75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.


IAC–24–D1.4B.5

# LEARNING-BASED TRAJECTORY OPTIMIZATION OF A SPACE MANIPULATOR POST TARGET-GRASPING

**Lorenzo Capra** [a*], **Matteo D'Ambrosio** [a], **Michèle Lavagna** [a]

[a] *Politecnico di Milano, Department of Aerospace Science and Technology, Via La Masa, 34, 20156, Milano - Italy*

[*] Corresponding Author: lorenzo.capra@polimi.it

The application of space manipulators for In Orbit Servicing (IOS) represents a paramount pursuit for leading space agencies, given the substantial threat posed by space debris to operational satellites and forthcoming space endeavours. A pivotal prerequisite for harnessing the potential of space servicing capabilities lies in the ability to repurpose space assets beyond their anticipated operational lifespan. Within this context, space robotics and heightened autonomy emerge as indispensable catalysts, unlocking a wealth of possibilities. The paper discusses innovative guidance and control strategies for the path-planning of a spacecraft equipped with a 7-dof robotic arm, in the peculiar case of post target-grasping. An uncooperative target of unknown inertial properties is captured by the space manipulator system and the overall stack dynamics is analysed. The trajectory of the resulting multi-body system is optimized with the goal of reaching a specified set pose, while maximizing a designed objective function and satisfying certain constraints. A learning-based approach is selected, due to its adaptivity to uncertainty in the surrounding environment and complex dynamics. Reinforcement Learning methods are here preferred thanks to the generalizing learning capabilities of these AI approaches, that are specifically exploited to gain flexibility and robustness in robotic guidance and control policies. These techniques emerge as a viable solution for effective decision-making, especially in problems where future dynamical states and uncertainty must be considered. The problem at hand focus on the post-grasping scenario of an IOS mission, in which an unknown target is captured by the space manipulator and the overall system needs to reconfigure according to a specified desired pose. By training the RL-based agent on simulation environment, the decision making is optimized to maximize the objective function. This function is designed considering the desired pose, via an Artificial Potential Field, and then augmented with a signal related to the robotic arm singularity, which is evaluated via singular value decomposition of the Generalized Jacobian Matrix. The resulting learning-based agent undergoes an extensive Montecarlo analysis to assess the effect of several randomized parameters; mainly the unknown properties of the target are randomized and varied across the simulations to evaluate the generalizing capabilities of the RL decision-making policy. The findings of this research highlight the applicability of RL in advancing the capabilities of space robotics for In Orbit Servicing.

**Keywords:** In Orbit Servicing, Space Robot, Learning-based guidance, Stack Dynamics, Reinforcement Learning

## 1. Introduction

Interest in the development of in-orbit servicing (IOS) technologies for satellites and space systems has surged recently. As the scope of space exploration and utilization widens, the need for effective methods to repair, refuel, and reposition satellites has become increasingly important. Central to IOS missions is the precise operation of space manipulators, robotic systems designed to perform these tasks in space. These manipulators are crucial during the close proximity phases of servicing missions, where their ability to execute complex tasks is essential. However, the unpredictable nature of working with uncooperative targets requires these systems to possess a high degree of motion control flexibility and adaptability to their environment [1]. Specifically after the grasping of an uncooperative object, the unknown inertial parameters of the latter poses quite a challenge when developing robust guidance and control algorithm for these systems.

The swift advancements in Artificial Intelligence are poised to significantly enhance the capabilities of autonomous Guidance, Navigation, and Control (GNC) systems. Among various AI techniques, Reinforcement Learning (RL) stands out as a promising approach for tackling complex decision-making problems modeled as Markov Decision Processes (MDPs). By merging the generalization power of neural networks with RL methods, Deep Reinforcement Learning (DRL) has emerged. DRL is extensively utilized in solving planning problems due to its proficiency in managing high-dimensional state and action spaces, as well as its

effectiveness in dealing with Partially Observable Markov Decision Processes (POMDPs) [2]. These methods have been recently adopted for the GNC of space systems, like for relative fly-around trajectories in [3, 4], landing operations [5], but also for space orbital robotics, though literature is still quite scarce in this regard. The main contributions are provided by [6, 7], which approximated the 7-DOF space robot decision making via a DRL agent for the motion synchronization with an uncooperative target, but also [8, 9, 10]. This work starts from the strategy developed by D'Ambrosio [6] *et al.* and investigates its capabilities in a different scenario. The work presented here explores the application of Deep Reinforcement Learning (DRL) for guiding a space robot after it has successfully grasped an uncooperative target and stabilized its motion. This scenario poses unique challenges, such as managing the dynamic interaction between the robot and the target, dealing with the uncertainties in the target's inertial parameters, and ensuring precise control in a highly unpredictable environment. The need for real-time decision-making and adaptability in the face of incomplete information adds to the complexity of the task, making DRL a suitable and promising approach for addressing these challenges. A similar scenario is investigated in [11], where a dual-arm space robot reconfigure the grasped target position with more classical control strategies. The main innovation of this work is indeed in the methodology, so testing and verifying the performance of a Deep Reinforcement Learning agent for this scenario and extending the future possible applicability of Artificial Intelligence strategies to improve the adaptivity and the flexibility of guidance and control algorithms.

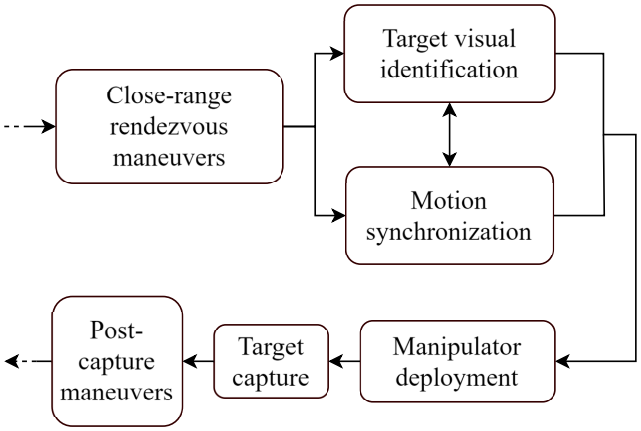


Fig. 1: Robotic IOS mission phases.

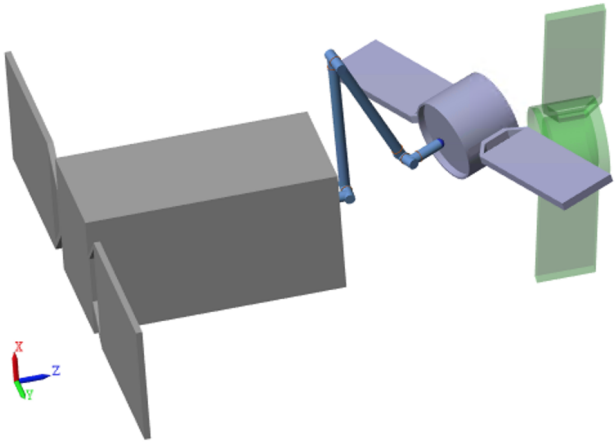

Fig. 2: Post-capture reconfiguration scenario.

## 2. Problem definition

Robotic In-Orbit Servicing operations refers to all those activities involving a *servicer*, that is the spacecraft equipped with a robotic manipulator, and a *client*, that could be a target object. These type of missions generally involves a similar set of phases, that are reported in Figure 1.

The scenario under investigation focuses on the last block in the diagram, so post-capture maneuvers. The works starts from the assumption of a target that has already being grasped by the end-effector of the robotic manipulator, as well as stabilized. The next phase, depending on the objective of the mission, would be to reconfigure the system, by moving the target to a desired pose, with both a reference position and attitude. The scenario is visualized in Figure 2.

The system consists of a 6-DOF platform, that is the spacecraft, equipped with a 7-DOF robotic arm and at its tip the target object is rigidly attached through a fixed joint. The target shape and dimension have been selected arbitrarily, but they could be easily generalized to any type of object and the formulation would be identical. The green part with a slightly lower opacity indicates the desired pose of the target and is indeed the reference for the guidance and control blocks of the GNC chain. Thus, the main objective is to reconfigure the system such that the space manipulator move and reorient the target in such a way that it matches the green desired pose. Navigation information, like the absolute state of the system and the reference one are assumed to be known, to simplify the process, but noise levels coming from the estimation process could be potentially integrated in the simulation tool.

*2.1 Space manipulator dynamics*

This section provides a brief overview of the equations of motion for a space manipulator with $N$ degrees of freedom. In this context, the multi-body system is described as free-flying, meaning that the spacecraft is actively controlled in both translation and rotation. This differs from the free-floating scenario, where only the manipulator moves while the spacecraft remains passive and unpowered [12]. Employing the direct path approach, the spacecraft's center of mass serves as the representative point for translational motion, and allow to derive the system's kinematics and dynamics. This approach leads to more concise equations [13]. The selected generalized variables, which compose the space manipulator state, are reported in Eq. 1:

$$q = [r_0, R_0, q_m] \tag{1}$$

where $r_0$ is the position vector of the base spacecraft in inertial frame, $R_0$ is the orientation of the base spacecraft with respect to the inertial frame, which employ a quaternion representation in this work, and $q_m$ contains the joint angles of the robotic arm. From these variables, thanks to the kinematic tree topology of the system it is possible to compute the kinematics variables of each element in the chain, so each link and joint until the end effector, as well as the target that could be considered as the last link of the kinematic chain. The same considerations are applicable to the differential kinematics to compute linear and angular velocities of all the elements. Specifically, the end effector linear velocity $\dot{r}_{ee}$ and angular velocity $\omega_{ee}$ can be expressed as a function of the generalized variables $q$ through the system jacobian matrix $J_E$, as shown in Eq. 2.

$$\begin{bmatrix} \dot{r}_{ee} \\ \omega_{ee} \end{bmatrix} = J_E(R_0, q_m)\, \dot{q} \tag{2}$$

The Generalized Jacobian Matrix (GJM), first introduced in [14], is a function of the spacecraft attitude ($R_0$) and the manipulator configuration ($q_m$). The variable $\dot{q}$ expresses the time derivative of the generalized variables and its extended form is reported in Eq. 3:

$$\dot{q} = [v_0, \omega_0, \dot{q}_m] \tag{3}$$

where $v_0$ is the linear velocity vector of the base spacecraft in inertial frame, $\omega_0$ is the angular velocity of the base spacecraft in inertial frame, and $\dot{q}_m$ contains the joints angular velocities.

Formulating the Newton-Euler equations of motion, one obtains the compact form expressed in Eq. 4.

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} = \tau \tag{4}$$

$H \in \mathbb{R}^{(6+N)\times(6+N)}$ is the symmetric, positive -definite Generalized Inertia Matrix (GIM), $C \in \mathbb{R}^{(6+N)\times(6+N)}$ is the Convective Inertia Matrix (CIM), containing the nonlinear contributions, the Coriolis and centrifugal forces, and $\tau \in \mathbb{R}^{(6+N)}$ is the vector of generalized forces in the joint space. The contributions of the base-spacecraft-link, of the manipulator and of their coupling can be made explicit writing the equations of motion as in Eq. 5.

$$\begin{bmatrix} H_0 & H_{0m} \\ H_{0m}^T & H_m \end{bmatrix} \begin{bmatrix} \ddot{q}_0 \\ \ddot{q}_m \end{bmatrix} + \begin{bmatrix} C_0 & C_{0m} \\ C_{m0} & C_m \end{bmatrix} \begin{bmatrix} \dot{q}_0 \\ \dot{q}_m \end{bmatrix} = \begin{bmatrix} \tau_0 \\ \tau_m \end{bmatrix} \tag{5}$$

The parameter $q_0$ in Eq. 5 represents the generalized variables associated to the spacecraft only, and its first and second derivatives are the velocity and acceleration contributions both in translation and rotation. Note that $\tau_0$ contains the effects of on board actuators like thrusters and momentum exchange devices, being the space manipulator in a free-flying mode.

To derive the kinematic and dynamic properties of the system the MATLAB library SPART (SPAce Robotic Toolkit) is used [15], a modeling and control software package for mobile-base robotic multi-body systems. It is based on a Newton-Euler approach, which makes use of the Decoupled Natural Orthogonal matrix to solve the forward and inverse dynamics in efficient and recursive fashion. To solve the equations of motion instead, a model of the space manipulator is implemented with the Simulink Simscape Multibody library, which provides a simulation environment for 3D mechanical systems, using blocks to represent bodies (links and joints), elements, connections, forces, and so on. The space manipulator considered is a 6-DOF spacecraft equipped with a 7-DOF redundant robotic arm, consisting of 3-DOF of shoulder, 1-DOF of elbow and 3-DOF of wrist. The geometry, configuration and connections between the different body elements are described in a URDF (Unified Robotic Description Format) file, which is a XML with tags for each characteristic of the elements composing the system. This file initializes the configuration and enters the SPART functions to compute the kinematics and dynamics. This initialization file also contains all the inertial properties of each body, like the center of mass, the mass and the inertia matrix. Hence, the accuracy of this file becomes of great importance when dealing with unknown and uncooperative objects. These parameters affects the computation of the GIM and CIM, which may then enter the formulation of the selected control algorithm. This work assumes that the mass of the target, and consequently its inertia matrix, are unknown, and are neglected inside the URDF file of the overall system. This in turn stresses the performance of the DRL agent, checking whether it can deal with this uncer-

tainty.

## 3. Guidance & Control strategy

This section elaborates on the guidance and control parts of the simulation. The goal of this two blocks is the online generation of a trajectory that the system shall follow and the required control actions to track it. It is important to clarify that the algorithm proposed in this work outputs only the guidance of the robotic arm, while in the control part the coupling between the spacecraft base and the manipulator is considered. A block diagram clarifying this point is reported in Fig. 3.

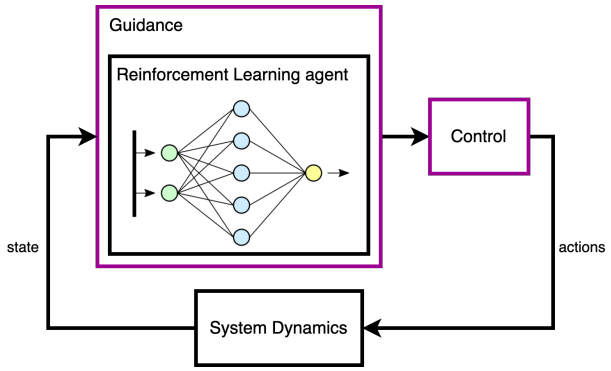


Fig. 3: Guidance and Control scheme.

The implementation of the two blocks is now described in greater details.

### *3.1 Reinforcement Learning guidance*

Within the expansive domain of Artificial Intelligence, Reinforcement Learning (RL) stands out as a compelling machine learning approach due to its effectiveness in addressing decision-making problems. RL is fundamentally based on an agent's interaction with its environment, where it continuously exchanges information, receives new observations, and obtains rewards—either positive or negative—based on its actions. This dynamic interaction enables the agent to refine its policy, thereby enhancing its decision-making capabilities to optimize performance according to a predefined objective function. When RL techniques are combined with the generalization power of Neural Networks, the result is Deep Reinforcement Learning (DRL), a potent tool extensively utilized in planning and optimization tasks. In DRL, the policy function, which maps states to actions, is developed through iterative interactions with the environment and optimized to maximize a reward-driven objective function. This policy function is approximated using Neural Networks, allowing for more sophisticated and scalable solutions.

#### *3.1.1 DRL Algorithm*

The selected Deep Reinforcement Learning method is Proximal Policy Optimization (PPO) [16]. It is a cutting-edge on-policy, model-free deep reinforcement learning (DRL) algorithm within the policy gradient methods family. Compared to its predecessor, Trust Region Policy Optimization (TRPO) [17], PPO offers a simpler implementation and higher sample efficiency, leading to faster training without sacrificing reliability. PPO excels in tackling complex, high-dimensional, and partially observable continuous control problems, outperforming many competitors in various benchmarks and ensuring high training stability. It operates based on the Actor-Critic framework, where the Actor dictates the agent's decision-making process (the policy $\pi$), and the Critic assesses the Actor's actions within the environment. The algorithm optimizes the agent's decision making by maximizing the reward signal it receives while interacting with the surrounding environment through its actions and observing its state. The optimal policy in an infinite-horizon problem is found through Equation (6), and provides the agent with the maximum reward when applied in the environment:

$$\pi^* = \underset{\pi}{\text{argmax}} E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R(a_k, s_k) \right] \tag{6}$$

where the discount factor $\gamma \in [0, 1]$ introduces a decay of rewards obtained distantly in time and measures whether the agent seeks short-term or cumulative rewards, and $R(a_k, s_k)$ is the reward signal, function of the action $a$ and the state $s$ at time step $k$. The improved robustness of PPO with respect to its predecessors comes from the way it handles the loss function, by clipping it and avoiding excessive variation of the network parameters during the training process. This would ideally provide increase robustness against catastrophic unlearning of the neural networks that are used to approximate both the actor and the critic and ensure a smoother learning.

#### *3.1.2 State and action space*

The state vector provides the agent with essential contextual information about its environment, enabling it to generate appropriate actions. This encompasses the current state of both the robotic arm and the reference pose, which is crucial for determining the manipulator's trajectory. By integrating this data, the agent can effectively plan and execute movements, ensuring precise and successful interac-

tions within the operational environment. The state vector is reported in Equation (7).

$$\vec{s} = [\vec{q}_m, \dot{\vec{q}}_m, \vec{\tilde{r}}, \vec{\tilde{v}}, \vec{\tilde{\theta}}, \vec{\tilde{\omega}}]^\top \tag{7}$$

where $(\vec{q}_m, \dot{\vec{q}}_m)$ are the joint angles and rates of the manipulator, $(\vec{\tilde{r}}, \vec{\tilde{v}}, \vec{\tilde{\theta}}, \vec{\tilde{\omega}})$ are the errors between the current and objective end-effector states.

The action space represents all of the possible actions the agent can produce, defined as the desired velocity of each joint, at each timestep. The selected action space is continuous, resulting in an agent that can freely generate $\dot{\phi}_i^*$ with any magnitude, as in Equation 8.

$$\vec{a} = \dot{\vec{q}}_m^* = [\dot{\phi}_1^*, \dot{\phi}_2^*, \dot{\phi}_3^*, \dot{\phi}_4^*, \dot{\phi}_5^*, \dot{\phi}_6^*, \dot{\phi}_7^*]^\top \tag{8}$$

The torques produced by the controller are bounded through control saturation. A Gaussian stochastic policy is used, resulting in an actor that generates a mean $\mu_i$ and standard deviation $\sigma_i$ for each action $\dot{\phi}_i^*$. During training, the action applied in the environment is sampled out of this distribution to increase agent exploration [2], since the agent could generate several different actions, even when provided with the same observation. Finally, the desired joint angles and accelerations are retrieved starting from the vector $\dot{\vec{q}}_m^*$, to provide them to the controller. The decision to have the decision-making policy outputting the angular velocities of the joints and not directly the torques to be applied is to simplify the learning process because only kinematic information are relevant inside the state vector.

#### *3.1.3 Reward function*

The primary method to enhance the performance of a manipulator's guidance system is through a system of rewards and penalties. Designing an appropriate reward function is crucial, as it significantly influences how quickly and effectively the policy converges to an optimal state, directly affecting the overall performance. Given the challenges of training with sparse rewards in high-dimensional state and action spaces, reward shaping has been implemented. This approach involves using an Artificial Potential Field (APF) [18, 8] to better guide the learning process. Its formulation is reported in Equations (9).

$$U_k = -\tilde{r} + \frac{10}{1+\tilde{r}_{ax}} + \frac{10}{1+\tilde{r}_{tx}} + \frac{10}{1+\tilde{\theta}} \tag{9}$$

where $\tilde{r}$ is the magnitude of the error between the desired and current positions of the end effector and connnected target, $\tilde{r}_{ax}$ and $\tilde{r}_{tx}$ are the projections of $\tilde{r}$ parallel and transverse to the X-axis of the space robot's body frame, and $\tilde{\theta}$ is the scalar error angle between the desired and current attitude of the end effector and connected target, in axis-angle representation. The reward is given as a function of the end effector's potential variation ($\Delta U$) between timesteps (Equations (10) and (11)):

$$\Delta U = U_k - U_{k-1} \tag{10}$$

$$R_k = \begin{cases} \Delta U & \text{if } \Delta U \geq 0 \\ 1.5\ \Delta U & \text{if } \Delta U < 0 \end{cases} \tag{11}$$

where the $1.5\times$ multiplier discourages the end effector from moving along equipotential surfaces. A bonus sparse reward of $+0.01$ is provided while $\tilde{r}_{ax}$, $\tilde{r}_{tx}$, and $\tilde{\theta}$ are simultaneously below a desired threshold. A time minimization objective is also implemented through a constant penalty of $-1/200$ at each timestep, received if the agent is outside of the same ($\tilde{r}_{ax}$, $\tilde{r}_{tx}$, $\tilde{\theta}$) tolerances. This reward component increases the overall performance of the agent and avoids episode failures caused by the robotic reconfiguration time exceeding the maximum episode length.

### *3.2 Control strategy*

A state-of-the-art Computed Torque Control algorithm is used as reference strategy. It revolves around a model-based feedback linearization of the equations of motion, consisting of an inner nonlinear compensation loop and an outer loop with an exogenous control signal employing a proportional-derivative (PD) strategy, and resulting in the following Eq. 12.

$$\tau = H(q)(\ddot{q} + K_d \dot{e}_q + K_p e_q) + C(q, \dot{q})\dot{q} \tag{12}$$

Note that the control action entering the space manipulator block contains both torques and forces, meaning that all the selected generalized variables in Eq. 1 can be commanded. In such a way, both a desired pose for the spacecraft and a desired configuration of the robotic arm can be obtained concurrently.

The limitation of this strategy is in the fact that the theoretical exact feedback linearization can never be achieved, because of the uncertainty on the dynamics parameters of the system. This effect leads to errors in the online evaluation of the inertia matrices $H$ and $C$, which appears in the control law. This is even more relevant when the correct value of these matrices cannot be retrieved, since the mass of the target attached rigidly to the end-effector is not known a priori. Moreover, the selection of the PD gains may not be optimal. The values assumed in this work are reported in Table. 1, but some other studies propose adaptive strategies to select them, as in [19].

Table 1: Tuned PD gains.

| Controller | $K_P$ | $K_D$ |
|---|---|---|
| Base | 0.4 | 0.3 |
| Manipulator | 2.5 | 1.25 |

## 4. Simulation Results

The agent is first trained by interacting with the environment which is now detailed. It encapsulates all the properties of the simulations, from the inertial parameters of the objects in play, to the initial conditions of all the variables affecting each run. In the scenario investigated, the target is assumed to be a cylinder with two solar panels rigidly connected to the end effector of the space manipulator. The inertial properties of the target are automatically computed from the selected mass and the given geometry. For the nominal case a mass of $M_{trg} = 20kg$ is chosen.

Regarding the initial conditions, these refers to the generalized variables in Eq. 1 and the desired pose in which the target should be set:

- the manipulator initial joint angles are sampled from a uniform distribution with mean $\vec{q}_m = [0, 285, 0, 210, 0, 75, 0]^\top$deg and a deviation of $[-15, 15]$ deg.
- the desired pose is sampled from a uniform distribution for both position and orientation. The position is randomly selected inside a sphere with $d = 50$ cm, starting from a nominal distance of 5 m from the space manipulator platform. The orientation instead is selected inside a cone with an aperture angle of 30 deg.

The simulations are terminated only when the manipulator's configuration become singular, to avoid mathematical issues that could cause the DRL algorithm to fail. To do so, the singular value decomposition of the Jacobian matrix is computed at each timestep and a lower bound threshold is defined on its output. This way the agent would learn to avoid these configurations, as it would stop receiving positive reward signals. Regarding the PPO hyperparameters, the agent's sample time is set to 0.3 seconds as a compromise between computational cost, convergence, and the space robot's reactivity. The set of hyperparameters used for the training process is the same as the one in [7].

The training profile is reflected in the average reward signal received by the agent during the simulations, which is reported in Fig. 4.

The agent is trained for 3000 episodes and the average reward signal increase during this process, suggesting also its performance are improving, as long as the reward function is designed properly.

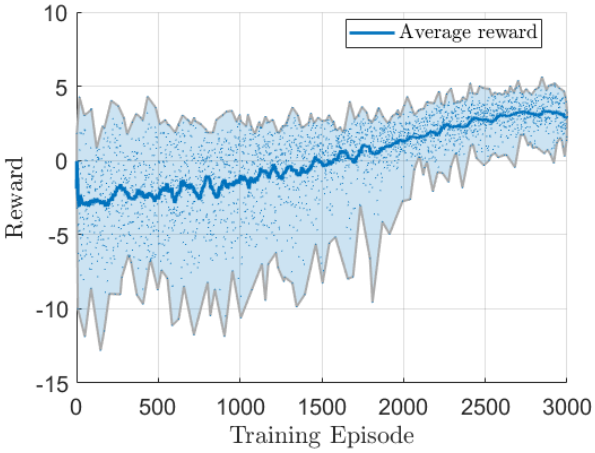


Fig. 4: Reward signal profile during training.

### *4.1 Agent testing*

Once trained, a Montecarlo simulation, with $N = 100$ runs, is set-up to evaluate the performance of the agent. The agent's success in the simulation is measured by its ability to maintain the end effector and the target attached to it within a specified tolerance from the target state, both in terms of position and orientation, continuously for at least $t_{min} = 30$ s. This contrasts with the common approach in much of the literature, where an episode is deemed successful and the simulation is terminated as soon as the end effector first enters the target threshold. In such highly dynamic scenarios, this method fails to demonstrate the end effector's ability to consistently stay aligned with the grasping position, and it can artificially inflate the agent's performance. The minimum error thresholds that defines an episode's success are $\tilde{r}_{ax}, \tilde{r}_{tx} < 5$ cm, and $\tilde{\theta} < 5$ deg. In the nominal case, the one for which the agent is trained, the results of the analysis are reported in Fig. 5, where each run lasts for 420 s.

In this case, the agent obtains a $100\%$ success rate, since in all the simulations the stack composed by end effector and target remains inside the desired thresholds for at least 30 s. Other metrics are also evaluated, like the first instant in which the stack enters the desired region and the maximum time in which it remains there. These results are presented in Fig. 6 and Fig. 7.

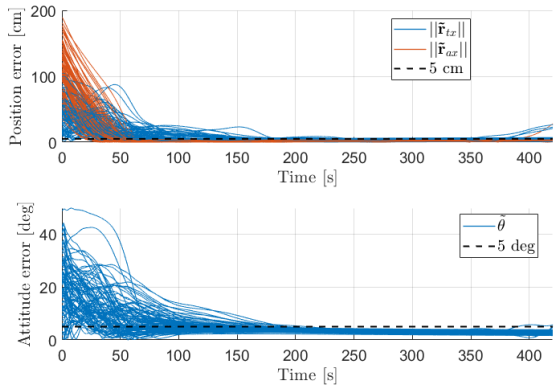


Fig. 5: Baseline results of the Montecarlo analysis on the agent performance.

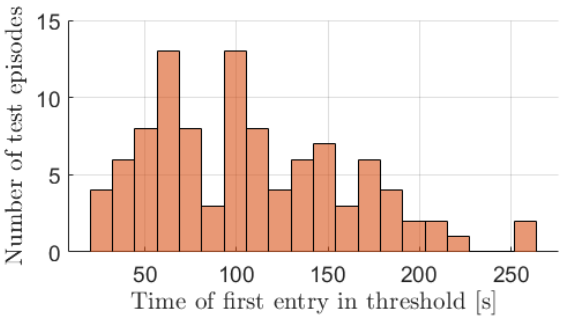


Fig. 6: Baseline results of the Montecarlo analysis on the stack first entry.

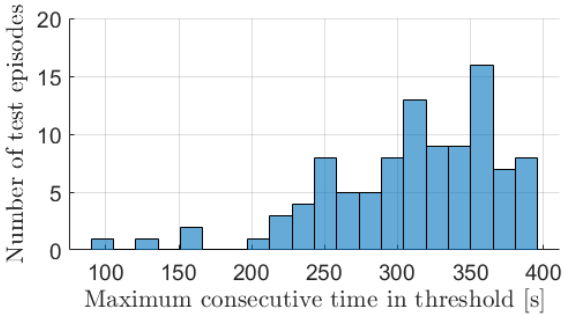


Fig. 7: Baseline results of the Montecarlo analysis on the maximum time in threshold of the stack.

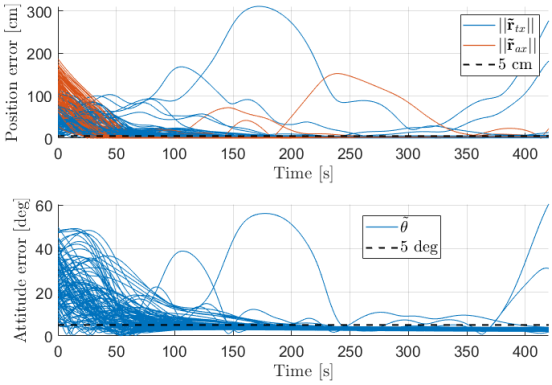


Fig. 8: Montecarlo analysis with random target mass.

### *4.2 Robustness analysis*

This section presents all the analysis performed starting from the nominal scenario and adding layers of complexity for the learning-based guidance. This is done by sequentially introducing modifications in the simulation set-up that the agent has not seen during the training phase. Such investigation is crucial to understand the generalizing capabilities of these types of approaches, since this can be one the main advantages above classical methods.

#### *4.2.1 Random target mass*

The first modification affects directly the inertial properties of the target, whose mass is now randomized and sampled from a uniform distribution $\in [10, 200]$ kg. The same Montecarlo analysis is performed and the results are compared to the baseline.

As visible in Fig. 9, this time 97% of the simulations are successful, but it seems like there is no real correlation to the mass of the target, as this value is quite different in the three failed episodes (red dots). So, the failing of certain episodes can be attributed to a superposition of different effects, i.e. the randomized initial conditions and target mass. Concerning the time of first entry and the maximum time inside the threshold, these results are similar to the baseline, meaning that the guidance and control reaction speed is not affected too much by the mass of the target. This is indeed expected, since the DRL agent only generates the desired kinematics of the manipulator, which is independent from the mass of the target.

#### *4.2.2 Flexible target connection*

In this case the assumption of rigid connection between the end effector and the target is relaxed. A spherical joint is added to connect the target to the manipulator's tip and the values for stiffness and damping are selected looking into literature. As the manipulator moves, this joint makes the target rotate around the point of attachment. Once again a Montecarlo analysis is set-up and the results are reported.

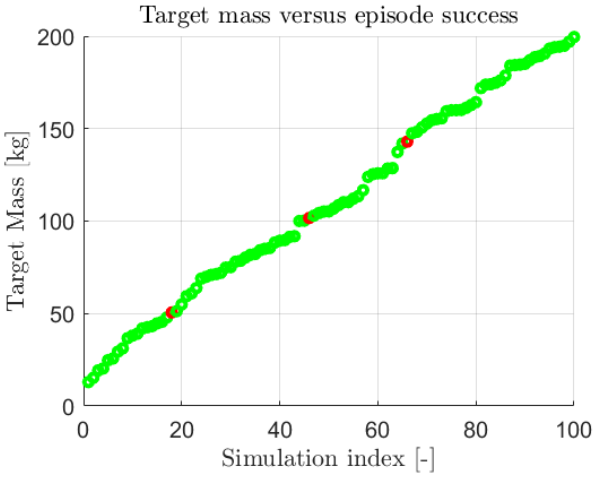


Fig. 9: Episodic success versus target mass.

This time, instead of reporting the attitude error relative to the rigid stack, it is reported the angle difference between the target and the end effector. This is due to the fact that the manipulator's tip motion is the same as in the previous cases, and the stack is inside the desires pose when the angle due to the spherical joint introduced is small.

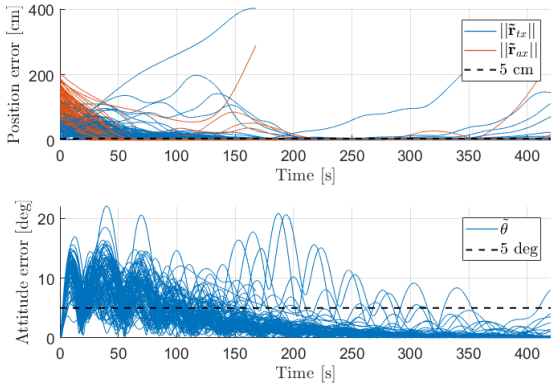


Fig. 10: Montecarlo analysis with random target mass and spherical joint

The success rate in this case is the same as the case before, but the main difference is visible in Fig. 11 and Fig. 12. The peak of the theoretical probability density function fitting these histograms is shifted with respect to the previous case. Indeed, it takes more time to move the stack inside the desired pose and, consequently, the maximum amount of time spent inside the thresholds reduces. This is imputed to the reaction of the spherical joint to the movement of the

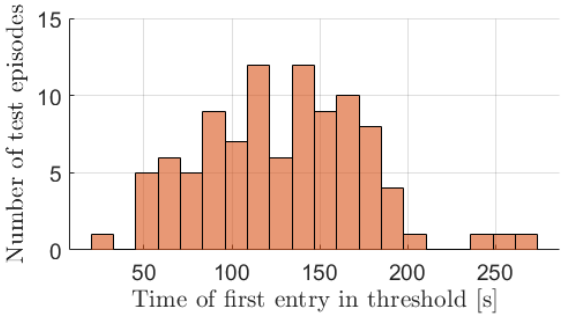


Fig. 11: Results of the Montecarlo analysis with spherical joint on the stack first entry.

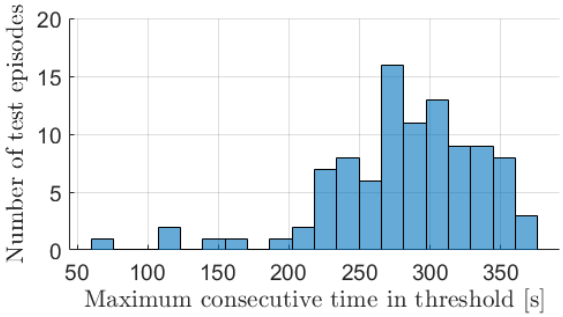


Fig. 12: Results of the Montecarlo analysis with spherical joint on the maximum time in threshold of the stack.

manipulator, which is to be damped to return to a scenario similar to the rigid case. This is also clearly visible in the bottom subplot of Fig. 10, that shows the oscillations of the target around the reference starting condition.

### *4.2.3 Residual target dynamics*

This last case adds a layer of complexity further to the previous analysis. After the target is grasped by the robotic arm, the next phase usually entails some form of rigidization, to erase any residual dynamics the target may exhibit. In this section it is assumed that is phase is not completed and some residual rotational dynamics is still present. Therefore, the target is initialized with a random angular velocity vector, whose components are sampled from a uniform distribution $\in [-5, +5]$ deg/s. The results of this case are reported in Fig. 13, Fig. 14, Fig. 15 and Fig. 16.

Due to the presence of an initial disturbance in the angular velocity of the target around the spherical joint, it takes more time to damp the oscillations, and, as such only 58% of the episodes are successful in the timespan of the simulation. However, the subplot in Fig. 14 clearly shows that the

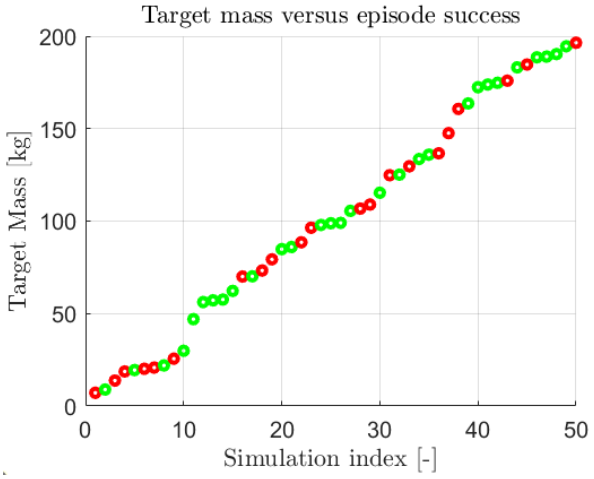


Fig. 13: Episodic success versus target mass with spherical joint and initial rotational dynamics.

behavior of the tracking error is similar to the other cases, meaning that with more time almost all the simulation run would end up successfully. Once again the peak of the fitting curve for both the first entry and the maximum time spent inside the desired pose moved right and left respectively, meaning it takes more time to align the stack with the reference.

## 5. Conclusions

The work proposes an innovative learning-based guidance strategy for the motion planning of the stack dynamical system formed by a space manipulator and an uncooperative target grasped by it. The inertial properties are unknown to the guidance and control system, so a Reinforcement Learning agent is trained to optimize the guidance of the 7-DOF robotic arm. so to place the target in a desired pose, that is randomized in each simulation run. The agent is then tested in nominal conditions, proving its performance, and then the robustness of this algorithm is analysed at different levels. First the target mass is randomized, then the assumption of rigid connection with the end effector is relaxed, and finally some residual rotational dynamics is assigned to the target.

### *5.1 Future developments*

Some further improvements are still to be carried out, like increasing the detailing of the contact dynamics between the manipulator's tip and the target and investigating how this affects the guidance and control chain. A validation test campaign should also be carried out to verify the performance of the algorithm not only at modeling level,

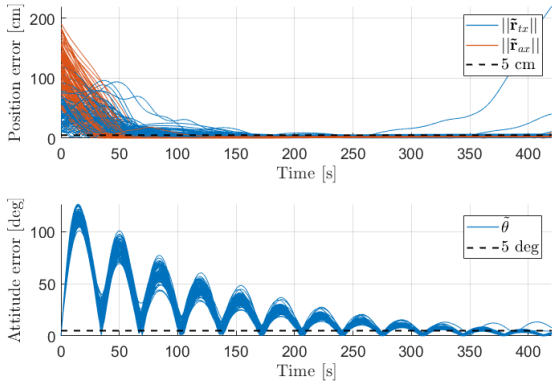


Fig. 14: Montecarlo analysis with random target mass, spherical joint and initial rotational dynamics.

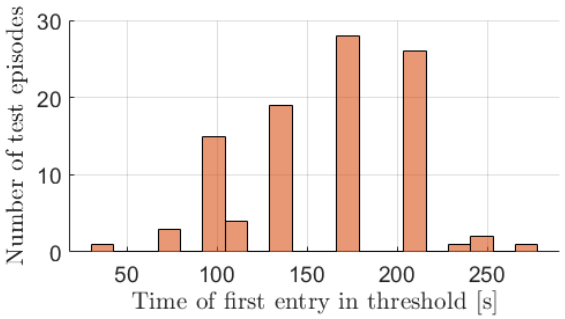


Fig. 15: Results of the Montecarlo analysis with spherical joint and initial rotational dynamics on the stack first entry.

but also at processor and hardware in the loop level. This aspect is critical in the development of AI-based strategies, to increase their reliability in highly-dynamical and complex scenarios.

## References

[1] Borna Monazzah Moghaddam and Robin Chhabra. On the guidance, navigation and control of in-orbit space robotic missions: A survey and prospective vision, 7 2021.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[3] Lorenzo Capra, Andrea Brandonisio, and Michèle Lavagna. Network architecture and action space analysis for deep reinforcement learning towards spacecraft

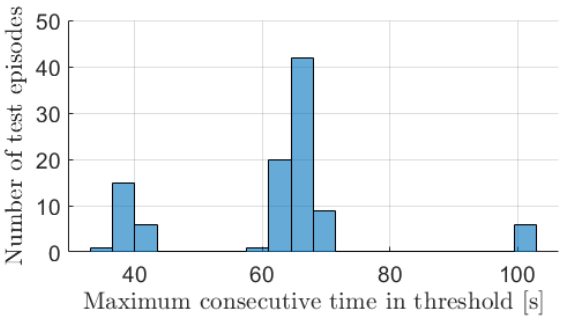


Fig. 16: Results of the Montecarlo analysis with spherical joint and initial rotational dynamics on the maximum time in threshold of the stack.

autonomous guidance. *Advances in Space Research*, 71, 2023.

[4] Andrea Brandonisio, Lorenzo Capra, and Michèle Lavagna. Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target. *Advances in Space Research*, 2023.

[5] Brian Gaudet, Richard Linares, and Roberto Furfaro. Deep reinforcement learning for six degree-of-freedom planetary landing. *Advances in Space Research*, 65(7):1723–1741, 2020.

[6] Matteo D'Ambrosio, Lorenzo Capra, Andrea Brandonisio, Stefano Silvestrini, and Michèle Lavagna. Redundant space manipulator autonomous guidance for in-orbit servicing via deep reinforcement learning. *Aerospace*, 11(5), 2024.

[7] Matteo D'Ambrosio, Lorenzo Capra, and Michèle Lavagna. Deep reinforcement learning for reactive IOS space manipulator operations. *17th Symposium on Advanced Space Technologies in Robotics and Automation*, 10 2023.

[8] Yinkang Li, Danyi Li, Wenshan Zhu, Jun Sun, Xiaolong Zhang, and Shuang Li. Constrained motion planning of 7-DOF space manipulator via deep reinforcement learning combined with artificial potential field. *Aerospace*, 9, 3 2022.

[9] Zeyuan Huang, Gang Chen, Yue Shen, Ruiquan Wang, Chuankai Liu, and Long Zhang. An obstacle-avoidance motion planning method for redundant space robot via reinforcement learning. *Actuators*, 12, 2 2023.

[10] Shengjie Wang, Xiang Zheng, Yuxue Cao, and Tao Zhang. A multi-target trajectory planning of a 6-dof free-floating space robot via reinforcement learning, 2021.

[11] Angelo Stolfi, Paolo Gasbarri, and Arun K. Misra. A two-arm flexible space manipulator system for post-grasping manipulation operations of a passive target object. *Acta Astronautica*, 175:66–78, 2020.

[12] Evangelos Papadopoulos, Farhad Aghili, Ou Ma, and Roberto Lampariello. Robotic manipulation and capture in space: A survey. *Frontiers in Robotics and AI*, 8, 7 2021.

[13] S. Ali A. Moosavian and Evangelos Papadopoulos. Explicit dynamics of space free-flyers with multiple manipulators via spacemaple. *Advanced Robotics*, 18:223–244, 2004.

[14] Continuous path control of space manipulators mounted on omv. *Acta Astronautica*, 15(12):981–986, 1987.

[15] Marcello Romano, Josep Virgili-Llop, and Jerry V Drew Ii. Spart spacecraft robotics toolkit: an open-source simulator for spacecraft robotic arm dynamic modeling and control. *6th International Conference on Astrodynamics Tools and Techniques*, 2016.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms (PPO). *arXiv:1707.06347*, 2017.

[17] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *arXiv:1502.05477*, pages 1889–1897, 2015.

[18] Visak Kumar, David Hoeller, Balakumar Sundaralingam, Jonathan Tremblay, and Stan Birchfield. Joint space control via deep reinforcement learning, 11 2020.

[19] Lorenzo Capra and Michèle Lavagna. Adaptive space robot motion synchronization towards tumbling uncooperative target grasping. *74th International Astronautical Congress*, 10 2023.