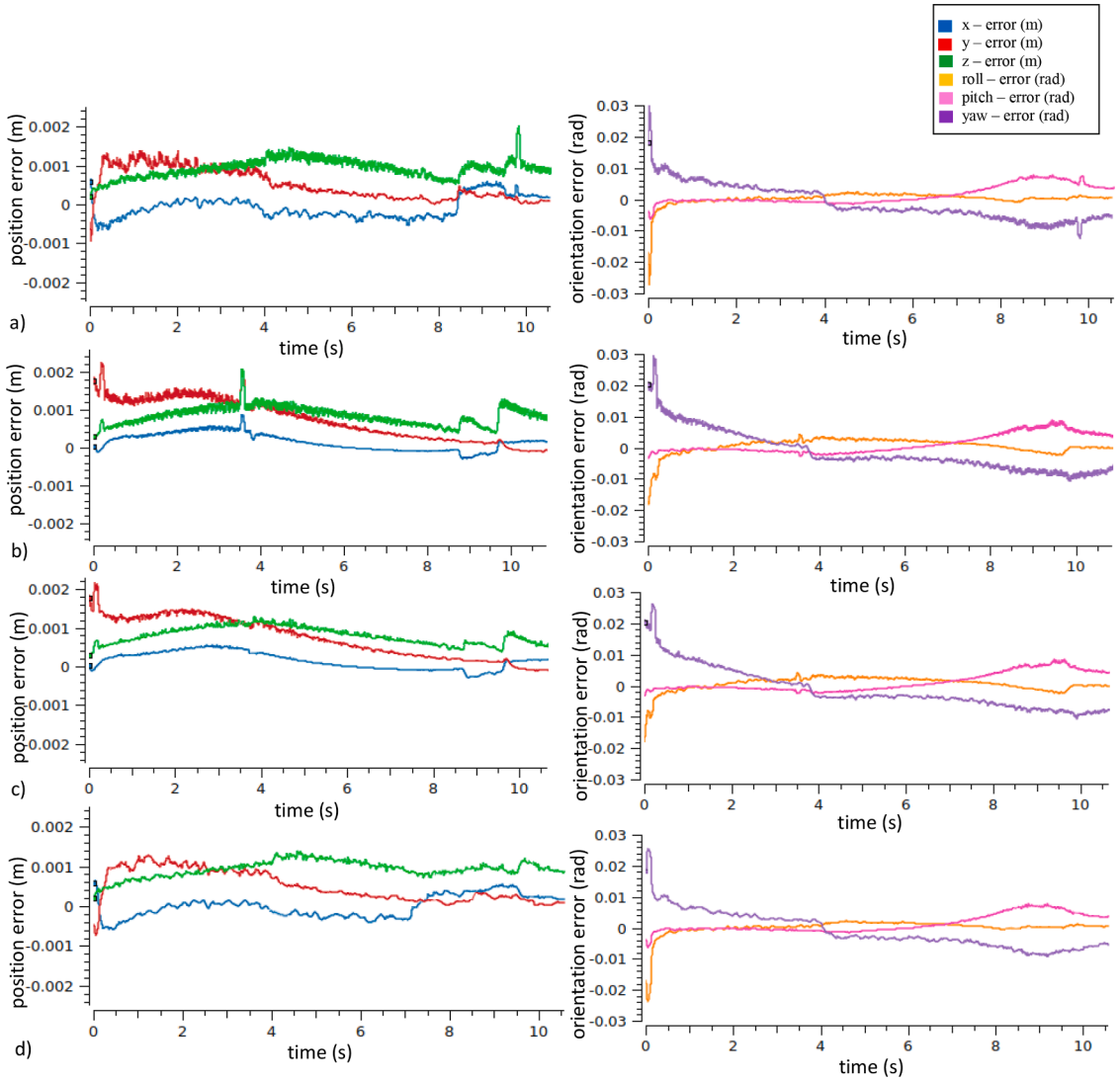


Fig. 12. Control error during the tracking. (a) Velocity-based controller. (b) Acceleration controller with inertia matrix pre-multiplication. (c) Acceleration controller without inertia matrix pre-multiplication. (d) Force-based controller.

measurements within the controllers. An acceleration controller without inertia matrix pre-multiplication was selected as the basis for implementing the proposed visual servoing system, given its more accurate performance in the simulation results compared to the other presented controllers.

Couplings between the relative orbit dynamics and the gravity gradient effects have been included in the simulation system. However, we are working on adding new features to OnOrbitROS to increase the realism of the simulations. Within these new features, it is worth mentioning the inclusion of other perturbations into the physical engine, such as the implementation of differential drag effects in the simulation. Additionally, in future works, the implementation of reinforcement learning approaches for robot guidance will also be considered.

Appendix A

This Appendix describes the implementation of the *Simple Orbit* module. This module obtains the frame position and orientation of $\boldsymbol{F}_{\mathrm{l}}$ with respect the inertial frame, $\boldsymbol{t}_l$ and $\boldsymbol{R}_{\mathrm{l}}$, and its linear velocity, $\boldsymbol{\nu}_L$, as a function of the typical Kepler parameters that define an orbit. *Simple orbit* is valid for elliptical orbits with eccentricity values equal to $0 \leq e < 1$. Earth orbits are considered, so the following constants are used in the calculations: the Kepler constant is defined as $\mu = 3.986004415 \times \frac{10^5\ \mathrm{Km}^3}{\mathrm{s}^2}$, the radius of the Earth is $r_{\mathrm{e}} = 6378$ Km, its angular velocity is defined as $om_e = 2\pi\ 3600 \times 24\ \frac{rad}{s}$ and the spherical harmonic coefficient of second degree is $j2 = 0.0010826269 \times 10^{-3}$. On the other hand, through the ROS Parameter Server the user can define the simulated orbit by indicating

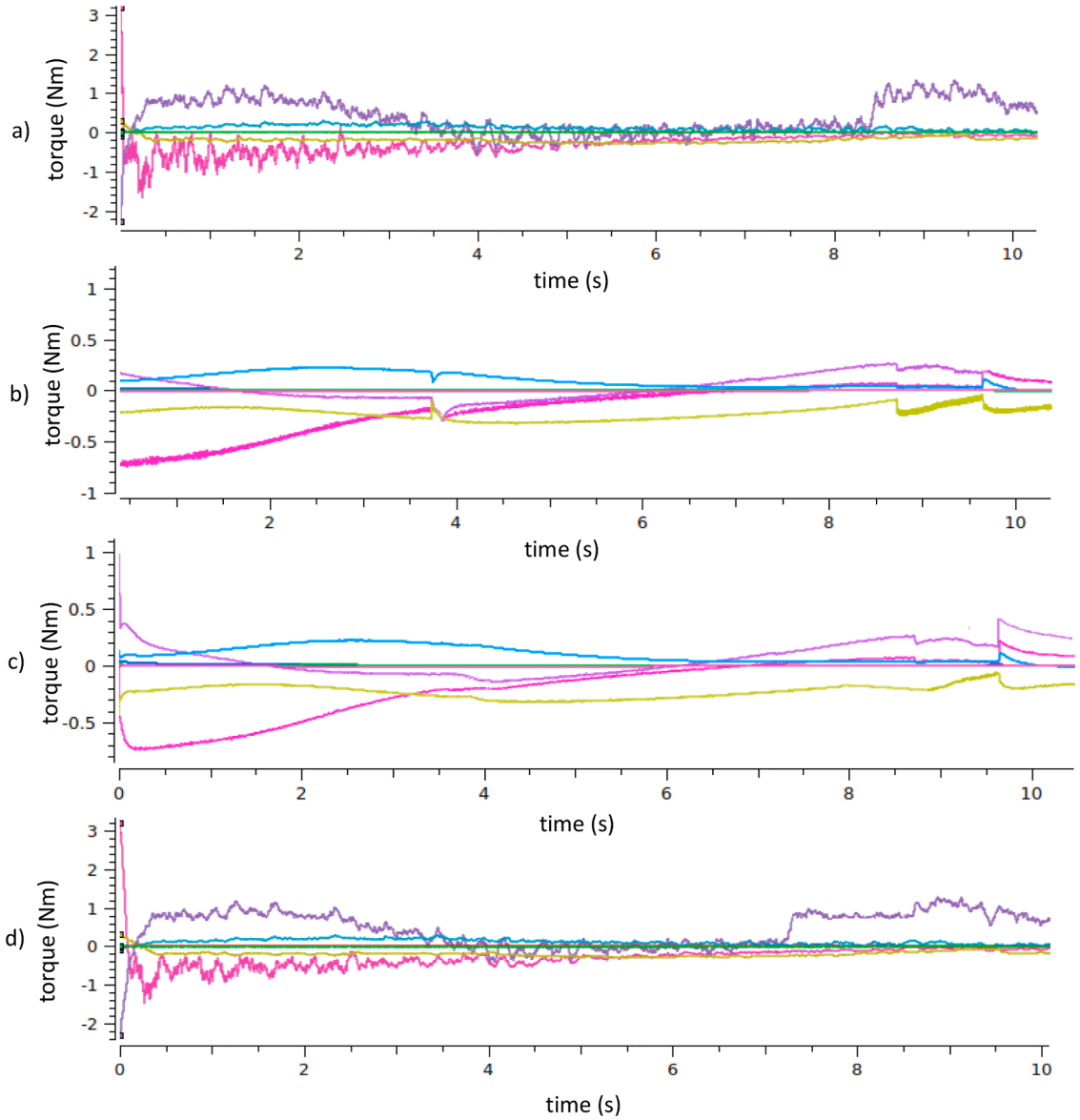


Fig. 13. Control actions during the tracking. (a) Velocity-based controller. (b) Acceleration controller with inertia matrix pre-multiplication. (c) Acceleration controller without inertia matrix pre-multiplication. (d) Force-based controller.

the following parameters corresponding to the described orbit $\{a,\ e,\ \omega,\ \Omega,\ i,\ t_p\}$. These parameters are the semi major axis, a, eccentricity, e, argument of periapsis, ω, right ascension of ascending node, Ω, inclination, i, and instant of time of perigee passage, t_p. Fig. 17 represents the notation of the main orbit parameters considered throughout the paper, where Θ is the true anomaly.

This module publishes, at each simulation time, the position, t_l, velocity, velocity, $\boldsymbol{\nu}_L$, and orientation $\boldsymbol{R}_l$, corresponding to the orbit parameters indicated by using the ROS Parameter Server. To offer a greater flexibility, two different time parameters, t_p and t_0, can also be indicated by using the ROS Parameter Server. The first one, t_p, is the time instant of passage through the perigee, and the second one, t_0, is an offset which corresponds to the duration, in seconds, from the theoretical passage of the satellite through perigee to the instant when the simulation starts. On the other hand, the time instant, t_a, is considered as the current simulation time. The relationship between the simulation time and the real time is configurable, so that it offers the possibility to simulate both, long periods of time in a few instants, as well as simulations of a few seconds at very high sampling and integration frequencies. The relationship between real and simulated time takes place in the physics engine so it is outside the scope of this module, but the parameter update frequency must be configured accordingly.

From the previously indicated parameters describing the orbit and the timing parameters (defined by the user in the ROS Parameter Server), the module *Simple Orbit* publishes, at each simulation time, the position, t_l, velocity, $\boldsymbol{\nu}_L$, and orientation $\boldsymbol{R}_l$, corresponding to the orbit parameters. To do this, first, the values necessary for the propagation of the orbit that do not depend on the time are

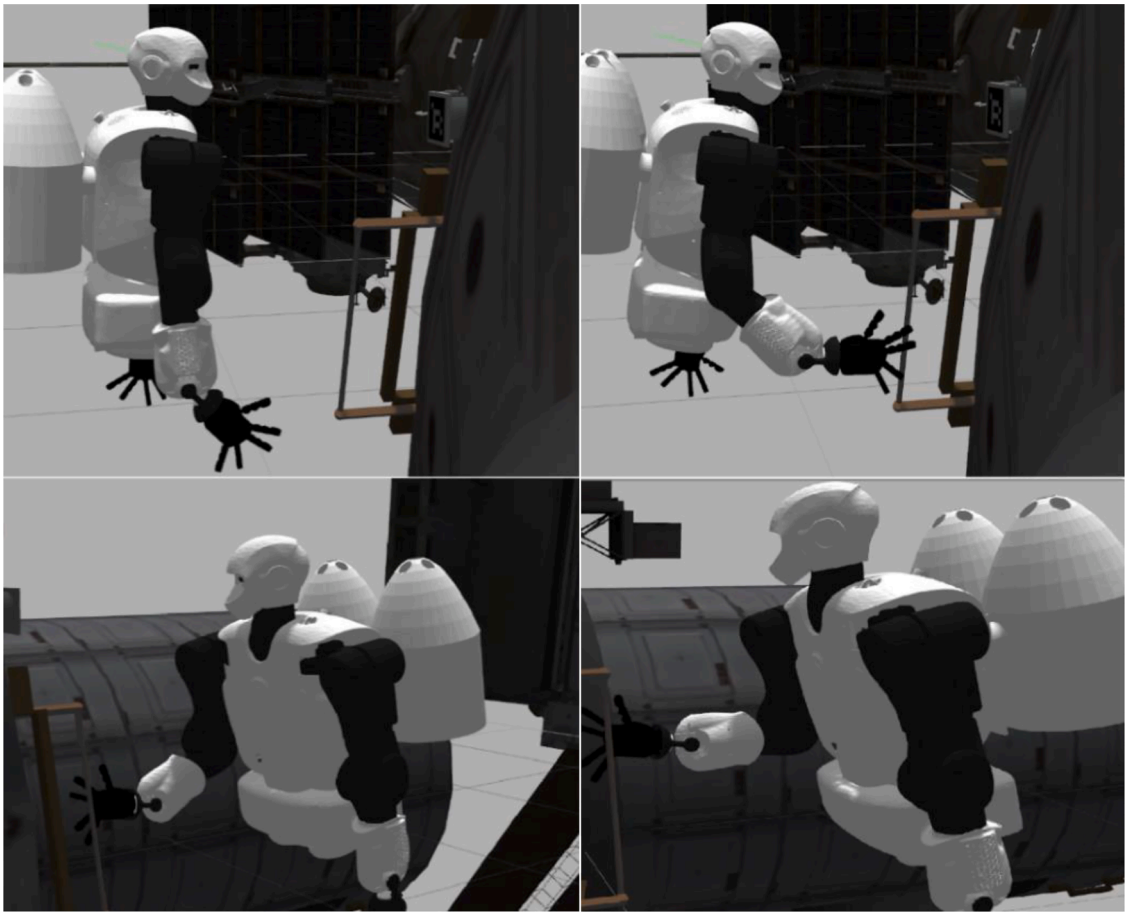


Fig. 14. OnOrbitROS simulation of the visual servoing task of a humanoid robot.

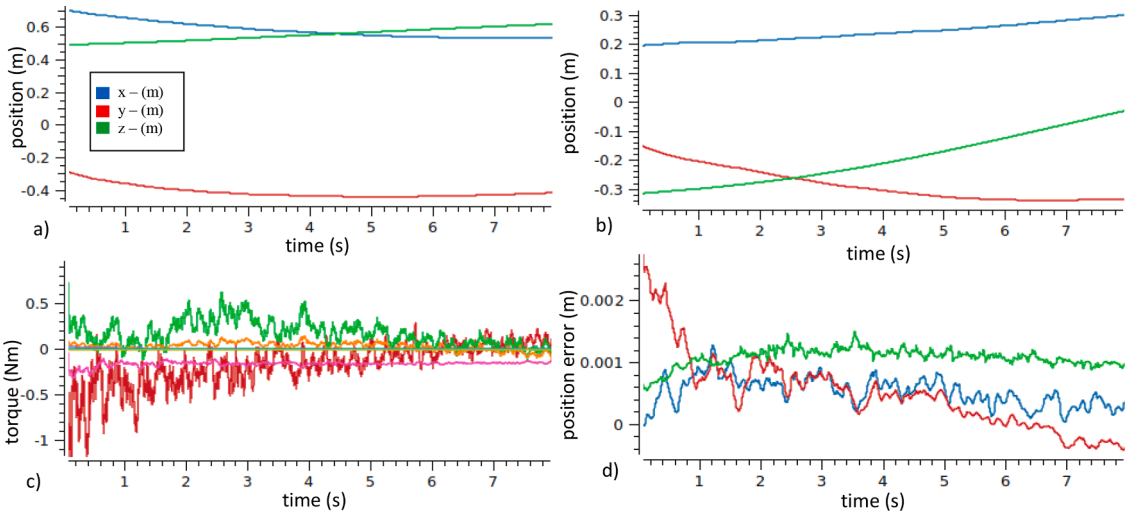


Fig. 15. (a) Position of the Aruco marker with respect the robot frame, ${}^{r}\boldsymbol{T}_{m}$. (b) End-effector trajectory during the tracking. (c) Control actions during the tracking. (d) Tracking error.

Table 5
Mean error obtained using different position-based visual servoing approaches.

Direct position-based visual servoing based on	Mean error (m)
Velocity-based controller	$1.5E-2$
Acceleration controller with inertia matrix pre-multiplication	$2.5E-3$
Acceleration controller without inertia matrix pre-multiplication	$1.1E-3$
Force-based controller	$4.2E-3$
Indirect position-based visual servoing	$4.7E-2$

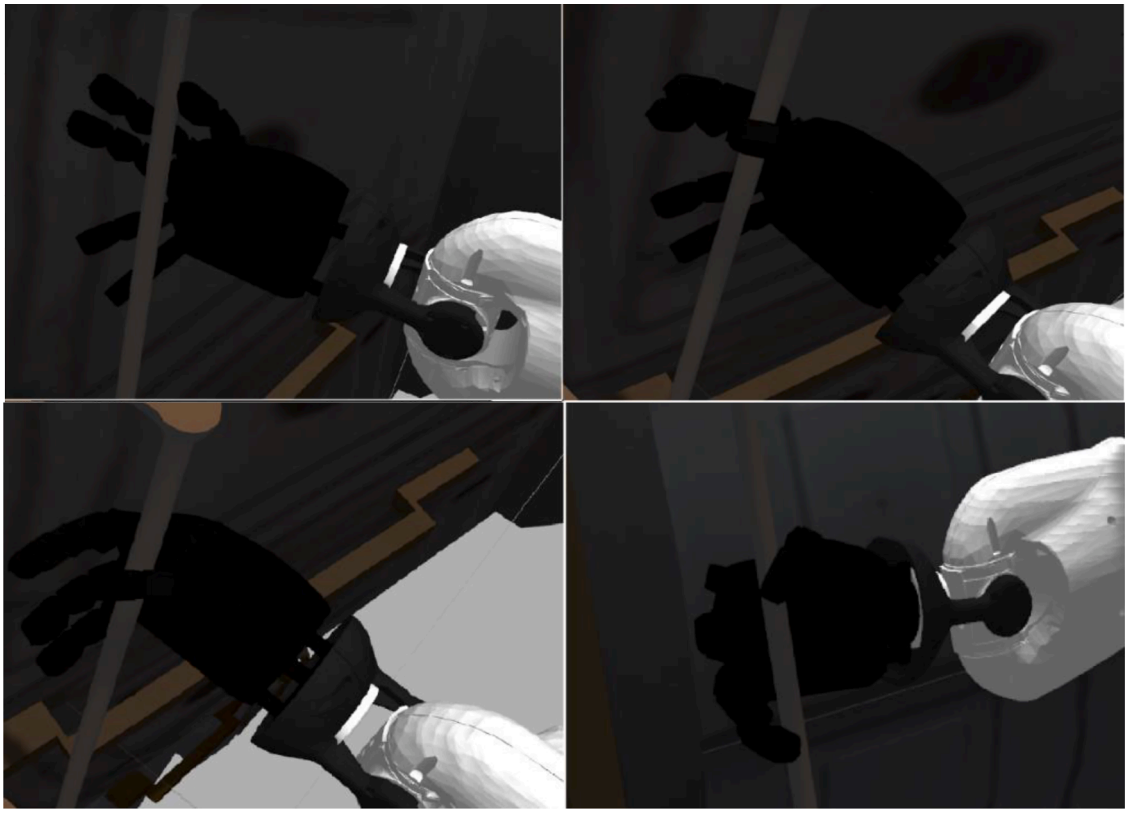


Fig. 16. Detail of the robot hand in the grasping position.

calculated when the node is created. This allows the use of this information in successive iterations for code optimisation. The mean orbital velocity, n, is obtained by using the following expression:

$$n = \sqrt{\frac{\mu}{a^3}} \tag{40}$$

To compensate for the difference between the real shape of the Earth with respect to a perfect sphere, the second harmonic $j2$ compensation has been introduced. Based on this a value closer to the real one of Ω and ω can be obtained. The ellipse semi-parameter and the time derivative of these last previous variables, $\dot{\Omega}$ and $\dot{\omega}$, can be obtained as:

$$\rho = a\left(1 - e^2\right) \tag{41}$$

$$\dot{\Omega} = -\frac{3}{2}\, J2\, \frac{r_e^2}{\rho} \cos i\, n \tag{42}$$

$$\dot{\omega} = \begin{cases} 0 & e \leq 1 \times 10^{-4} \\ \frac{3}{2}\, \frac{J2}{e\left(\frac{r_e}{\rho}\right)^2}\, 2 - \frac{5}{2}\, \sin^2 in & e > 1 \times 10^{-4} \end{cases} \tag{43}$$

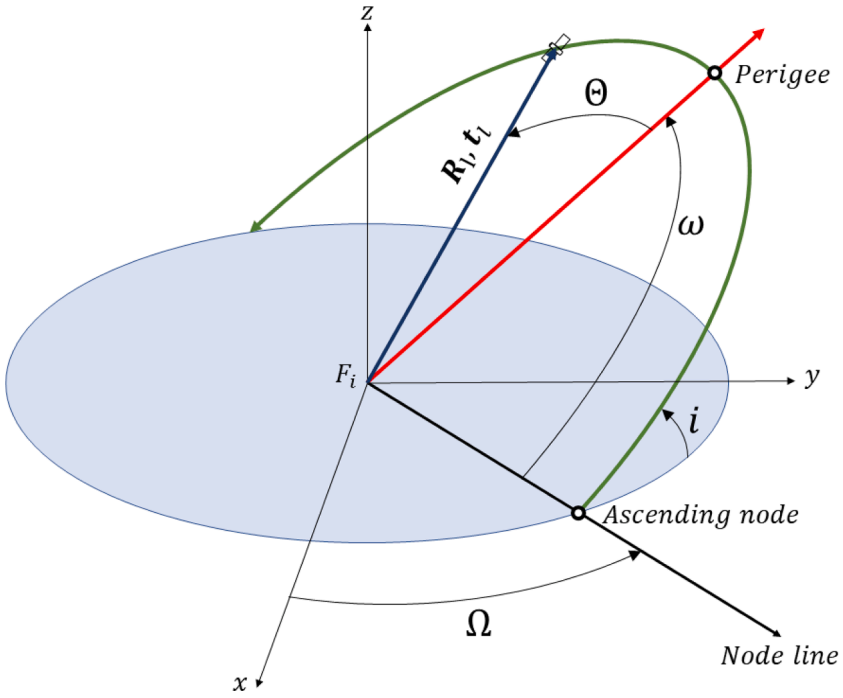


Fig. 17. Orbit parameters.

After obtaining the parameters that do not vary over time, the time-depending computations are iteratively performed as is described in the next paragraphs. The simulation time, t_a, is obtained from the simulation clock. This fact is very important to synchronise the simulations when the simultaneous simulations of several spacecraft are performed. Therefore, the current mean anomaly, $m(t)$, is:

$$m(t) = m(t_0) + n\left(t_a - t_p\right) \tag{44}$$

where $m(t_0)$ is the initial mean anomaly. The method used to obtain the eccentric anomaly $\epsilon(t)$ is the one described in [50] which is mainly based on the Kepler expression:

$$m(t) = \epsilon(t) - e\ \sin\epsilon(t) \tag{45}$$

Eq. (45) is a transcendental equation, so the value of $\epsilon(t)$ should be iteratively obtained. This method is based on starting from an initial value (as approximate as possible) and then iteratively approaching it more and more to its real value. A simple approximation would be to consider that for values of eccentricity close to zero, the value of $\epsilon(t)$ will be equal to $m(t)$. Function (45) is redefined to obtain values closer to the real one in an iterative way:

$$\epsilon_k(t) = m(t) + e\ \sin\epsilon_{k-1}(t) \tag{46}$$

considering then $\epsilon_0(t) = m(t)$ and were $\epsilon_1(t) = m(t) + e\ \sin m(t)$ and continue iterating until the following expression is obtained:

$$\epsilon(t) = m(t) + e\ \sin m(t) + e^2 \sin m(t)\cos m(t) + \frac{1}{2}\, e^3 \sin m(t)\left(3cos^2 m(t) - 1\right) \tag{47}$$

Once an initial value has been obtained, an iterative method should be defined that tries to minimise the error ε in the approximation of $\epsilon(t)$, that relates $\epsilon_k(t) = \epsilon_{k-1}(t) - \varepsilon$. In order to get $\epsilon(t)$ to converge more quickly, it is used the first three terms of the Taylor series of the function $f(x) = x - e\sin x - m(t)$ about $x = \epsilon(t)$, where $f(\epsilon(t)) = (x - \varepsilon) - e\sin(x - \varepsilon) - m(t)$. In this way we use as initial value of x_0 the value of ϵ(t) obtained in the expression (47) and the next values are obtained from $x_k = x_{k-1} - \varepsilon_k$ and the values of ε_k comes from the first three terms of the Taylor series:

$$\varepsilon_{k+1} = \frac{x_k - e\sin x_k - m(t)}{1 - e\cos x_k - \frac{1}{2}\left(e\sin x_k - \frac{1}{3}\, e\cos x_k\ \varepsilon_k\right)\varepsilon_k} \tag{48}$$

When the error ε_k is lower than a defined threshold, the algorithm ends by returning the value of ϵ_t with which the true anomaly value, Θ_t is obtained by using the well know expression:

$$\Theta_t = tan^{-1}\left(\frac{\frac{sin\epsilon_t\sqrt{1-e^2}}{1-ecos\epsilon_t}}{\frac{cos\epsilon_t - e}{1-ecos\epsilon_t}}\right) \tag{49}$$

The distance between the centre of the Earth and the current position of the spacecraft is defined as $\boldsymbol{t}_l$ whose module is equal to:

$$\| \boldsymbol{t}_l \| = a\ (1 - ecos\epsilon_t) \tag{50}$$

To position the vector $\boldsymbol{t}_l$ in the orbital plane, the angle at which it is located with respect to the initial perigee is defined as:

$$\xi(t) = \theta(t) + \omega_0 + \dot{\omega}\ (t_a - t_p) \tag{51}$$

Where Cartesian coordinates of $\boldsymbol{t}_l$ in the orbital plane are as follows:

$$x_p = \| \boldsymbol{t}_l \| cos\xi(t) \tag{52}$$

$$y_p = \| \boldsymbol{t}_l \| sin\xi(t) \tag{53}$$

The corresponding velocity based on angular momentum is equal to:

$$h = \sqrt{\mu\ a\ (1 - e^2)} \tag{54}$$

$$\dot{x}_p = \frac{\mu}{h} - sin\theta(t) \tag{55}$$

$$\dot{y}_p = \frac{\mu}{h}(cos\theta(t) + e) \tag{56}$$

From the previous position and velocity obtained in the orbital plane, it is possible to obtain the value of $\boldsymbol{R}_l$ and $\boldsymbol{t}_l$ applying the required rotations. To do this, first, the resulting rotation matrix from Ω and i can be computed as:

$$\boldsymbol{R}_p = \boldsymbol{R}_z(\Omega)\boldsymbol{R}_x(i) \tag{57}$$

From this rotation matrix, the translation of the LVLH frame, $\boldsymbol{t}_l$, and the linear velocity, $\boldsymbol{\nu}_l$, is:

$$\boldsymbol{t}_l = \boldsymbol{R}_p \begin{bmatrix} x_p \\ y_p \\ 0 \end{bmatrix} \tag{58}$$

$$\boldsymbol{v}_l = \boldsymbol{R}_p \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ 0 \end{bmatrix} \tag{59}$$

Finally, the rotation matrix that represents the attitude of $\boldsymbol{F}_l$ with respect the inertial frame is equal to:

$$\boldsymbol{R}_l = \boldsymbol{R}_p\boldsymbol{R}_\xi \tag{60}$$

References

[1] C. Blackerby, A. Okamoto, S. Iizuka, Y. Kobayashi, K. Fujimoto, Y. Seto, S. Fujita, T. Iwai, N. Okada, J. Forshaw, A. Bradford, The ELSA-d end-of-life debris removal mission: preparing for launch, in: Proceedings of the 70th International Astronautical Congress (IAC), Washington DC, US, 2019.

[2] R. Biesbroek, S. Aziz, A. Wolahan, S. Cipolla, M. Richard-Noca, L. Piguet, The clearspace-1 mission: ESA and clearspace team up to remove debris, in: Proceedings of the 8th European Conference on Space Debris, Darmstadt, Germany, 2021, pp. 20–23.

[3] N.T. Redd, Bringing satellites back from the dead: mission extension vehicles give defunct spacecraft a new lease on life - [News], IEEE Spectr. 57 (2020) 6–7, https://doi.org/10.1109/mspec.2020.9150540.

[4] B.R. Sullivan, J. Parrish, G. Roesler, Upgrading in-service spacecraft with on-orbit attachable capabilities, in: Proceedings of the AIAA SPACE and Astronautics Forum and Exposition, Reston, Virginia, American Institute of Aeronautics and Astronautics, 2018.

[5] M.A. Shoemaker, M. Vavrina, D.E. Gaylor, R. Mcintosh, M. Volle, J. Jacobsohn, OSAM-1 decommissioning orbit design, in: Proceedings of the AAS AIAA Astrodynamics Specialist Conference, 2020, pp. 20–460.

[6] F. Rems, H. Frei, E.A. Risse, M. Burri, 10-year anniversary of the European proximity operations simulator 2.0—looking back at test campaigns, rendezvous research and facility improvements, Aerospace 8 (2021) 235, https://doi.org/10.3390/aerospace8090235.

[7] M. Romano, D.A. Friedman, T.J. Shay, Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target, J. Spacecr. Rockets. 44 (2007) 164–173, https://doi.org/10.2514/1.22092.

[8] M. Wilde, B. Kaplinger, T. Go, H. Gutierrez, D. Kirk, ORION: a simulation environment for spacecraft formation flight, capture, and orbital robotics, in: Proceedings of the IEEE Aerospace Conference, IEEE, 2016.

[9] M. Wilde, C. Clark, M. Romano, Historical survey of kinematic and dynamic spacecraft simulators for laboratory experimentation of on-orbit proximity maneuvers, Prog. Aerosp. Sci. 110 (2019), 100552, https://doi.org/10.1016/j.paerosci.2019.100552.
[10] K.R. da S. Santos, E. Villani, W.R. de Oliveira, A. Dttman, Comparison of visual servoing technologies for robotised aerospace structural assembly and inspection, Robot. Comput. Integr. Manuf. 73 (2022), 102237, https://doi.org/10.1016/j.rcim.2021.102237.
[11] K. Yoshida, The SpaceDyn: a MATLAB toolbox for space and mobile robots, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289), IEEE, 2003.
[12] M. Toso, V. Rossi, M. Robinson, F. Battie, DCAP: assessment of a multi-payload insertion problem by means of multibody dynamics, in: Proceedings of the European Conference on Spacecraft Structures, Materials and Environmental Testing, The Netherlands, ESTEC, 2018.
[13] A. Oliveira, G. Baldesi, D. Sciacovelli, A space application of a data recovery procedure based on direct enforced motion using a multi-body dynamics software (DCAP). 581. 45, in: Proceedings of the European Conference on Spacecraft Structures, Materials and Mechanical Testing 2005 (ESA SP-581), Noordwijk, The Netherlands, 2005, pp. 10–12.
[14] G. Smet, M. Yorck, G. Baldesi, M. Palladino, Multi-body dynamics software tool: two case studies, in: Proceedings of the ESAMATS : 13th European Space Mechanisms and Tribology Symposium, Vienna, Austria; Noordwijk, the Netherlands, ESA Communication Production Office, 2009.
[15] P. Santini, P. Gasbarri, Dynamics of multibody systems in space environment; Lagrangian vs. Eulerian approach, Acta Astronaut. 54 (2004) 1–24, https://doi.org/10.1016/s0094-5765(02)00277-1.
[16] D.I.M. Forehand, R. Khanin, M.P. Cartmell, A Lagrangian multibody code for deriving the symbolic state-space equations of motion for open-loop systems containing flexible beams, Math. Comput. Simul. 67 (2004) 85–98, https://doi.org/10.1016/j.matcom.2004.05.010.
[17] W. Kong, Q. Tian, Dynamics of fluid-filled space multibody systems considering the microgravity effects, Mech. Mach. Theory 148 (2020), 103809, https://doi.org/10.1016/j.mechmachtheory.2020.103809.
[18] C. Henshaw, F. Tasker, Managing contact dynamics for orbital robotic servicing missions, in: Proceedings of the AIAA SPACE Conference & Exposition, Reston, Virigina, American Institute of Aeronautics and Astronautics, 2008.
[19] D.N. Nenchev, K. Yoshida, P. Vichitkulsawat, M. Uchiyama, Reaction null-space control of flexible structure mounted manipulator systems, IEEE Trans. Rob. Autom. 15 (1999) 1011–1023, https://doi.org/10.1109/70.817666.
[20] Z. Vafa, S. Dubowsky, On the dynamics of manipulators in space using the virtual manipulator approach, in: Proceedings of the IEEE International Conference on Robotics and Automation, Institute of Electrical and Electronics Engineers, 2005.
[21] J.L. Ramón, J. Pomares, L. Felicetti, Direct visual servoing and interaction control for a two-arms on-orbit servicing spacecraft, Acta Astronaut. 192 (2022) 368–378, https://doi.org/10.1016/j.actaastro.2021.12.045.
[22] S. Abiko, R. Lampariello, G. Hirzinger, Impedance control for a free-floating robot in the grasping of a tumbling target with parameter uncertainty, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006.
[23] A. Farley, J. Wang, J.A. Marshall, How to pick a mobile robot simulator: a quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion, Simul. Model. Pract. Theory 120 (2022), 102629, https://doi.org/10.1016/j.simpat.2022.102629.
[24] P. Anggraeni, I. Rokhim, R.M. Salam, Design and development of multiple mobile manipulator robots using gazebo-ROS, in: Proceedings of the International Conference on Applied Science and Technology (ICAST), IEEE, 2020.
[25] T. Haldankar, S. Kedia, R. Panchmatia, D. Parmar, D. Sawant, Design of robotic manipulator for welding using ROS and Gazebo, in: Proceedings of the IEEE Delhi Section Conference (DELCON), IEEE, 2022.
[26] J.L. Ramón, J. Pomares, L. Felicetti, ROS Framework for on Orbiting Space Robots, OnOrbitROS, 2022. https://github.com/OnOrbitROS.
[27] Amazon Web Services 2022, Amazon web services. AWS RoboMaker: Developer Guide, Amazon WebServices, Inc., 2022. https://docs.aws.amazon.com/robomaker/latest/dg/aws-robomaker-dg.pdf.
[28] S. Abiko, K. Yoshida, Post flight analysis of ETS-VII space robotic experiments, in: Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: I-SAIRAS, St-Hubert, Quebec, Canada, 2001.
[29] K. Yoshida, Engineering test satellite VII flight experiments for space robot dynamics and control: theories on laboratory test beds ten years ago, now in orbit, Int. J. Rob. Res. 22 (2003) 321–335, https://doi.org/10.1177/0278364903022005003.
[30] C. Marchionne, M. Sabatini, P. Gasbarri, GNC architecture solutions for robust operations of a free-floating space manipulator via image based visual servoing, Acta Astronaut 180 (2021) 218–231, https://doi.org/10.1016/j.actaastro.2020.11.049.
[31] E. Marchand, F. Chaumette, T. Chabot, K. Kanani, A. Pollini, RemoveDebris vision-based navigation preliminary results, in: Proceedings of the 70th International Astronautical Congress (IAC), Washington DC, US, 2019.
[32] F. Yu, Z. He, B. Qiao, X. Yu, Stereo-vision-based relative pose estimation for the rendezvous and docking of noncooperative satellites, Math. Probl. Eng. 2014 (2014) 1–12, https://doi.org/10.1155/2014/461283.
[33] L. Song, Z. Li, X. Ma, Autonomous rendezvous and docking of an unknown tumbling space target with a monocular camera, in: Proceedings of the IEEE Chinese Guidance, Navigation and Control Conference, IEEE, 2014.
[34] G. Zhang, M. Kontitsis, N. Filipe, P. Tsiotras, P.A. Vela, Cooperative relative navigation for space rendezvous and proximity operations using controlled active vision: cooperative relative navigation for space rendezvous and proximity operations, J. Field Robot. 33 (2016) 205–228, https://doi.org/10.1002/rob.21575.
[35] B.E. Tweddle, T.P. Setterfield, A. Saenz-Otero, D.W. Miller, An open research facility for vision-based navigation onboard the international space station: an open research facility for vision-based, J. Field Robot. 33 (2016) 157–186, https://doi.org/10.1002/rob.21622.
[36] G. Ma, Z. Jiang, H. Li, J. Gao, Z. Yu, X. Chen, Y.H. Liu, Q. Huang, Hand-eye servo and impedance control for manipulator arm to capture target satellite safely, Robotica 33 (2015) 848–864, https://doi.org/10.1017/s0263574714000587.
[37] J.P. Alepuz, M.R. Emami, J. Pomares, Direct image-based visual servoing of free-floating space manipulators, Aerosp. Sci. Technol. 55 (2016) 1–9, https://doi.org/10.1016/j.ast.2016.05.012.
[38] J. Pomares, L. Felicetti, J. Pérez, M.R. Emami, Concurrent image-based visual servoing with adaptive zooming for non-cooperative rendezvous maneuvers, Adv. Space Res. 61 (2018) 862–878, https://doi.org/10.1016/j.asr.2017.10.054.
[39] G. Dong, Z.H. Zhu, Position-based visual servo control of autonomous robotic manipulators, Acta Astronaut. 115 (2015) 291–302, https://doi.org/10.1016/j.actaastro.2015.05.036.
[40] ROS: Home, Ros.org. (n.d.). https://www.ros.org (accessed 18 October 2022).
[41] Thomas, D., Scholz, D., Blasdel, A., Jeronimo, M., ROS visualization, 2012. https://github.com/ros-visualization/rqt (accessed 18 October 2022).
[42] D. Vallado, P. Crawford, SGP4 orbit determination, in: Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Reston, Virigina, American Institute of Aeronautics and Astronautics, 2008.
[43] D. Vallado, P. Crawford, R. Hujsak, T.S. Kelso, AIAA 2006-6753, in: Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, 2006.
[44] KDL, Orocos.org. https://www.orocos.org/kdl.html (accessed 18 October 2022).
[45] P.C. Hughes, Spacecraft Attitude Dynamics, Dover Publications, Mineola, NY, Estados Unidos de América, 2004.
[46] O. Khatib, A unified approach for motion and force control of robot manipulators: the operational space formulation, IEEE J. Robot. Autom. 3 (1987) 43–53.
[47] Y. Liu, Z. Xie, H. Liu, Three-line structured light vision system for non-cooperative satellites in proximity operations, Chin. J. Aeronaut. 33 (2020) 1494–1504, https://doi.org/10.1016/j.cja.2019.08.024.
[48] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, M.J. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognit. 47 (2014) 2280–2292.
[49] J.L. Ramón, J. Pomares, L. Felicetti, Visual servoing - acceleration-based controller without inertia matrix pre-multiplication, Youtube, 2022. https://youtu.be/7VmVWi5QVgo (accessed 18 October 2022).
[50] M.A. Murison, A practical method for solving the Kepler equation, (2006). 10.13140/2.1.5019.6808.

2023-06-17

Task space control for on-orbit space robotics using a new ROS-based framework

Ramón, José Luis

Elsevier

Ramón JL, Pomares J, Felicetti L. (2023) Task space control for on-orbit space robotics using a new ROS-based framework. Simulation Modelling Practice and Theory, Volume 127, September 2023, Article number 102790

https://doi.org/10.1016/j.simpat.2023.102790