Journal of Spacecraft and Rockets
Vol. 62, No. 3, May–June 2025

Meta-Reinforcement Learning for Spacecraft Proximity Operations Guidance and Control in Cislunar Space

Giovanni Fereoli*
University of Colorado Boulder, Boulder, CO 80309
Hanspeter Schaub†
University of Colorado Boulder, Boulder, CO 80309
and
Pierluigi Di Lizia‡
Politecnico di Milano, Milano 20156, Italy

https://doi.org/10.2514/1.A36100

Innovative lightweight and model-free guidance algorithms are essential to achieve full autonomy of spacecraft and address future space exploration challenges. In the near future, this type of technology will be essential for cislunar space proximity operations, such as NASA's Artemis program and its Lunar Gateway project. In this scenario, a meta-reinforcement learning algorithm is employed to address the real-time optimal guidance problem of a spacecraft in the cislunar environment. Non-Keplerian orbits pose complex dynamics, where classic control theory may be less adaptable and more computationally expensive compared to machine learning approaches. Meta-RL stands out for its ability to *learn how to learn* through experience, training on various tasks to enhance efficiency, and effectiveness in tackling new ones. By modeling a stochastic optimal control problem within the circular restricted three-body problem framework as a Markov decision process, an LSTM-based network agent is trained using proximal policy optimization, considering operational constraints and stochastic effects for safety and robustness evaluation. Additionally, an MLP-based agent and an optimal control pseudo-spectral solution are assessed for comparison. The resulting tool autonomously guides spacecraft in cislunar proximity operations, approximating the optimal control solution with a versatile algorithmic framework. This ensures both robustness and computational efficiency.

Nomenclature

$\mathcal{A}$	=	action space
A_t, a	=	action at time step t
$\mathcal{B}$	=	spacecraft body reference frame
$\mathcal{C}$	=	relative synodic reference frame
G_t	=	discounted return at time step t
I_{sp}	=	specific impulse
m	=	mass
q	=	action-value function
$\mathcal{R}$	=	reward space
R_t, r	=	reward at time step t
$\mathcal{S}$	=	state space
S_r	=	success rate
S_t, s	=	state at time step t
u	=	control action
V	=	candidate Lyapunov function
v	=	state-value function
$\boldsymbol{x}$	=	state vector
α	=	learning rate
β	=	approach corridor half-angle
γ	=	discount factor
$\delta\boldsymbol{x}$	=	relative state vector
ε	=	clip parameter
θ, w	=	actor and critic neural networks parameters
λ	=	GAE lambda
μ	=	mass parameter
π	=	policy function
$\boldsymbol{\rho}$	=	relative position vector between chaser and target
$\dot{\boldsymbol{\rho}}$	=	relative velocity vector between chaser and target
$\boldsymbol{\rho}^*$	=	pseudo-relative state vector
ϕ	=	generative transition function

Superscripts and Subscripts

C	=	chaser spacecraft
f	=	terminal state
T	=	target spacecraft
0	=	initial state
1	=	first primary
2	=	second primary
$*$	=	optimal

I. Introduction

Rendezvous and docking (RVD) maneuvers play a critical role in various missions in space, including crew transfer, cargo exchange, repairs, and structural assembly. For more than five decades, various countries have conducted missions of this nature [1], and various chaser spacecraft, originating with Gemini and Apollo, have successfully performed these tasks. Some contemporary examples, such as the demonstration of autonomous rendezvous technology, the automated transfer vehicle (ATV) of the European Space Agency, and the H-II Transfer Vehicle (HTV) from Japan, are now unmanned and fully automated. This progress, considerably more robust and secure than previous manual astronaut maneuvers, required the adoption of new autonomous guidance, navigation, and control (GNC) algorithms. In addition, these techniques have become indispensable for the most intricate spacecraft proximity operations, including on-orbit servicing, active space debris removal,

Received 15 April 2024; revision received 2 August 2024; accepted for publication 10 September 2024; published online 15 October 2024. Copyright © 2024 by Giovanni Fereoli. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 1533-6794 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Graduate Research Assistant, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 431 UCB, Colorado Center for Astrodynamics Research; giovanni.fereoli@colorado.edu. AAS Member. Member AIAA (Corresponding Author).

†Professor and Department Chair, Schaden Leadership Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 431 UCB, Colorado Center for Astrodynamics Research. AAS Fellow. Fellow AIAA.

‡Associate Professor, Department of Aerospace Science and Technology, Via Privata Giuseppe La Masa 34. AAS Member. Member AIAA.

and other missions. This type of problem can be formulated as an optimal control problem (OCP) and solved with active set or interior point techniques after transcription [2]. However, these methods are open-loop, unable to manage unpredictable circumstances without a controller, and too computationally expensive to be executed on-board. Autonomous guidance and control (G&C) algorithms require two fundamental features: robustness, which should be achieved by satisfying numerous requirements, such as closed-loop control, for mission safety; and computational efficiency for on-board implementation. Today, to acquire this capability, the high-level task is usually transformed into a precalculated reference trajectory (guidance) and then tracked by a controller (control). Several effective algorithms suitable for implementation on-board have been proposed, as documented by [3]. However, additional criteria can be introduced. First, to handle more intricate dynamical environments where engineering modeling would be impractical, these algorithms should be model-free. Second, they should incorporate an integrated G&C law, eliminating the need to break down the high-level task into separate processes for reference computation and tracking.

Reinforcement learning (RL) possesses the aforementioned characteristics [4]. RL agents have the capability to undergo training using high-fidelity simulators. This training allows them to learn the optimal closed-loop policy, rather than being explicitly designed. When the policy is tested on-board, it only needs minimal computational resources. Additionally, RL can directly optimize the task-level objective and use domain randomization to handle model uncertainty. This allows the identification of control responses that are more empirically reliable [5]. Deep-reinforcement learning (Deep-RL) has demonstrated its effectiveness in cutting-edge applications, including robotics [6], automotive [7], and aerospace [8]. Employing agents equipped with artificial neural networks (ANNs), Deep-RL is adept at addressing sequential decision-making problems, known as Markov decision processes (MDPs), through interactive engagement with the environment. Neural networks excel in accurately estimating functions, which makes them well suited to learn optimal closed-loop G&C policies by estimating the solution to the Hamilton–Jacobi–Bellman equations [9]. The development of GNC autonomous systems presents a major challenge in terms of robustness to uncertain spacecraft models and environments. RL deep agents, such as feedforward ANNs, are usually successful in learning within the training distributions, but often have difficulty extrapolating beyond them (i.e., low generalization capability). This can lead to an unstable guidance law when the spacecraft encounters states that are not part of the training distribution. Moreover, traditional RL requires a large amount of experience to even learn basic tasks (i.e., sample inefficiency). In recent years, numerous papers in the aerospace literature have explored spacecraft GNC for RVD scenarios using Deep-RL methods. For instance, [10] examined RL for guidance purposes only, necessitating a separate controller design. They demonstrated its effectiveness in autonomously handling docking in the *xy* plane with a close-proximity spinning target in low-earth orbit, without considering control cost in the reward function. Oestreich et al. [11] conducted a similar study, exploring RL for both guidance and control tasks in a 6-degree-of-freedom (DOF) scenario over a few tens of meters for rendezvous, while also attempting to minimize fuel consumption. However, this study based its reward engineering on a predesigned LQR controller. Scorsoglio et al. [12] developed an adaptive ZEM/ZEV algorithm using RL in cislunar space with nonlinear equations, demonstrating the method's effectiveness with various path constraints, from simple cone constraints to obstacle avoidance. Federici et al. [13] investigated RL for an *xy*-plane rendezvous scenario using Clohessy–Wiltshire (CW) equations and a visibility cone constraint. To date, no research has addressed the use of RL for spacecraft proximity operations integrating G&C that simultaneously considers fuel efficiency, three-dimensional motion in complex nonlinear dynamics, and safety requirements, without relying on classical optimal control theory or parametric guidance laws.

Recent advances in Meta-RL have effectively addressed the two aforementioned main weaknesses of RL. Meta-RL is a machine learning (ML) technique in which an agent is taught a range of tasks (that is, randomized environment parameters) instead of just one, allowing it to create a meta-policy that can quickly adjust to new and unseen tasks with minimal experience [14]. This transfer learning ability has been referred to as *learn to learn* by [15]. There are numerous approaches to implementing these characteristics in practice, such as model-agnostic meta-learning, task-agnostic meta-learning, REPTILE, Meta-SGD, and many more [16]. One of the most popular approaches, as proposed by [17], is to employ long short-term memory (LSTM) networks within a RL agent trained through gradient-based methods, referred to as meta-recurrent neural networks (Meta-RNNs). In the aerospace literature, Meta-RNN has been more successful than traditional RL in handling unmodeled dynamics and actuator failures [18], multitask scenarios with uncertain initial conditions [19], and partially observable dynamics [20]. Moreover, Meta-RNN has shown effectiveness in various spacecraft GNC applications, integrating not only guidance and control tasks but also navigation, where the policy directly maps observations to control actions. These applications include planetary landing [21,22], underactuated CubeSats [23], trajectory design [19], rendezvous missions [24], and asteroid proximity operations [25–27]. All of these studies employ the same actor–critic gradient-based training algorithm, proximal policy optimization (PPO), which has proven highly effective for continuous control applications and is currently state-of-the-art [28]. The strength of Meta-RNN lies in the hidden states of LSTM networks, which act as internal memory for storing temporal data. This allows LSTMs to maintain internal dynamics that is continuously updated by new observations along the trajectory. This capability improves learning in high-uncertainty environments during training, as the memory enhances sample efficiency by facilitating the transfer of learning between training episodes. It also enables an adaptive policy during testing. Consequently, the resulting policy demonstrates greater overall robustness from a spacecraft operations perspective, especially after on-board deployment. Unlike multilayer perceptrons (MLPs), which are fundamental units in typical feedforward ANNs, LSTMs can dynamically modify their hidden state and adjust in real time to capture unmodeled data along the controlled trajectory, depending on the observations from the environment. This capability allows LSTMs to account for external factors not considered during training while testing. This makes them ideal for scenarios involving complex, time-varying dynamics, uncertain models, and the possibility of failure. The hidden state can store information and learn during training how to adjust the internal dynamics of the LSTMs, generating the most suitable output for the task-level goal during testing.

This study will address, through the proposed Meta-RL solution, the challenge of autonomous relative G&C in cislunar space. This is especially pertinent in the context of NASA's ARTEMIS project, which intends to establish a Lunar Gateway, a cislunar space station, in the next decade [29]. Positioned at the Southern 9:2 Resonant L_2 near-rectilinear halo orbit (NRHO) of the Earth–Moon system, the station's construction and operations necessitate rendezvous and docking missions. Autonomous rendezvous, proximity operations, and docking capabilities in cislunar space have the potential to significantly improve the deployment and functionality of the Lunar Gateway. The dynamics within non-Keplerian environments, characterized by high nonlinearity and chaoticity, still demand a thorough examination [30]. Traditional protocols employed in the Apollo and Shuttle programs and automated ISS operations of ATV/HTV, which were originally designed for central gravitational fields, prove inadequate for the complexity of cislunar space. Consequently, to overcome these challenges, there is a pressing need for the design of innovative RVD algorithms and procedures. Recent research [30,31] has partially addressed this need by formalizing the various safety constraints that should be applied in this scenario (e.g., keep-out sphere, approach corridors, etc.). Researchers also analyzed the effectiveness of different equations of motion, highlighting that nonlinear equations of motion are necessary for designing accurate G&C solutions. The works of [32,33] introduced a combination of stable and unstable manifold exploitation with impulsive control for passive safety during far-range rendezvous. In addition, they used traditional G&C algorithms, such as

PID, SDRE, etc., with continuous control for active safety in close-range rendezvous and docking procedures. Despite these contributions, there is still a notable scarcity of research on GNC algorithms specifically designed for relative multibody dynamics. This work aims to address this gap as well.

In summary, this study explores the potential of using a specific Meta-RL method, known as Meta-RNN, as an autonomous spacecraft G&C algorithm to manage relative motion within the chaotic cislunar dynamical environment. In this context, comprehensive analysis from a controlled motion perspective is lacking, and previous methods may either falter or become computationally intensive. The study evaluates whether this Meta-RL algorithm meets safety requirements, such as terminal phase accuracy and approach corridor collision avoidance, while maintaining robustness and computational efficiency for on-board implementation. Additionally, it investigates the benefits of using LSTM networks over basic MLP networks in RL agents for spacecraft G&C applications, thereby assessing the advantages of adaptive policies in environments with high parameter randomization. This paper aims to build on the existing Meta-RL literature for spacecraft GNC by presenting new results, such as a new stability analysis, enhanced robustness evaluation of the policy considering failures and significant parameter uncertainty, and a comparison with classical RL based on feedforward networks. These are among the most interesting and innovative features of this paper and warrant thorough discussion.

II. Cislunar Space Relative Dynamics

This section presents a summary of the equations of motion and the reference frame employed in this study to model the relative motion between a target and a chaser spacecraft in the cislunar space environment.

A. Relative Circular Restricted Three-Body Problem

The motion of a massless particle influenced by two massive bodies is described by the restricted three-body problem (R3BP). This model considers the mass of the particle m_B and the masses of massive bodies m_1 and m_2, assuming $m_B \ll m_1, m_2$ and $m_1 > m_2$. The problem is simplified by representing the equations in a rotating reference frame, also known as the synodic reference frame, which is described in the following section for clarity. Additionally, assuming circular motion of the primary bodies further simplifies the problem, leading to the circular restricted three-body problem (CR3BP) [34]. This problem is typically studied in its normalized form; for more details on the scaling parameters, refer to [35]. The CR3BP is an autonomous system of equations, where the only parameter defining the three-body system is the mass parameter μ. In the Earth–Moon case, this parameter is denoted as $\mu = 0.012150584269542$.

The equations of motion for the relative circular restricted three-body problem (RCR3BP) [12] are obtained by subtracting the absolute CR3BP equations of motion of the chaser and the target, resulting in $\delta\boldsymbol{x} = \boldsymbol{x}^C - \boldsymbol{x}^T$. In the context of the relative synodic Earth–Moon frame (refer to Fig. 1), these nonautonomous equations are presented in Eq. (3). Here, $\boldsymbol{\rho}$ represents the vector $[\delta x, \delta y, \delta z]$, whereas $\boldsymbol{r}_{1T}$ and $\boldsymbol{r}_{2T}$ are defined as $[x^T + \mu, y^T, z^T]$ and $[x^T + \mu - 1, y^T, z^T]$, respectively. The RCR3BP is also known as the nonlinear relative (NLR) equations in the relative synodic reference frame. However, it is referred to as RCR3BP here to avoid ambiguity, given the multiple NLR formulations. Various types of relative equations of motion for NRHOs have been explored in the literature. Lizy-Destrez et al. [30] explored the CW equations and linearized relative equations, in addition to the NLR equations in the relative synodic reference frame. Meanwhile, [36] proposed a set of NLR equations in the local-vertical/local-horizon (LVLH) frame, necessitating the computation of the target's angular velocity and acceleration vectors during its orbit around the Moon. However, the RCR3BP equations stand out for their generalization and simplicity:

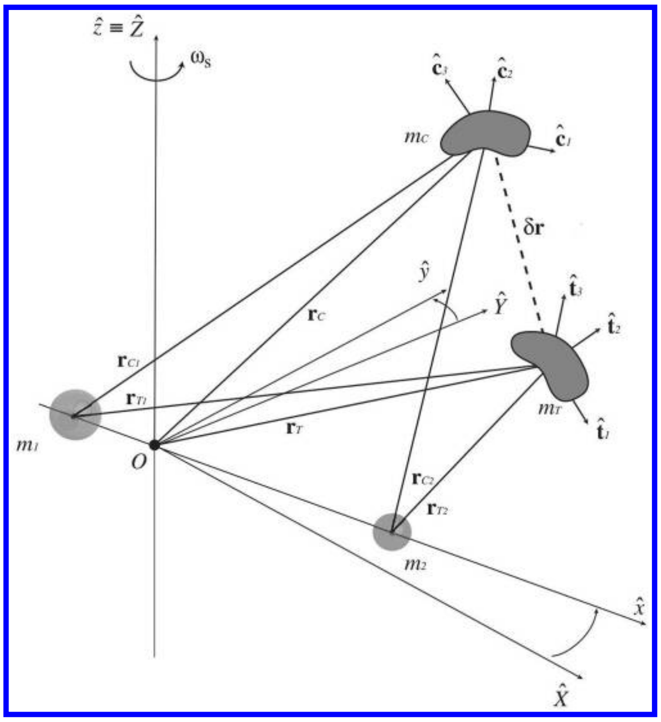


Fig. 1 Synodic reference frame centered on the barycenter of the primary bodies in the CR3BP. Source: [37].

$$
\begin{aligned}
\delta\ddot{x} &= 2\delta\dot{y} + \delta x + (1-\mu)\left[\frac{x^T+\mu}{\|\boldsymbol{r}_{1T}\|^3} - \frac{x^T+\delta x+\mu}{\|\boldsymbol{r}_{1T}+\boldsymbol{\rho}\|^3}\right] \\
&\quad + \mu\left[\frac{x^T-\mu-1}{\|\boldsymbol{r}_{2T}\|^3} - \frac{x^T+\delta x+\mu-1}{\|\boldsymbol{r}_{2T}+\boldsymbol{\rho}\|^3}\right] \\
\delta\ddot{y} &= -2\delta\dot{x} + \delta y + (1-\mu)\left[\frac{y^T}{\|\boldsymbol{r}_{1T}\|^3} - \frac{y^T+\delta y}{\|\boldsymbol{r}_{1T}+\boldsymbol{\rho}\|^3}\right] \\
&\quad + \mu\left[\frac{y^T}{\|\boldsymbol{r}_{2T}\|^3} - \frac{y^T+\delta y}{\|\boldsymbol{r}_{2T}+\boldsymbol{\rho}\|^3}\right] \\
\delta\ddot{z} &= (1-\mu)\left[\frac{z^T}{\|\boldsymbol{r}_{1T}\|^3} - \frac{z^T+\delta z}{\|\boldsymbol{r}_{1T}+\boldsymbol{\rho}\|^3}\right] + \mu\left[\frac{z^T}{\|\boldsymbol{r}_{2T}\|^3} - \frac{z^T+\delta z}{\|\boldsymbol{r}_{2T}+\boldsymbol{\rho}\|^3}\right]
\end{aligned} \tag{1}
$$

B. Relative Synodic Reference Frame

The CR3BP can be described using a rotating reference frame, also known as the synodic reference frame, that is centered on the barycenter of two massive bodies. This frame rotates with a constant angular velocity ω_s and has a right-handed coordinate system with the $\hat{\boldsymbol{x}}$ axis pointing from the larger body to the smaller one, the $\hat{z}$ axis aligned with the system's angular momentum, and the $\hat{\boldsymbol{y}}$ axis completing the right-handed coordinate system. As seen in Fig. 1 (from [37]), this establishes $\mathcal{C}_O : \{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$.

The relative motion between a target and a chaser spacecraft in the CR3BP can be expressed in this frame, with masses m_T and m_C. The assumed origin of the three axes, taking into account $\delta\boldsymbol{x}$, is now positioned at the opening of the target's docking port. This yields $\mathcal{C}_T : \{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$. The target's body-frame is denoted as $\mathcal{B}_T : \{\hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2, \hat{\boldsymbol{t}}_3\}$, and, similarly, the chaser's body-frame is represented by $\mathcal{B}_C : \{\hat{\boldsymbol{c}}_1, \hat{\boldsymbol{c}}_2, \hat{\boldsymbol{c}}_3\}$.

III. Reinforcement Learning

This section reviews RL with a focus on the MDP as the modeling framework. The discussion then highlights PPO as a key gradient-based RL technique.

A. Markov Decision Process

Reinforcement learning (RL) is a form of learning that associates actions with situations in order to maximize a reward signal. It is formulated on the basis of the principles of dynamical systems, particularly using MDPs to model sequential decision-making. An agent interacts with an environment over discrete time steps ($t = 0, 1, 2, 3, \ldots$). The agent receives a state representation $S_t \in \mathcal{S}$, selects an action $A_t \in \mathcal{A}(s)$, and obtains a numerical reward $R_{t+1} \in \mathcal{R}$. This interaction creates a trajectory starting with S_0, A_0, R_1, S_1, A_1, R_2, etc. (Fig. 2, from [9]).

To employ this modeling framework, the foundational assumptions encompass the complete observability of the state, finite sets of states $\mathcal{S}$ and actions $\mathcal{A}$, and rewards $\mathcal{R}$, along with the Markov property. The latter asserts that the probability of each possible value for S_t and R_t is only influenced by S_{t-1} and R_{t-1}, and not by any previous states and actions. The MDP environment is described by a discrete probability distribution $p: \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, which determines the dynamics of the system as

$$p(s', r|s, a) \doteq Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a) \tag{2}$$

For all $s' \in \mathcal{S}$, $s \in \mathcal{S}$, $r \in \mathcal{R}$, and $a \in \mathcal{A}(s)$. In this paper, $A|B$ is used to denote the conditional dependence of event A given event B, so $P(A|B)$ represents a conditional probability. This notation is essential in MDPs as it captures the probabilistic dependency of any required random variable, such as future states and rewards, on current states and actions. The agent's objective is to maximize the long-term cumulative reward rather than the immediate rewards $R_t \in \mathbb{R}$. Typically, the interaction between an agent and its environment is divided into sequences known as episodes. At the conclusion of each episode, a specific state called the terminal state is reached, which signifies the completion or noncompletion of the task. Following this, the system is restored to either a default initial state or a randomly chosen initial state from a standard distribution. To avoid excessive cumulative rewards and promote convergence, the agent usually employs the idea of discounting to determine which actions will maximize the total benefits it will receive in the future. Specifically, it selects the action A_t to maximize the discounted return, represented as G_t in Eq. (5):

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = R_{t+1} + \gamma G_{t+1} \\ &= \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \end{aligned} \tag{3}$$

where $\gamma \in [0, 1]$ is the discount rate. The discount rate determines the current value of a future reward. A policy π can be formally expressed as a mapping of states to the likelihood of selecting each potential action; specifically, $\pi(a|s)$ is the probability that $A_t = a$ when $S_t = s$. The expected discounted return when starting in a state $s \in \mathcal{S}$ and following policy π is known as the state-value function v_π. This is defined as follows:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] \tag{4}$$

The expected value of a random variable is denoted as $\mathbb{E}_\pi[\cdot]$ when the agent follows the policy π. The expected discounted return when beginning in a state $s \in \mathcal{S}$ and taking an action $a \in \mathcal{A}(s)$ according to policy π is denoted by the action-value function q_π:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{5}$$

The primary objective in reinforcement learning is to find a policy that maximizes the expected discounted return. A policy π is considered better than or equal to another policy π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all states $s \in \mathcal{S}$. The most favorable policy is referred to as the optimal policy denoted by π_*. Its optimal state-value function v_* is defined as

$$v_*(s) = \max_\pi v_\pi(s) \tag{6}$$

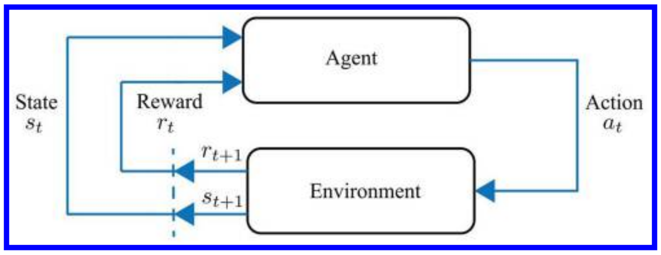


Fig. 2 Agent–environment interaction at each discrete time step in a Markov decision process (MDP). Source: [9].

B. Proximal Policy Optimization

Policy gradient methods, including PPO, are a type of approximated method solution that are commonly used to solve RL problems in large state spaces. These methods are preferred because they offer greater stability and the assurance of convergence [4]. They achieve this by using parameterized functions, such as ANNs, to approximate functions, which makes them well suited for dealing with high-dimensional, continuous-state, and continuous-action spaces. Unlike traditional value-based methods, such as Q-learning, policy gradient methods calculate the policy directly without requiring a model. However, they are generally less data-efficient. An actor–critic framework, which PPO also uses, can be applied to solve this issue. This approach helps reduce the variance in the estimated policy gradient, making it suitable for online applications. These methods acquire a parameterized policy[§] and make action selections without the need to compute a value function. The parameter vector is expressed as $\boldsymbol{\theta} \in \mathbb{R}^d$ and the policy as $\pi_{\boldsymbol{\theta}}(a|s)$. The algorithms to solve the policy parameters are based on the gradient of a scalar performance measure $J(\boldsymbol{\theta})$. They are designed to maximize performance, so the update of the parameters in the training step k follows the gradient ascent of J:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \widehat{\nabla J(\boldsymbol{\theta}_k)} \tag{7}$$

where $\widehat{\nabla J(\boldsymbol{\theta}_k)}$ is an estimate of the gradient of the performance metric with respect to the current policy parameters $\boldsymbol{\theta}_k$, and the step size is determined by the learning rate α. PPO is renowned as a state-of-the-art algorithm for continuous control in reinforcement learning [28]. It is a first-order approximation of the thrust region policy optimization (TRPO) method. As an actor–critic method, PPO introduces bias through a bootstrapping critic and uses a learned weight vector $\boldsymbol{w} \in \mathbb{R}^m$ to estimate the state-value function $\hat{v}_{\boldsymbol{w}}$. In PPO, a clipped surrogate objective function is used to produce a pessimistic and conservative approximation of policy performance. The corresponding figure of merit is displayed subsequently:

$$\max_{\boldsymbol{\theta}} \hat{\mathbb{E}}_\pi[\min(r_t(\boldsymbol{\theta})\hat{A}_t, \operatorname{clip}(r_t(\boldsymbol{\theta}), 1 - \varepsilon, 1 + \varepsilon)\hat{A}_t)] \tag{8}$$

The probability ratio r_t is expressed as

$$r_t(\boldsymbol{\theta}) = \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta,\text{old}}(A_t|S_t)} \tag{9}$$

The advantage function, denoted as $A_t(s, a) = q_\pi(S_t, A_t) - v_{\boldsymbol{w}}(S_t)$, serves as an indicator of the additional reward that the agent can take when returning a specific action from a given state. In practical PPO implementations, it is crucial to estimate both the advantage function $\hat{A}_t$ (dependent on the unknown environment) and the expectation operator $\hat{\mathbb{E}}_\pi$.

[§]The policy $\pi_{\boldsymbol{\theta}}(a|s)$ is typically designed to be stochastic for exploration purposes. In the context of continuous action spaces, the policy can be formulated as a normal Probability Density Function (PDF). As learning progresses, the focus shifts from exploration to exploitation, resulting in the standard deviation of the PDF converging to zero.

IV. Meta-Reinforcement Learning

Meta-Reinforcement Learning (Meta-RL), as implemented in the Meta-Recurrent Network (Meta-RNN) [17], integrates LSTM [38] into the RL agent. LSTM networks, a type of RNN, use gates to prevent gradients from vanishing or becoming too large, enabling selective updating and control of information flow. For detailed information on LSTM structure and mathematics, refer to [38]. Unlike MLPs, all RNNs, including LSTMs, have feedback loops in their layer interactions, forming directed graphs.

The LSTM-agent undergoes training using gradient-based techniques such as PPO, making it well-suited for models with numerous free parameters and enhancing the framework's simplicity compared to other meta-learning algorithms. The Meta-RNN approach excels in handling highly nonstationary time-series and has demonstrated its applicability to learning and autonomous systems.

A. Training Algorithm

The PPO algorithm, which is a model-free method, has been previously discussed, and it has been noted that it requires the estimation of the advantage function in order to operate. To address this, the generalized advantage estimate (GAE) [39] is used and can be adjusted using the coefficient λ to strike a balance between bias and variance. The GAE formula, as shown in Eq. (12), is truncated at the end of each episode ($t = T$) to accommodate LSTM-based networks [40]:

$$\hat{A}_t^{\mathrm{GAE}} = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}^v \tag{10}$$

The definition of the residual temporal-difference [41] for the discounted value function can be expressed as $\delta_t^v = R_t + \gamma v_{\boldsymbol{\pi}}(S_{t+1}) - v_{\boldsymbol{\pi}}(S_t)$. To estimate $\hat{A}_t^{\mathrm{GAE}}$, it is necessary to estimate the state-value function. When a nonlinear function approximator is used to represent the state-value function, a commonly employed approach involves solving a nonlinear regression problem by minimizing the mean square error (MSE) in the following manner:

$$J^{\mathrm{MSE}}(\boldsymbol{w}) = \hat{\mathbb{E}}_\pi\left[\left(v_{\boldsymbol{w}}(S_t) - \sum_{k=t}^{T} \gamma^{k-t} R_t\right)^2\right] \tag{11}$$

Batch learning is a machine learning technique used to estimate expectancy operators. It works by generating a batch of trajectories (or roll-outs) before each training cycle, with the number of trajectories controlled by the hyperparameter called batch size. After processing the entire training dataset, the model is updated in a single iteration, adjusting all parameters at once. For a more detailed explanation of the algorithm's architecture, see Fig. 3 (from [4]).

Gradient ascent is subsequently applied to both unrolled LSTM-based actor and critic networks, and backpropagation through time (BPTT) is used to compute the gradients of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$. In each epoch k of the gradient-based optimizer, such as the adaptive moment (ADAM) estimation method [42], the update equations [similar to Eq. (9)] are implemented as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \beta_\theta \widehat{\nabla J(\boldsymbol{\theta}_k)} \tag{12}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \beta_w \widehat{\nabla J(\boldsymbol{w}_k)} \tag{13}$$

The difference in the sign of the update arises because J^{PPO} is maximized when optimizing for $\boldsymbol{\theta}$, whereas J^{MSE} is minimized when optimizing for $\boldsymbol{w}$. For each set of roll-out data, the update step is applied a certain number of times, called epochs. The algorithm stops when the maximum number of predetermined learning steps is reached. The weights in the networks are initialized using an orthogonal approach. An incorrect initialization can lead to issues, such as all layers learning the same feature or vanishing and exploding gradients. This method mitigates the issue of ill-conditioned gradients by initializing ANNs using matrices with eigenvalues having absolute unitary values.

The algorithm described previously, due to the specific version of GAE implemented and the use of BPTT, produces a version of PPO that is suitable for use with LSTM networks, called recurrent proximal policy optimization (Recurrent PPO) [43]. The algorithm is detailed in Algorithm 1. The implementation of Recurrent PPO, and PPO in general, involves a number of complex optimizations that are not discussed in depth or even mentioned by [28]. However, these optimizations have been found to have a significant effect on the efficacy of PPO [44]. To ensure robustness, consistency, and high

Algorithm 1: Recurrent PPO

```
1.  Initialization of neural network parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$
2.  for step = $1, 2, \ldots,$ learning steps do
3.    Reset environment
4.    for episode = $1, 2, \ldots,$ batch size do
5.      Run policy $\pi_{\mathrm{old}}$ in environment until done
6.      Compute advantage estimate $\hat{A}_t^{\mathrm{GAE}}$
7.      Store trajectory into batch roll-out
8.    end for
9.    for iteration = $1, 2, \ldots,$ epochs do
10.     Unroll agent LSTMs
11.     Optimize Actor $J^{\mathrm{PPO}}$ wrt $\boldsymbol{\theta}$ and Critic $J^{\mathrm{MSE}}$ wrt $\boldsymbol{w}$
12.     $\boldsymbol{\theta}_{\mathrm{old}} \leftarrow \boldsymbol{\theta}$, $\boldsymbol{w}_{\mathrm{old}} \leftarrow \boldsymbol{w}$
13.   end for
14. end for
```

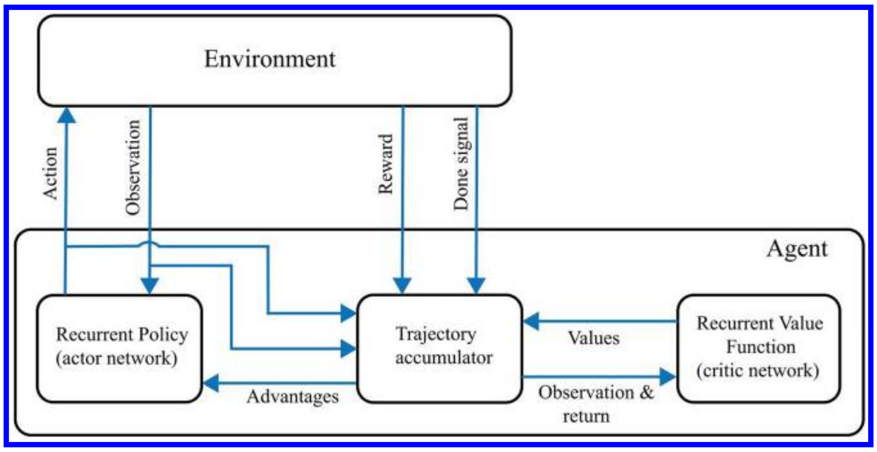
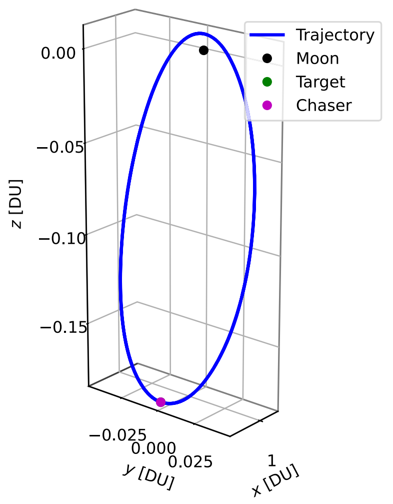
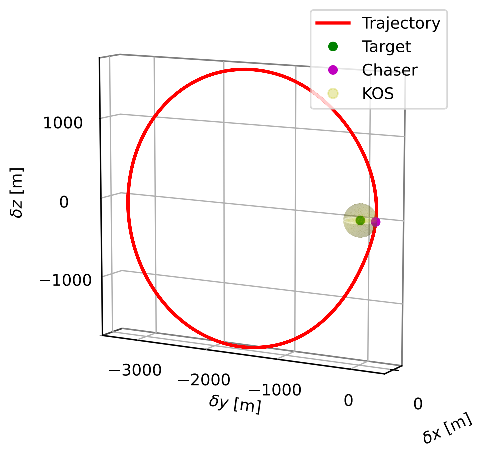


Fig. 3 Meta-reinforcement learning (Meta-RL) training architecture for recurrent proximal policy optimization (Recurrent PPO) algorithm. Source: [4].



a) Absolute synodic reference frame



b) Relative synodic reference frame

Fig. 4 Chaser and target spacecraft on the Southern 9:2 Resonant Near-Rectilinear Halo Orbit (NRHO) of the Earth–Moon system apolune at a relative distance of 200 m, keep-out-sphere (KOS) edge, during one orbital motion.

flexibility, this work uses the `Stable-Baselines3 (SB3)`[¶] library in Python, an open-source machine learning library created by the Facebook AI Research (FAIR) Laboratory.

V. Problem Formulation

This project focuses on developing an autonomous G&C algorithm for these final approaches and docking of a spacecraft in cislunar space, with emphasis on continuous control and active collision avoidance. The chaser spacecraft initiates the maneuver from a holding point at the edge of the target keep-out-sphere (KOS) with a radius of 200 m. It is also assumed that the chaser spacecraft is already situated at the apolune of the Southern L_2 9:2 Resonant NRHO. The choice of this location for the final rendezvous stages is based on its favorable slower dynamics, ensuring passive safety and fuel efficiency [30]. Figure 4 illustrates this scenario, showing the initial absolute and relative conditions of the NRHO, together with the natural motion along one orbit. These specific initial conditions can also serve as a holding point until the chaser spacecraft receives the *GO Final Approach*. Using this strategy ensures passive avoidance of collisions with the target, thanks to the relative periodic central manifold. Furthermore, in this study, the Lunar Gateway is assumed to be the target, and the chaser is the Orion spacecraft. As a result, the main characteristics of the Orion spacecraft, including $m = 21000$ kg, $u_{\max} = 29.3$ kN, and $I_{\text{sp}} = 310$ s, are incorporated. The spacecraft is modeled with 3DOF, focusing solely on the guidance and control of the center of mass. The feasibility of the control action profile should be further examined with respect to the throttling capabilities of the thrusters and analyzed in a 6DOF environment for attitude control suitability. However, these investigations are currently outside the scope of this paper. Here, it is assumed that maximum thrust is available in all directions simultaneously, achievable with six thrusters of equal capacity oriented in opposite directions.

To ensure the safety of the approach and proper docking, the RVD maneuver must meet specific requirements. Therefore, the algorithm must take into account the following list of constraints:

1) **Docking Port**: the final relative position and velocity shall be $\rho_f <$ 1 m and $\dot{\rho}_f < 0.1$ m/s, respectively;

2) **Approach Corridor**: the chaser spacecraft shall remain within a truncated cone (Fig. 5) with a half angle of $\beta = 20°$ for safety and vision-based navigation purposes. Assuming that the target docking port is located in its positive $\hat{\boldsymbol{t}}_2$ direction, the requirement for a truncated cone is expressed as follows:

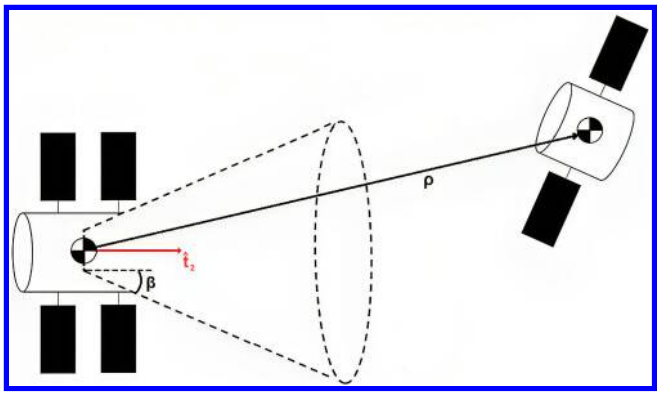


Fig. 5 The approach corridor is represented by a truncated cone with a maximum semiangle of β and a small frustum base designed to satisfy the docking port requirements.

$$\boldsymbol{\rho}^*(t)\cdot\hat{\boldsymbol{t}}_2 - \rho^*(t)\cos(\beta) > 0 \quad \forall\delta y \geq 0, \quad \forall t \tag{14}$$

$$\boldsymbol{\rho}^*(t) = \boldsymbol{\rho}(t) + \left[0, \frac{\rho_{f,\max}}{\tan\beta}, 0\right]^T \tag{15}$$

Using a truncated cone, also known as a frustum [29], provides a more realistic representation of the docking port [1]. Moreover, this model avoids singularities, which improves the convergence properties of the algorithm. The constraint involves an alternative relative position $\boldsymbol{\rho}^*$ from a point in the $-\hat{\boldsymbol{t}}_2$ direction.

The $\boldsymbol{\rho}^*$ vector is designed to meet the docking port requirements at the small frustum base, considering an angular distance from the approach axis of β. Additionally, because this requirement is defined in the target's body frame and is attitude-dependent, it is assumed that the target remains cooperative and aligned with the Earth–Moon synodic frame, with $\hat{\boldsymbol{t}}_2 \equiv \hat{\boldsymbol{y}}$. The attitude of the chaser should be adjusted in accordance with its docking port's position in its body frame.

3) **Maximum Time-Of-Flight**: the maneuver shall be accomplished within a restricted time of flight; therefore $t < ToF_{\max} = 100$ s must be satisfied;

4) **Thrusters' Performance**: the maximum thrust value specified in the data sheet shall be followed to formulate a feasible control action profile, ensuring that $u < u_{\max} = 23.9$ kN.

A. Optimal Control Problem Formulation

Taking into account the requirements mentioned previously, it is simple to design an OCP: *find the control action profile that minimizes*

[¶]Data available online at https://stable-baselines3.readthedocs.io/en/master/index.html *control effort and such that dynamics, initial conditions, maximum control action, maximum time of flight, approach corridor, and final docking port requirements are respected.* It can be expressed mathematically as follows:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & J = \int_{t_0}^{t_f} \|\boldsymbol{u}(t)\|\, \mathrm{d}t \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))\ \forall t \\ & \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \\ & u(t) < u_{\max}\ \forall t \\ & t < ToF_{\max} \\ & \boldsymbol{\rho}^*(t) \cdot \hat{\boldsymbol{t}}_2 - \rho^*(t)\cos(\beta) > 0\ \forall \delta y \geq 0,\ \forall t \\ & \rho(t_f) - \rho_{f,\max} < 0 \\ & \dot{\rho}(t_f) - \dot{\rho}_{f,\max} < 0 \end{aligned} \tag{16}$$

The system's equations of motion, represented by $\boldsymbol{f}$, encompass the absolute target state, the relative chaser state, and its associated mass. To address this OCP using Meta-RL, it is essential to transform it into an MDP, as explained in the following subsection.

B. Markov Decision Process Formulation

The agent's goal in a MDP is *find the control policy π^* that maximizes the discounted expected return of rewards received along a trajectory and such that the policy, the environment's dynamics, and the initial conditions are respected*. The MDP is mathematically expressed as follows:

$$\begin{aligned} \max_{\pi} \quad & J = \mathbb{E}_\pi[G_t | \boldsymbol{S}_t] \\ \text{s.t.} \quad & \boldsymbol{A}_t = \boldsymbol{\pi}(\boldsymbol{S}_t) \\ & \boldsymbol{S}_{t+1} = \boldsymbol{\phi}(\boldsymbol{S}_t, \boldsymbol{A}_t) \\ & \boldsymbol{S}_0 \sim \mathcal{N}(\hat{\boldsymbol{S}}_0, \boldsymbol{C}) \end{aligned} \tag{17}$$

The second constraint considers the dynamics of the MDP environment, resulting in the update of its state $\boldsymbol{S}_t$ at each time step t. In contrast to the straightforward OCP formulation, the MDP can leverage domain randomization to improve the robustness of the algorithm [5]. In this scenario, the dynamics of the spacecraft will be modified to include random processes and stochastic effects. Additionally, uncertainty will be introduced in the initial state by sampling it from a normal distribution, just as in the final restriction of Eq. (19). Moreover, it is crucial to note that operational constraints will be *translated* through the definition of the reward function. MDPs face challenges in handling constraints compared to OCPs. In MDPs, constraints should be represented as significant positive or negative rewards or as the conclusion of an episode.

1. State and Action Spaces

The agent's neural networks use the state space as input to generate the action (actor) or predict the state-value function (critic). In spacecraft RVD problems, it is advantageous to incorporate both the absolute target and relative chaser states. The inclusion of the chaser's mass becomes necessary for optimizing fuel consumption in the reward function. To learn to comply with time constraints, the remaining flight time $\Delta T_t = ToF_{\max} - t$ is also taken into account. Moreover, observations include the action and reward from the previous time step, enabling LSTMs to update their dynamics [17]. Consequently, the state space $\boldsymbol{S} \in \mathbb{R}^{18}$ encompasses

$$\boldsymbol{S}_t = \left[\boldsymbol{x}_t^T, \delta\boldsymbol{x}_t, m_t, \Delta T_t, \boldsymbol{u}_{t-1}, R_{t-1}\right] \tag{18}$$

For actual deployment, using a state space of this kind means assuming, in particular, that the chaser knows its relative position to the target and the target's absolute position. This is a reasonable assumption because the chaser would perform relative navigation on-board while approaching the target and can obtain the target's absolute position through its absolute navigation or via an intersatellite link.

Normalizing the inputs of neural networks is essential to bring all input features to a comparable scale and improve numerical stability. Therefore, a simple min–max normalization is applied. The state is normalized to be within $\boldsymbol{S}_t^* \in [-1, +1]$ using the following method:

$$S_{t,i}^* = 2\left(\frac{S_{t,i} - S_{\min,i}}{S_{\max,i} - S_{\min,i}}\right) - 1 \tag{19}$$

Once the problem is defined, the extremal components of $\boldsymbol{S}_t$ are identified. The actor network generates a set of all possible actions that the agent can perform in the environment, termed the action space. In this specific study, which focuses on a 3DOF spacecraft model, the action space is intended to depict thrust control. The suitable representation for this research case [45] includes an action space $\boldsymbol{A}_t \in \mathbb{R}^3$ and its unscaled control action $\boldsymbol{u}_t$, defined as

$$\boldsymbol{A}_t = [\tilde{u}_{x,t}, \tilde{u}_{y,t}, \tilde{u}_{z,t}] \quad \boldsymbol{u}_t = \sigma\tilde{\boldsymbol{u}}_t \tag{20}$$

The control action $\boldsymbol{u}_t$ serves as input to the spacecraft equations of motion. To maintain adherence to the maximum thrust value, the scaling coefficient is $\sigma = u_{\max}/\|\boldsymbol{1}_3\|$, because the output of the ANN is limited to the range of $[-1, 1]$.

2. Reward Function

The reward signal's purpose is to convey the desired outcome to the agent, rather than specifying how it should be achieved. In scenarios where rewards are inherently sparse, such as the one described, RL tends to exhibit subpar performance. To improve them, it is recommended to employ reward shaping techniques [46]. In such cases, one can make use of nonlinear functions, such as logarithmic and exponential functions, leveraging their characteristic of increasing the first derivative near attractive or repulsive states. This contributes to a smoother appearance of the reward landscape. Additionally, squaring is used to enhance convergence performance [47]. The specific reward function employed is described** subsequently:

$$\begin{aligned} R_t = {} & \alpha\left[\ln\left(\frac{\|\delta\boldsymbol{x}_t \oslash \delta\boldsymbol{x}_{\max}\|}{\|\boldsymbol{1}_6\|}\right)\right]^2 - \lambda\left[\exp\left(\frac{\arccos(\hat{\boldsymbol{\rho}}_t^* \cdot \hat{\boldsymbol{t}}_2)}{\pi}\right)\right]^2 \\ & - \gamma\left[\exp\left(\frac{u_t}{u_{\max}}\right)\right]^2 + (\rho_t < \rho_{f,\max} \wedge \dot{\rho}_t < \dot{\rho}_{f,\max} \Rightarrow)\zeta \\ & - (\boldsymbol{\rho}_t^* \cdot \hat{\boldsymbol{t}}_2 - \rho_t^* \cos(\beta) < 0 \Rightarrow)\kappa \end{aligned} \tag{21}$$

The reward function consists of two main parts:

1) The dense rewards provide ongoing feedback within each episode, featuring a primary bonus component for approaching the goal state $\delta\boldsymbol{x} = \boldsymbol{0}$, a penalty for the distance from the approach corridor as the second component, and a penalty for the control effort as the third. All inputs to these nonlinear functions are normalized to be within [0, 1]. Particular attention has been paid during reward engineering to the first logarithmic component. The incorporation of the Hadamard operator, L_2-norm, and division by $\|\boldsymbol{1}_6\|$ aims to generate a scalar value that ensures equal importance for each state component and offers no reward for the greatest distances and high velocities.

2) Episodic rewards deliver feedback at the conclusion of a task, featuring a primary bonus component when the docking port requirements are satisfied and a secondary penalty component when a collision with the approach corridor occurs. These components also signal the completion of the episode to the environment.

The reward shaping logic, using logarithmic and exponential functions (with numerical values provided solely for demonstration), is summarized and illustrated in Fig. 6. Through an iterative process, the values used in this work are determined to be $\alpha = 0.02$, $\lambda = 0.1$, $\gamma = 0.01$,

**In the reward function, the right arrow ($\Rightarrow$) denotes implication. For example, condition $\Rightarrow$ reward means "if condition is satisfied, then this is the reward."

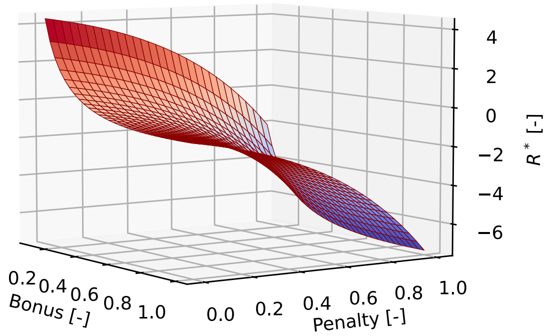


Fig. 6 Graphic illustration of the reward shaping employing logarithmic and exponential functions. The bonus and penalty axes represent their input arguments.

$\zeta = 100$, and $\kappa = 30$. For future work, an extensive hyperparameter search should be conducted, potentially using state-of-the-art methods such as Bayesian optimization.

3. Environment's Dynamics

In the realm of an RVD problem, given its continuous nature, it is more appropriate to represent the transition function using a generative model, expressed as $\boldsymbol{S}_{t+1} = \phi(\boldsymbol{S}_t, \boldsymbol{A}_t)$. This model facilitates updates for both the absolute target and the relative chaser state by integrating the CR3BP and the RCR3BP. The equations of motion, denoted by $\boldsymbol{f}$, undergo discretization into discrete steps through integration, taking into account a constant control action. Additionally, the model includes updates for the residual flight time, as well as the action and reward from the preceding time step. At the beginning of each MDP episode, a random initial state $\boldsymbol{S}_0$ is selected. This randomness specifically influences the initial conditions of the relative chaser dynamics $\delta\boldsymbol{x}_0 = [\boldsymbol{\rho}, \dot{\boldsymbol{\rho}}]$ and its initial mass m_0, derived from three as broad normal distributions as possible. Standard deviations are configured as follows: $\sigma_{\rho_0} = 0.1\rho_0$, $\sigma_{\dot{\rho}_0} = 0.5$ m/s, and $\sigma_{m_0} = 100$ kg. Furthermore, the environment sends out a done when the flight time exceeds its highest value ($ToF_{\max}$). Process noise is included in the chaser spacecraft dynamics to account for unmodeled accelerations (e.g., SRP, the gravity of other celestial bodies, thrust uncertainty, etc.). Hence, the equations of motion for the chaser, taking into account $\boldsymbol{h}$ as the nonautonomous RCR3BP, can be expressed as

$$\delta\dot{\boldsymbol{x}}(t) = \boldsymbol{h}(\delta\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{x}^T(t)) + \boldsymbol{w}(t) \quad \boldsymbol{w}(t) \sim \mathcal{N}(\boldsymbol{0}, \operatorname{diag}(\boldsymbol{0}_3, \boldsymbol{10}_3^{-8})) \tag{22}$$

$\mathcal{N}$ represents a Gaussian white noise, featuring a covariance that matches the nondimensional magnitude ($\sigma = 10^{-4}$) of typical NRHO disturbances. The robustness of the algorithm is further enhanced by taking into account and modeling the possibility of a random failure in the control action. At the beginning of each episode, the environment randomly selects a control direction in which to reduce the magnitude of the thrust by 50% or have no failure. Each scenario has an equal probability of 25%, as illustrated in Table 1.

C. Artificial Neural Networks Architecture and Hyperparameters

The architecture of the model is inspired by `Stable-Baseline3 (SB3) Recurrent PPO`, featuring layers of LSTM followed by layers of MLP extractor, as shown in the Table 2. The use of a combination of LSTM networks and MLP extractor layers is a common approach in RL [48]. This is because LSTMs are particularly adept at capturing sequential dependencies, but may not be able to provide a suitable final representation. An MLP-layer at the end can be used to condense the variable-length sequence of LSTM outputs into a fixed-size representation, making it easier to make decisions or take actions. This combination of sequence modeling (handled by LSTMs) and representation learning (handled by MLPs) allows the model to effectively use sequential information in RL tasks. In this study, the size of the neural network is determined through experimentation to guarantee a high level of generalization and to accurately fit the optimal solution. The hyperparameters used in this study are described in Table 3. These selected hyperparameters closely align with the default settings [9] and required only minor adjustments determined by trial and error. It is crucial to note that gradient clipping is a significant hyperparameter for LSTM networks, given their susceptibility to catastrophic forgetting [49].

Table 1 Multinomial discrete distribution modeling a breakdown in 6DOF control

Failure type	$0.5 \cdot u_x$	$0.5 \cdot u_y$	$0.5 \cdot u_z$	No failure
Probability	0.25	0.25	0.25	0.25

Table 2 Architecture of the artificial neural network implemented in the actor (A) and critic (C) networks of the agent

Layer	Neurons (A/C)	Activation	Type
Hidden 1	256/256	Tanh	LSTM
Hidden 2	256/256	Tanh	LSTM
Hidden 3	64/64	Tanh	MLP
Hidden 4	64/64	Tanh	MLP

Table 3 Hyperparameters for actor's training using proximal policy optimization and critic's training through MSE algorithms

	Actor PPO	Critic MSE
Batch size	64	64
Number epochs	10	10
GAE lambda (λ)	1	
Discount factor (γ)	0.99	
Clip parameter (ε)	0.1	
Learning rate (α)	0.00005	0.00005
Entropy coefficient	0.0001	
Clip gradient	0.1	0.1

VI. Numerical Results

RL encompasses two primary phases: training and testing. In the context of spacecraft G&C applications, ground-based high-fidelity simulators should be used for training, with testing of the policy occurring directly on-board during operations. The equations of motion are solved in the MDP environment with the `scipy.integrate` library in `Python`, using the nonstiff Adam predictor-corrector, called LSODA, method with relative and absolute tolerances of $2.22 \cdot 10^{-14}$. The integration time step to convert continuous dynamics to an MDP is set to $dt = 0.5$ s. The device used for this work is a Laptop PC with an Intel(R) Core(TM) i7-8565U CPU 1.99 GHz and 16.0 GB of RAM.

A. Training

The algorithm is trained for 7 million time steps, which corresponds to 40.5 days in the simulated environment. This training duration spans 3.2 days in real time, and the resulting training curve, which illustrates the mean discounted cumulative reward of trajectory roll-outs in relation to the learning step, is shown in Fig. 7. A standard PPO is also used to train a simple fully connected MLP-agent, serving as a benchmark for the LSTM-agent in Meta-RL performance. The MLP-agent in this comparison shares the same environmental setup, network architecture (number of layers and layer widths), and hyperparameters as the LSTM-agent. This ensures a fair comparison. Having the same architecture also means that both models have the same number of cells overall. Although LSTMs have more parameters due to their different internal cell architecture, comparing models with the same number of cells is more meaningful.

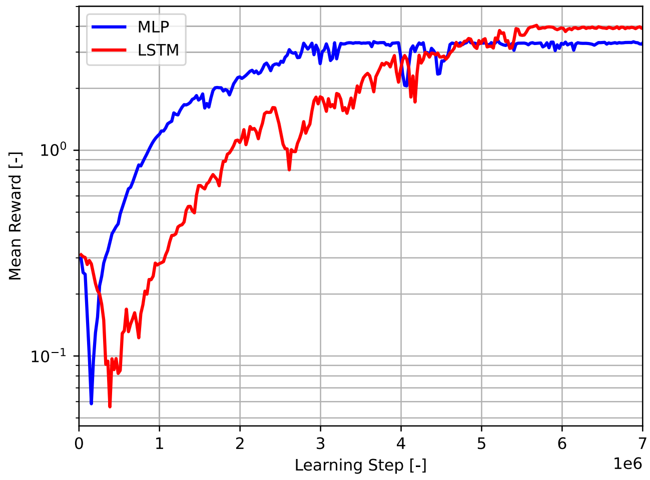


Fig. 7 Mean discounted cumulative rewards of the trajectory roll-outs during the training phases.

The Meta-RL capability of LSTM-agents is attributed to the internal structure of their cells (composed of gates, closed loops, etc.) rather than the number of parameters alone.

Both learning curves exhibit a plateau, indicating the effectiveness of the training. The LSTM-agent, which has six times the parameters of the MLP-agent, clearly requires more training steps and wall-time. However, it achieves higher cumulative rewards. Meta-RL employs a recurrent agent, which allows the policy to adapt its actions within the same episode based on the given inputs. Consequently, during training, an MLP-policy acquires a generally effective strategy, whereas an LSTM-policy also learns to dynamically adjust itself to optimize performance for each task in a specific way.

B. Testing

The trained LSTM-policy is evaluated in the environment taking into account the same uncertainty regarding the initial conditions of the MDP. The deployment results, depicted in Figs. 8 and 9, are derived from a single episode.

Figure 8 shows that the chaser spacecraft, originating from a random initial condition, effectively reaches the target ($\delta \boldsymbol{x}_f = [0.612\ \mathrm{m}, 0.085\ \mathrm{m/s}]$), fulfilling safety requirements. The confirmation of adherence to the maximum flight-time constraint is visible in Figs. 9a and 9b. Figure 9c specifically displays the mass profile over time, and, according to Tsiolkovsky's equation, a $\Delta V = 13.113\ \mathrm{m/s}$ is required. Furthermore, Fig. 9d shows compliance with the restriction on maximum control action. It is essential to emphasize that the policy is adept at handling unforeseen thrust failures by learning to use half of the maximum thrust.

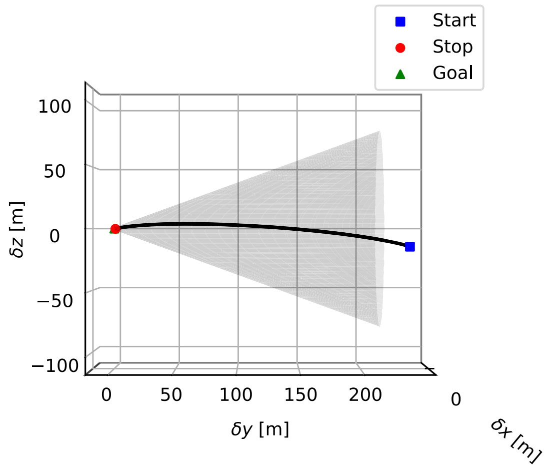


Fig. 8 Trained LSTM-policy deployed in the environment and tested for a single episode. Trajectory inside the approach corridor starting from a randomized initial condition.

A Monte Carlo analysis is performed on the trained LSTM-policy and MLP-policy, generating 500 trajectories, each with a different initial state due to the randomness of the MDP. This Monte Carlo campaign aims to evaluate the robustness and computational efficiency of Meta-RL as an autonomous G&C algorithm for on-board deployment. A performance comparison with the nonrecurrent policy is also performed. A summary of the results is presented in Table 4, whereas the trajectories are shown in Fig. 10. To evaluate the fuel optimality of the Meta-RL approach in this study, a state-of-the-art OCP direct pseudospectral method [50] is used as a reference. The results of the Monte Carlo campaign with the OCP method, considering the same initial condition uncertainties as the MDP but not accounting for process noise and actuator malfunctions, are presented in the same table. Additionally, during the Monte Carlo campaign, the CPU time for each policy evaluation step is measured and reported in the same table. For a more detailed discussion of the computational efficiency results, including a plot and considerations regarding computational efficiency when running the policy on a real spacecraft on-board computer (OBC), please refer to Appendix VIII.

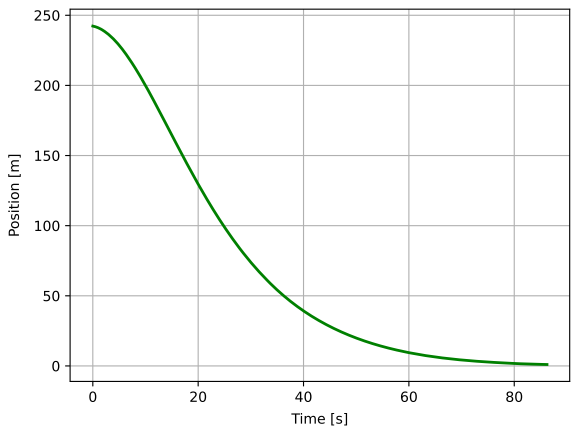
The LSTM-policy and the MLP-policy demonstrate success in all situations, obtaining a $S_r = 100\%$, where S_r is the proportion of trajectories that meet all requirements. Despite the numerous uncertainties and the potential for failure, the policies have effectively guided the spacecraft to the intended final state with minimal deviation in every case. However, the MLP-policy has much higher fuel consumption, whereas the LSTM-policy is near-optimal. Recurrent layers generate an adaptive policy that can more effectively optimize the trajectory of each episode. Therefore, the LSTM policy showed robustness, achieving the highest possible success rate, near-optimality, and computational efficiency ($dt_{\mathrm{CPU}} = 2.33$ ms, as illustrated more clearly in the figure in Appendix VIII) in Monte Carlo simulations. This makes it well-suited for real-time on-board applications. RL methods are highly computationally efficient for on-line operations because, after on-ground training, the on-board policy evaluation requires only a few matrix multiplications, as the policy is merely a function approximator.

The trained LSTM-policy is now subjected to an additional testing phase, encompassing a set of 500 trajectories in an enhanced environment similar to the one described. The enhanced environment eliminates process noise and introduces into the chaser dynamics the following elements: the Sun's fourth-body gravitational effect, considering the bicircular restricted four-body problem [51], and the solar radiation pressure (SRP), modeled with the cannonball method [52]. The objective is to show that the deployed policy can withstand unmodeled accelerations by introducing noise during training. The results are in Fig. 11 and Table 5, which demonstrate that the LSTM-policy is effective in handling unmodeled accelerations.

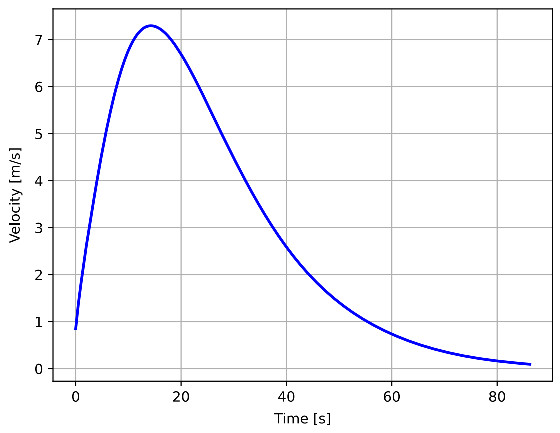
VII. Stability Analysis

A G&C algorithm should ensure the asymptotic stability, as per Lyapunov's definition [53], of the controlled system to achieve mission success, avoiding any erratic behavior and guaranteeing predictable vehicle performance, while also avoiding potential accidents or deviations from the intended trajectory. In the literature, there are a few analytical methods [41] that address the convergence of RL algorithms with function approximators. RL methods involve optimizing a criterion, inherently constraining states along the optimal trajectory, and stabilizing the system. However, in this section, Lyapunov's direct method [34] is employed to highlight system stability from the standpoint of nonlinear equations of motion. This approach, based on Lyapunov's second stability theorem [53], is advantageous because it enables the assessment of the stability of the controlled system without the need for linearization around an equilibrium solution.

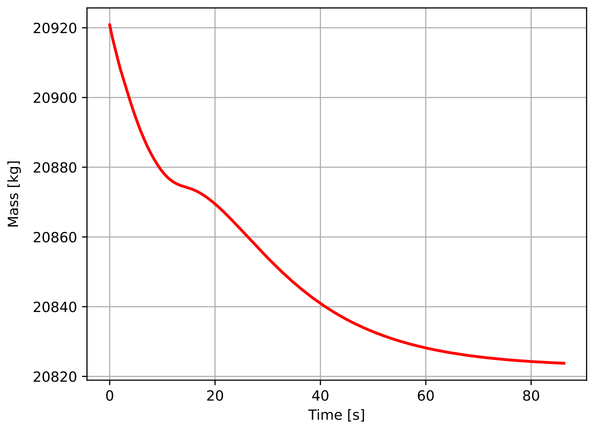
In this setting, the RCR3BP is assumed to be an autonomous dynamical system. This assumption holds provided that the target's absolute state $\boldsymbol{x}^T$ remains constant during the maneuver. This feasibility arises from the chaser's time of flight being considered



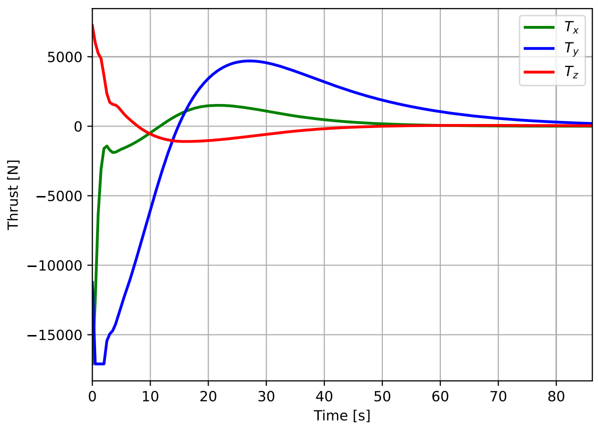
a) Relative position in time



b) Relative velocity in time



c) Total mass in time



d) Control action in time

Fig. 9 Trained LSTM-policy deployed in the environment and tested for a single episode. Relative position, relative velocity, mass, and control action along the trajectory of Fig. 8.

Table 4 Monte Carlo performance of the trained LSTM-policy and trained MLP-policy tested in the environment, alongside the OCP pseudospectral solution

	LSTM-policy: $\mu \pm 3\sigma$	MLP-policy: $\mu \pm 3\sigma$	OCP: $\mu \pm 3\sigma$
Success rate: S_r [%]	100	100	
Final position ρ_f [m]	0.561 ± 0.288	0.719 ± 0.153	
Final velocity $\dot{\rho}_f$ [m/s]	0.082 ± 0.012	0.088 ± 0.009	
Fuel consumption ΔV [m/s]	11.981 ± 1.485	18.669 ± 1.611	11.153 ± 2.316
Time of flight ToF [s]	82.571 ± 4.713	61.571 ± 3.822	76.619 ± 6.345
CPU-time step dt_{CPU} [ms]	2.334 ± 0.414		

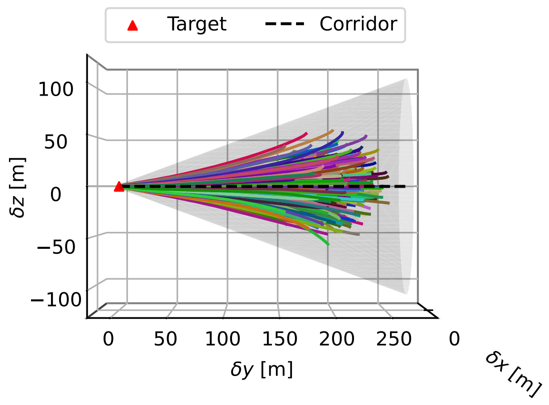


Fig. 10 Trained LSTM-policy deployed in the environment and tested for a batch of episodes. Trajectories inside the approach corridor start from a randomized initial condition.

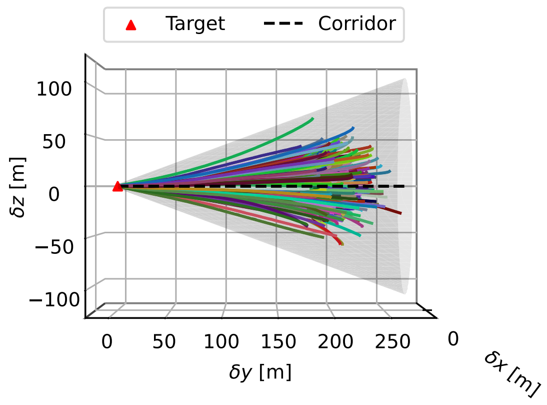


Fig. 11 Trained LSTM-policy deployed in an environment with unmodeled accelerations and tested for a batch of episodes. Trajectories inside the approach corridor start from a randomized initial condition.

Table 5 Monte Carlo performance of the trained LSTM-policy tested with unmodeled accelerations

Success rate: $S_r = 100\%$	Results MC: $\mu \pm 3\sigma$
Final position ρ_f [m]	0.497 ± 0.252
Final velocity $\dot{\rho}_f$ [m/s]	0.094 ± 0.003
Fuel consumption ΔV [m/s]	12.263 ± 1.572
Time of flight ToF [s]	81.713 ± 2.865

negligible in comparison to the orbital target period. With these assumptions, a positive-definite quadratic, *energy-error-like*, candidate Lyapunov function in $\mathbb{R}^6$ can be employed as follows:

$$V(\delta x) = \delta x^T P \delta x \tag{23}$$

The diagonal weight matrix considered is $\boldsymbol{P} = \mathrm{diag}(\mathbf{0.5}_6)$, and the equilibrium point is $\delta x^* = \mathbf{0}$, representing the desired outcome of the RVD maneuver. This equilibrium point would result in $\boldsymbol{h}(\delta x^*, \boldsymbol{u}) = \mathbf{0}$ in the RCR3BP equations. The stability assessment uses a Monte Carlo technique by randomly selecting the chaser's initial conditions, as defined for the MDP. A set of 500 trajectories is simulated using the trained LSTM-policy deployed in the environment. The numerical values of $V(\delta x)$ and $\dot{V}(\delta x)$ are determined along these trajectories. The former is calculated using the formula in Eq. (25), whereas the latter, its time derivative, is obtained numerically using a simple forward Euler finite difference method. The results are shown in Fig. 12, where the L_2-norm of the relative state vector serves as the abscissa in each graph. The developed LSTM-policy demonstrates asymptotic stability, as indicated by the identification of a candidate Lyapunov function that meets the criteria outlined in Lyapunov's second stability theorem.

VIII. Conclusions

This work demonstrates the efficacy of Meta-RL in effectively handling the formulated MDP. More specifically, it is applied to address the final approach and docking scenario of a spacecraft within a Southern L_2 9:2 Resonant NRHO in the Earth–Moon system. Despite a great deal of uncertainty in initial conditions, process noise, and actuator malfunctions, all objectives have been achieved. The learning curve shows that the LSTM-agent training phase was successful, as it reached the expected plateau. During the testing phase, the LSTM-agent is evaluated through a Monte Carlo campaign. The results demonstrate that the trained policy consistently meets all the requirements imposed on the entire set of generated trajectories. As a result, the Meta-RL algorithm has proven its ability to effectively learn a G&C policy tailored for RVD maneuvers within the cislunar space. Furthermore, the reliability of the method has been confirmed through Monte Carlo simulations conducted not only in the training environment, but also in an enhanced setting that incorporates unmodeled dynamics not present during the learning phase.

The computational efficiency for on-board execution of the LSTM-policy has been demonstrated, its fuel optimality has been verified through a comparison with a state-of-the-art OCP pseudospectral direct solution, and the asymptotical stability of the controlled system has been established from a nonlinear equations of motion perspective. Consequently, it can be considered a promising solution for autonomous G&C applications.

For the sake of comparison, a fully MLP-agent has also been trained. It successfully learns the proposed MDP, reaching the expected plateau during training, and achieving the highest possible success rate in the Monte Carlo testing campaign. However, the LSTM-policy outperforms the MLP-policy by acquiring greater cumulative rewards during training and demonstrating significantly higher fuel efficiency during testing. The internal memory of the LSTM-policy enables it to generate an adaptive G&C policy better suited to the optimal solution of the problem. This highlights the evident success of recurrent policies over nonrecurrent ones when addressing tasks characterized by significant parameter randomization.

Appendix : Computational Efficiency

During the Monte Carlo simulation illustrated in Fig. 10, the computational efficiency of the policy was evaluated at each step and is presented in Fig. A1. The policy evaluation had an average CPU time

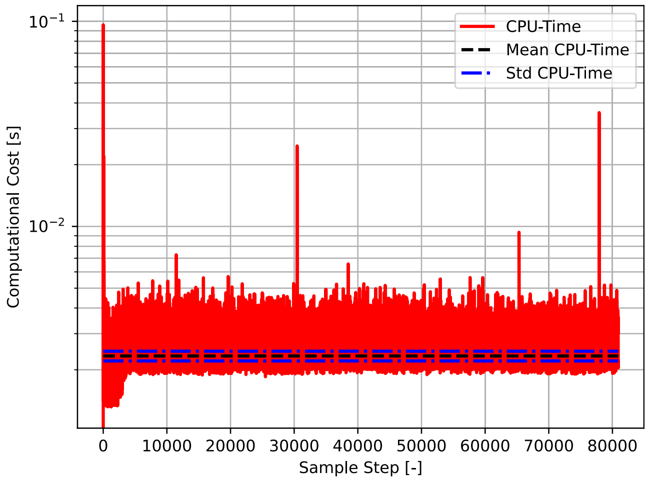
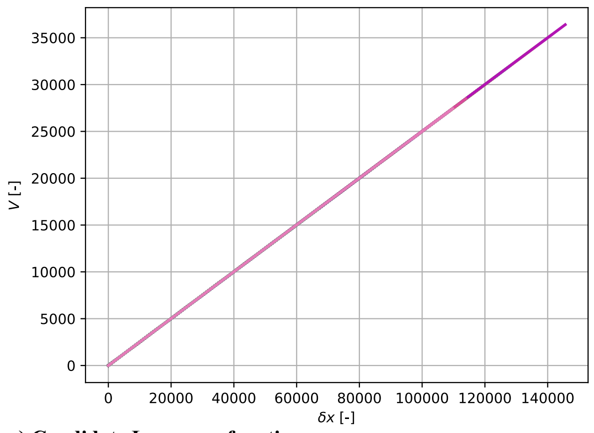
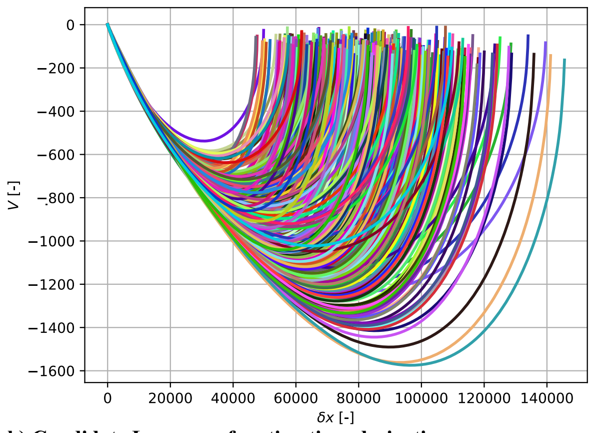


Fig. A1 Trained LSTM-policy deployed in the environment and tested for a batch of episodes. CPU-Time for each policy evaluation step during testing.



a) Candidate Lyapunov function



b) Candidate Lyapunov function time-derivative

Fig. 12 Trained LSTM-policy deployed in the environment and tested for a batch of episodes. Candidate Lyapunov function and its time-derivative with regard to the distance from the equilibrium point.

per step of 2.334 ms. A very simplified formula for estimating CPU time between two different machines is detailed by [54]: $\mathrm{CPU}_{\mathrm{time}} = \mathrm{CPI} \cdot I/R = I/\mathrm{FLOPS}_{\mathrm{core}}$, where CPI represents cycles per instruction, I is the total number of instructions, and R denotes the clock rate. However, in real operations, CPU time is highly dependent on the implementation of the flight software and the specific hardware characteristics of the OBC. Nevertheless, this formula can provide a rough estimate. In this paper, for the conversion, it is assumed that the number of instructions remains constant between the two machines and that single-thread usage is applied. The Intel(R) Core(TM) i7-8565U processor used in this work achieves 3.26 GFLOPS per core. When considering a clock rate of 250 MHz and a floating-point operations per second (FLOPS) per core of 100 MFLOPS on a LEON3FT SPARC V8 [55] on-board spacecraft processor, this translates to 76.088 ms (equivalent to 13.142 Hz). The method's computational efficiency is due to the fact that, once the agent has completed the learning phase on a high-fidelity ground simulator, the on-board execution of the policy for guidance and control during flight requires only a small number of matrix multiplications.

Acknowledgments

The first author acknowledges Prof. John R. Martin, Dr. Adam Hermann, and Mr. Mark Stephenson for their invaluable insights into reinforcement learning. It has been wonderful to be welcomed into your ML meetings every week and to learn from you. Special thanks to Prof. Hanspeter Schaub from the Ann & H.J. Smead Department of Aerospace Engineering Sciences for the opportunity to conduct a visiting research period at the AVS laboratory. Thanks to the University of Colorado Boulder for the Europe-Colorado Program scholarship and to Politecnico di Milano for the *Thesis Abroad* scholarship. Lastly, heartfelt thanks to my former advisor, Prof. Pierluigi Di Lizia, for his unwavering support and trust.

References

[1] Fehse, W., *Automated Rendezvous and Docking of Spacecraft*, Cambridge Univ. Press, Cambridge, England, U.K., 2003.
https://doi.org/10.1017/CBO9780511543388

[2] Xie, Y., Chen, C., Liu, T., and Wang, M., *Guidance, Navigation, and Control for Spacecraft Rendezvous and Docking: Theory and Methods*, Springer Singapore, Singapore, 2021.
https://doi.org/10.1007/978-981-15-6990-6

[3] Capello, E., Dabbene, F., Guglieri, G., and Punta, E., "“Flyable” Guidance and Control Algorithms for Orbital Rendezvous Maneuver," *SICE Journal of Control, Measurement, and System Integration*, Vol. 11, No. 1, 2018, pp. 14–24.
https://doi.org/10.9746/jcmsi.11.14

[4] Tipaldi, M., Iervolino, R., and Massenio, P. R., "Reinforcement Learning in Spacecraft Control Applications: Advances, Prospects, and Challenges," *Annual Reviews in Control*, Vol. 54, Jan. 2022, pp. 1–23.
https://doi.org/10.1016/j.arcontrol.2022.07.004

[5] Song, Y., Romero, A., Müller, M., Koltun, V., and Scaramuzza, D., "Reaching the Limit in Autonomous Racing: Optimal Control Versus Reinforcement Learning," *Science Robotics*, Vol. 8, No. 82, 2023, p. eadg1462.
https://doi.org/10.1126/scirobotics.adg1462

[6] Pierson, H. A., and Gashler, M. S., "Deep Learning in Robotics: A Review of Recent Research," *Advanced Robotics*, Vol. 31, No. 16, 2017, pp. 821–835.
https://doi.org/10.1080/01691864.2017.1365009

[7] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., and Perez, P., "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 6, 2022, pp. 4909–4926.
https://doi.org/10.1109/TITS.2021.3054625

[8] Izzo, D., Märtens, M., and Pan, B., "A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control," *Astrodynamics*, Vol. 3, No. 4, 2019, pp. 287–299.
https://doi.org/10.1007/s42064-018-0053-6

[9] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, Cambridge, MA, 2018.
https://doi.org/10.5555/3312046

[10] Hovell, K., and Ulrich, S., "Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 254–264.
https://doi.org/10.2514/1.a34838

[11] Oestreich, C. E., Linares, R., and Gondhalekar, R., "Autonomous Six-Degree-of-Freedom Spacecraft Docking Maneuvers via Reinforcement Learning," ArXiv, 2020.
https://doi.org/10.2514/1.I010914

[12] Scorsoglio, A., Furfaro, R., Linares, R., and Massari, M., "Relative Motion Guidance for Near-Rectilinear Lunar Orbits with Path Constraints via Actor-Critic Reinforcement Learning," *Advances in Space Research*, Vol. 71, No. 1, 2023, pp. 316–335.
https://doi.org/10.1016/j.asr.2022.08.002

[13] Federici, L., Benedikter, B., and Zavoli, A., "Deep Learning Techniques for Autonomous Spacecraft Guidance During Proximity Operations," *Journal of Spacecraft and Rockets*, Vol. 58, No. 6, 2021, pp. 1774–1785.
https://doi.org/10.2514/1.a35076

[14] Schweighofer, N., and Doya, K., "Meta-Learning in Reinforcement Learning," *Neural Networks*, Vol. 16, No. 1, 2003, pp. 5–9.
https://doi.org/10.1016/s0893-6080(02)00228-9

[15] Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M., "Learning to Reinforcement Learn," ArXiv, 2016.
https://doi.org/10.48550/arXiv.1611.05763

[16] Vuorio, J. B. R., Liu, E. Z., Xiong, Z., Zintgraf, L., Finn, C., and Whiteson, S., "A Survey of Meta-Reinforcement Learning," ArXiv, 2023.
https://doi.org/10.48550/arXiv.2301.08028

[17] Hochreiter, S., Younger, A. S., and Conwell, P. R., "Learning to Learn Using Gradient Descent," Lecture Notes in Computer Science, Vol. 2130, Springer–Verlag, Berlin, 2001, pp. 87–94.
https://doi.org/10.1007/3-540-44668-0_13

[18] Gaudet, B., Linares, R., and Furfaro, R., "Deep Reinforcement Learning for six Degree-of-Freedom Planetary Landing," *Advances in Space Research*, Vol. 65, No. 7, 2020, pp. 1723–1741.
https://doi.org/10.1016/j.asr.2019.12.030

[19] Federici, L., and Zavoli, A., "Robust Design of Interplanetary Trajectories Under Severe Uncertainty via Meta-Reinforcement Learning," *Proceedings of the International Astronautical Congress (IAC), Volume 2022-September, 73rd International Astronautical Congress, IAC 2022*, Code 190266, 2022.

[20] Gaudet, B., and Linares, R., "Adaptive Guidance with Reinforcement Meta-Learning," ArXiv, 2019.
https://doi.org/10.48550/arXiv.1901.04473

[21] Andrea, S., Andrea, D., Luca, G., Brian, G., Curti, Fabio, and Roberto, F., "Image-Based Deep Reinforcement Meta-Learning for Autonomous Lunar Landing," *Journal of Spacecraft and Rockets*, Vol. 59, No. 1, 2021, pp. 153–165.
https://doi.org/10.2514/1.A35072

[22] Gaudet, B., Linares, R., and Furfaro, R., "Adaptive Guidance and Integrated Navigation with Reinforcement Meta-Learning," *Acta Astronautica*, Vol. 169, April 2020, pp. 180–190.
https://doi.org/10.1016/j.actaastro.2020.01.007

[23] Calabrò, G., "Adaptive Guidance via Meta-Reinforcement Learning: ARPOD for an Under-Actuated CubeSat," Master's Thesis, Politecnico Di Milano, Milan, Italy, 2022.

[24] Federici, L., Scorsoglio, A., Zavoli, A., and Furfaro, R., "Meta-Reinforcement Learning for Adaptive Spacecraft Guidance During Finite-Thrust Rendezvous Missions," *Acta Astronautica*, Vol. 201, April 2022, pp. 129–141.
https://doi.org/10.1016/j.actaastro.2022.08.047

[25] Gaudet, B., Linares, R., and Furfaro, R., "Six Degree-of-Freedom Hovering over an Asteroid with Unknown Environmental Dynamics via Reinforcement Learning," *AIAA Scitech 2020 Forum*, AIAA Paper 2020-0953, 2020.
https://doi.org/10.2514/6.2020-0953

[26] Gaudet, B., Linares, R., and Furfaro, R., "Terminal Adaptive Guidance via Reinforcement Meta-Learning: Applications to Autonomous Asteroid Close-Proximity Operations," *Acta Astronautica*, Vol. 171, June 2020, pp. 1–13.
https://doi.org/10.1016/j.actaastro.2020.02.036

[27] Federici, L., Scorsoglio, A., Ghilardi, L., D'Ambrosio, A., Benedikter, B., Zavoli, A., and Furfaro, R., "Image-Based Meta-Reinforcement Learning for Autonomous Guidance of an Asteroid Impactor," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 11, 2022, pp. 2013–2028.
https://doi.org/10.2514/1.G006832

[28] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," ArXiv, 2017.
https://doi.org/10.48550/arXiv.1707.06347

[29] Lee, D. E., "Gateway Destination Orbit Model: A Continuous 15 Year NRHO Reference Trajectory," NASA Johnson Space Center TR JSC-E-DAA-TN72594, Aug. 2019.

[30] Lizy-Destrez, S., Le Bihan, B., Campolo, A., and Manglativi, S., "Safety Analysis for Near Rectilinear Orbit Close Approach Rendezvous in the Circular Restricted Three-Body Problem," *68th Annual International Astronautical Congress (IAC 2017)*, 2017, pp. 25–29, https://hal.science/hal-01755252.

[31] Colombi, F., Colagrossi, A., and Lavagna, M., "Characterisation of 6DOF Natural and Controlled Relative Dynamics in Cislunar Space," *Acta Astronautica*, Vol. 196, July 2022, pp. 369–379. https://doi.org/10.1016/j.actaastro.2021.01.017

[32] Bucchioni, G., and Innocenti, M., "Rendezvous in Cis-Lunar Space Near Rectilinear Halo Orbit: Dynamics and Control Issues," *Aerospace*, Vol. 8, No. 3, 2021, p. 68. https://doi.org/10.3390/aerospace8030068

[33] Lorenzo Bucci, A. C., and Lavagna, M., "Rendezvous in Lunar Near Rectilinear Halo Orbits," *Advances in Astronautics Science and Technology*, Vol. 1, Sept. 2018, pp. 39–43. https://doi.org/10.1007/s42423-018-0012-6

[34] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, AIAA Education Series, Reston, VA, 2003. https://doi.org/10.2514/4.861550

[35] Topputo, F., "On Optimal Two-Impulse Earth-Moon Transfers in a Four-Body Model," *Celestial Mechanics and Dynamical Astronomy*, Vol. 117, No. 3, 2013, pp. 1–34. https://doi.org/10.1007/s10569-013-9513-8

[36] Franzini, G., and Innocenti, M., "Relative Motion Dynamics in the Restricted Three-Body Problem," *Journal of Spacecraft and Rockets*, Vol. 56, No. 5, 2019, pp. 1322–1337. https://doi.org/10.2514/1.a34390

[37] Colagrossi, A., Lavagna, M., Biggs, J. D., and Masarati, P., "Absolute and Relative 6DOF Dynamics, Guidance and Control for Large Space Structures in Cislunar Environment," Ph.D. Thesis, Politecnico di Milano, Milan, Italy, 2019.

[38] Hochreiter, S., and Schmidhuber, J. U., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[39] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," ArXiv, 2015. https://doi.org/10.48550/arXiv.1506.02438

[40] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous Methods for Deep Reinforcement Learning," ArXiv, 2016. https://doi.org/10.48550/arXiv.1602.01783

[41] Sutton, R. S., Mcallester, D., Singh, S., and Mansour, Y., "Policy Gradient Methods for Reinforcement Learning with Function Approximation," ArXiv, 2015. https://doi.org/10.48550/arXiv.1706.06643

[42] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *3rd International Conference for Learning Representations*, San Diego, CA, May 2015. https://doi.org/10.48550/arXiv.1412.6980

[43] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, Vol. 22, No. 268, 2021, pp. 1–8, http://jmlr.org/papers/v22/20-1364.html.

[44] Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A., "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO," ArXiv, 2020. https://doi.org/10.48550/arXiv.2005.12729

[45] Bonasera, S., Bosanac, N., Sullivan, C. J., Elliott, I., Ahmed, N., and McMahon, J. W., "Designing Sun-Earth L2 Halo Orbit Stationkeeping Maneuvers via Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 2, 2023, pp. 301–311. https://doi.org/10.2514/1.g006783

[46] Ng, A., Harada, D., and Russell, S. J., "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," *Proceedings of the 16th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1999, pp. 278–287.

[47] Bonasera, S., "Incorporating Machine Learning into Trajectory Design Strategies in Multi-Body Systems," Ph.D. Thesis, Univ. of Colorado Boulder, Boulder, CO, 2022.

[48] Sak, H., Senior, A., and Beaufays, F., "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," ArXiv, 2014. https://doi.org/10.48550/arXiv.1402.1128

[49] Monika, S., and Alexander, G., "A Study on Catastrophic Forgetting in Deep LSTM Networks," *Artificial Neural Networks and Machine Learning—ICANN 2019: Deep Learning*, Springer International Publ., Cham, Switzerland, 2019, pp. 714–728. https://doi.org/10.1007/978-3-030-30484-3_56

[50] Ross, I. M., and Karpenko, M., "A Review of Pseudospectral Optimal Control: From Theory to Flight," *Annual Reviews in Control*, Vol. 36, No. 2, 2012, pp. 182–197. https://doi.org/10.1016/j.arcontrol.2012.09.002

[51] Mccarthy, B. P., and Howell, K. C., "Quasi-Periodic Orbits in the Sun-Earth-Moon Bicircular Rrestricted Four-Body Problem," *31st AAS/AIAA Space Flight Mechanics Meeting*, Paper AAS 21-270, 2021.

[52] Vallado, D. A., and McClain, W. D., *Fundamentals of Astrodynamics and Applications*, Microcosm Press, Torrance, CA, 2001.

[53] Khalil, H., *Nonlinear Systems*, Pearson Education, Prentice Hall, Hoboken, NJ, 2002.

[54] Patterson, D. A., and Hennessy, J. L., *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Francisco, CA, 1990.

[55] Habinc, S., "GR712RC Dual-Core LEON3FT SPARC V8 Processor," 2023, https://www.gaisler.com/index.php/products/boards/gr712rc-board?task=view&id=364 [accessed 19 July 2024].

R. Linares
Associate Editor