ROBUST CONTROL OF FREE-FLYING SPACE MANIPULATOR FOR CAPTURING UNCONTROLLED TUMBLING OBJECTS

Davide Invernizzi [(1)], Pietro Ghignoni [(1,2)], Lorenzo Ticozzi [(1,2)], Mauro Massari [(1)], Marco Lovera [(1)], Pedro Simplicio[(3)], Irene Huertas Garcia [(3)]

(1) Politecnico di Milano, Department of Aerospace Science and Technology, Via La Masa 34, 20156, Milano, Italy, corresponding author: davide.invernizzi@polimi.it
(2) Pietro Ghignoni and Lorenzo Ticozzi were affiliated to Politecnico di Milano during the development of this work.
(3) ESA–ESTEC, European Space Agency, Keplerlaan 1,Noordwijk, NL–2200, The Netherlands

ABSTRACT

In-orbit servicing (IOS) and active debris removal (ADR) are crucial for extending the life of satellites and addressing space debris. Autonomous spacecraft equipped with robotic arms can capture and detumble resident space objects. A combined strategy that controls the spacecraft base and robotic arm together is proposed to address the pre-capture phase, where the goal is to approach with the arm end-effector a point on an uncontrolled tumbling spacecraft. A recursive method is applied to the dynamic model of the space robot to fully capture the dynamic coupling between its elements in a computationally efficient way. The proposed control design consists in using nonlinear control laws, based on extensions of the well–known computed torque controller to space robots, together with a systematic tuning procedure based on the structured H_∞ framework. As the control law is designed in the joint space but the control objective is formulated in the task space, a closed-loop inverse kinematic solver is included in design. A robustness analysis is performed with respect to rigid-body uncertainties and sloshing effects. The performance of the proposed controller is evaluated on a representative scenario for capturing an uncontrolled tumbling object using a nonlinear dynamical model.

1 INTRODUCTION

The potential of In-Orbit Servicing (IOS) to extend the operational life of satellites and the need to implement Active Debris Removal (ADR) to effectively tackle the space debris problem are well known among the space community [1], [2]. Research on technical solutions to enable this class of missions is thriving, also pushed by the development of new control systems. Several solutions have been proposed over the years to safely capture orbital objects, the majority of which rely on robotic systems [3]. A promising solution is the employment of an autonomous spacecraft (chaser) equipped with a highly dexterous robotic arm able to perform the berthing with a resident space object. In this respect, the design of an effective, reliable, and robust Guidance Navigation and Control (GNC) system, for which several architectures and hardware configurations are possible, plays a key role to ensure a safe mission execution. The problem addressed in this work concerns the design of control laws suitable for capturing an uncontrolled tumbling spacecraft using a space robot equipped with a redundant manipulator. When developing the control and guidance functions, we assume that there are navigation functions available that estimate the necessary quantities. The integration and testing of the proposed design in a full GNC simulator is reported in [4].

Space robots are characterized by a high level of complexity due to the kinematic and dynamic coupling between its elements [5]. The motion of a single body (be it the base or one of the links) is transmitted downstream (in the direction of the end-effector) according to the properties of the kinematic chain, while it dynamically affects the system also in the upstream direction (towards the satellite-base). From a practical standpoint this means that, unlike ground-fixed robots, the motion of the manipulator causes a motion of the base. Keeping these aspects in mind, the dynamic model of a space robot is mostly built upon the traditional theory of rigid multibody system. In this work a recursive method is applied to the system of interconnected rigid bodies which, unlike the direct equivalent, models the interconnections in terms of forces and kinematic constraints acting at a single-body level. This results in a large set of equations which can be solved efficiently by exploiting recurrence relations descending from the tree-like structure of the system [6].

To address the capturing problem, we propose a combined control approach wherein base and manipulator states are controlled together, following ideas recently proposed in the literature. As shown by recent works [7], a combined architecture has several advantages over decoupled control strategies, from fuel efficiency improvement to performance improvement. The specific approach developed in this work consists in using nonlinear control laws, based on extensions of the well–known computed torque controller to space robots, together with a systematic tuning procedure based on the H_∞ framework. Indeed, while computed torque controllers deliver good tracking performance in a large domain of operating conditions, they suffer from modelling uncertainty (being based on feedback linearization) and no rule is given to tune the gains of the feedback component of the control law, which is typically based on a (nonlinear) Proportional Derivative (PD) law. Hence, trial and error procedures are often employed in practice to select the gains and achieve acceptable performance. Such an approach is made more challenging by the large number of states of space robots. Therefore, the following systematic tuning approach is considered: first, both the plant and the control law are linearized about a nominal operating point and a linear uncertain description of the closed–loop system is derived; then, the gains of the control law are tuned by leveraging the structured H_∞ framework [8]. In this manner, the control law handles by design the rigid body nonlinearities while performance requirements can be imposed in the neighbourhood of the desired configurations when tuning the gains. The proposed synthesis approach allows accounting for dynamics effects at synthesis time, such as sloshing, actuator dynamics, flexibility, orbital dynamics, which are neglected when deriving the nonlinear control law.

The proposed control law is designed using a joint space formulation and thus requires computing a reference trajectory in the joint space. However, the capture of uncontrolled tumbling objects poses requirements to the trajectory generation in the task space. Hence, a trajectory generation for the end effector in task space is proposed together with an inverse kinematic approach which exploits the manipulator redundancy to locally optimize the manipulability index. As the target is in an uncontrolled tumbling state, the reference trajectory generation is generated propagating forward in time the target motion, but continuously updating the propagation with information on the current state of the target coming from the Navigation System.

After the controller synthesis, a robustness analysis with respect to rigid-body geometry and inertial uncertainties and sloshing has been performed, while the performance of the proposed controller is evaluated in a representative scenario for the capturing of an uncontrolled tumbling object using a full nonlinear model of the dynamics.

2 CONTROL-ORIENTED MODELING of SPACE ROBOTS

In this section, we discuss the approach developed to derive a mathematical model of space robots suitable for control design which is also sufficiently accurate for a preliminary assessment of the closed-loop performance.

2.1 Configuration and kinematics

A brief description of the geometry, frames definition and kinematics of the space robot is reported next. It should be noted that the kinematics of a spacecraft-mounted manipulator shares significant similarities with its ground-fixed counterpart; however, as highlighted in the following, the satellite-base position and orientation must be accounted for in the formulation of the problem.

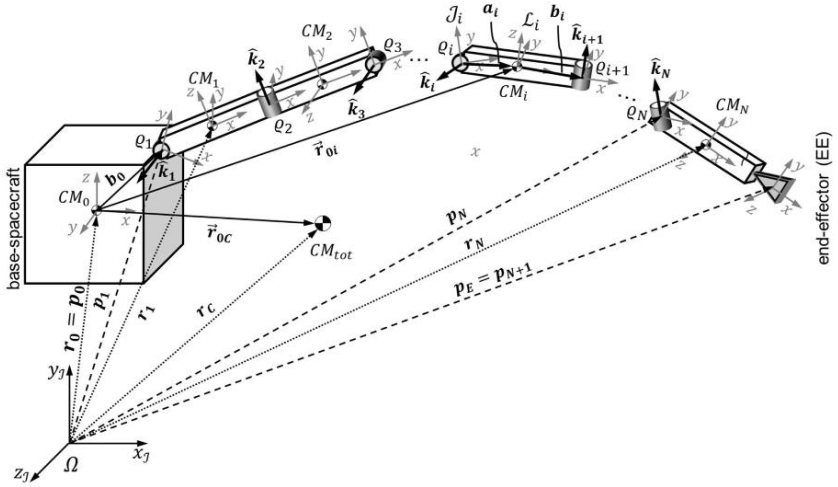

Figure 1 Geometry of a spacecraft-manipulator system - Credits [9].

The spacecraft-manipulator system represented in Figure 1 can be thought of as a tree-like structure where the kinematic relationship between two generic adjacent bodies is defined by the type of interconnection between the bodies themselves. A geometric model of the system provides a description of the positions of the interconnections, the joints, onto each body. Frames $F_{i,j}$ identify the position of joint j onto body i, while each body has its own local *body frame* F_i, also known as *link frame*. Frame I is the reference coordinate system, while frame F_0 is located on the base body (when describing a space robot, F_0 identifies the main satellite body). The transform from body frame F_i to joint frame $F_{i,j}$ is given by the constant spatial transform ${}^{i,j}X_i$, while the position and attitude of the each link can be described either with respect to the satellite-base (${}^{i}X_0$ transforms) or to the reference coordinate system (${}^{i}X_I$ transforms). Moreover, each joint defines a joint transform X_{J_i} which connects joint frame $F_{i,j}$ to the subsequent body frame F_j; this transform depends on the joint's position variable, q_i. The forward kinematics of the space manipulator can be solved by recursively exploiting the above transformations up to the desired element of the system. Given a space robot made up by a chain of $N_B + 1$ bodies (satellite base + N_B links) connected by N_B joints, the pose transformation from the satellite-base frame F_0 to the last link frame F_{N_B} can be expressed as follows,

$$ {}^{N_B}X_0 = X_{J_{N_B}}\left(q_{N_B}\right)^{N_B-1,N_B}X_{N_B-1}\cdots X_{J_1}(q_1)^{0,1}X_0. \qquad (2.1) $$

If one wishes to express the pose of the last body (end-effector) with respect to the reference coordinate system I, it is sufficient to post-multiply the above expression by the pose transform between frame I and frame F_0, ${}^{N_B}X_I = {}^{N_B}X_0 * {}^{0}X_I$.

The forward kinematics of the space robot is tightly related to the nature of the motion allowed by the joints of the manipulator. In fact, each joint can be thought of as a motion constraint between its adjacent bodies; different joint types (revolute, prismatic, helical, etc.) correspond to different motion constraints. The generic joint transform X_{J_i} describes the spatial motion of a body frame F_j with respect to its predecessor joint frame, $F_{i,j}$. This transform can be identified by the joint type and by

its position coordinate, q_i. The joint coordinate of a revolute joint is therefore given by the angle of rotation of F_i with respect to $F_{i,j}$ along a common axis, known as *joint axis.* By convention, we consider this rotation to take place along the revolute joint's local $z-$ axis. Remembering that revolute joints never allow a relative translation between their predecessor and successor frames, the following holds,

$$X_{J_i}(q_i) = rot(E_i) * transl(r_i), \tag{2.2}$$

$transl(r_i) = \begin{bmatrix} I_3 & 0_3 \\ -r_i \times & I_3 \end{bmatrix}$, $E_i = blkdiag\big(rotz(q_i), rotz(q_i)\big)$, $rotz(q_i) = \begin{bmatrix} \cos q_i & \sin q_i & 0 \\ -\sin q_i & \cos q_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and I_3, 0_3 are, respectively, the 3×3 identity matrices and the 3×3 matrix filled with zeros. Note that, for a revolute joint, $r_i = [0,0,0]^T$. Moreover, since joints always express a motion constraint, each joint type is featured by its own *motion subspace,* identified by a matrix. In particular, the motion subspace matrix of a revolute joint is expressed as $S_r = [0,0,1,0,0,0]^T$.

Solving the forward kinematics of the system also allows to express the angular and linear velocities of each element with respect to a known reference frame. By recalling the previous concepts applied to the revolute joint case, we can define the velocity across joint i as the difference of the velocities of its successor and predecessor elements, $v_{J_i} = v_i - v_{i-1}$. Since the motion must take place accordingly to the joint's motion subspace, the following is also valid, $v_{J_i} = S_r \dot{q}_i$. Therefore, we obtain the equation to compute the velocity of each body in the system as follows,

$$v_i = v_{i-1} + S_{r_i} \dot{q}_i. \tag{2.3}$$

Note that this expression can be effectively used in a *forward* manner, *i.e.*, moving outwards with respect to the satellite base. In fact, once the 6 elements of the base velocity v_0 are known along with the joints velocity variables $\dot{q}_i$, *(2.3)* can be recursively applied to find the velocity of each body in the system,

$$v_i = v_0 + \sum_{j=1}^{i} S_{r_j} \dot{q}_j = v_0 + J_i(q)\dot{q}, \tag{2.4}$$

where J_i is the *i-th* body Jacobian and $\dot{q}$ is the joint-space velocity vector, highlighting the linear dependence of v_i on the joint velocity variables.

2.2 Dynamics

To describe the space robot dynamics, we have developed a mathematical model using the floating version of the Recursive Newton-Euler Algorithm (RNEA), which leverages the efficient Composite Rigid-Body Algorithm (CRBA) for the computation of the inertial terms, as suggested in [6]. The considered approach allows dealing with arbitrarily complex configurations while maintaining computational efficiency. In addition, an orbital disturbance term was included in the model to evaluate possible undesired effects resulting from the coupling between multibody and orbital dynamics that are usually ignored in previous studies.

The dynamic equations of motion for the system can then be compactly represented as follows:

$$H(q) \begin{Bmatrix} \dot{\omega}_b \\ \dot{v}_b \\ \ddot{q} \end{Bmatrix} + C\big(q, \omega_b, v_b, \dot{q}, \eta_b, r_{b\backslash t}\big) = \begin{Bmatrix} M_b \\ F_b \\ \tau \end{Bmatrix}, \tag{2.5}$$

where ω_b, $v_b \in \mathbb{R}^3$ represent the base angular and linear velocity respectively, $q \in \mathbb{R}^7$ is the vector of joint angles, $\eta_b \in \mathbb{S}^3$ is the unit quaternion describing the base attitude with $\mathbb{S}^n = \{v \in \mathbb{R}^{n+1}: \|v\| = 1\}$, $r_{b/t} \in \mathbb{R}^3$ is the position of the base with respect to the target, while M_b, $F_b \in \mathbb{R}^3$ and $\tau \in \mathbb{R}^7$ are base torque, force and joint motor torque, respectively. The joint space inertia matrix is denoted as $H \in \mathbb{R}^{13x13}$, while $C \in \mathbb{R}^{13}$ represents the Coriolis/centrifugal term that accounts also for terms related to the relative orbital dynamics.

3 CONTROL ARCHITECTURE and DESIGN

This section is devoted to presenting the combined control architecture proposed to solve the pre-capturing phase addressed in this work, where the goal is to accurately track the target's position with the manipulator's end effector. To this aim, we have considered using a joint space formulation of the control law combined with a closed-loop inverse kinematic solver to convert the end effector desired trajectory into joint profiles.

3.1 Control Architecture

The determination of the control framework primarily relies on the available control inputs, measurements, state estimation, mission phase (such as reach and capture or stabilization), and the chosen control strategy. The joint space architecture depicted in Figure 1 has been used, where the input and output terms are reported.

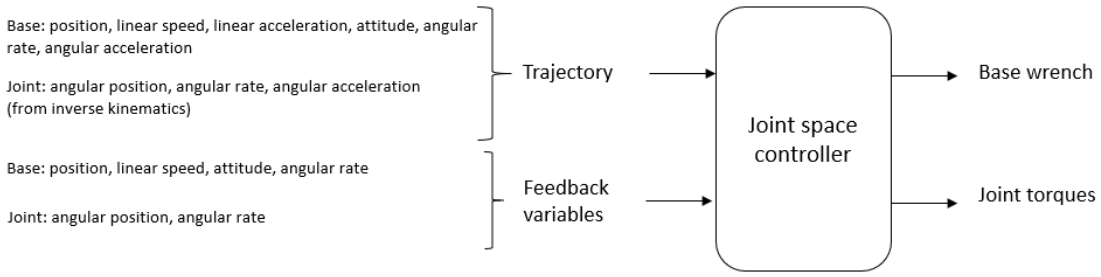

Figure 2: Joint space control architecture.

The desired trajectory comprises desired base states and manipulator states (joint angles and velocities). The inverse kinematics problem is solved to obtain the manipulator desired states for a desired profile of the end effector's pose (provided by a dedicated guidance function described in Section 0). Additionally, the desired acceleration is required to compute feedforward control actions without resorting to numerical differentiation. Both the base and manipulator states are utilized for feedback control.

3.2 Combined control law design

The concept of computed torque controllers, initially proposed for ground-based robots with fixed bases has been extended to encompass the control of space robots in [10]. The structure of this control law is simple: the estimated robot states are used to calculate estimates of the Coriolis/centrifugal terms and the mass matrix; a PD controller acts upon the control error, which may be nonlinear, while feedforward action is employed to enhance system tracking capabilities. Such a control law ensures asymptotic tracking of the error (assuming exact knowledge of the system parameters and ideal measurements) and is commonly employed in robotics. The potential increase in complexity is not a critical concern since efficient recursive methods exist to compute the quantities required by the controller. It is important to note that for small control errors and low speeds, typically encountered

in close proximity operations where Coriolis/centrifugal terms are negligible, the controller behaves approximately like a simple PD controller.
The goal of the control law is to follow a desired attitude trajectory $\eta_b^{sp}, \omega_b^{sp}$, a base position trajectory in the LVLH frame frame $v_b^{sp}, r_{b\backslash t}^{sp}$, and joint trajectories obtained through the inverse kinematics function $q^{sp}, \dot{q}^{sp}$. The expression for the computed torque control law is given by [4]:

$$\begin{Bmatrix} M_b \\ F_b \\ \tau \end{Bmatrix} = \hat{C}(q, \omega_b, v_b, \dot{q}) + \hat{H}(q) \begin{Bmatrix} a_\omega \\ a_v \\ a_q \end{Bmatrix}, \tag{3.1}$$

where a_ω, a_v, a_q represent virtual input variables, defined next. The attitude virtual input is

$$a_\omega = -K_{d,att}\omega_e + K_{p,att}\eta_{e,b,vec}H_{/\eta} + R_e\dot{\omega}^{sp} - \omega_e \times R_e\omega^{sp}, \tag{3.2}$$

where $\eta_{e,b} = \begin{Bmatrix} \eta_{e,b,vec} \\ \eta_{e,b,sc} \end{Bmatrix} = \eta_b \otimes \eta_b^{sp*}$, is the attitude error between the reference quaternion η_b^{sp} and the current quaternion computed using the quaternion product $\otimes$, $R_e = R(\eta_b)R\left(\eta_b^{sp\ *}\right)$ is the attitude error expressed using rotation matrices, $H_{/\eta} = -2\eta_{e,b,sc}$ is a function introduced to avoid unwinding [11] when using quaternions in the feedback law, $\omega_e = \omega_b - R_e\omega_b^{sp}$ is the angular velocity error expressed in the body frame. $K_{d,att}, K_{p,att} \in \mathbb{R}^{3\times3}$ are the proportional and derivative gains for the attitude, respectively. The position virtual input is defined as

$$a_v = R(\eta_b)\left[\dot{v}_b^{sp} - K_{d,pos}\left(R^T(\eta_b)v_b - v_b^{sp}\right) - K_{p,pos}\left(r_{b\backslash t} - r_{b\backslash t}^{sp}\right)\right] - \omega_b \times v_b \tag{3.3}$$

with $K_{d,pos}, K_{p,pos} \in \mathbb{R}^{3\times3}$. Finally, the joint virtual input is

$$a_q = \ddot{q}_{sp} - K_{d,joint}\left(\dot{q} - \dot{q}_{sp}\right) - K_{p,joint}\left(q - q_{sp}\right), \tag{3.4}$$

where $K_{d,joint}, K_{d,joint} \in \mathbb{R}^{7\times7}$. We can collect the free parameters of the controller in block-diagonal matrices as $K_p = diag\left(K_{p,att}, K_{p,pos}, K_{p,joint}\right), K_d = diag\left(K_{d,att}, K_{d,pos}, K_{d,joint}\right)$.

3.3 Guidance and inverse kinematics

The aim of the guidance strategy is to generate suitable trajectories for the capturing task while keeping it as simple as possible. To achieve this, the strategy does not include collision avoidance or interference checks during guidance. Instead, the consistency with collision-free operation is verified after the simulation. The capturing maneuver is divided into four main instants: 1) when the manipulator starts moving (t_{start}), 2) from a specific time point (t_{point}) to the time of grasping (t_{grasp}), where the end-effector camera is directed towards the grasping point using the guidance design, 3) at the time of grasping (t_{grasp}) when the end-effector is brought onto the grasping point, and 4) when the capture maneuver is completed (t_{end}), as shown in Figure 3.

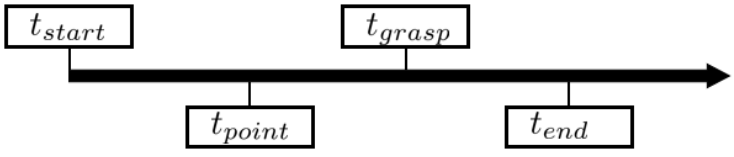

Figure 3 Relevant time instants for trajectory design.

The reference trajectory is computed based on a desired trajectory of the end-effector and then translated into reference trajectories in the joint space using the inverse kinematic solution described next. The guidance of the base is obtained by selecting a proper fixed position in the LVLH frame while the trajectory for the arm is computed using a 5th order polynomial to match the initial and final states (including their derivatives) of the end-effector. The desired final state of the end-effector depends on the location of the grasping point at a specific time in the future. This location is predicted by propagating the target motion forward until the desired time, and then the inverse kinematic is solved to compute the joint coordinates guidance. Since the motion prediction of the target is only accurate for a short time horizon, the coefficients of the 5th order polynomial interpolation are updated at each step using current measurements of both the chaser and target states obtained from the navigation function. This allows for a new prediction of the future location of the grasping point. This approach compensates for variations in the expected location of the chaser's center of mass and the grasping point. As the time for grasping approaches, the prediction horizon for the grasping point location decreases, but the guidance is able to compensate for any mismatch in the final location of the grasping point.

The forward kinematics relationships in equation *(2.1)* and can be used to describe the end-effector's position, attitude and velocity in the Cartesian space; therefore, they require as input the pose of the base and the joints' position and velocity coordinates (i.e., q_i and $\dot{q}_i$). In practice, when planning a robot's task, the joint position and velocity variables are not known a priori, but they must be reconstructed starting from the knowledge of the desired end-effector Cartesian position and velocity. This process, known as *inverse kinematics*, allows the generation of a setpoint trajectory in the joint-space of the manipulator and, therefore, the synthesis of joint-space controllers. Unlike forward kinematics, whose solution is straightforward once the system's configuration is known (see Section 2.1), inverse kinematics is featured by a higher degree of complexity. In general, solving the inverse kinematics means assigning a desired value ${}^{EE}X_I{}^{d}$ to the pose of the robot's end-effector and then solving equation *(2.1)* for $q_1^d, \dots, q_{N_B}^d$, the desired joint variables. Analytical approaches leading to closed-form solutions can only be applied to certain classes of robots and are indeed unsuitable to redundant manipulators [12].

The approach we choose consists instead in solving the inverse kinematics at the joints velocity level by expressing the linear relationship *(2.4)* between the desired end-effector Cartesian velocity and the joint-space velocity as

$$v_{EE}^d = v_0 + J_{EE}(q)\dot{q}_d, \tag{3.1}$$

and then solving for $\dot{q}_d$,

$$\dot{q}_d = J_{EE}^{\#}(q)\left(v_{EE}^d - v_0\right), \tag{3.2}$$

where $J_{EE}^{\#}(q)$ is the right (Moore-Penrose) pseudo inverse of the end-effector Jacobian matrix, and v_0 must be considered since the satellite-base may have a nonzero velocity. It should be noted that, when the manipulator is redundant, has infinite solutions, and the one expressed in *(3.2)* is the one which minimizes the cost function $g(\dot{q}) = \frac{1}{2}\dot{q}^T\dot{q}$.

Due to redundancy, there exists a subset of the joint-space velocities, the *null space,* which does not yield a velocity in the task space. In the particular case with $v_0 = 0$, the joint velocities in the null space are a solution to the following homogeneous equation

$$J_{EE}(q)\dot{q}_0 = 0. \tag{3.3}$$

According to the formulation in [13], the null space motion can be included in the inverse kinematics solution as

$$\dot{q}_d = J_{EE}^{\#}(q) v_{EE/0}^{d} + \left(I_{N_B} - J_{EE}^{\#}(q) J_{EE}(q)\right) \dot{q}_0, \tag{3.4}$$

where $v_{EE/0}^{d} = v_{EE}^{d} - v_0$ is the end-effector velocity with respect to the satellite-base, and $\left(I_{N_B} - J_{EE}^{\#}(q) J_{EE}(q)\right)$ is a projector of the joint velocity vector $\dot{q}_0$ onto the null space [14]. Vector $\dot{q}_0$ is typically specified according to the projected gradient method, as in

$$\dot{q}_0 = k_0 \left(\frac{\partial w(q)}{\partial q}\right)^T, \tag{3.5}$$

where $k_0 > 0$ and $w(q)$ is a differentiable objective function. Depending on the choice of $w(q)$, the inverse kinematics solution in equation *(3.5)* locally maximizes a performance index which may be related to robot dexterity, joint limits avoidance, obstacle avoidance, etc. Within this study, we have considered two different objective functions,

$$w_1(q) = \sqrt{\det\left(J_{EE}(q) J_{EE}^T(q)\right)}, \qquad w_2(q) = -\frac{1}{2N_B} \sum_{i=1}^{N_B} \left(\frac{q_i - \bar{q}_i}{q_{i,M} - q_{i,m}}\right)^2, \tag{3.6}$$

where $w_1(q)$ coincides with the manipulability index of the manipulator and therefore optimizes the null motion in order to maximize the robot dexterity [15], while $w_2(q)$ is a performance index related to avoidance of mechanical joint limits, $q_{i,M}(q_{i,m})$ denotes the maximum (minimum) limit for q_i and $\bar{q}_i$ is the middle value of the joint range [16].
Hence, the inverse kinematics solution in *(3.4)* guarantees that $J_{EE}(q)\dot{q}_d = v_{EE/0}^{d}$, while also exploiting redundancy to locally optimize a selected performance index. However, this solution does not compensate for a mismatch between the current and desired end-effector location and attitude. Therefore, a feedback term dependent on the pose error $p_{EE}^{e} = ({}^{EE}X_I{}^{d})^{-1}{}^{EE}X_I$, is added to make up for possible location and attitude errors, according to the Closed-Loop Inverse Kinematics (CLIK) algorithm proposed in [14]. Equation *(3.4)* is then transformed into the following expression:

$$\dot{q}_d = J_{EE}^{\#}(q) \left(v_{EE/0}^{d} + K_P(p_{EE}^{e})\right) + \left(I_{N_B} - J_{EE}^{\#}(q) J_{EE}(q)\right) \dot{q}_0. \tag{3.7}$$

4 CONTROL LAW SYNTHESIS

Due to their reliance on feedback linearization, computed torque controllers can only provide limited robustness in the presence of uncertainties, usually in the form of local boundedness of the states. Additionally, no systematic tuning procedure is available for selecting the gains. A promising approach to address this limitation is to employ robust control theory for the systematic and optimal tuning of the controllers.
Figure 4 illustrates the robustification process of nonlinear controllers. The starting point is the nonlinear system interconnected with a nonlinear controller, which comprises a nonlinear component and tunable parameters. To find appropriate parameters for these tunable elements, the linearization of both the plant and controller is performed. As depicted in the right part of the figure, the closed-loop system is transformed into a linearized plant (possibly accounting for uncertainties in Linear Fractional Transformation (LFT) form) and a linearized control law including tunable parameters like

PD gains. Robust control methods and structured routines can then be employed to determine optimal gains for the tunable parts. Notably, it allows for the consideration of orbital dynamics effects, which are typically disregarded in other works on space robotics.

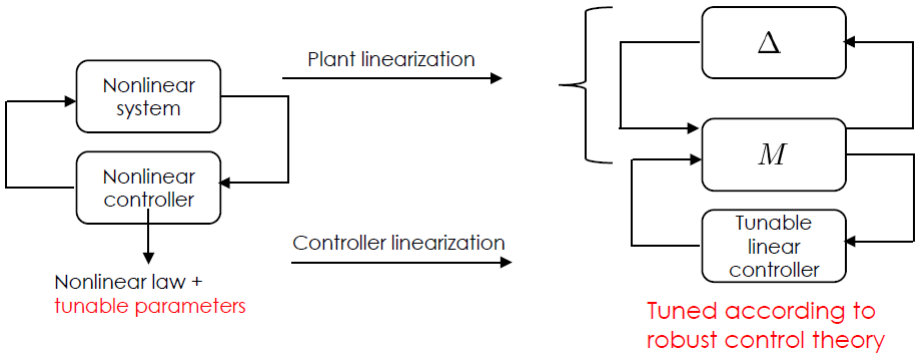

Figure 4 Robustification of nonlinear controllers.

In following subsections, we discuss the approach followed for the linearization of the equations of motion and of the control law, as well as the dependence of the model on uncertain parameters using the Linear Fractional Transformation (LFT) form.

4.1 Plant and control law linearization

We consider the set of dynamic equations in *(2.5)* together with the kinematics of the base described in terms of quaternions for the attitude and the relative position between the chaser and the target. In the following equations, $\delta\star$ denotes the perturbation of the variable $\star$ with respect to a reference configuration. To facilitate control gain tuning, the plant and controller will be linearized around a reference trajectory. Considering the expected slow maneuvers from the guidance, a constant reference configuration is used for linearization, which is consistent with existing approaches in space robotics. It should be noted that the uncertainties in mass, moments of inertia, and center of mass position account for variations in the robotic arm configuration.
The linearization of the kinematics of the base is given by:

$$\delta\dot{\eta}_b = \frac{1}{2}\begin{Bmatrix}\delta\omega_b \\ 0\end{Bmatrix}, \tag{4.1}$$

$$\delta\dot{r}_{b\backslash t} = R_{b\backslash t}(\bar{\eta}_b)\delta v_b, \tag{4.2}$$

where $\delta\eta_b$ denotes the quaternion perturbation, *i.e.*, $\eta_b = \delta\eta_b \otimes \bar{\eta}_b$ and $R_{b\backslash t}(\bar{\eta}_b)$ is the rotation matrix from body to LVLH frame corresponding to quaternion $\bar{\eta}_b$.
In the absence of external perturbations, the linearized form of *(2.5)* is given by:

$$H(\bar{q})\begin{Bmatrix}\delta\dot{\omega}_b \\ \delta\dot{v}_b \\ \delta\ddot{q}\end{Bmatrix} = \begin{Bmatrix}\delta M_b \\ \delta F_b \\ \delta\tau\end{Bmatrix}. \tag{4.3}$$

Let $x = \left[\delta\eta_{b,vec}, \delta r_{b\backslash t}, \delta q, \delta\omega_b, \delta v_b, \delta\dot{q}\right]^{\mathsf{T}}$ be the vector of small variations with respect to the reference configuration, $A_{kin} = blkdiag(\frac{1}{2}I_3,\ R_{b\backslash t}, I_n)$ and $u = [\delta M_b, \delta F_b, \delta\tau]^{\mathsf{T}}$. When the relative orbital dynamics is considered, the linearized equation can be written as

$$\dot{x} = \begin{bmatrix}0 & A_{kin} \\ -H^{-1}C_{\backslash x_{kin}} & -H^{-1}C_{\backslash x_{dyn}}\end{bmatrix}x + \begin{bmatrix}0 \\ H^{-1}\end{bmatrix}u, \tag{4.4}$$

In the above equation, $C_{\backslash x_{kin}}, C_{\backslash x_{dyn}}$ denote the partial derivatives of the Coriolis/centrifugal term C with respect to the sets of variables $[\delta\eta_{b,vec}, \delta r_{b\backslash t}, \delta q]$ and $[\delta\omega_b, \delta v_b, \delta\dot{q}]$, respectively. Similar to point mass systems, where the relative orbital motion is described by the Yamanaka-Ankersen equations, the presence of orbital effects shifts the eigenvalues from the origin. Unfortunately, no analytical methods exist for computing the terms $C_{\backslash x_{kin}}, C_{\backslash x_{dyn}}$ when orbital perturbations are considered. Consequently, the part of the dynamic matrix A in equation *(4.4)* that contains those terms must be computed numerically.

Once the linear models are obtained, uncertainties are incorporated into the model using the Linear Fractional Transformation (LFT) form. Specifically, during the reach and capture phase of the mission, the uncertainties related to mass, moments of inertia, products of inertia and center of mass position of the base spacecraft are included in the model. Uncertainties related to the robotic arm are not considered, as it is assumed that accurate calibrations can be performed before operations, which is a standard assumption in the robotics literature.
The uncertain parameters can be incorporated into equation *(4.4)* using routines implemented in the MATLAB robust control toolbox. This leads to the uncertain dynamics, where the matrices become functions of the uncertain parameters δ:

$$\dot{x} = A(\delta)x + B(\delta)u. \qquad (4.5)$$

To simplify the complexity of the uncertain model, a procedure based on the evaluation of the Vinnicombe metric has been adopted. This results in the following model $\dot{x} = Ax + B(\delta)u$, where only $B(\delta)$ is considered uncertain. The corresponding LFT representation (for the 7-link robot considered in the numerical example in Section 5) contains 27 parameters, including repetitions.

Up to this point, the system has been modeled with a focus on rigid bodies, which is a commonly adopted approach for control design as it accurately captures the primary behavior of the system. However, the actual behavior of the system is influenced by additional second-order effects such as sloshing and flexibility. Despite their significance, these effects are often overlooked in control design practices, as indicated by the literature review. Nevertheless, within this work, we have developed more sophisticated models to enable stability analysis of the system whenever feasible.
To incorporate sloshing, we have included a spring-mass-damper equivalent model, as presented in [17], in the linearized equations of motion. This sloshing model requires several parameters for its implementation, including sloshing frequency, damping, mass, and tank positions. The state space formulation for the capture phase varies depending on the number of tanks of the chaser.
As for the controller linearization, after linearization, one obtains

$$\begin{Bmatrix} \delta M_b \\ \delta F_b \\ \delta\tau \end{Bmatrix} = \widehat{H} \begin{bmatrix} I_3 & 0 & 0 \\ 0 & R(\bar{\eta}_b) & 0 \\ 0 & 0 & I_7 \end{bmatrix} [K_p \quad K_d] \begin{Bmatrix} 2\eta^*_{e,b,vec} \\ r^{sp}_{b\backslash t} - r_{b\backslash t} \\ q^{sp} - q \\ \omega^{sp}_b - \omega_b \\ v^{sp}_b - R^T(\bar{\eta}_b)v_b \\ \dot{q}^{sp} - \dot{q} \end{Bmatrix} \qquad (4.6)$$

where $\eta^*_{e,b,vec} = \delta\eta^{sp}_{b,vec} - \delta\eta_{b,vec}$ is a three-dimensional vector containing small attitude errors of the base. By inspecting *(4.6)*, one immediately sees that locally the controller behaves as a PD control law, with tunable gains contained in the matrices K_p, K_d.

4.2 Gain tuning using structured mixed-sensitivity H_∞ synthesis

As mentioned in the Introduction, the controller synthesis is performed using on H_∞. We provide a brief summary of the key components of the mixed-sensitivity approach, as presented in [18].
Let $K(s,\theta)$ be a structured linear controller represented in the Laplace domain, dependent on a vector of tunable parameters $\theta \in \mathbb{R}^{n_\theta}$, and let $J_i(\theta), i = 1, \dots, n$ be a set of n objectives

$$J_i(\theta) = \|W_i(s)S_i(s,\theta)\|_\infty \tag{4.7}$$

where $S_i(s,\theta)$ is the transfer function from some performance input to some performance output of the closed-loop plant and $W_i(s)$ is the corresponding frequency weight; in general, these are MIMO transfer matrices defined to embed desirable requirements. In the SISO case, the inverse of the magnitude of weight can be interpreted as the desired shape, or template, for the magnitude of the transfer function to be shaped.
The mixed-sensitivity synthesis can then be cast as the problem of finding the parameter vector θ which solves the following multi-objective optimization problem

$$\begin{aligned} &\min_{\theta} \max_{i=1,\dots,n} J_i(\theta) \\ &\text{subject to} \\ &\max_{j=1,\dots,c} H_j(\theta) \end{aligned} \tag{4.8}$$

while guaranteeing that the closed-loop system is stable. The constrained optimization *(4.9)* accounts for c inequality constraints of the form

$$H_j(\theta) = \left\|W_j(s)S_j(s,\theta)\right\|_\infty, \qquad H_j(\theta) \le 1. \tag{4.9}$$

The optimization problem stated above is coherent with the one presented in [17] and implemented in the MATLAB *systune* routine. Given an open-loop plant $G(s)$, to ensure consistency in the computation of the norm of a linear multivariable system, it is important to normalize the inputs and outputs of $G(s)$. When constructing the open-loop plant is typically replaced with $G'(s)$, given by $G'(s) = Y^{-1}G(s)\,U$. Here, Y and U are constant diagonal matrices of appropriate dimensions, and their inverses scale the inputs and outputs of G, respectively. In particular, U is selected so that the maximum allowed control input is normalized to 1. This is achieved by setting the diagonal entries of U as the maximum available control input for the corresponding channel. Y normalizes the output to 1, and its precise value is determined through an iterative procedure. The objective of the synthesis process is to find a controller $K'(s)$ such that $u = K'(s)y$ solves the optimization problem *(4.9)*. The controller to be implemented on the unscaled plant is obtained as $K = UK'Y^{-1}$.

5 CASE STUDY: REACH and CAPTURE of a NON-COOPERATIVE TARGET

5.1 Scenario definition

The considered scenario involves a servicing mission in Low Earth Orbit (LEO) specifically designed for a small platform within a large constellation, such as the Airbus Arrow platform or OneWeb constellation. The target satellite is in a circular orbit at an altitude of $1210km$ with an inclination of 88 degrees. This scenario is representative of other constellation satellite configurations like IRIDIUM, GLOBALSAR, and Starlink. The target satellite is prepared for servicing with a grapple interface and fiducial markers for navigation. During the mission phase, the target is considered non-collaborative. Under nominal conditions, the target satellite is tumbling mostly about Y-axis of the

body frame, with angular rates of up to 2.5 deg/s. To perform reach and capture operations, the chaser spacecraft synchronizes its motion with the target's approach corridor in the Local Vertical/ Local Horizontal (LVLH) reference frame to minimize relative motion between the chaser and the grapple fixture on the target. The considered simulations do not account for the synchronization maneuvers as they focus on the capture phase. The chaser platform is selected based on a literature review and simple system design criteria with the main inertial parameters reported in Table 2.

5.2 Control synthesis results

Prior to the synthesis, appropriate control requirements must be defined when using the mixed-sensitivity H_∞ approach. The synthesis of the gains is carried out for the fixed controller structure in equation *(4.6)* as described in Section 4.2 using the MATLAB routine *"systune"* which enables the simultaneous imposition of multiple objectives.

Concerning the control requirements, because of the high tumbling rate of the target, the manipulator and base of the chaser have both to track fast reference signals; this translates into imposing sufficiently high control bandwidths. The multi-objective optimization problem has been formulated considering the following requirements:

- Req. 1: Tracking performance requirement from base pose setpoint to base pose error.
- Req. 2: Tracking performance requirement from joint angles setpoint to joint angles.
- Req. 3: Control effort moderation from base and joint setpoints to base control wrench.
- Req. 4: Control effort moderation from base and joint setpoints to arm torque commands.

The high-level objectives can be translated using a block-diagonal frequency weight $W_S = blkdiag(W_{S,att}, W_{S,pos}, W_{S,joint})$. We have used first order filters in the form $w_{S,\star} = A\,\frac{1+sT}{1+s\tau}$ implemented using the MATLAB function *makeweight* and shown in Figure 5. We have imposed the same requirements as in [3] for the base attitude and position closed loops, namely 0.3 and 0.2 rad/s, respectively. The bandwidth for the joint angles closed loops is set to 3 rad/s, almost one decade faster than the one used for tracking cooperative targets. The DC gain of the weight has been set to $40dB$ to impose a small steady state error. However, since the plant dynamics includes a double integrator (approximately, due to the presence of low frequency orbital modes), the value of this number is not particularly influent. The high frequency gain is set to 0.9 to reduce the overshoot in the response. For this parameter, a trial and error procedure has been applied to find the final value. The corresponding functions are shown in Figure 5. The synthesis on the nominal system (no uncertainty is considered) yields the final values of the objective functions reported in the first four columns of Table 1. By inspecting Req. 3 and Req. 4, we can see that the main limitation to the achieved performance is given by the limited control authority of the base (4 RWs, 0.248 Nm each). Robust stability in the presence of uncertainty (considering the values reported in Table 2) has been done using the *robstab* routine implemented in the MATLAB robust control toolbox (last three columns of Table 1). A value of the stability margin equal to x means that the controller can tolerate x times the considered uncertainties. In this sense, lower bounds greater than 1 prove robust stability; upper bounds smaller than 1 prove that the controller is not robust to the uncertainties in the considered set. Exploiting the capabilities of the *systune* routine, a retuning with uncertainties has been done, yielding almost the same cost function values and control parameters.

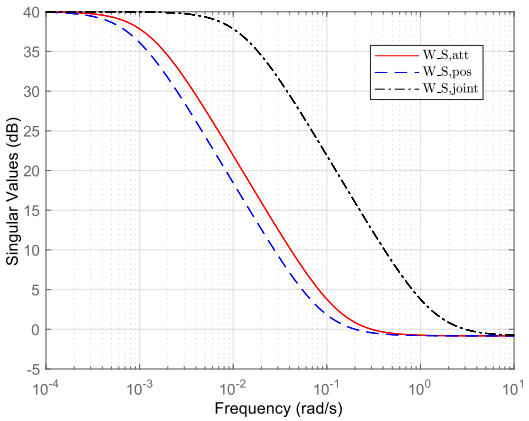

Figure 5 Frequency weights for sensitivity functions.

Table 1 Achieved cost function values and robustness margins.

Req. 1	Req. 2	Req. 3	Req. 4	Lower bound	Upper bound	Critical frequency [rad/s]
1.55	1.55	1.55	0.007	8.56	8.98	∞

Table 2 Chaser parameters and corresponding uncertainties.

	Nominal	Uncertainty
Mass [kg]	372	$\pm 5\%$
Moments of inertia (xx, yy, zz) [kg m^2]	(200, 200, 140)	$\pm 10\%$
Products of inertia (xy, yz, zx) [kg m^2]	0	± 1
Center of mass position (x,y,z) [m]	0	± 0.05

The robustness analysis in presence of sloshing effects has been considered as well, assuming the presence of one tank modelled as a mass-spring-damper system with the values reported in Table 3.

Table 3 Chaser sloshing modelling.

	Tank 1
Mass [kg]	74
Frequency [rad/s]	0.0063 (uncertainty $\pm 20\%$)
Damping	0.001 (uncertainty $\pm 40\%$)

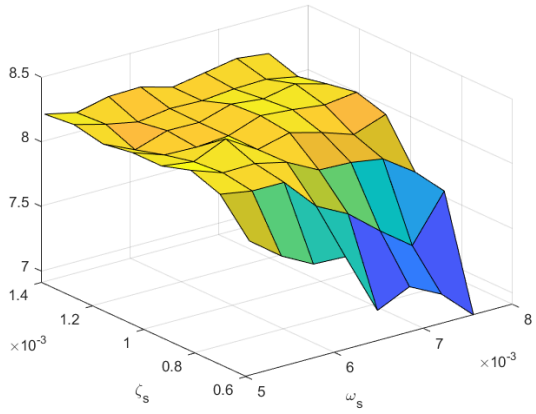

Figure 6 Lower bound of the stability margin for the selected grid points (ζ_s= damping, ω_s = frequency).

Using the *robstab* routine in MATLAB, the stability margins have been computed also in this case, yielding a lower bound of approximately 0.3 and an upper bound of 2.5. These values provide insights into robust stability, although no definitive conclusions can be drawn solely based on them. To address this numerical limitation, a discretization approach has been employed for the sloshing frequency and damping parameters within the uncertainty set. Subsequently, for each point on the grid, a robust analysis has been performed while considering the remaining parameters as uncertain. In the specific case being investigated, an $8x8$ grid with equally spaced points has been selected. This translates to a total of 64 stability margins to be computed (i.e., 64 evaluations of the structured singular value). It is important to note that this analysis is computationally intensive and feasible only for a limited number of parameters to be gridded, as the number of evaluations grows exponentially. The results are shown in Figure 6, thanks to which robust stability can be assessed. According to this analysis, sloshing is not affecting the performance of the closed-loop system.

The validity of the solution has been assessed via nonlinear simulations considering the nonlinear rigid body dynamics of the space robot with actuators saturation and thrusters' discretization. The parameters for the simulation have been randomly sampled in the uncertain set. The base performs station keeping with respect to the target, while the end effector moves to grasp the target as shown in Figure 7, in which the error between end-effector gripper and target grasping point is reported.

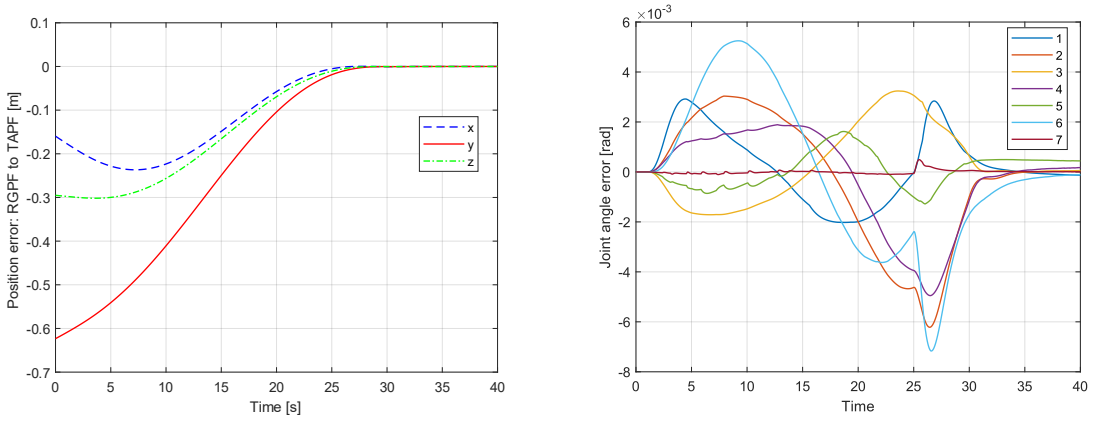

Figure 7 Relative end-effector to grasping point distance time

Even if the robot has to track a fast-moving target, the error for the joint variables (closely related to the end-effector performance) is very small (Figure 7, right). The performance achieved by the control law are good and the end-effector error converges to zero according to the performance requirements and respecting the actuator limits. The effect of thrusters discretization can be seen on the position error dynamics of the base (discontinuities on the derivative in Figure 8).

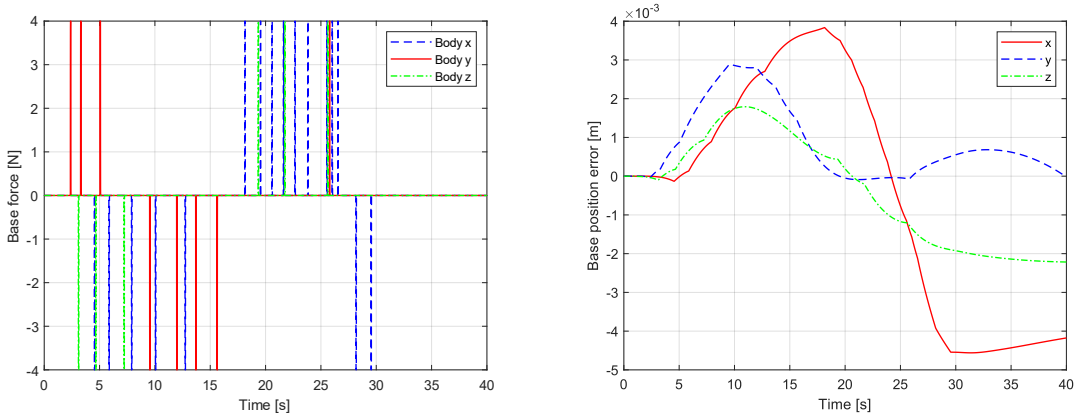

Figure 8 Thrusters force time history.

6 CONCLUSIONS

This paper has presented the development and numerical validation of a combined control design for an autonomous spacecraft equipped with a robotic arm to perform the capture of a target tumbling object. The control law is based on the concept of computed-torque control for space-robots using a robust tuning procedure that allows imposing desirable performance and control moderation requirements. The simulation results have shown excellent performance of the developed algorithm in preliminary nonlinear simulation. Of note, the proposed controller has been tested successfully in a more realistic simulation environment [4] developed by a consortium of Italian universities within the project “Preparation of enabling space technologies and building blocks: GNC and Robotic Arm Combined Control” funded by ESA.

ACKNOWLEDGMENTS

This work was supported by the European Space Agency (Contract NO. 4000132611/20/NL/CRS) in the context of the Open Invitations To Tender (ITT): ESA ITT 1-10250/20/NL/CRS.

References

[1] A. Flores-Abad, O. Ma, K. Pham e S. Ulrich, «A review of space robotics technologies for on–orbit servicing,» *Progress in Aerospace Sciences,* vol. 68, p. 1–26, 2014.

[2] European Space Agency, «Preparation of enabling space technologies and building blocks: GNC and robotic arm combined control,» 2020, Accessed online May 2023 https://nebula.esa.int/content/preparation-enabling-space-technologies-and-building-blocks-gnc-and-robotic-arm-combined.

[3] J. Teelar, E. S., M. De Stefano, W. Rackl, R. Lampariello, F. Ankersen e J. Gil-Fernandez, «Coupled control of chaser platform and robot arm for the e.Deorbit mission,» in *10th International ESA Conference on Guidance, Navigation and Control*, 2017.

[4] F. Basana et al, «GNC and robotic arm combined control for spacecraft close proximity operations,» Submitted to *Multibody System Dynamics,* 2023.

[5] S. Dubowsky e E. Papadopoulos, «The Kinematics, Dynamics, and Control of Free-Flying and Free-Floating Space Robotic Systems,» *IEEE Transactions on Robotics and Automation,* vol. 9, n. 5, pp. 531-543, 1993.

[6] R. Featherstone, «Rigid Body Dynamics Algorithms,» Springer, 2008.

[7] A. Giordano, A. Dietrich, C. Ott e A. Albu-Schaffer, «Coordination of thrusters, reaction wheels, and arm in orbital robots,» *Robotics and Autonomous Systems,* vol. 131, 2020.

[8] P. Apkarian e D. Noll, «Nonsmooth H-infinity synthesis,» *IEEE Transactions on Automatic Control ,* vol. 51, n. 1, pp. 71-86, 2006.

[9] M. Wilde, S. Kwok Choon, A. Grompone e M. Romano, «Equations of Motion of Free-Floating Spacecraft-Manipulator Systems: An Engineer's Tutorial,» *Frontiers in Robotics and AI,* vol. 5, p. 41, 2018.

[10] E. Papadopoulos e S. Dubowsky, «Coordinated manipulator/spacecraft motion control for space robotic systems,» in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, 1991.

[11] D. Invernizzi, M. Lovera e L. Zaccarian, «Global robust attitude tracking with torque disturbance rejection via dynamic hybrid feedback,» *Automatica,* vol. 144, 2022.

[12] B. Siciliano, L. Sciavicco, L. Villani e G. Oriolo, «Robotics: Modelling, Planning and Control,» London: Springer-Verlag London, 2009.

[13] A. Liegeois, «Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms,» *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 7, n. 12, pp. 868-871, 1977.

[14] J. Wang, Y. Li e X. Zhao, «Inverse Kinematics and Control of a 7-DOF Redundant Manipulator Based on the Closed-Loop Algorithm,» *International Journal of Advanced Robotics Systems,* vol. 7, n. 4, pp. 1-10, 2010.

[15] T. Yoshikawa, «Manipulability of Robotic Mechanisms,» *The International Journal of Robotics Research,* vol. 4, n. 2, pp. 3-9, 1985.

[16] P. Rocco, «Control of industrial robots - Kinematic redundancy,» *Lecture Notes,* Politecnico di Milano.

[17] F. Ankersen, «Guidance, Navigation, Control and Relative Dynamics for Spacecraft Proximity Maneuver», Phd Thesis, 2010.

[18] P. Apkarian,, P. Gahinet e C. Buhr, «Multi-model, multi-objective tuning of fixed-structure controllers», in *European Control Conference (ECC)*, 2014.