Annual Reviews in Control xxx (xxxx) xxx

Contents lists available at ScienceDirect

Annual Reviews in Control

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

Review article

Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges

Massimo Tipaldi [a,b,*], Raffaele Iervolino [c], Paolo Roberto Massenio [b]

[a] University of Sannio, Department of Engineering, 82100 Benevento, Italy
[b] Polytechnic University of Bari, Department of Electrical and Information Engineering, 70126 Bari, Italy
[c] University of Naples, Department of Electrical Engineering and Information Technology, 80125 Napoli, Italy

ARTICLE INFO

Keywords:
Reinforcement learning
Spacecraft control applications
Agent environment interface
Adaptive guidance and control
Policy gradient methods
Deep reinforcement learning

ABSTRACT

This paper presents and analyzes Reinforcement Learning (RL) based approaches to solve spacecraft control problems. Different application fields are considered, e.g., guidance, navigation and control systems for spacecraft landing on celestial bodies, constellation orbital control, and maneuver planning in orbit transfers. It is discussed how RL solutions can address the emerging needs of designing spacecraft with highly autonomous on-board capabilities and implementing controllers (i.e., RL agents) robust to system uncertainties and adaptive to changing environments. For each application field, the RL framework core elements (e.g., the reward function, the RL algorithm and the environment model used for the RL agent training) are discussed with the aim of providing some guidelines in the formulation of spacecraft control problems via a RL framework. At the same time, the adoption of RL in real space projects is also analyzed. Different open points are identified and discussed, e.g., the availability of high-fidelity simulators for the RL agent training and the verification of RL-based solutions. This way, recommendations for future work are proposed with the aim of reducing the technological gap between the solutions proposed by the academic community and the needs/requirements of the space industry.

1. Introduction

In the upcoming space missions, the need of designing spacecraft with highly autonomous on-board capabilities is an emerging trend (Frost, Butt, & Silva, 2010; McGovern & Wagstaff, 2011; Shirobokov, Trofimov, & Ovchinnikov, 2021; Tipaldi & Glielmo, 2018). Such capabilities rely on the usage of tools, mainly implemented on-board the spacecraft (Eickhoff, 2011), aimed at reducing the human-in-the-loop intervention, while dealing with uncertain and complex environments (Frost et al., 2010). The potential payoffs are considerable, such as the overall improvement of spacecraft availability and reliability and the reduction of costs in ground segment operations. In addition to this, any improvement in on-board autonomy and decision making for remote spacecraft can enable entirely new kinds of missions that are not currently possible (McGovern & Wagstaff, 2011). For instance, the next generation of landers for large and small planetary bodies (e.g., Mars and small asteroids) will require more advanced and integrated Guidance, Navigation, and Control (GNC) systems, able to satisfy increasingly stringent accuracy requirements (driven by the ambition of exploring regions having the potential to yield challenging scientific returns) and endowed with unprecedented levels of on-board autonomy (Gaudet & Furfaro, 2014; Scorsoglio et al., 2021; Tipaldi & Glielmo, 2018). Considering the strong interest in sending humans to Mars within the next few decades, as well as the renewed interest in building infrastructures in the Earth–Moon system for the easy access to the Lunar surface (Whitley & Martinez, 2016), the GNC system technology will need to make progress to satisfy the demand for more strict requirements. The Mars robotic landing systems have already experienced a relevant improvement from the 100 km required by the Phoenix mission (Shotwell, 2005) to 5 km of Sky Crane delivered by the Mars Science Laboratory (MSL) to the martian surface in 2012 (Singh, SanMartin, & Wong, 2007). Whilst acknowledging such improvement, future missions will need to deliver cargo in specified location with the pinpoint accuracy of 10–100 m (Furfaro, Scorsoglio, Linares, & Massari, 2020). Another example of unprecedented levels of on-board autonomy is given by distributed space systems composed of several microsatellites flying in formation, which are becoming increasingly appealing to the space community (Di Mauro, Lawn, & Bevilacqua, 2018). As the number of satellites grows, there is a need of both autonomous

* Corresponding author at: University of Sannio, Department of Engineering, 82100 Benevento, Italy.
E-mail addresses: mtipaldi@unisannio.it, massimo.tipaldi@poliba.it (M. Tipaldi), rafierv@unina.it (R. Iervolino), paoloroberto.massenio@poliba.it (P.R. Massenio).

https://doi.org/10.1016/j.arcontrol.2022.07.004
Received 27 May 2022; Received in revised form 28 July 2022; Accepted 30 July 2022

GNC functions (Silvestrini & Lavagna, 2021; Smith et al., 2021) and decision making and planning capabilities (Tipaldi & Glielmo, 2018). Autonomous trajectory planning capabilities can be also desirable for modern spacecraft so as to reduce the human effort with possibly faster reactions towards hazards and modified mission goals (Elliott, Bosanac, Ahmed, & McMahon, 2020; LaFarge, Miller, Howell, & Linares, 2021).

Space missions usually operate in challenging and changing environments with an extremely high cost of failure (McGovern & Wagstaff, 2011). Many space missions actually take place in environments with complex and time-varying dynamics that may be incompletely modeled during the mission design phase. For example, in orbital refueling missions, the inertia tensor of each of the two spacecraft can change significantly as fuel is transferred from one spacecraft to the other (Gaudet, Linares, & Furfaro, 2020a). Traditional optimization-based control approaches are dependent on the accuracy of the underlying dynamic models, and can fail if such models do not include correct gravitational models, disturbances, and other nonlinear terms (Gaudet, Linares, & Furfaro, 2018; Silvestrini & Lavagna, 2021). As a consequence, there is a need of using robust and adaptive approaches, which can cope with the typical issues of space missions, i.e., partially known environment and the limited on-board memory and computational resources (Furano et al., 2020; McGovern & Wagstaff, 2011; Tipaldi & Glielmo, 2018).

Reinforcement Learning (RL) based approaches can address such issues, and have become truly powerful lately, also thanks to their generalization capabilities (Gaudet et al., 2020a; Yang, Zhao, Li and Zomaya, 2020; Yoo, Byun, Han, & Lee, 2021). RL is a sub-discipline of Machine Learning (ML), and provides a principled mathematical framework to goal-directed learning, where the agents learn how to solve a task (like navigation or planning) by interacting directly with their own potentially changing environment (Bertsekas, 2019a; Glavic, 2019; Izzo, Märtens, & Pan, 2019; Oche, Ewa, & Ibekwe, 2021; Sutton & Barto, 2018). Its main idea is to produce fully autonomous agents, which learn a suitable control strategy/policy (i.e., a function mapping states, or observations, into actions) over time and through trial and error. More specifically, the agents improve their behavior by maximizing the expected cumulative reward (called return) when they interact with their own environment. The reward function has to be shaped according to the objectives the agent has to accomplish, and its proper definition is a key aspect for the RL agent learning process to work (Silver, Singh, Precup, & Sutton, 2021). Unlike supervised machine learning approaches (Labrèche et al., 2022; Shirobokov et al., 2021), no labeled data as reference to the solution is required during the training process (Shirobokov et al., 2021). The usage of deep Neural Networks (NNs) in RL approaches has given rise to the Deep Reinforcement Learning (DRL), thus allowing RL-based solutions to scale to problems with high-dimensional and continuous state and action spaces by representing them via compact low-dimensional function approximations (Arulkumaran, Deisenroth, Brundage, & Bharath, 2017). NNs have been used in control for decades in a variety of ways, mainly to handle system uncertainties and to implement online adaptive controllers (Emami, Castaldi, & Banazadeh, 2022; Fazlyab, Fani Saberi, & Kabganian, 2016; Massenio, Rizzello, Comitangelo, Naso, & Seelecke, 2021; Wilson & Riccardi, 2021). As a result, the resulting learnt policy proves to be robust to external disturbances/noise and adaptive to new scenarios, e.g., see Gaudet et al. (2020a) and Scorsoglio et al. (2021). For instance, as shown later in the paper, RL can enable new possibilities for closed-loop autonomous GNC systems by solving pinpoint powered descent and landing tasks and dealing with uncertain and complex environmental dynamics and off-nominal conditions (Gaudet, Linares, & Furfaro, 2020c; Scorsoglio et al., 2021). The integrated RL-based GNC systems could perform both the targeting and the trajectory-following algorithms on-board. Indeed, during the offline training phase of RL agents, RL algorithms can be used to compute policies addressing wide operational scenarios. In addition to that, the resulting trained policies can also exhibit generalization capabilities, being adaptive in real-time to changing environmental conditions and dynamics (Gaudet & Furfaro, 2014; Gaudet et al., 2020a).

In this survey, we present and analyze the RL-based approaches used to solve spacecraft control problems. The following application fields are addressed: GNC systems for spacecraft landing on celestial bodies (Section 3), maneuver planning for GTO/GEO transfer, cislunar transfer and interplanetary mission trajectory (Section 4), spacecraft attitude control systems (Section 5), guidance and proximity maneuvers in rendezvous and docking scenarios (Section 6), constellation orbital control (Section 7), on-board decision-making for spacecraft operations scheduling and rover path planning (Section 8). This work focuses mainly on peer-reviewed works solving problems in the above-mentioned application fields, giving evidence of the benefits of RL-based approaches in complex spacecraft control applications, but also highlighting the current limitations and developing some recommendations for future research. To the best of our knowledge, this is the first thorough survey focusing only on RL solutions and their applications to spacecraft control problems. The following themes are recurrently addressed throughout the paper:

- For each application field, the RL framework core elements (e.g., the reward function, the RL algorithm and the environment model used for the training) are discussed with the aim of providing some guidelines in the formulation of control problems in this context.
- The application of RL in a real industrial setting is also discussed. Even though RL-based solutions can be the enabler of new possibilities and up-to-now inconceivable space missions, their adoption in real space missions is still hindered by relevant open issues.

In Section 9, by analyzing the opportunities, challenges, and prospects of RL techniques, we outline some future developments aimed at reducing the technological gap between the solutions proposed by the academic community and the needs/requirements of the space industry. Finally, Section 2 provides an outlook on RL and shows the notation used, while Section 10 concludes the paper.

2. An outlook on Reinforcement Learning

RL is an effective algorithmic framework where an agent (i.e., the controller) learns a proper control strategy (called "policy") fulfilling specific objectives through interactions with its own environment (i.e., the controlled system or plant) (Sutton & Barto, 2018). RL agents are either directly trained in their environment itself or in a simulation of it (Bertsekas, 2019a, 2019b). As shown in this survey, the latter option is usually applied to spacecraft control problems. During each single episode (i.e., a sequence of steps from the initialization state up to a terminal state), the RL agent iteratively acquires a representation of the environment state, and uses such information and the current policy to select an action (i.e., the control signal) that is sent back to the environment. Upon the execution of the selected action, the environment enters a new state and generates a scalar reward signal. The reward and the representation corresponding to the next state are then passed back to the agent. The episode terminates when the agent completes its task (e.g., the lander has reached the terminal position on a celestial body) or some constraints (e.g., the compatibility of the lander attitude with its sensor field of view) are violated, see Gaudet et al. (2020a) and Gaudet, Linares, and Furfaro (2020d). After that, a new episode can be initialized by randomly generating the environment initial state variables and configurations, e.g., the asteroid rotational speed and its shape in a lander powered descent experiment (Gaudet et al., 2020c). By using all this information gathered during the execution of each episode, RL agents try to compute a policy maximizing the expected cumulative reward (or equivalently, minimizing the expected cumulative cost). For the learning process to work, it is fundamental to define a suitable reward (or cost) function

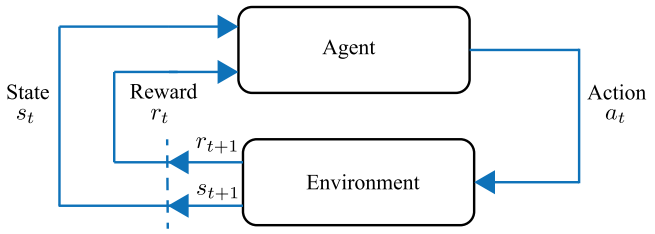

Fig. 1. The agent-environment interaction.
Source: Adopted and modified from Sutton and Barto (2018).

encapsulating the desired performance of the agent, after interacting with its environment, in accomplishing its prescribed task, and choose a proper RL method/algorithm to process all the information collected during the training phase (Bertsekas, 2019a; Sutton & Barto, 2018).

Being a classical formalization of sequential decision making problems, Markov Decision Processes (MDPs) offer a straightforward framing for RL-based approaches (Bertsekas, 2019a; Sutton & Barto, 2018). An MDP can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}, R, P \rangle$, where $\mathcal{S}$ is the environment state space (with $s \in \mathcal{S}$ its generic state), $\mathcal{A}$ is the action space (with $a \in \mathcal{A}$ its generic action), $r = R(s,a)$ is the reward function (r is the reward signal), and $P(s'|s,a)$ is the state transition probability distribution (i.e., it gives the probability of entering the state s' when the action a is performed in the state s). The agent policy $\pi(\cdot|s) := \mathcal{S} \to \Delta_{\mathcal{A}}$ is a mapping from states to the probability distribution, say $\Delta_{\mathcal{A}}$, over the whole action space $\mathcal{A}$, completely determining the control strategy. Note that this definition includes the case of a deterministic policy $\pi(\cdot) := \mathcal{S} \to \mathcal{A}$.

More in general, we can also consider partially observable MDPs (POMDPs), where the state s becomes a hidden state and what is actually available to the agent is its observation $o \in \mathcal{O}$ through an observation function $O(s)$ (Gaudet et al., 2020a; Scorsoglio et al., 2021). The POMDP formulation can be useful when the agent can only estimate a subset of the environment full state, as well as in case the observation consists of raw sensor outputs such as LIDAR readings or sequential camera images (Gaudet et al., 2020a). As for POMDPs, the agent policy becomes $\pi(\cdot|o) := \mathcal{O} \to \Delta_{\mathcal{A}}$.

Fig. 1 shows the agent-environment interaction for an environment modeled as an MDP (Sutton & Barto, 2018). Given a specific episode and at each time step $t = 1, 2, \ldots$, the agent receives a representation of the environment state $s_t \in \mathcal{S}$, and then selects an action $a_t \sim \pi(\cdot|s_t)$ (where the symbol $\sim$ stands for 'belonging to'). Upon its execution, the agent receives a reward signal $r_{t+1} = R(s_t, a_t)$, while the environment enters a new state $s_{t+1} \sim P\left(\cdot \middle| s_t, a_t\right)$. A similar pattern can be applied to the case of environments modeled as POMDP (the only difference is that the agent selects the action a_t based on the observation o_t).

As mentioned above, by updating the policy π during the learning process, the agent tries to maximize its (discounted) cumulative reward. Generally speaking, two kinds of value functions can be used to measure the quality of a given policy π, i.e., the state-value function $V^{\pi}(s)$ and the action-value function $Q^{\pi}(s,a)$ (Sutton & Barto, 2018). For a specific policy π, the state-value function $V^{\pi}(s)$ can be defined as follows

$$V^{\pi}(s) = \mathbf{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\Big], \tag{1}$$

where $\mathbf{E}_{\pi}[\cdot]$ is the expectation operator calculated when applying the policy π, $\gamma \in [0,1)$ is the discount factor (used to reduce the effect of future rewards while keeping the cumulative rewards over infinite horizon bounded), and s_0 is the initial state. The action-value function $Q^{\pi}(s,a)$ can be defined as follows

$$Q^{\pi}(s,a) = \mathbf{E}_{\pi}\Big[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a\Big]. \tag{2}$$

It represents the expected (discounted) cumulative reward when the agent takes the action a in state s, and then follows the given policy π. The optimal policy π^* is the one maximizing the value functions (1) and (2), while the corresponding optimal value functions are denoted with $V^*(s)$ and $Q^*(s,a)$, respectively.

In simple environments, policies and value functions can be represented in a tabular form. However, when the state and/or action spaces are large or continuous, the tabular representation becomes impractical (Bertsekas, 2019a; Forootani, Iervolino, & Tipaldi, 2019; Sutton & Barto, 2018). DRL methods address such issue by combining the RL paradigm with Deep Learning frameworks (Arulkumaran et al., 2017). More specifically, such methods allow RL-based solutions to scale to problems with high-dimensional or continuous state and action spaces by using Neural Networks (NNs) to approximate control policies and to learn their expected return, which are improved incrementally through interaction with the environment. As shown in this survey, DRL-based solutions are becoming increasingly common in spacecraft control problems thanks to their robustness to system uncertainties, their adaptability to new scenarios, and the limited computational and memory resources need for their on-board execution, e.g., see Gaudet, Linares, and Furfaro (2020b).

RL methods are basically model-free since the agent learns the optimal policy by interacting with its environment (or with a computer simulator that mimics its behavior and that includes adverse effects such as environmental noise and sensor/actuator noise) without knowing the state transition probability distribution and the reward function (Bertsekas, 2019a; Sutton & Barto, 2018). Closed-form analytical/mathematical equations of the whole environment are not needed (Bertsekas, 2019a). Such computer simulators (also called environment models Sutton & Barto, 2018) shall have the capability of generating state transitions and rewards used to train policies (Bertsekas, 2019b). As for spacecraft control applications, they can be thought as collection of SW modules simulating the spacecraft units, its dynamics, and its operational context (Eickhoff, 2011; Zoppi, Tipaldi, & Di Cerbo, 2018). Among other things, these SW modules can incorporate all the stochastic and dynamic models relevant for the agent to be trained.

RL approaches can be categorized into two major classes: value function methods and policy gradient methods. The former aims at findings good estimates for the value functions. We can mention Monte Carlo and Temporal Difference (TD) based methods (e.g., the Q-learning, the SARSA), as well as Deep-Q Learning and Deep-Q Network (DQN), where neural networks are used as value function approximators (Arulkumaran et al., 2017). As for RL value function methods, all the possible actions must be theoretically considered, thus these algorithms only work with discrete action spaces (Gaudet et al., 2020b; Sutton & Barto, 2018).

However, many applications (in particular, well-known spacecraft control problems, e.g., see Ciabatti, Daftry, and Capobianco (2021), Gaudet and Furfaro (2014) and Gaudet et al. (2020b)) are featured by continuous control spaces. Therefore, policy gradient learning methods have to be used (Bertsekas, 2019a; Sutton & Barto, 2018). Such methods learn a parameterized policy $\pi_{\theta}(\cdot) := \pi(a|s,\theta)$ with $\theta \in \mathbb{R}^d$ as the policy parameter vector. Such policy gives the probability that the action a is taken when the environment is in state s with the policy parameter vector set to θ. Unlike value function methods, policy gradient approaches select actions without consulting a value function, although value functions may be used to learn a proper policy parameter θ. Generally speaking, the main idea behind policy gradient methods is to run iteratively a batch of episodes and collect samples from the environment by applying a specific policy π_{θ_k} (note that k denotes the episode index). Then, the policy parameter vector is updated by taking a gradient step in the ascending direction

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} V^{\pi_{\theta_k}}, \tag{3}$$

where α is the learning rate, θ_k is the policy parameter vector considered at the iteration k, and $V^{\pi_{\theta_k}}$ denotes the state-value function computed when the policy π_{θ_k} is applied. In literature, there exist different types of policy gradient methods, each providing a different approach for the computation of the gradient term in (3). Amongst other things, we can mention the REINFORCE algorithm (Williams, 1992), the Actor–Critic methods (e.g., the Advantage Actor–Critic (A2C) algorithm Mnih et al., 2016), and the Proximal Policy Optimization (PPO) algorithm (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). Neural networks are widely used in all policy gradient methods to formulate policies and value functions. For instance, in actor–critic approaches, an actor neural network is used to define parameterized policies and outputs actions to the environment based on the observed state of the environment, while a critic network outputs the value function based on the state and the rewards collected from the environment (Sutton & Barto, 2018; Yang, Zhao et al., 2020).

For the sake of brevity, this paper is not meant to provide any further details on RL-based algorithms, which can be found in the referenced literature. However, relevant aspects about their usage in spacecraft control problems are highlighted in the next sections.

3. Design of guidance and control systems for spacecraft landing on celestial bodies

This section addresses RL-based approaches for the control problem of a spacecraft landing on the surface of celestial bodies with and without an atmosphere, e.g., Mars or an asteroid. In case of celestial bodies with an atmosphere, the overall landing process consists of two phases, i.e., the entry phase, where the vehicle relies only on aerodynamics for deceleration and trajectory corrections, and the powered descent phase, where the retro-propulsion system is activated to steer the spacecraft trajectory towards the desired location on the planetary surface (Furfaro et al., 2020; Jiang, Li, & Furfaro, 2019). Combining the entry guidance with the powered descent guidance (and making them work collaboratively) is a challenging task from a control design perspective, but anyway needed to fulfill two competing objectives, i.e., landing on a selected terminal point with pinpoint accuracy while autonomously flying fuel-efficient trajectories (Gaudet & Furfaro, 2014; Jiang et al., 2019). The handover point between these two phases can belong to a wide region (the so-called deployment ellipse Gaudet et al., 2020b), and such aspect has to be considered in the design of the overall robust integrated guidance and control system (Gaudet et al., 2020a; Jiang et al., 2019; Wilson & Riccardi, 2021). Moreover, environmental parameters cannot be easily determined. In this regard, we can mention the lander wet mass, the environmental disturbance acceleration, and the solar radiation pressure. As for landing on asteroids, current practices for close proximity maneuvers require an accurate characterization of the environmental dynamics and precise spacecraft positioning before starting the maneuver from a given deployment region (Gaudet et al., 2020c, 2020d).

As highlighted in several articles (see, e.g., Gaudet et al. (2020a, 2020c, 2020d)) and outlined in this Section, RL-based solutions show promising performance for landing applications on celestial bodies, where the training of the policy can be carried out over a wide range of randomized system and environment parameters (e.g., the gravity acceleration in an asteroid landing scenario) as well as initial conditions (e.g., the above-mentioned handover point). For instance, in Gaudet et al. (2020a), the authors use a meta-Reinforcement Learning (meta-RL) approach to formulate two adaptive guidance laws, one suitable for controlling a lander for a Mars powered descent phase, and the other suitable for landing on small celestial bodies without an atmosphere, i.e., an asteroid. Meta-RL can be considered as an implementation of the idea of learning to learn: the agent learns to act effectively not just in one task (but in a series of tasks) and, when dealing with a new task that has never been encountered during training (e.g., in online mode), it is able to adapt to it (Shirobokov et al., 2021). In literature, there exist several examples of spacecraft landing applications solved by means of meta-RL approaches, with recurrent neural networks used both for the actor and critic parts (e.g., see Gaudet et al. (2020b, 2020c, 2020d) and Scorsoglio et al. (2021)). For instance, an image-based meta-RL approach is proposed in Scorsoglio et al. (2021) to solve the lunar pinpoint powered descent and landing task with uncertain dynamical parameters and actuator failure conditions. The trained closed-loop meta-RL policy is able to process real-time images and ranging observations (acquired during the simulated descent) by mapping them directly to thruster commands (i.e., sensor-to-action policy), while errors in the order of meters in different scenarios are achieved.

Recurrent neural networks represent an interesting class of intelligent and adaptive algorithms and have the capabilities of learning the time-varying and complex dynamics either in off-line mode (i.e., during the training phases) or in on-line/real-time mode (i.e., during the testing phase) (Gaudet & Furfaro, 2014). In other words, trained policies can adapt in real-time to changing dynamics and internal/external disturbances. The computation/training of policies involves a simulated interaction between the agent and environment over many episodes with randomly generated initial conditions and environmental parameters from suitable ranges, which demonstrates the ability of RL-based approaches to learn policies over a large “theater of operations” (Gaudet et al., 2020a). This has the potential benefit to simplify mission planning and execution activities (Tipaldi & Glielmo, 2018).

In order to run the episodes needed for the policy training, an RL framework has to be instantiated for the application at hand. In other words, the following core elements have to be defined: the environment model (i.e., something that mimics the behavior of the environment Sutton & Barto, 2018), the reward function, the state/observation space, the action space, the RL training algorithm, and the policy definition. This aspect has been analyzed in the referenced papers and is discussed shortly in the remaining part of this Section.

Fig. 2 shows the RL framework definition for addressing the adaptive landing problem on a celestial body. It has been derived from some relevant works, e.g., Gaudet et al. (2018, 2020a, 2020c, 2020d). Different environment models can be found in literature, depending on the addressed celestial body (e.g., Mars or an asteroid) and the number of Degree-of-Freedom of the lander (DOF), which also affects the dimension of the state and action spaces. More specifically, in case of 3-DOF landing environments, only the lander translational position and velocity are considered (Gaudet et al., 2020a; Wilson & Riccardi, 2021), whereas the lander attitude and rotational velocity are also modeled in a 6-DOF landing environment (Gaudet et al., 2020b, 2020c).

For the sake of clarity and brevity, we outline hereafter the 3-DOF Mars landing environment model reported in Gaudet et al. (2020a) (used for the RL agent training and testing), where the translational motion of the lander is formulated by the following equations

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{4a}$$

$$\dot{\mathbf{v}} = \frac{\mathbf{T}}{m} + \mathbf{g} + \mathbf{a}_{env}, \tag{4b}$$

$$\dot{m} = -\frac{\|\mathbf{T}\|}{I_{sp} g_{ref}}, \tag{4c}$$

where $\mathbf{r}$ is the lander position in the target centered reference frame, $\mathbf{T}$ is the lander thrust vector (whose components are usually constrained to be within a given range), m is the lander wet mass (which can be set over a random distribution of its nominal value in each training episode), a_{env} is the environmental disturbance acceleration (again, to be set over a given random distribution in each episode), $\mathbf{g}$ is the Mars gravity acceleration vector, I_{sp} is the specific impulse, and g_{ref} is the Earth gravitational acceleration. The environment model becomes more complex if the 6-DOF landing problem on an asteroid is considered. The reader can refer to the cited papers (e.g., Gaudet et al. (2020d)) for further information on landing environment modeling approaches.

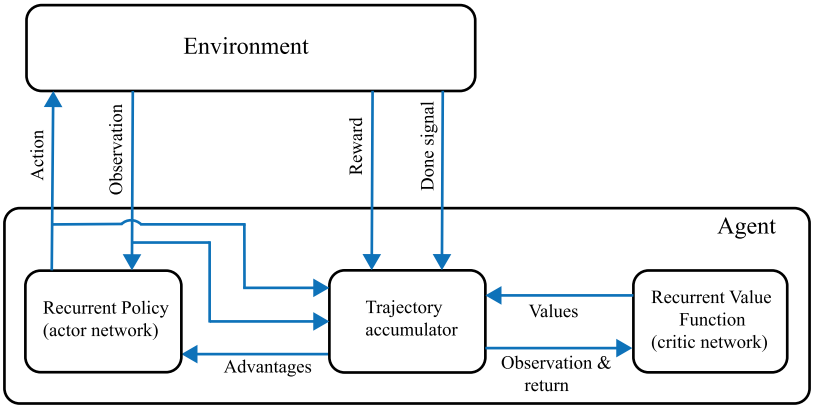

Fig. 2. RL framework definition for training the adaptive landing policy.
Source: Adopted and modified from Gaudet et al. (2020a, 2020c, 2020d).

The reward function is usually defined as a combination of shaping and terminal rewards, e.g., see Gaudet et al. (2020a), Scorsoglio et al. (2021) and Wilson and Riccardi (2021). The former provides to the agent information about how well it is performing at each time step during an episode. This information can be formulated as the velocity tracking error $\mathbf{v}_{err} = \|\mathbf{v} - \mathbf{v}_{targ}\|$, where $\mathbf{v}_{targ}$ is a vector field mapping the lander position to a target velocity aligned with the line of sight vector. This way, the lander first tries to target a specific location above the desired landing site, and afterwards a vertical descent towards it is encouraged. The terminal reward is a piece-wise bonus given if the final position and velocity (and, in case of a 6-DOF landing problem, if all the components of the lander angular velocity) are within specified limits. Moreover, a term penalizing the control effort, a term penalizing constraint violations (so that constraints are accounted during the training), and a constant positive term η (encouraging the agent to make progress along the trajectory) are usually added into the reward function. Thus, the reward signal r can be generally defined as follows

$$r = \beta \mathbf{v}_{err} + \lambda \parallel \mathbf{T} \parallel + \eta + \zeta(\text{good landing}) + \kappa(\text{constraint violation}), \tag{5}$$

where β, λ, ζ, and κ are weighting parameters. The different terms of the reward signal definition (5) can be shaped according to the objectives the agent has to accomplish and the environment model complexity.

The state/observation vector also depends on the specific landing problem (Mars vs. asteroid landing and 3-DOF vs. 6-DOF). For instance, in Gaudet et al. (2020b), the observation vector given to the agent during learning and testing is defined as follows

$$o = [\mathbf{v}_{err}, \mathbf{q}, \boldsymbol{\omega}, r_z, t_{go}]', \tag{6}$$

where $\mathbf{q}$ is the estimated lander attitude, $\boldsymbol{\omega}$ is its estimated rotational velocity vector, r_z is its estimated altitude, and t_{go} is the time-to-go (note that the symbol $'$ denotes the transpose operator). The latter is computed as the ratio between the distance from the terminal position $\mathbf{r}$ (measured by the on-board LIDAR altimeter) and the magnitude of the lander velocity (which can be estimated by filtering $\mathbf{r}$ Gaudet et al., 2020d). The action generated by the agent is the thrust vector $\mathbf{T} = \sum_{i=1}^{k} \mathbf{T}_i$ (with $\mathbf{T}_i \in \mathbb{R}^3$), where k is the number of thrusters and the thrust values are usually constrained to be within a given range (Gaudet et al., 2020a, 2020b; Scorsoglio et al., 2021).

As for the RL algorithm used to train the policy, the PPO algorithm (Schulman et al., 2017) with recurrent neural networks is usually adopted for training adaptive meta-RL policies in landing problems (e.g., see Gaudet et al., 2020a, 2020b; Scorsoglio et al., 2021). The training is performed via simulation, and Fig. 2 shows the related RL framework set-up for the PPO algorithm. Both the policy (mapping the estimated lander state into a commanded thrust vector) and the value function are implemented via recurrent neural networks, while the advantage function is computed as the difference between the sum of discounted rewards (collected during an episode) and the estimated state-value function provided by the critic network. Fig. 3 shows how the trained recurrent policy can be deployed into the lander system and interfaces with the peripheral lander components during the powered descent phase. After training, although the recurrent policy network weights are frozen, the hidden state will continue to evolve in response to a sequence of observations and actions, thus making the policy adaptive to new environment configurations and robust to external disturbances and thruster failure conditions. This is the main feature of meta-RL approaches, in particular meta-RL policies are able to adapt to new scenarios and learn new tasks by exploiting their previous knowledge (Scorsoglio et al., 2021).

The literature referenced herein presents a lot of simulated landing scenarios demonstrating the capabilities of the trained RL policies in achieving the desired performance, robustness and adaptability objectives, e.g., see Gaudet et al. (2020a, 2020b, 2020c, 2020d). Moreover, some interesting comparisons with more traditional approaches are also shown. For instance, in Gaudet et al. (2020a), the learnt RL policy for a 3-DOF simulation environment outperforms a traditional energy-optimal closed-loop guidance algorithm (Battin, 1999) (called "DR/DV policy"), as well as can adapt to new environmental conditions, e.g., engine failure conditions, a larger lander mass variation during the powered descent phase, and a higher lander state estimation bias. In Gaudet et al. (2020b), the authors demonstrate the RL agent robustness to noise and system parameter uncertainty presented in a 6-DOF simulation environment for the powered descent problem on a large planetary body such as Mars. Moreover, numerical results comparing the RL-based closed-loop policy with the solution of an open-loop optimal trajectory derived using the Gauss Pseudospectral Optimal Control Solver (GPOPS) (Rao et al., 2010) and the DR/DV policy are reported. In particular, although fuel efficiency is not optimal, the RL approach has the advantages of delivering better landing accuracy performance over the theater of operations than both the GPOPS solution and the DR/DV policy.

Finally, we also highlight that RL-based approaches can be combined with more traditional solutions to overcome some of their limitations. For instance, in Furfaro et al. (2020), the analytical Zero-Effort-Miss/Zero-Effort-Velocity (ZEM/ZEV) feedback guidance algorithm (derived by a straightforward application of the optimal control theory to the powered descent landing problem Guo, Hawkins, & Wie, 2013) is generalized and made adaptive. The overall structure of the guidance algorithm is unchanged with respect to the classical ZEM/ZEV, but the optimal guidance gains are determined at each time step as function of the state via a parameterized learnt policy. The resulting guidance algorithm generates closed-loop trajectories that are quasi-optimal (w.r.t. the fuel-efficiency) and satisfies flight constraints (e.g., thrust constraints).

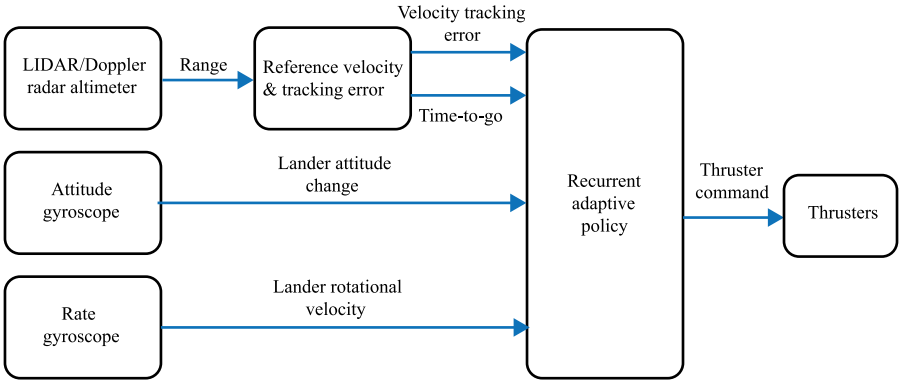

Fig. 3. Deployed RL policy and its interfaces with the lander components
Source: Adopted and modified from Gaudet et al. (2020c, 2020d).

4. GTO/GEO transfer, maneuver planning in multi-body environment, and interplanetary trajectory design

In the most recent years, (solar) electric propulsion (EP) systems are replacing traditional chemical propulsion systems in various space missions, including orbit raising maneuvers, maneuver planning in multi-body environment and interplanetary transfer (Miller, Englander, & Linares, 2020). The main reason relies on the much higher specific impulse I_{sp} that allows more payload for scientific equipment or cargo (Miller et al., 2020). On the other side, the low-thrust characteristic of EP systems implies continuous thrust operations, with burns measured in days or weeks, different from the impulsive (and discontinuous) thrust feature, with burns measured in seconds or minutes, of chemical propulsion. Therefore, the main challenge consists in selecting the optimal sequence of thrust levels and directions (action space) over long and continuous periods of time. As matter of fact, traditional optimization techniques cannot be applied satisfactorily, since they can easily suffer from the computational burden, unless sub-optimal solutions, with poorer performance values, are accepted. In this regard, there are recent research articles which address this issue by bridging traditional optimal control techniques with RL methods, as shown in the following subsections.

4.1. GTO/GEO transfer

Orbit transfers from geostationary transfer orbits (GTO) to geosynchronous equatorial orbit (GEO), or, simply, geostationary orbit, usually named *orbit raising maneuvers*, are of major importance in many Earth satellites applications (Curtis, 2010). Indeed, every satellite directed to GEO needs to be previously placed into a GTO, by means of the high-thrust engines of its launch vehicle, and then, successively, transferred to GEO by using its on-board low-thrust engines. Especially for high-inclined GTO, the transfer is accomplished by several middle steps, i.e., several intermediate orbits have to be planned and tracked (Junkins & Taheri, 2018).

The availability of low-thrust engines, such as EP systems, characterized by a near continuous thrust, has called for low-thrust many-revolution trajectory design and orbit transfer planning techniques, no longer based on impulsive maneuvers approximation (Junkins & Taheri, 2018). Being complex optimal control problems, conventional optimization methods can be employed to tackle this issue, in principle, with the main aim of minimizing the transfer time (Arora & Dutta, 2020). However, such methods cannot actually be implemented as on-board guidance law since they involve a relevant computational burden (even though such issue could be addressed via proper robust analytical approaches as explained in Locoche (2021)), provide only solutions valid for specific initial (operational) conditions, and suffer from time limitations and difficulties in ensuring convergence (Holt et al., 2021; Holt, Armellin, Scorsoglio, & Furfaro, 2020). On the other hand, heuristic control approaches, such as the closed-loop feedback-driven control, entail low computational costs, but at the price of low performance values and sub-optimal solutions (Arora & Dutta, 2020).

In orbit raising maneuvers planning, the action selection (from a set of feasible actions) relies on the current state only, i.e., it does not depend on previous choices of planning variables in a preceding optimization step (Arora & Dutta, 2020). For this reason, MDP based frameworks (and RL algorithms to solve them) are natural candidates to formalize orbit raising problem. The combination of RL with existing control methodologies to solve orbit transfer problems has been recently proposed by some authors to achieve near optimal guidance laws with acceptable performance levels (Arora & Dutta, 2020; Harris, Teil, & Schaub, 2019; Holt et al., 2021, 2020). All of these articles share a common methodology to cope with the sequential orbit raising problem issues, which basically consists of a two-level optimization framework with

1. A low-level optimization problem, used to determine the orbit raising maneuvers up to the GEO orbit by breaking the multi-revolution large-scale optimal control problem into a sequence of optimization sub-problems.
2. A high-level planning problem (formulated via RL frameworks), used to modify the objective function weights of the low-level optimization sub-problems in order to minimize the orbit transfer time or the propellant usage.

In particular, in Holt et al. (2021, 2020) the authors propose to incorporate an RL actor–critic method into the Lyapunov-based Q-law controller (Petropoulos, 2005) (used for the low-level optimization problem) to solve orbit raising problems. Such actor–critic RL approach is used to make the parameters of the Lyapunov-based Q-law state-dependent, thus ensuring that the controller can adapt as the dynamics evolve during a transfer. This way, it is possible to address the issue of Lyapunov controllers being inherently sub-optimal with user-defined parameters which significantly affect their performance. In the above-mentioned papers, the state vector χ includes all the classical orbital elements with the exception of the true anomaly (i.e., semi-major axis, eccentricity, inclination, right ascension of the ascending node and argument of periapsis), while the action is a resultant perturbing acceleration. The candidate Lyapunov function, named Q-law, is chosen as the weighted, squared summation of the difference between the states χ and the target state χ_T. Such weights are assumed to be piecewise constant during the whole orbital transfer, which is decomposed into a number of small intervals of time where the spacecraft is placed into an intermediate orbit (Holt et al., 2020).[1]

[1] In Holt et al. (2020) and in Holt et al. (2021), a subdivision into 1 day and 0.25 days intervals is used, respectively.

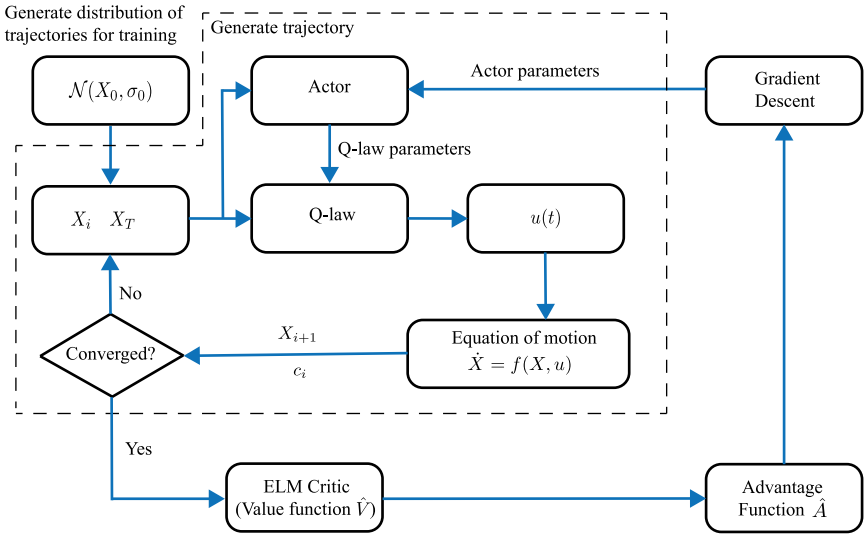

Fig. 4. The control scheme with actor–critic algorithm to compute the Q-law parameters.
Source: Adopted and modified from Holt et al. (2020).

The RL environment model includes the dynamics of the system with Gauss's Variational Equations and J_2 perturbation effects. The RL agent is trained by simulating trajectories generated in the environment and receiving feedback based on the performance. More specifically, the policy, implemented via an actor neural network, maps states into Q-law parameters. The policy parameters θ (i.e., the weights and biases of its associated actor neural network) are determined during the RL training phase. To do that, the critic network, which is an outer learning loop consisting of a single layer feed-forward network, also known as Extreme Learning Machine (ELM), is employed to improve the policy parameters estimate, and, hence, the policy itself. By generating a batch of trajectories of prescribed length, under the same policy, the critic network computes the approximate value for each visited state, by using the costs c_i (based on the performance of the ith trajectory), the final time-of-flight (ToF) for the transfer, and a penalty term used to ensure convergence within a maximum iteration time (Holt et al., 2020). Then, the advantage function is computed, and used to update the policy parameters via gradient descent. All these computations are performed when the episode ends, i.e., when the current trajectory converges to the target orbit within a convergence criteria, or when the maximum simulation time-steps have been reached.

Fig. 4 schematizes the RL actor–critic Q-law framework, as proposed in Holt et al. (2020). A bunch of trajectories is generated starting from random initial conditions, normally distributed with average X_0 and standard deviation σ_0. In Holt et al. (2021), the RL update strategy is slightly modified and improved, in terms of optimality and robustness, through the PPO algorithm (Schulman et al., 2017), which saturates the objective function in order to avoid that the updated policy is too far away from the current one, thus inhibiting possible updating errors. Both in Holt et al. (2021, 2020), a particle swarm optimization (PSO) approach (Kennedy & Eberhart, 1995) is used to generate time-dependent Q-laws which act as a suitable benchmark for comparison with the state-dependent RL Q-law approach.

Similarly to Holt et al. (2021, 2020), in Arora and Dutta (2020) the multi-revolution low-thrust trajectory design is first split into a finite number of optimizations (low-level optimization problems), where the number of total revolutions itself is an optimization variable. The low-level objective function weights are then determined by solving a high-level planning problem. The state vector is composed by $(h, h_X, h_Y, e_x, e_y, \phi)$, where h is the specific angular momentum, h_X and h_Y are the components of the vector h in a $X-Y$ Earth-centered inertial reference frame, e_x and e_y are the components of the eccentricity vector in a $x-y$ non-inertial frame, and ϕ is an angle providing the angular position of the spacecraft along its orbit, counted from the inertial $X-Z$ plane. In the low-level optimization, the objective function for each sub-problem measures how close to GEO the current orbit is, at the end of each revolution. In order to apply RL methods, a reward associated to each revolution has to be also identified, as in Arora and Dutta (2020), where the reduction in transfer time over each planning horizon is chosen. For the planar transfer case, the reward r_k corresponding to the kth revolution, i.e., the transfer time over each planning horizon, is[2]

$$r_k = \frac{h_{k+1}^3}{\mu^2} \sqrt{\frac{1}{1 - e_{x,k+1}^2 - e_{y,k+1}^2}}, \tag{7}$$

where μ is the gravitational parameter. The selection of the weights in the objective function is performed at the high level optimization step, for each planning horizon. By defining a finite set of feasible weight values for each revolution, the learnt policy becomes a mapping which computes the new weights from the orbit parameters and the weights of the previous orbit. Associated to each action, there is a corresponding reward, which depends on the actions and the planning variables, made of the orbit main parameters (i.e., h_k, e_k and inclination i_k), together with the optimization weights. With that said, the objective of the high-level planning problem is to choose the action that leads to the smallest transfer time for the orbit-raising maneuver. In Arora and Dutta (2020), to handle the resulting high dimensional state space, the DQN algorithm is used to train the RL agent with episodes defined from orbit raising scenarios. Moreover, different deep NNs are trained with different numbers of hidden neurons, to find out the least number of hidden neurons with the best achievable performance (Arora & Dutta, 2020).

4.2. Maneuver planning in multi-body systems: cislunar transfer

The advent of miniaturizing technology has extended the employability of small satellites from low Earth orbits to multi-body gravitational environments, as in cislunar space missions (LaFarge et al., 2021; Sullivan & Bosanac, 2020). In this context, the trajectory and maneuvers design problems become high dimensional, whereas additional constraints are to be taken into account, such as the small satellite actuators thrust limitations. Traditional optimal control methods,

[2] Note that in this case the reward is not constant and depends on the current iteration step.

based on approximate multi-body dynamic models, do not guarantee autonomous/robust on-board trajectory and maneuvers profiles determination in multi-body scenarios (even though the computational burden of on-board classical guidance algorithms could be tackled via non-convex/sequentially convex programming approaches, see Contini, Zamaro, Marinas, and Casasco (2021)). On the contrary, RL methods are arousing an increasing interest in such application field for their intrinsic adaptation capabilities (Elliott et al., 2020; LaFarge et al., 2021; Sullivan & Bosanac, 2020). In particular, the transfer trajectory design through RL algorithms in the context of Circular Restricted Three-Body Problem (CR3BP) is the subject of several papers (Elliott et al., 2020; Federici, Scorsoglio, Zavoli and Furfaro, 2021; LaFarge et al., 2021; Sullivan & Bosanac, 2020; Sullivan, Bosanac, Anderson, Mashiku, & Stuart, 2021). In the CR3BP model, the dynamics of a negligible mass spacecraft is considered in presence of two bigger bodies, i.e., the Earth and the Moon, which are rotating in circular orbits around their mutual center of gravity (Koon, Lo, Marsden, & Ross, 2006). Such a multi-body model admits 5 equilibrium points, named $L_1, \dots, L_5$, among which the most interesting periodic orbits, from an application point of view, are the L_1 and L_2 planar Lyapunov orbits and halo orbit families (Koon et al., 2006; Sullivan et al., 2021).

In LaFarge et al. (2021) and Sullivan and Bosanac (2020), an L_2 (planar) Lyapunov orbit is specifically set as target orbit, while a reference trajectory starting close to the Moon and asymptotically approaching it through a stable manifold is selected. In order to reach one of the states associated to the reference trajectory, a controlled transfer from the actual spacecraft state is needed. As for the RL environment model, the proposed DRL approach in LaFarge et al. (2021) and Sullivan and Bosanac (2020) considers a CR3BP dynamics model enriched with the small satellite mass and the acceleration dynamics related to the low-thrust engine (a low-thrust enabled CR3BP Sullivan & Bosanac, 2020), whose states are the spacecraft position and velocity vectors in the Earth–Moon rotating frame, the spacecraft mass, and the time elapsed from the initial epoch. The DLR network follows a typical Actor–Critic structure, with the PPO algorithm employed for the networks training (Schulman et al., 2017), the latter used to avoid large updates in the weights and the policy. Like all the RL frameworks, the environment model also provides the reward function. In Sullivan and Bosanac (2020), it is defined in three different ways: (1) by rewarding the spacecraft when close to a state of the reference trajectory at the same instant of time, measured from a common initial epoch (isochronous correspondence Szebehely, 1967); (2) by rewarding the spacecraft when its state matches the nearest state along the reference trajectory (normal correspondence Szebehely, 1967); (3) as in (1), but with an additional constraint on the angular variation of the thrust vector orientation. For the first and the last choice, the spacecraft has to track a moving target, while in the second case any point along the reference trajectory can be targeted. At a general level, the scalar reward functions are anyway defined as the weighted sum of the norm of the errors between the spacecraft state and the reference trajectory position and velocity, with an additional term penalizing changes in thrust direction (Sullivan & Bosanac, 2020), i.e.,

$$r_t(s_t, a_t) = -c_d |d_{ref}(t + \Delta t) - d_{sc}(t + \Delta t)| - c_v |v_{ref}(t + \Delta t) - v_{sc}(t + \Delta t)| - c_\theta \theta(t), \tag{8}$$

where $d_{ref}(t)$, $d_{sc}(t)$ are the position vectors, as well as $v_{ref}(t)$, $v_{sc}(t)$ are the velocity vectors, at a given time step t, for the reference and the spacecraft trajectory, respectively, $\theta(t)$ represents the angle between the current and the previous thrust vector, while c_d, c_v, and c_θ are penalizing weights. Numerical simulations are performed in the environment by starting from random initial conditions. The authors in Sullivan and Bosanac (2020) claim that the lowest tracking errors can be achieved with the second choice of the reward function, while the third choice produces more feasible maneuvers at the price of larger errors in spacecraft position and velocity.

A very similar approach can be found in LaFarge et al. (2021), whose main goal is to develop an autonomous onboard guidance close-loop controller for a low-thrust engine spacecraft transfers between Lyapunov orbits in the Earth–Moon system. The main difference consists in the choice of the reward function, which is assumed to be the Euclidean norm of the error vector, defined by the actual spacecraft position and velocity deviation from the nearest neighbor position and velocity, along the given reference trajectory path, at a given time step (i.e., a normal correspondence) (LaFarge et al., 2021). Moreover, in LaFarge et al. (2021) the problem about how to measure the nearness to the reference trajectory in a computationally efficient manner is addressed. Indeed, by sampling a sufficiently high number of states along the reference trajectory first, a searching problem arises when looking for the nearest state in the path. The authors in LaFarge et al. (2021) propose to adopt a K-dimensional tree (KD-Tree) data structure to reduce the search algorithm complexity.

In Federici, Scorsoglio et al. (2021) and Sullivan et al. (2021), low-thrust cislunar transfers between planar libration point orbits (LPOs) in Earth–Moon CR3BP problem are considered. To cope with the high dimensional state and actions spaces, the authors adopt a PPO based approach to implement a Multi-Objective Deep Reinforcement Learning (MODRL) technique for guidance law optimization, including constraints on the size of the policy updates. Being the reward a scalar function, multiple problem objectives, related to, e.g., path and terminal constraints, can be taken into account in the reward function as weighted penalty terms. The choice of such weights is crucial, since they represent the relative importance of the various terms. In Federici, Scorsoglio et al. (2021), a delayed reward function (i.e., a reward that is obtained at the end of each episode), which combines the minimum distance from the target orbit and the fuel consumption through some preliminary tuned weights, is proposed. On the other hand, in Sullivan et al. (2021), a more general Multi-Reward Proximal Policy Optimization (MRPPO) method is presented. It is a MODRL algorithm with the peculiarity of training multiple policies simultaneously, each with the correspondent reward weights scaling, thus reducing computational burden and training times. All the reward functions are the weighted difference of three objectives, given by the spacecraft position and velocity errors w.r.t. the target orbit, the flight time and the propellant mass consumption, each with distinct weights for representing distinct priorities. The state–action propagation data from the environment model are shared by the multiple policies to save training time. Each policy generates a distinct reward and learns how to maximize the long-term reward from its peculiar reward function. The generation of the episodes is performed by modifying the mass usage term. Once a suitable number of state–action pairs has been stored in the memory, they are used to evaluate the rewards (defined as in LaFarge et al. (2021), i.e., following a normal correspondence) for each policy and, hence, to train the deep neural networks that model the policies. The policy are consequently updated until a terminal condition is reached.

A completely different paradigm is discussed in Elliott et al. (2020). Indeed, it is known that the entire maneuvers design process is practically performed by a team of flight dynamics engineers. However, autonomous trajectory planning capabilities that somehow reproduce the human skills are indisputably desirable for modern spacecraft, in view of reducing the human effort and the operational costs, with, in addition, faster reactions towards hazards and modified mission goals (Elliott et al., 2020; LaFarge et al., 2021). In this context, an alternative and innovative approach can be found in Elliott et al. (2020), which consists in mimicking the flight dynamics experts reasoning to plan robust, efficient and safe maneuvers. However, it is a very difficult task for flight dynamics engineers to express all the goals and constraints they consider in maneuvers planning through a single mathematical relation. Inverse Reinforcement Learning (IRL) techniques (Ng & Russell, 2000) can provide a possible solution to learn the reward function embedded in the expert-designed maneuvers/policy. A common issue of IRL approaches is the ambiguity in

the uncovered reward function (Zhifei & Joo, 2012): indeed, there might exist various combinations of weights and features in the reward function that uncover the expert reward function, or, correspondingly, the expert policies might optimize different rewards.

To cope with this ambiguity issue, in Elliott et al. (2020), an IRL related apprenticeship learning is proposed to approximate the behavior of the maneuver planner expert, in the sense that, instead of learning the experts reward function, an estimation of a policy that produces the same trajectories produced by an expert policy is sought (Zhifei & Joo, 2012). More specifically, a Linear Quadratic Regulator (LQR) controller is used to define the expert policy as a simple case study for an effectiveness analysis of the apprenticeship method. The environment model for the spacecraft dynamics is again the CR3BP, for which a particularly relevant Near-Rectilinear Halo Orbit (NRHO) close to L_2 equilibrium point is set as reference trajectory.[3] The LQR controller computes the command inputs, and, hence, the commanded trajectories, for two scenarios, i.e., station-keeping and rendezvous (Elliott et al., 2020), for several initial conditions by perturbing the NHRO states. The IRL algorithm uses such a set of expert maneuvers and related trajectories to learn a policy that produces similar results to the LQR control policy. By doing so, the apprenticeship algorithm can also estimate the reward function optimized by the apprentice policy, which, although possibly different from the actual expert reward function, allows to guess the priorities of the LQR control strategy. More specifically, the apprenticeship learning algorithm proposed in Elliott et al. (2020) estimates the weights of a reward function assumed to be a linear-weighted combination of some feature vector components

$$r(s,a) = w^T \phi(s,a), \tag{9}$$

where ϕ is the feature vector and w the scaling weights vector. In order to cope with the above-mentioned ambiguity issue, the authors in Elliott et al. (2020) propose to adopt the maximum margin IRL approach (Abbeel & Ng, 2004), which recovers a set of reward weights corresponding to the expert policy with a greater expected value than the expected value of other, non-expert policies.

4.3. Interplanetary trajectory design

As in the case of cislunar transfer applications, the literature reports examples of interplanetary missions with small- or micro- spacecraft (e.g., the cubesats used in the Mars Cube One mission by NASA Asmar & Matousek, 2014, in the Miniaturized Asteroid Remote Geophysical Observer by ESA Walker, Koschny, Bramanti, & Carnelli, 2017, and in Lunar missions Bosanac, Cox, Howell, & Folta, 2018), which are all based on low-thrust electric propulsion technology (Zavoli & Federici, 2021) to guarantee reduced costs and development time. However, interplanetary low-thrust trajectories are characterized by long thrusting periods, usually in a not well-known environment, which makes the spacecraft guidance and navigation problem more challenging for traditional optimal control methods. As matter of fact, conventional optimization approaches turn out to be computationally expensive and subject to fall in local minima (Miller et al., 2020). RL provides an interesting alternative for the robust design of interplanetary trajectories, as shown in recent articles (Miller et al., 2020; Zavoli & Federici, 2021).

Both in Zavoli and Federici (2021) and in Miller et al. (2020), the authors discuss the usage of the PPO algorithm for the robust design of a low-thrust minimum-propellant trajectory which leads, in presence of structured and unstructured uncertainties, the spacecraft from the Earth to Mars. In Zavoli and Federici (2021), the RL state variables selected are the mass of the spacecraft together with its inertial position and velocity with respect to the Sun, discretized into N time steps along the whole mission. In the time interval between steps, the trajectory is approximated through a ballistic arc initialized by an impulsive velocity variation Δv (Sims & Flanagan, 2000), which is actually the commanded action a_t at each time step. The deterministic model provided by Sims and Flanagan (2000) is updated in order to include the different sources of uncertainty, related to unmodeled dynamics (unmodeled accelerations and uncertain parameters), measurements noise, and control execution errors (thrust orientation and modulation), including the possible occurrence of a Missed Thrust Event (MTE), with variable duration. All the uncertainties and measurements noise are, for the sake of simplicity, assumed to be represented by an additive Gaussian noise. Control execution errors are modeled as perturbations in the commanded vector Δv orientation and modulus. Vice-versa, the MTE is depicted as a null thrust event, assumed to occur only once during the mission, at a random time step sampled from a uniform probability distribution in $(0, N]$ at the beginning of each episode, and to last a given finite number of steps.

The above described RL environment model, implemented through an orbital mechanics simulator, is completed by a reward function definition. The goal of the optimization procedure is to maximize the (expected) final mass of the spacecraft, while ensuring the compliance with terminal rendezvous constraints on position and velocity. For this reason, the reward r_t obtained at time t is defined as (Zavoli & Federici, 2021)

$$r_t = -\mu_t - \lambda_{e_u} e_{u,t-1} - \lambda_{e_s} \max\{0, e_{s,t} - \epsilon\}, \tag{10}$$

with

$$\mu_t = \begin{cases} (m_{t-1} - m_t)/\bar{m} & \text{if} \quad t < N, \\ (m_{N-1} - m_f)/\bar{m} & \text{if} \quad t = N, \end{cases} \tag{11}$$

$$e_{u,t} = \max\{0, (|a_t| - \Delta v_{\max,t})/\bar{v}\}, \tag{12}$$

$$e_{t,k} = \begin{cases} 0 & \text{if} \quad t < N, \\ \max\{e_r, e_v\} & \text{if} \quad t = N, \end{cases} \tag{13}$$

where μ_t is the cost function, that is, the (nondimensional) consumed propellant mass at time t, $e_{u,t}$ is the (nondimensional) violation of the constraint on the maximum Δv magnitude, and $e_{t,N}$ is the maximum value between the final position error e_r and velocity error e_v. Moreover, ϵ is a tolerance on terminal constraint violation, λ_{e_u} and λ_{e_s} and are weighting factors. The episodes end when some termination conditions are reached.

Almost similarly, in Miller et al. (2020), the fixed-length Earth to Mars reference trajectories are considered, sampled into a finite number of states that are used, together with the corresponding actions and rewards, to train the RL agent. Simulations are generated by initial conditions randomly chosen from a set of post-launch vectors spread over a period of 2 weeks. The reward is computed based a weighted sum of errors, according to a structure similar to LaFarge et al. (2021) (i.e., with a normal correspondence). However, the authors admit the results obtained for the RL trained agent are not yet comparable with the ones provided by the interplanetary trajectory optimization tool used as a benchmark.

5. Design of spacecraft attitude control systems

The problem of orienting a spacecraft toward a specific target point has been addressed since a long ago. As a rule, satellites require an Attitude Determination System (ADS) and an Attitude Control System (ACS) to establish, and then govern their orientation after the launch (Wertz, 2012). The satellite attitude can be determined by using orbit propagators, models (e.g., the International Geomagnetic Reference Field (IGRF) for the Earth's main magnetic field), and different types of on-board sensors (e.g., sun sensors, magnetometers, and earth sensors). In ADS systems, neural networks can be used to implement

[3] Such a periodic orbit has low perilune and high apolune, and is nearly polar. It will be adopted for the future Lunar Orbital Platform Gateway (Whitley & Martinez, 2016).

orbit propagators, see Yadava, Hosangadi, Krishna, Paliwal, and Jain (2018), using GPS data.

Once the attitude of the spacecraft has been estimated, the required controlled torque has to be computed to track the reference attitude. And here, RL solutions can come into play. There exists two main types of ACS, depending on the actuators employed, namely momentum management and momentum exchange actuators (Vedant, Allison, West, & Ghosh, 2019). The former category includes attitude control thrusters and magnetic torque coils, whereas the latter is represented by reaction wheels and control moment gyroscopes (Vedant et al., 2019). Conventional feedback control methods consist of two nested loops, the outer dedicated to local trajectory optimization, while the inner to a state-feedback tracking control. However, other than the difficulties of a full state-feedback, such methods go in trouble as far as long-term planning and adaptive behavior are concerned. According to the authors of Dong, Zhao, and Yang (2021), Elkins, Sood, and Rumpf (2020), Gao, Zhang, Li, and Gao (2020), Su, Wu, and Zhao (2019), Vedant et al. (2019) and Yadava et al. (2018), the chosen RL approaches outperform, both in software and hardware-in-the-loop simulation, more traditional control methods (e.g., PID controller) in terms of industry performance specifications, robustness and feasibility requirements over the addressed theater of operations, meaning, more specifically, improvements in terms of industry-standard pointing accuracy (Elkins et al., 2020; Su et al., 2019), no susceptibility to noise in sensors (Yadava et al., 2018) or to disturbances (Dong et al., 2021), achievable reward (Vedant et al., 2019), control time and energy consumption (Dong et al., 2021; Gao et al., 2020).

In Elkins et al. (2020), the problem of orienting the spacecraft to a reference attitude by means of a discretized set of torque values for each of the principal body-frame axes is considered. The authors propose to use a variant of the Proximal Policy Optimization algorithm for a discrete control scenario, named Twin Delayed Deep Deterministic Policy Gradient (TD3) method, to train a neural attitude controller. Two separate loss functions for the value and policy functions are trained through minibatches of states, actions, and rewards, generated by the interaction, for a number of episodes, between the agent and the (simulated) environment (see Fig. 5). The environment model implements the rigid body model of the Lockheed Martin Corporation LM50 satellite. The satellite attitude is expressed through the error quaternions (Wertz, 2012), i.e., the difference between the current body-fixed reference frame orientation and the desired one. The attitude controller (agent) must perform large-angle slews and stabilize the spacecraft at rest about the desired orientation, within industry standard pointing accuracy, until the end of the episode. As a lesson learned, the authors in Elkins et al. (2020) claim that an improvement in the agent training can be observed if the state vector is normalized (to make the input data comparable) and built using relative (instead of absolute) representations, as it occurs when choosing the normalized error quaternions representation. Each episode is initialized with the spacecraft at rest and an initial error quaternion vector, sampled from a uniform distribution. Then, the error quaternion is propagated through time by integrating the quaternion kinematics. Moreover, the following assumptions are made for the attitude controller (agent) design: (i) the attitude controller can provide near-impulsive torques of three magnitudes only; (ii) the control action can be exerted on only one body-axis at time; (iii) the agent selects an action to rotate the spacecraft once every 20 time steps (i.e., a variant of frameskipping is implemented).[4] For both the value and policy networks, the reward function is defined as follows

$$r = \begin{cases} r_a + 9 & \text{if } \phi \le 0.25, \\ r_a & \text{otherwise,} \end{cases} \tag{14}$$

[4] RL algorithms are usually tested by playing an Atari arcade game, and frameskipping consists in taking actions at a given number of frames only Braylan, Hollenbeck, Meyerson, and Miikkulainen (2015).

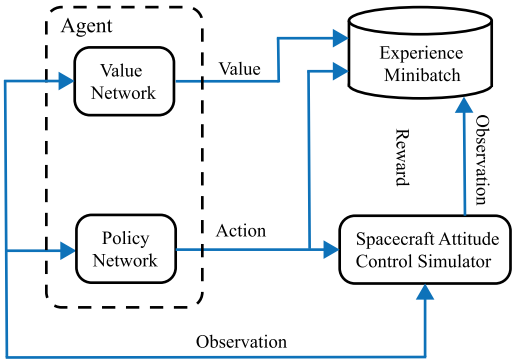

Fig. 5. Diagram of agent-environment interaction for ACS applications and with minibatches of experience used in agent training.
Source: Adopted and modified from Elkins et al. (2020).

where ϕ is the angular error and r_a is an intermediate reward given by

$$r_a = \begin{cases} \exp\left(\frac{-\phi}{0.28\pi}\right), & q_{s,t} > q_{s,t-1}, \\ \exp\left(\frac{-\phi}{0.28\pi}\right) - 1, & q_{s,t} \le q_{s,t-1}, \end{cases} \tag{15}$$

where $q_{s,t}$ is the scalar part of the error quaternion at the current time step and $q_{s,t-1}$ is the scalar part of the error quaternion at the previous time step.

Similarly to Elkins et al. (2020), the authors in Vedant et al. (2019) propose an RL method for the design of the ACS by using the PPO algorithm. The environment model implements the rigid body model of the spacecraft as well, and the state space is defined by the error quaternion vectors. Moreover, in order to obtain a robust ACS design, suitable for a variety of missions and spacecraft, the inertia of the spacecraft and the peak attitude control torque are assumed to be uncertain and randomly chosen within a design polytope before each simulation (Vedant et al., 2019). As a result, the attitude control problem is modeled through a POMDP, which, anyway, does not imply any particular modifications to the RL algorithm. The ACS problem is solved by assuming a continuous action space, for both the momentum management and momentum exchange scenario.

Still in Vedant et al. (2019), constraints on control signal magnitude and limitations on control effort are taken into account by means of penalties in the reward function as follows

$$r = -\alpha_q q_{er} - \alpha_\omega \|\omega_e\|_2 - c, \tag{16}$$

where q_{er} is the absolute value of the last error quaternion component minus 1, ω_e is connected to the angular velocity error, α_q and α_ω are weights for the system response, and c is a conditional reward representing the control constraints. Since the attitude and the angular velocity dynamics are coupled, a curriculum learning-based method is used to facilitate the RL training phase. Roughly speaking, the agent is first trained with an environment initial state chosen close to the target state, and then a hardness parameter is linearly increased through the episodes to reinforce the problem difficulty. Remarkably, the RL method is compared to a classical tuned quaternion rate feedback, showing slightly better attitude control performance in all the test cases (Vedant et al., 2019).

Analogous good results are obtained in Gao et al. (2020), Su et al. (2019) and Yadava et al. (2018), where RL approaches produce better performance with respect to PID controllers in terms of rise time and susceptibility to sensors noise. In particular, while in Yadava et al. (2018) an orientation tracking problem is addressed, in Su et al. (2019) and Gao et al. (2020) the stabilization of the satellite attitude, after a perturbation, is considered instead.

More specifically, in Yadava et al. (2018), the ACS consists of two neural networks, one for the angular rate control and one for the orientation control. The angular rate controller intervenes first to guarantee an angular rate below a certain threshold. As soon as the spacecraft angular rate falls below that threshold, the orientation controller is activated to achieve the desired attitude angles. Both controllers are trained via the Deep Deterministic Policy Gradient (DDPG) method (Lillicrap et al., 2015), a variant of the Actor–Critic algorithm. The reward for the angular rate (the attitude) controller is a positive number if the current angular rate error value (the current error quaternion value) is closer to zero than the previous error value. Similarly, in Gao et al. (2020) and Su et al. (2019), the DDPG algorithm is employed. As for Su et al. (2019), it is worth noting the inclusion of actuator faults in the simulated environment and the usage of the simulated annealing algorithm (Yu, Redi, Hidayat, & Wibowo, 2017) to reduce the training time without compromising robustness, thus improving the computational efficiency. In Gao et al. (2020), even though the simulation results show the feasibility of applying RL methods to ACS problem, the authors recognize the issue concerning the stability issue of the agent-environment system, see also paragraph 9.2.

A complete discussion on practical ACS design problems can be found in Dong et al. (2021), where the presence of some state constraints is also addressed. In particular, the attitude angle and the angular velocity limits are considered. Indeed, there exist some particular payloads, such as infrared telescopes, that cannot be pointed towards direct sunlight during the satellite orientation. Moreover, the attitude maneuvers should keep the angular velocity below the maximum angular velocity measured by the on-board gyros, for safety reasons. To this aim, a novel RL algorithm is proposed to take into account the state constraints (angular and angular velocity limits) by introducing suitable barrier functions (Baccari, Tipaldi, Iannelli, & Vasca, 2012; Baccari, Vasca, Tipaldi, & Iannelli, 2017) into the cost function to encode the information of attitude forbidden zones and angular-velocity limits. The cost function also includes the weighted sum of angular position and angular velocity errors. More specifically, a critic-only neural network approximates the optimal cost function and control policy (Dong et al., 2021). Both real time and past measurements data are concurrently used to update the NN weights during the online learning process. The authors also emphasize the reduced computational complexity of the updating law of the NN weights when compared with MPC methods. Remarkably, the effectiveness of the approach is demonstrated through an hardware-in-the-loop testbed composed of a real-time simulation computer and a turntable, together with a real time controller implemented in a ARM-based micro-controller.

6. Spacecraft guidance and maneuvers in rendezvous and docking scenarios

Contemporary space missions, such as Restore-L (Vavrina et al., 2019) and DARPA Robotic Space Servicer (Sullivan, Kelm, Roesler, & Henshaw, 2015), have been dedicated to satellite on-orbit servicing, inspection, assembly, and orbital debris removal, which require strict spacecraft autonomous rendezvous performance. Indeed, many constraints and challenges have to be faced, including moving and possibly uncooperative targets (e.g., asteroids, abandoned satellites, debris), collisions avoidance, uncertain environment dynamics, actuators faults, and so on. Rendezvous missions are commonly divided into 5 operational stages: launch, phasing, far range rendezvous, close range rendezvous, and mating (Fehse, 2003). Nevertheless, the authors in Wang, Wang, Chen, and Xie (2020) refer to a simplified 3 phases rendezvous mission decomposition: remote guidance, short-range search, proximity and docking. Remote guidance requires the assistance by the ground-station, while the short-range search starts when the distance between the chaser and the target falls between 10 and 100 km (Wang et al., 2020). In this phase, the chaser synchronizes its translational and rotational motions with those of the target. Finally, a docking or capturing phase follows (Hovell & Ulrich, 2021). There are several recent papers dedicated to this topic, but some of them have to be particularly mentioned for the novelty of the contribution and the results presented.

In Oestreich, Linares, and Gondhalekar (2021), the RL approach is applied to the translational and attitude control design for a 6-DOF docking maneuvers problem, with possibly rotating targets along their spin axes. The environment model consists of a nonlinear 6-DOF system containing the translational and rotational motion of the chaser and the rotational motion of the target, assumed to spin around a fixed axis. The initial conditions of each episode are randomly chosen from a subset of the state space (training range). The docking policy is obtained through PPO, where the policy function and the state-value function (advantage function) are represented by feedforward neural networks, trained simultaneously. More specifically, the authors in Oestreich et al. (2021) propose to tune the network weights by taking into account a desired Kullback–Leibler (KL) divergence value when updating the policy (Kullback & Leibler, 1951). The chosen cost function is a weighted sum of state tracking errors and control effort, with penalties on collisions, and bonus on successful docks. The translational acceleration error is evaluated with respect to a reference acceleration computed by a LQR controller, designed offline to provide trajectories with a fixed time duration. Such a trick allows enforcing in the PPO development hard time constraints on the dock trajectory. A comparison with conventional optimal control techniques by means of a realistic simulator proves the effectiveness of the method.

An analogous accurate environment model, with the formalism of dual-quaternions, can be found in Hu, Yang, Dong, and Zhao (2021). Even though the dynamics model is similar, the proposed control approach is completely different. By assuming a continuous torque action, the authors in Hu et al. (2021) solve an optimal control problem in presence of field of view and approach corridor constraints, whose cost function is defined as the integral of a reward function. Such cost function is approximated through a neural network whose weights are determined by using an actor–critic based RL technique. Once these weights have been estimated, the control action is given by solving the associated Hamiltonian equation. The chosen reward function takes into account both desired and undesired states, depending of dual quaternions and angular velocity error and the chaser distance from the boundaries of the field-of-view cone and of the approach corridor cone. As initial policy prior to the training, a PD-like controller is chosen for its simplicity and to guarantee asymptotic convergence of the error variables. Remarkably, a Lyapunov analysis is also performed to check the closed-loop system asymptotic stability.

An RL approach based on PPO algorithm to the design of the autonomous guidance control of a spacecraft chaser in a terminal rendezvous maneuver with a non-cooperative target can be also found in Federici, Benedikter, and Zavoli (2021a). The control task starts when the chaser is sufficiently close to the target, and it is performed, until the target is reached, through a series of (saturated) impulses taken at discrete time-steps. Moreover, the environment model consists of the linearized dynamics of the chaser motion with respect to the target, described via the Hill–Clohessy–Wiltshire equations (Clohessy & Wiltshire, 1960). With the constraints of being in the approach corridor and of having the target always in the visibility cone, the chaser reference trajectory has to lead it from an initial condition to a prescribed final one. To account for the stochastic effects, in Federici et al. (2021a) the initial condition is allowed to vary randomly around the nominal values, and the chaser dynamics includes uncertainties by adding a random velocity perturbation at each time-step. A comparison of the RL algorithm with another (supervised) deep learning method, named Behavioral Cloning (BC) approach, has been also implemented in Federici et al. (2021a) by means of Monte Carlo simulations. RL-designed policies appear to be more robust, with, however, longer training times and greater susceptibility to the choice of the optimization hyperparameters (e.g., learning rate and penalty weights). In the

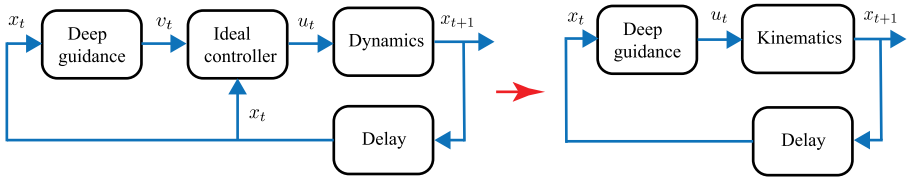

Fig. 6. Deep guidance with an ideal controller for training purposes in simulation.
Source: Adopted and modified from Hovell and Ulrich (2021).

authors' opinion, a good compromise seems to be the mix of the two methods, with the BC method used to provide a promising starting point for the RL algorithm in order to reduce the training times.

An important issue related to the simulation-to-reality gap is discussed in Hovell and Ulrich (2021). Indeed, as many authors declare, RL policies are trained through interactions with a simulated, although accurate, environment, being the training on-board the spacecraft unfeasible due to, e.g., fuel, time and computation constraints. Although good results can be obtained by training RL agents via simulations, it may happen that the resulting RL policies do not generalize well when operated in the real environment (Kober, Bagnell, & Peters, 2013). To address this point, in Hovell and Ulrich (2021), unlike all the others approaches in the field, where both the guidance and the control logic are designed through RL algorithms, it is proposed to use deep RL to train a guidance policy only, whose outputs are intended to be the reference inputs for a conventional (inner) controller. Indeed, RL algorithm performs well when a guidance policy has to be discovered at a higher level, while classical control strategies can be adopted to guarantee formal robustness to system uncertainties, and, hence, able to deal with discrepancies between simulation and reality. The combination of deep RL for guidance with conventional control approaches, named deep guidance strategy (Hovell & Ulrich, 2021), can be deployed to the real system without further fine-tuning. In Fig. 6, the adopted mixed scheme is represented, where x_t is the current state, v_t is the desired velocity, and u_t the control action applied to the environmental dynamics. Note that when the deep guidance is trained, an ideal controller is adopted, to make it independent of the specific controller model employed. The chaser motion is approximated via a double integrator dynamics, and the reward function, at a given time step, is defined as $r_t = \|K(|e_{t-1}| - |e_t|)\| - c\frac{\|v_t - v_{ref}\|}{\|e_t\|+\eta}$, where e_t is the error between the desired state and the current state, c is a penalty weight on the velocity error, and v_{ref} is the velocity of the docking point. Moreover, if the distance between the chaser and the target is less than a given threshold, an additional collision penalty can be considered. An experimental campaign on a small-scale planar chaser and target docking system has proven the ability of the proposed approach to solve the simulation-to-reality problem.

7. Constellation orbital control

Favored by recent advances in satellite miniaturization technologies, satellite constellations are becoming a new trend in spacecraft design thanks to the fact that they can offer higher performance, lower development costs, higher flexibility, better fault tolerance, and enhanced reconfigurability (Shi, Ren, & Du, 2020; Silvestrini & Lavagna, 2021; Tipaldi & Glielmo, 2018; Yang, Zhang and Gao, 2021). They can also enable new types of scientific applications (e.g., distributed sensing for solar and extra-terrestrial observatories Joshi & Padhi, 2014), and can also provide infrastructures for high-quality services (e.g., high-bandwidth and low-latency Internet services with global coverage Liu, Zhao, Xin, Su, & Ou, 2021). Satellite constellations are composed of small spacecraft collaborating all together to achieve shared mission objectives. For a constellation to accomplish its mission objectives, relative phasing mechanisms are required to actively maintain specific configurations and precise relative positions despite orbital perturbations (Joshi & Padhi, 2014; Smith et al., 2021).

In Smith et al. (2021), the applicability of RL approaches to satellite phasing problems is investigated. In particular, in such work, the authors address the propellantless planar phasing maneuver of three spacecraft operating in the same coplanar orbit. Generally speaking, the propellantless planar phasing of multiple spacecraft is a common maneuver that occurs to establish or change the relative angle between any two spacecraft operating within the same orbital plane by using accelerations generated from interactions with the space environment, e.g., arising from aerodynamic drags (Smith, Capon, & Brown, 2019). In Smith et al. (2021), three different cross-sectional drag configurations are considered for each satellite, which imply an action space $\mathcal{A}$ composed of 27 elements. The authors evaluate the performance of the A2C and PPO algorithms in solving such propellantless planar phasing problem, while orbit propagators are used as environment model. More specifically, both the A2C and PPO RL agents are trained by using a surrogate model of a high-fidelity numerical orbit propagator (with the aim of reducing the RL agent training times), while the trained A2C and PPO policies are tested in an environment simulated by the high-fidelity orbit propagator itself. As highlighted in Smith et al. (2021), simulating the spacecraft dynamics with computationally intensive orbit propagators is actually unattractive for the RL agent training process in constellation orbital control applications since the time taken to train an agent is sensitive to the latency of the orbit propagation. And this remark is even more important when the RL framework has to be instantiated for new constellation orbital control problems (e.g., the goodness of the defined reward function and the training setup has to be assessed). The low-latency surrogate model proposed in Smith et al. (2021) is four orders of magnitude faster than the high-fidelity numerical propagator, and this translates to much faster times taken to train the RL agent. The reward function is defined so that it has to provide incentive for the RL agent to separate its relative planar phase angle during the initial period of the formation maneuver, and then stabilize it within the later stages. In particular, the following reward function definition is defined in (Smith et al., 2021)

$$r = \Phi + \Phi^{(2-\frac{N}{30})}\Theta, \tag{17}$$

with

$$\Phi = \frac{1}{\sqrt{1 + E'_{\text{maol}} E_{\text{maol}}}},$$
$$\Theta = \frac{1}{\sqrt{1 + E'_{\text{sma}} E_{\text{sma}}}},$$

where E_{maol} provides the error between the desired and current relative phase angle between any two satellites of the constellation, E_{sma} is the error between the zeroed relative semi-major axes and the current relative semi-major axes of the satellites, and N is the number of days passed since the initial epoch of the orbit propagator (note that the maximum length of an episode is set to 30 days). Thanks to the reward definition (17), the RL agent is able to learn a policy establishing the desired relative phase angles during the first days of operations, and then minimizing the relative semi-major axes as N increases. For the A2C, the success criteria of the propellantless planar phasing of three satellites is achieved via simulation in 16 days, while 24 days are needed for the PPO. As stated by the authors in Smith et al. (2021), the purpose of that work is to gain some insight into solving satellite phasing

problems via RL-based solutions. Further developments are expected, such as assessing the limits of the maximum number of spacecraft that can be practically handled by using a centralized RL control approach (where the RL agent takes observations from each spacecraft, and subsequently sets the actions of all the spacecraft belonging to the constellation) and the inclusion of a time parameter into the reward function to minimize the maneuver time.

The scenario of collision-free phasing problems for constellations composed of several spacecraft (each of them with different target states) can challenge a centralized RL paradigm. In such a case, one can resort to a decentralized RL agent architecture, where each spacecraft can measure relative states of its neighboring satellites, predict their future trajectories, and perform its own control policy. However, an extensive training campaign with thousands of episodes can be needed to learn an appropriate distributed policy. In Silvestrini and Lavagna (2021), an hybrid and distributed solution based on both RL and classical optimal control methods is presented. More specifically, the trajectory generation and control actuation of each spacecraft is performed by a local MPC-based algorithm, which exploits the predicted trajectories of concurring agents (i.e., neighboring satellites). Such predicted trajectories are computed by a Long–Short Term Memory (LSTM) network (for the short-term horizon) and an IRL network (for the longer-term horizon). As suggested in Silvestrini and Lavagna (2021), the adoption of model-based RL approaches could sort out the limitations of both RL and more traditional control methods at the same time, e.g., by providing adaptive solutions able to cope with partially known environments and trained with a reasonable number of episodes.

8. On-board decision-making for spacecraft operations scheduling and rover path planning

In the upcoming space missions, the need of designing spacecraft with highly autonomous on-board decision-making capabilities is an emerging trend (Frost et al., 2010; Tipaldi & Glielmo, 2018). Such capabilities rely on the usage of automated planning tools to reduce the human-in-the-loop intervention, while dealing with uncertain and complex environments. On-board planning mechanisms for spacecraft mission operations would bring relevant benefits, such as the improvement of spacecraft availability and reliability and the reduction of costs in ground segment operations. A great opportunity for implementing highly autonomous on-board planning mechanism is offered by RL-based approaches (Harris et al., 2019; Oche et al., 2021). In principle, planning problems can be formulated using the constrained optimization framework (Ghallab, Nau, & Traverso, 2004). Due to the aforementioned high uncertainty and environment complexity, such problems may be difficult or impossible to solve using traditional tools. In the next two subsections, a comprehensive review on the latest developments of RL-based planning solutions in space missions is provided. In particular, the scheduling of spacecraft operations is addressed in Section 8.1. Instead, Section 8.2 deals with the optimal path planning of planetary rovers. In both cases, a description of the examined strategies will be provided, highlighting also major advantages and drawbacks.

8.1. On-board operations scheduling in space missions

In this paragraph, RL solutions for on-board operations scheduling services in Earth observation applications and Active Debris Removal (ADR) space missions are presented. Earth observation satellites (EOSs) are Earth-orbiting spacecraft with sensors used to collect imagery and measurements of the Earth's surface. They are widely employed for military reconnaissance, ocean monitoring, and disaster surveillance (Bianchessi, Cordeau, Desrosiers, Laporte, & Raymond, 2007). To meet the increasing demand of multi-user scenarios, efficient scheduling approaches for observation tasks are needed. The corresponding task scheduling problems can be expressed in terms

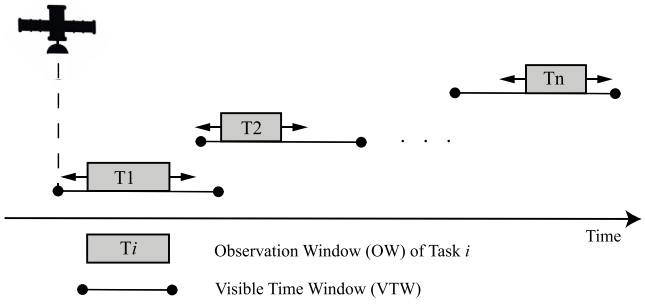

Fig. 7. EOS observation task scheduling.
Source: Adopted and modified from Huang et al. (2021).

of Visible Time Windows (VTWs), i.e., the period when a specific target is observable, and Observation Window (OW), i.e., the period when the actual observation is executed (Huang, Mu, Wu, Cui, & Duan, 2021; Wei, Chen, Chen, & Chen, 2021; Zhao, Wang, & Zheng, 2020). A visual representation of the observation task scheduling is reported in Fig. 7. The decision variables are the continuous variable representing the OW start time for a given target, and a binary variable representing whether the observation task is actually scheduled or not. The scheduling problem aims at allocating OWs within each VTW to maximize a given observation profit, subject to some constraints. That is, time window constraints require that each observation task is executed within the corresponding VTW, while overlap constraints avoid overlapping between two consecutive observations (by also taking into account switching maneuver times), and memory constraints limit the maximum usage of on-board data storage devices. Then, the objective function (and thus, the reward in the RL formulation) can be defined according to priority values assigned to each target. The EOS task scheduling problem is a well-known example of a NP-hard multi-objective combinatorial problem (Wolfe & Sorensen, 2000). Such problem could be effectively solved by DRL as it provides task scheduling solutions for combinatorial problems (Huang et al., 2021; Wang, Yang, Zhou, & Li, 2019).

The training process of RL agents and the performance of the resulting trained RL agents can be improved by applying clustering techniques for the targets to be observed. In Huang et al. (2021), a clustering of nearby targets prior the RL agent training is adopted, i.e., multiple close targets are observed simultaneously, and thus treated as a single, but longer, task within the same OW. This can improve the efficiency of the overall scheduling policy and the utilization of EOS on-board resources. The clustering model is built using the graph theory. Each observation task is represented as a vertex of an undirected graph, and each clique of the graph defines a clustering task. An off-policy DDPG algorithm (Lillicrap et al., 2015) is implemented via an actor–critic structure with deep neural networks. The objective function of the scheduling problem is employed as reward, while constraint violations are checked after executing each DDPG episode. Simulation studies with typical orbital elements of a low orbit observation satellite show how the DDPG policy with a prior clustering outperforms traditional heuristic approaches. The scheduling is executed in a batch-wise manner, i.e., if the target list changes a re-training is needed. A viable solution to this issue is provided in Wang et al. (2019), where the dynamic real-time scheduling of image satellites is presented. The observation tasks arrive dynamically with an associated reward, unknown prior to their arrival. Then, each task can be accepted or rejected according to a policy as long as on-board data storage and timing constraints are fulfilled. The dynamical scheduling is formulated as a Dynamic and Stochastic Knapsack Problem (Kleywegt & Papastavrou, 1998), and solved by means of DRL methods. The RL scheduling policy is implemented by a three-layer fully connected neural network. Arrival time and profit of the new task along with remaining storage and time

are the inputs of the network, while the outputs are the expected profit and the decision policy of whether accepting or not the new task. The RL agent training is performed using the Asynchronous Advantage Actor–Critic algorithm (A3C) (Babaeizadeh, Frosio, Tyree, Clemons, & Kautz, 2017). The scheduling performance values are assessed through simulations, while comparisons with other heuristic approaches show improvements in terms of both total profit and execution times.

Agile Earth Observing Satellites (AEOSs) are the next generation of EOSs. Thanks to their precise and fast attitude control systems, AEOSs can experience longer VTWs for each target, thus allowing more observations within an orbit (Lemaıtre, Verfaillie, Jouhaud, Lachiver, & Bataille, 2002; Wei et al., 2021; Zhao et al., 2020). Each attitude change has a bidirectional influence on the current and future observation tasks, resulting in a time-dependent scheduling problem whose complexity is increased if compared with EOS systems (Lemaıtre et al., 2002). In Wei et al. (2021) and Zhao et al. (2020), DRL is used to solve the AEOS scheduling problem. A multi-objective optimization problem is addressed in both cases. In Zhao et al. (2020), the total observation profit and the image quality are maximized. These are two competing objectives since the former increases as the number of scheduled targets increases, while the latter limits the number of observations inside a given VTW. A two-phase algorithm is proposed. During the first phase, a recurrent neural network is trained to learn a stochastic scheduling policy aimed at defining a proper set of observation tasks. As for the second phase, a DDPG algorithm learns the optimal selection of OWs to maximize the image quality. Experimental studies on synthetic datasets compare the proposed two-phase approach with other non-RL heuristic methods. In general, the heuristics were outperformed especially in the OW selection phase. However, the heuristic methods provide faster training times. This constitutes an important limitation in Zhao et al. (2020), as a re-training would be necessary if some task attributes change. A more efficient approach is proposed in Wei et al. (2021), where the double objective encodes the failure rate of observation requests and the timeliness of the scheduled requests. The key idea is to divide the original problem into two sub-problems solved using DRL. Each sub-problem is defined according to each single objective. The trained network parameters are transferred between the sub-problems (Taylor & Stone, 2009), so that both convergence speed and computational efficiency are enhanced. The actor–critic algorithm trains the neural networks employed in each sub-problem. Simulation experiments highlight better performance values and accelerated training speed if compared with standard heuristic tools, thus allowing faster re-training if needed. Nevertheless, the practical application of Wei et al. (2021) is highly limited by the simplifications made, e.g., only point targets are considered.

The EOS scheduling problem becomes more complex when multi-satellite systems (i.e., satellite constellations) are considered. The authors of Wang, Li, Jing, Wang, and Chen (2011) present a first attempt in tackling the multi-satellite cooperative task scheduling via RL. A set of satellites is considered, each of them with different on-board resource capabilities (e.g., memory, energy). Observation requests are assumed to occur randomly. Then, based on the available individual resources, the proposed approach aims at allocating each task to a specific satellite. A hybrid learning is developed to improve the cooperative RL policies of each satellite via a genetic approach. Training speed is increased thanks to a transfer learning approach (Taylor & Stone, 2009). Although good performance values are obtained in terms of cooperative optimization, the solution is far from being practical. In fact, the scheduling policy defines only whether a task should be accepted (and by which satellite) or not, without defining the OW for the actual task execution.

Besides Earth observation applications, another area of great interest is the observation of small celestial bodies, e.g., asteroids. Currently, the observation task scheduling for small celestial bodies relies on human expertise, resulting in a highly time-consuming effort (Gaskell et al., 2008). Such approach may be unfeasible for near future missions, where the exploration of an increasing number of deep-space celestial bodies will be required. In such cases, the work in Chan and Aghamohammadi (2019) sheds the lights on the advantages offered by DRL to accomplish an autonomous planning of such operations. In particular, the goal is to plan the best trajectories allowing a spacecraft to explore the small body by providing high-quality images of its surface. First, a detailed and realistic POMDP is built to formalize the planning problem. The state space is defined by including the position, attitude, velocity, and acceleration of both the body and spacecraft. Available on-board memory, camera modes (on/off), downlink modes (on/off), body trajectory, and nearest star position estimating the lighting conditions are also included in the state space formulation. The action space is constituted by the thruster commands and camera controls, i.e., taking or downlinking images to free-up memory. To limit the problem complexity, instead of considering continuous thrust values, thrusters can only be turned on or off. Different constraints are also added to the action space to make the RL formulation closer to the real application. The reward model encourages high quality images, while penalizing the use of thrusters. The angles between the surface's facet being examined and the spacecraft, and between the facet and incoming starlight provide a measure of the image quality. The optimal trajectories are planned using a policy-gradient approach with a deep policy network, i.e., the REINFORCE algorithm. Simulation studies are conducted on a custom-developed simulator with real asteroid models. The resulting REINFORCE policy outperforms three different benchmarking algorithms, showing promising potential in the adoption of RL-based approaches for solving autonomous planning problems in deep-space exploration missions.

To conclude this paragraph, we discuss the adoption of RL-based solutions in ADR space missions. The growing number of spatial debris, such as abandoned spacecraft and non-functional satellites, represents a potential flaw for future space missions, especially if operated in low Earth orbits (Liou & Johnson, 2006). ADR is a mitigation measure for the space debris problem (Forshaw et al., 2020). It is a newly developed process conducted through proper spacecraft, namely orbital transfer vehicles, having the objective of dragging debris into reentry orbits. To reduce costs and increase ADR mission effectiveness, an efficient operations planning is necessary to remove multiple debris within a single ADR mission. Since debris move according to their orbits, the debris removal planning problem of ADR missions is an example of a time-dependent Traveling Salesman Problem, whose complexity explodes as the number of debris objects increases (Cerf, 2012). Machine learning approaches provide effective tools in tackling such a complex problem (Viavattene et al., 2022; Yang, Hou, Hu, Liu and Pan, 2020). And, the planning of ADR space missions particularly fits with RL approaches, as shown for the first time in Yang, Hou et al. (2020). In particular, given a set of debris to be removed, the goal is to choose a subset of debris and their sequence in order to maximize a certain reward, while fulfilling maximum time and propulsion budget constraints. In Yang, Hou et al. (2020), priority values are given to each debris, e.g., based on their danger level. The state vector includes the number of debris left to remove, the total impulse propulsion left, the total mission time left, and a binary flag for each debris representing whether it has been removed or not. The action vector identifies the next debris to remove, the reward function is the sum of the removed debris' priorities, while the RL environment model is deterministic. Then, the best removal sequence is computed by using a modified version of the AlphaGo Zero algorithm (Silver et al., 2017), which is based on a policy iteration approach. In particular, a search tree is established for optimization, and an Upper Confidence bound Tree (UCT) search algorithm is developed for the proposed RL scheme. The proposed algorithm uses a deep neural network for evaluating the state-value and policy of new tree nodes. Such neural network assists the selection and expansion procedures of new tree nodes, thus enhancing the exploration capabilities of the proposed UCT search algorithm. When no propulsion or time is left, a terminal state of the search tree

is reached. Finally, experiments using the real data of the Iridium 33 debris cloud demonstrate fast convergence to the optimal scheduling solution. However, the developed method still needs improvement, e.g., to demonstrate its ability to scale up the problem size.

8.2. Planetary rover path planning

The exploration of planets surfaces is mainly performed by means of appropriate vehicles deployed on them Quadrelli et al. (2015). These planetary vehicles, such as the rovers currently exploring Mars, face substantial difficulties both in long- and short-range navigation (Pflueger, Agha, & Sukhatme, 2019; Yan, Xiao, Guo, Bai, & Zheng, 2019). While the high delay in round-trip communications with Earth (e.g., up to 24 h for Mars) restricts long-range navigation planning (Pflueger et al., 2019), the unknown unstructured environment (in which rovers operate) also affects the short-range exploration performances (Yan et al., 2019).

In principle, effective long-range path planning can be achieved using orbital imagery as they provide wider views than those of rover cameras. Based on these images, traditional long-range planners typically use complex heuristic rules or embed terrain features with exhaustive and time-consuming labeling (supervised-based) processes, thus requiring large amounts of both images and paths. Alternatively, in Pflueger et al. (2019) an IRL approach finds suitable value functions based on a single human expert-designed path. Given a specific terrain map, the rover $x - y$ coordinates within the grid constitutes the state space, while the action space includes the movements towards the 8 adjacent grid boxes or stay in place. Orbital image data, target point data as one-hot goal map, and any other relevant terrain data (e.g., elevation maps) are the inputs of a convolutional network implementing the reward function to be computed via the IRL. A value iteration module provides incremental estimations of both the state-value and action-value functions based on the current reward network. Then, the difference between the expert-designed actions and those obtained using the estimated action-value function drives the reward network training. Thus, the IRL algorithm is forced to learn the reward function embedded in the expert-designed path. The obtained value function determines optimal paths starting from any location on the specific terrain map as well as it can be used to evaluate the value of a landing site. Experimental validations using synthetic datasets based on real Mars images show how the determined paths avoid unfavorable areas as the expert-designed path does on the training route. However, it can be noted that such method does not provide a fully autonomous solution as it requires re-training if the terrain map changes. Thus, a continuous interaction with the ground segment is still needed. Also, due to the low resolution of orbital images, such approach should be used together with a more accurate short range planning, which highly depends on the knowledge of the complex planetary surface. In this regard, single vehicle performance values can be enhanced by collaborative behaviors between two or more vehicles, with the aim of gathering/sharing more information about their operational environment (and thus, improving surface awareness and short-range planning). In Yan et al. (2019), a deep-RL algorithm enables self-learning cooperative detection between two vehicles exploring an unknown surface. Each vehicle is equipped with LIDAR sensors and visual cameras to explore the area and detect obstacles. The area is divided into a grid, while the distance and angle between the vehicle and each grid cell, the distance between the two vehicles, and the detection coverage rate define the state. For each vehicle, the action space consists of the angular and linear velocity commands, while the reward function encodes three objectives: (1) maximizing the overall coverage rate in the shortest time; (2) avoiding collision with the other vehicle or obstacles by defining a minimum safety distance; (3) maintaining communication without exceeding a maximum distance with the other vehicle. For the training process, a conventional deep Q learning algorithm is employed. The approximation of the optimal action-value function Q^* is achieved through a deep Q-Network with convolutional layers to process laser, map, and motion data. The collaborative experience is implemented by letting the vehicles to store the detecting information in a same shared dataset through communication. Simulations show good performance values in learning effective strategies to detect unknown areas quickly and without collisions. However, such approach would fail in presence of dynamic environments, where surfaces may change after the detection is completed.

Small celestial bodies and asteroids also play a significant role in terms of scientific relevance, with space missions requiring effective and accurate exploration (Dietze et al., 2010; Hockman & Pavone, 2019; Jiang, Zeng, Guzzetti, & You, 2020). Due to their highly irregular surfaces and reduced gravity fields, the exploration with conventional vehicles such as wheeled rovers may be difficult. Hopping vehicles constitute a valid alternative due to their capabilities in traversing long distances over rough terrains with limited energy. An example is given by the hoppers currently employed on the Ryugu asteroid (Dietze et al., 2010). Unlike traditional rovers, hopping systems lack of continuous interaction with their environment and full control over trajectories. Moreover, given the high complexity and uncertainty of their operational environment, planning hopping sequences is a highly challenging task. Existing studies developed planning strategies by considering simplified and deterministic dynamics for hopping rover path planning, e.g., smooth and spherical asteroids (Bellerose & Scheeres, 2008). Instead, in Hockman and Pavone (2019) the authors introduce uncertainty in the gravity field, control accuracy, and rover position. Then, RL is used to deal with such uncertainty and provides a sequence of consecutive hops to reach a target area in the minimum amount of time. More specifically, the hop planning problem is formulated as a stochastic optimization problem over an infinite horizon, where the rover's state is its position vector on the irregular asteroid's surface and the action space is constituted by the velocity vector. The reward function penalizes the time and energy required for each hop as well as the approach to a set of hazardous regions, while incentivizing the target regions. The off-policy Least Squares Policy Iteration (LSPI) method (Lagoudakis & Parr, 2003) is adopted to compute a linear approximation of the optimal action-value function Q^* by using datasets generated through heuristic sub-optimal policies or high fidelity simulators. Approximately five million trajectories are simulated over a broad range of the state–action space to train the RL agent via the LSPI algorithm. It is noted how the major advantage relies in the off-policy approach, which enables the reuse of large datasets to re-learn an optimal policy on the fly when the reward structure changes. Nevertheless, the learnt policies highly depend on the asteroid environment simulator, wherein the gravity field and contact models have to be estimated more accurately, e.g., by exploiting remote observations.

As for wheeled vehicles, short-range planning is required when the exploration area is small. In such cases, an alternative representation of the surface's terrain is given in terms of height levels over a reference altitude (Jiang et al., 2020). Landing probes or mother spacecraft can provide topology-surface data useful for height-level representation. In Jiang et al. (2020), a DQN algorithm finds optimal trajectories when the state space is defined according to a grid of cells, with each of them having its own height. The rover actions allow moving along the four directions resulting in rolling motion when moving from higher to lower levels, walking motion when moving on the same level, and hopping motion when moving from lower to higher levels. To minimize the required energy, the reward function penalizes hopping and rolling motions, and incentivizes walking motions. The DQN algorithm is then validated through simulations, showing good performance values in finding minimum-energy trajectories also when small terrain changes occur. However, such approach is limited by the static map employed during the training phase. Therefore, the learnt approximated trajectory is specific to the provided terrain map with no generalization capabilities.

9. Opportunities, challenges and prospects

This section summarizes the major aspects concerning the application of RL-based approaches to spacecraft control problems. Besides the opportunities and the prospects offered by such techniques, we outline the main issues to be solved for the adoption of RL-based solutions in an industrial space context. Such issues pave the way for interesting future developments from both a theoretical and implementation standpoint. As highlighted in this section, such future developments are also meant to reduce the technological gap between the solutions proposed by the academic community and the needs/requirements of the space industry.

9.1. Advantages and prospects

As highlighted in literature (e.g., Gaudet et al. (2020c, 2020d)), there are many benefits to using RL approaches in spacecraft control applications, which are summarized below.

- RL frameworks can be used to design and synthesize fully integrated solutions (e.g., GNC systems Gaudet et al., 2020a), where a trained global closed-loop policy can directly map raw sensor data (or the navigation system estimates of the spacecraft states) directly to actuator commands. Traditionally, this problem has been solved using guidance algorithms to generate the nominal trajectory for the inner control loops (Federici, Benedikter, & Zavoli, 2021b). For example, in planetary landing applications, traditional GNC systems can generate on-board a reference trajectory before the actual start of the powered descent phase, and then use a separate controller to track the trajectory towards the desired landing site. In this regard, we can mention the MSL case (Gaudet & Furfaro, 2014; Singh et al., 2007), where, after the unguided parachute deceleration phase, a powered descent 5th order polynomial trajectory was computed on-board before the start of the powered landing phase. Contrary to this, integrated GNC systems based on RL solutions would be able to learn a global closed-loop policy over the region of state space defined by the deployment ellipse and potential landing sites (Gaudet et al., 2018, 2020b), and would also allow retargeting while en-route to guarantee safety during landing (Gaudet & Furfaro, 2014). Unlike the above-mentioned traditional approaches, no reference trajectory computation (or re-computation) is actually needed at the beginning of the powered descent phase thanks to the fact that, during the RL training, the agent learns to autonomously land using the current position and velocity information as provided by the navigation system over the whole theater of operations (Gaudet & Furfaro, 2014). Similar considerations can be made for other kinds of applications, e.g., the GTO/GEO transfer problems (Arora & Dutta, 2020; Holt et al., 2021, 2020). And, as explained in Gaudet et al. (2020a) and Gaudet et al. (2020d), this capability has the potential to dramatically simplify the planning and the execution of the operations needed for the spacecraft landing.
- Unlike classical control theory, closed-form and explicit mathematical equations of the whole environment and conservative/stringent assumptions are not needed to solve control problems. RL-based approaches do not take care of the environment model directly. Indeed, they learn the environment dynamics by training the RL policies, e.g., the NN weights, via simulations. The agent can be trained via high fidelity computer simulators, capable of reproducing the behavior of its environment over a large theater of operations (e.g., the deployment ellipse and the potential landing site in planetary landing applications), the interplay between the agent and its own environment, while taking into account uncertainties and failure conditions (e.g., sensor noise/biases, actuator failures Gaudet et al., 2020d). For instance, the configuration parameters of the simulators can be varied over the whole theater of operations to define and run episodes used both for the training and the testing of RL agents, e.g., see Gaudet et al. (2020a). Moreover, RL agents can be also trained by using synthetic or flight data, reproducing the behavior of their own environment (Liu et al., 2021). In this regard, on-line training can be also carried out by exploiting data generated during the actual operational phase (Elkins et al., 2020).
- Building analytical environment models in spacecraft control applications can be very difficult, e.g., see Gaudet et al. (2020a, 2020d) for planetary landing applications and Miller et al. (2020) for interplanetary low-thrust trajectory design. In addition to that, model-based control approaches (e.g., MPC) with a precise modeling of the environment need extensive on-board computation, which is impractical due to the limited computational capability of typical flight processors (Furano et al., 2020; Gankidi & Thangavelautham, 2017). On the other hand, trained RL policies implemented via neural networks turn out to be computationally efficient, see Dong et al. (2021), Gaudet and Furfaro (2014) and Smith et al. (2021). RL approaches can also be used in case the environmental dynamics are too complex to be modeled deterministically, and must be approximated by means of stochastic models (Bishop, 2006), tuned by using simulated or real spacecraft telemetry data (Tipaldi, Feruglio, Denis, & D'Angelo, 2020). Some of the papers cited in this survey provide SW repository links to SW packages/libraries implementing RL environment simulators used to train and test RL policies, e.g., see Gaudet et al. (2020b).
- Although sub-optimal compared to the solutions computed by traditional approaches (e.g., the GPOPS solver Rao et al., 2010) in ideal/nominal conditions (Gaudet et al., 2020a, 2020b), the trained RL policies are able to cover a wide theater of operations and are robust to system uncertainties and off-nominal conditions. Moreover, some RL solutions (e.g., meta-RL) can also exhibit adaptability to changing system dynamics aspects or uncertainties (see Gaudet et al. (2020a, 2020b, 2020d)) and even to novel environments (Gaudet et al., 2020c). This can be a key aspect for space missions featured by complex operational environments (e.g., deep space exploration missions), as well as for the possible cost-effective reuse of RL frameworks (and the related implementation) from one application to another (Gaudet et al., 2020b).
- Although the training phase can be computationally expensive, limited memory and computational resources are required to run the resulting trained policies (as their execution implies a relatively small number of matrix multiplications and activation function evaluations). Thus, such policies should be easily implemented and executed on the current generation of spacecraft flight computers (Furano et al., 2020; Tipaldi et al., 2018). For instance, in Gaudet et al. (2020b), it took less than 1 ms for the 6-DOF planetary landing policy (implemented in Tensorflow/Python) to map an observation to an action on a 2.3 GHz processor, which means around 23 ms on a typical flight processor running at 100 MHz (Furano et al., 2020; Gankidi & Thangavelautham, 2017). These computational times are in line with typical guidance cycles, see also Gaudet and Furfaro (2014) and Gaudet et al. (2020d). The feasibility of implementing DRL agents on-board is also highlighted in Smith et al. (2021), where a CubeSat constellation phasing problem is addressed. Moreover, it is worth highlighting the usage of space-grade FPGAs for implementing DRL-based algorithms. In this regard, the work in Gankidi and Thangavelautham (2017) is noteworthy as it paves the way to the implementation of DRL algorithms on-board space vehicles by presenting an FPGA implementation of the Q-learning algorithm with neural networks.

Table 1
A comparison of optimal control and RL.
Source: Derived from Gaudet et al. (2018).

Optimal control	Reinforcement learning
Single trajectory (except for trivial cases where Hamilton–Jacobi–Bellman (HJB) equations can be solved)	Global over the whole theater of operations
Unbounded run time except for special cases such as convex constraints	Extremely fast run time for trained policy
Hybrid dynamics requires special treatment	Hybrid Dynamics handled seamlessly
Dynamics need to be represented as ordinary differential equations	Agent can learn in a high fidelity simulator
Open Loop when using the Pontryagin's Maximum Principle (requires a controller to track the optimal trajectory)	Closed Loop (e.g., integrated guidance and control systems)
Output feedback an open problem for nonlinear systems	Can learn from raw sensor inputs allowing fully integrated GNC (e.g, pixels to actuator commands)
Elegantly handles constraints	Constraints handled via large negative rewards and episode termination
Deterministic	Stochastic, learning does not converge every time, may need to run multiple policy optimizations

Table 2
Summary table of the RL framework core elements chosen in the most relevant papers reported in this survey (value function methods).

RL algorithm	Planetary landing	Orbit transfer	Attitude control systems	Rendezvous & docking	Constellation orbital control	Operations scheduling & rover
Q-learning	Wilson and Riccardi (2021) (d/s)	–	–	–	–	–
DQN	–	Arora and Dutta (2020) (c/s)	–	–	–	–
Deep-Q learning	–	–	–	–	–	Yan et al. (2019) (d/s) Yang, Hou et al. (2020) (d/s)
Policy iteration based	–	–	–	–	–	Wang et al. (2011) (d/s) Yang, Hou et al. (2020) (d/dt)
LSPI	–	–	–	–	–	Hockman and Pavone (2019) (c/dt)

The resolution of constrained optimization problems need considerable computational power, which actually precludes their implementation on-board spacecraft and CubeSats (Furano et al., 2020; Gankidi & Thangavelautham, 2017; Smith et al., 2021). An interesting comparison between RL and traditional optimal control approaches related to GNC systems (and actually applicable to the other types of applications analyzed in this survey) is provided in Table 1.

The usage of RL-based approaches can also bring significant benefits to the space mission planning and operation activities. For instance, current practices for asteroid close proximity maneuvers require an accurate characterization of the environmental dynamics and precise spacecraft positioning before starting the maneuver. This can create a delay of several months between the spacecraft arrival and the ability to safely complete close proximity maneuvers (Gaudet et al., 2020d). Adaptive integrated GNC systems implemented as meta-RL policies have the potential to complete these maneuvers in environments with unknown dynamics, with initial conditions spanning over a large deployment region, and without a shape model of the asteroid. As a consequence, mission planning and operation tasks can become more straightforward (and thus less expensive) since close proximity operations to asteroids might be undertaken without the need of accurately modeling, e.g., the asteroid gravitational field and the local solar radiation pressure (Gaudet et al., 2020a, 2020d).

As for the most relevant papers reported in this survey, Tables 2 and 3 provide an overview of the RL algorithms for value function methods and policy gradient methods, respectively. The type of action space ('c' for continuous and 'd' for discrete) and the type of RL environment models ('s' for computer simulator and 'dt' for real/synthetic data) are also reported.

9.2. Challenges and future developments

Even though RL-based solutions can enable new possibilities and pave the way for challenging space applications, their adoption in real space missions is still hindered by relevant open issues, as hereafter reported.

- Formulating a real-world problem into an RL framework and defining its core elements is nontrivial task, e.g., see Elkins et al. (2020). The training times and the performance of the trained RL agents depend on, e.g., the state representation, the reward function, the episode definition (including the termination conditions), the state space exploration approach. The selection of a proper RL framework formulation depends on different factors (e.g., the objectives to be achieved and the problem constraints) and requires care and thought. In addition to that, relying on a single RL formulation of the problem at hand may not be a robust strategy during operations. In particular, one may deploy different RL agents, and select one of them based on the current operational phase.
- A potential drawback of RL-based approaches is the lack of local or global stability guarantees. While remarkable progress has been made in RL algorithms, an unresolved fundamental issue is indeed how to analyze or certify the stability of the agent-environment system (Busoniu, de Bruin, Tolic, Kober, & Palunko, 2018; Garcıa & Fernández, 2015). However, as claimed in Gaudet et al. (2018), the stability issue also concerns other more traditional approaches, e.g., optimal control, since they have stability guarantees only when the environment acts in nominal conditions, used to synthesize the corresponding control law. It is interesting to note how the combination of the RL with more traditional approaches can help. In Holt et al. (2021), the authors state that combining a Lyapunov controller with an RL architecture can eliminate the drawbacks from each approach, i.e., the sub-optimality of Lyapunov controllers and the unknown stability of RL methods. In this regard, it is also worth mentioning some interesting results concerning the stability and robustness analysis of data-driven control systems, which can be exploited to train RL policies with provable stability and safety certificates,

Table 3
Summary table of the RL framework core elements chosen in the most relevant papers reported in this survey (policy gradient methods).

RL algorithm	Planetary landing	Orbit transfer	Attitude control systems	Rendezvous & docking	Constellation orbital control	Operations scheduling & rover
(Advantage) Actor–critic	Furfaro et al. (2020) (c/s)	Holt et al. (2020) (c/s)	Dong et al. (2021) (c/s + dt)	Hu et al. (2021) (c/s) Hovell and Ulrich (2021) (c/s + dt)	Smith et al. (2021) (d/s)	Wang et al. (2019) (d/s) Wei et al. (2021) (c/s)
PPO	Scorsoglio et al. (2021) (c/s) Gaudet et al. (2020a) (c/s) Gaudet et al. (2018) (c/s) Gaudet et al. (2020b) (c/s) Gaudet et al. (2020c) (d/s) Gaudet et al. (2020d) (d/s)	Holt et al. (2021) (c/s) LaFarge et al. (2021) (c/s) Sullivan and Bosanac (2020) (c/s) Sullivan et al. (2021) (c/s) Federici, Scorsoglio et al. (2021) (c/s) Zavoli and Federici (2021) (c/s) Miller et al. (2020) (c/s)	Elkins et al. (2020) (d/s) Vedant et al. (2019) (c/s)	Oestreich et al. (2021) (c/s) Federici et al. (2021a) (c/s)	Smith et al. (2021) (d/s)	–
DDPG	–	–	Yadava et al. (2018) (c/s) Su et al. (2019) (c/s) Gao et al. (2020) (c/s)	–	–	Huang et al. (2021) (c/s) Zhao et al. (2020) (c/dt)
REINFORCE	–	–	–	–	–	Chan and Agha-mohammadi (2019) (d/s)
IRL	–	Elliott et al. (2020) (c/s)	–	–	Silvestrini and Lavagna (2021) (c/s)	Pflueger et al. (2019) (d/dt)

see Donti, Roderick, Fazlyab, and Kolter (2020), Fazlyab, Morari, and Pappas (2022) and Yin, Seiler, and Arcak (2022).

- Another issue with RL is how to guarantee the constraints on states and actions. In literature, two types of constraints are considered, i.e., hard and soft constraints. The former can be imposed by terminating the episode with a negative reward, whereas the latter can be formulated through negative rewards, but without premature termination of the episode (Gaudet et al., 2018, 2020b). This can require a certain amount of trial and error for the reward function definition.
- Solving control problems via RL is also challenging because of the long training times (Gaudet & Furfaro, 2014; Smith et al., 2021; Zavoli & Federici, 2021) and the sensitivity of the achieved performance to the selected hyperparameters (e.g., the number of nodes in a NN layer Arora & Dutta, 2020, the number of steps to unroll a recurrent NN in the forward pass Gaudet et al., 2020a, as well as the learning rate and the penalty weights in the reward function Federici et al., 2021a). As shown in Gaudet and Furfaro (2014), the training times can be shortened by using the apprenticeship learning technique, which can be employed to initialize the neural network weights by using some prior knowledge (e.g., open-loop optimal trajectories computed by the GPOPS solver Rao et al., 2010 under ideal conditions). Similarly, an IRL approach could be adopted to autonomously extract a reward function from human expert solutions, thus improving the effectiveness of the training process on the considered problem (Zavoli & Federici, 2021). In Wilson and Riccardi (2021), the RL agent training times in a 3-DOF powered descent problem are reduced by discretizing the action space. Still in Wilson and Riccardi (2021), the authors investigate the usage of two methods for supporting the hyperparameter tuning process, i.e., the area-optimized approach (which considers the smallest area under the learning curve) and the reward-optimized approach (which considers the total reward at the end of training). In Smith et al. (2021), a grid search method (Liashchynskyi & Liashchynskyi, 2019) is used to select the hyperparameters of both the actor and critic NNs for solving a satellite phasing problem.
- Besides the complexity of the problem at hand, the issue of the long training times can also depend on the training data quality. Indeed, the RL agent training process can be slow down when the training data coming from the simulators are correlated to each other. Such a correlation is undesirable since, in case the agent starts collecting poor quality data (e.g., trajectories with no rewards), the agent does not know how to improve his performance (thus continuing to accumulate bad trajectories Federici et al., 2021a).
- The verification of RL-based solutions (and in general, of AI-based approaches) is a topic of paramount importance towards the adoption of these algorithms in space missions, where safety and reliability aspects are strongly considered (Tavallali, Karumanchi, Bowkett, Reid, & Kennedy, 2020; Tipaldi et al., 2020; Wesel & Goodloe, 2017). RL-based solutions tend to be very sensitive to their environment, and thus their behavior over a vast range of plausible conditions should be assessed. As a consequence, it becomes necessary to employ advanced verification techniques, which are largely based on analytical methods (Nardone, Santone, Tipaldi, Liuzza, & Glielmo, 2019; Tipaldi et al., 2020). As suggested by Wesel and Goodloe (2017), one should combine knowledge from the domains of formal methods, dependable systems engineering, and artificial intelligence to develop new formal methods for the verification of RL-based solutions. This topic is an unexplored research area, and it is not expected to have one single verification method applicable to all the possible RL algorithms (Wesel & Goodloe, 2017).
- In spacecraft control applications, confidence about on-board RL-based solutions can not only be achieved through verification by test, but also through review of design (ECSS, 2018), which means providing evidence that mission and system-level

requirements are consistent with the related implemented RL-based solutions. Being usually implemented via NNs, the latter cannot be adequately reviewed by system engineers, resulting in a potential loss of desired system behavior between system-level requirements and their actual implementation (Cancro, Innanen, Turner, Monaco, & Trela, 2007). Moreover, the systems-level impact of any modification/tuning of RL-based solutions (e.g., the NN weights) during the spacecraft operational phase cannot properly assessed (Tipaldi et al., 2020). This calls for a disentanglement of RL-based solutions from the typical interpretability issues of NNs. The reader can refer to Amarasinghe, Kenney, and Manic (2018) and Yang, Zhang and Sudjianto (2021) for possible solutions.

- In connection with the above two bullets, the decision surfaces of neural networks are, in general, complex and unpredictable. Hence, verifying that the networks operate as expected under every possible condition is difficult. Therefore, control policies developed through RL may be risky for a space mission. In Chan and Agha-mohammadi (2019), an example about integrating an RL policy within a flight planning and execution system, namely MEXEC, is provided. As outlined in Section 8.1, the authors aim at computing the best trajectories for the exploration of small celestial bodies. Instead of actuating those best trajectories via the RL policy, an agile and safe implementation within the MEXEC module provides the current trajectory to the trained RL agent. Then, such agent is used to compute the best times that an image can be taken along the current trajectory. This way, the spacecraft control is decoupled from the RL agent, which is only used for evaluating the best observation times. In other words, Chan and Agha-mohammadi (2019) shows how a safe RL implementation with proper integration processes is possible.
- After being trained and tested, RL policies have to be deployed into space-grade target environments. Depending on the type of application, space grade processors/microcontrollers or FPGA can be adopted (Furano et al., 2020; Gankidi & Thangavelautham, 2017). Generally speaking, the development of spacecraft on-board ML-based applications has lagged behind terrestrial applications for several reasons. e.g., limited processing power of space qualified computers and stringent reliability/quality requirements. Powerful frameworks now exist to deploy trained NNs onto a range of hardware targets. However, these tools are aimed at terrestrial hardware devices. The usage of automatic code generator tools and libraries for space projects is important to foster RL-based applications on-board. Lightweight RL libraries need to be adopted with a limited memory footprint and computational needs. Moreover, RL-agent deployment tools have to produce human editable code, which can be used as basis to develop formal flight on-board SW fulfilling flight SW quality requirements (Blacker, Bridges, & Hadfield, 2019; Furano et al., 2020). In Engstrom et al. (2020), the main implementation issues of deep RL algorithms, with reference to PPO and TRPO, are discussed. In particular, it is stated that the use of fully tested libraries is crucial, because the results are often significantly impacted by the so-called code-level optimizations, i.e., small implementation details seemingly of secondary importance (Engstrom et al., 2020). At the same time, the reproducibility of the results should be guaranteed to the scientific community by making use of open-source libraries.

In addition to the above open challenges, we can also mention the following aspects and considerations which can lead to interesting future developments, the latter still connected with the adoption of RL approaches in real space missions.

- Even though the environment models/simulators used in literature provide good means to train RL agents and prove the effectiveness of RL-based approaches, it is necessary to focus on the system engineering aspects in order to implement simulators with more realistic models (to better emulate, e.g., the lander aerodynamics or sensor distortion effects Gaudet et al., 2018), and able to cover an entire space mission phase (e.g., the whole Entry, Descent & Landing phase Ciabatti et al., 2021). As highlighted in Gaudet et al. (2020b), developing a real guidance and control system would require high fidelity models of peripheral systems. This system engineering effort is needed for the adoption of RL-based solutions in an industrial Space context (Gaudet et al., 2020d).
- As stated above, RL policies for spacecraft control applications are usually trained in a simulated environment. Although good results can be obtained by using computer simulators, this approach may not produce RL policies generalizing well when operated in the real environment, especially when the computer simulators do not accurately reproduce its complex dynamics. This issue is known as the simulation-to-reality gap (Kober et al., 2013). To address it, one could randomize the environment model parameters for each simulation to force the policy to become more robust to environmental changes. However, this workaround, known as domain randomization, can significantly increase the training times. Other efforts to solve the simulation-to-reality problem can imply training RL agents continuously, once deployed into the real environment (Hovell & Ulrich, 2021; Tipaldi et al., 2020).
- Multi agent RL approaches (Busoniu, Babuska, & De Schutter, 2008) can be adopted in case of complex systems (or Systems of Systems), where a centralized RL approach cannot be feasible. A typical case is a mega-constellation of satellites, in which each satellite can be regarded as an independent agents exchanging information with the environment and taking decision on its own (Liu et al., 2021; Silvestrini & Lavagna, 2021).
- RL agent performance objectives are widely assessed and discussed in the literature, and interesting ideas for further improvements are also provided. For instance, the usage of model-based RL approaches (e.g., the Dyna architecture Sutton & Barto, 2018) is proposed in Gaudet et al. (2018) as future work. In particular, the RL agent could make use of a learnt dynamics model to look forward over a short horizon and select the action that results in the highest advantages over such horizon. As claimed in Gaudet et al. (2018), this approach could bring some relevant benefits to planetary landing applications, such as the reduction in the fuel consumption during the powered descent phase.
- Both the state space formulation and the reward function definition are fundamental in RL (Silver et al., 2021; Sutton & Barto, 2018). The selected reward function can have an impact on the fuel efficiency, e.g., the exponential decrease in the magnitude of the target velocity can induce a sub-optimal behavior of the trained policies in planetary landing applications (Gaudet et al., 2020b) (indeed, an optimal trajectory with an initial position far from the target landing site should accelerate towards the target in order to reduce trajectory time and use less fuel to maintain altitude). As also highlighted in Wilson and Riccardi (2021), very poor performance of RL policies can be achieved in case the state space formulation and the reward function are not strongly connected. Moreover, some links between control theory and RL can also be investigated for the reward function designation. For instance, in Dong, Tang, and Yuan (2020), a Lyapunov function based approach is used to shape the reward function, which can effectively shorten the training times of the RL agents.

10. Conclusion

This survey paper has examined Reinforcement Learning (RL) based approaches for solving spacecraft control problems. The analysis has been focused on the usage of RL techniques in specific application fields, i.e., guidance, navigation and control systems for spacecraft

landing on celestial bodies, maneuver planning for orbit transfers and interplanetary mission trajectory, spacecraft attitude control systems, guidance and proximity maneuvers in rendezvous and docking scenarios, constellation orbital control, on-board decision-making for spacecraft operations scheduling and rover path planning. Throughout the paper, it has been highlighted how RL solutions can address the emerging needs of designing spacecraft with highly autonomous on-board capabilities and implementing controllers (i.e., RL agents) robust to system uncertainties and adaptive to changing environments. It has been also shown how the usage of deep neural networks allows scaling to control problems with high-dimensional and continuous state and action spaces, implementing integrated guidance and control systems, and addressing wide operational scenarios (all within limited on-board memory and computational resources).

Finally, by assessing the adoption of the reported RL solutions in real space projects, some challenging issues have been identified. In this regard, we can mention the availability of high-fidelity simulators for the RL agent training, the sensitivity of the achieved RL agent performance to the selected hyperparameters (e.g., the number of nodes in a neural network layer), as well as the verification and dependability requirements to be fulfilled by the resulting trained RL agents. This way, recommendations for future work have been proposed with the aim of reducing the technological gap between the solutions provided by the academic community and the needs/requirements of the space industry. As a result, this survey can be regarded as a comprehensive road map for all researchers and practitioners interested in the design and development of RL-based solutions, particularly in the field of spacecraft control applications.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on machine learning* (p. 1). ACM, http://dx.doi.org/10.1145/1015330.1015430.

Amarasinghe, K., Kenney, K., & Manic, M. (2018). Toward explainable deep neural network based anomaly detection. In *2018 11th international conference on human system interaction (HSI)* (pp. 311–317). http://dx.doi.org/10.1109/HSI.2018.8430788.

Arora, L., & Dutta, A. (2020). Reinforcement learning for sequential low-thrust orbit raising problem. In *AIAA scitech 2020 forum*. http://dx.doi.org/10.2514/6.2020-2186.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38. http://dx.doi.org/10.1109/MSP.2017.2743240.

Asmar, S., & Matousek, S. (2014). Mars Cube One (MarCO): The first planetary cubesat mission. In *Proceedings of the mars CubeSat/NanoSat workshop* (pp. 1–21).

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., & Kautz, J. (2017). Reinforcement learning through asynchronous advantage actor-critic on a GPU. In *ICLR*.

Baccari, S., Tipaldi, M., Iannelli, L., & Vasca, F. (2012). Photoelectrothermal model predictive control for light emitting diodes. In *2012 IEEE 51st IEEE conference on decision and control (CDC)* (pp. 394–399). http://dx.doi.org/10.1109/CDC.2012.6426679.

Baccari, S., Vasca, F., Tipaldi, M., & Iannelli, L. (2017). Model predictive control for luminous flux tracking in light-emitting diodes. *IEEE Transactions on Control Systems Technology*, *25*(2), 695–703. http://dx.doi.org/10.1109/TCST.2016.2560122.

Battin, R. H. (1999). *An introduction to the mathematics and methods of astrodynamics*. American Institute of Aeronautics and Astronautics.

Bellerose, J., & Scheeres, D. (2008). Dynamics and control for surface exploration of small bodies. In *AIAA/AAS astrodynamics specialist conference and exhibit*. American Institute of Aeronautics and Astronautics, http://dx.doi.org/10.2514/6.2008-6251.

Bertsekas, D. (2019a). *Reinforcement learning and optimal control*. Athena Scientific.

Bertsekas, D. P. (2019b). Feature-based aggregation and deep reinforcement learning: a survey and some new implementations. *IEEE/CAA Journal of Automatica Sinica*, *6*(1), 1–31. http://dx.doi.org/10.1109/JAS.2018.7511249.

Bianchessi, N., Cordeau, J.-F., Desrosiers, J., Laporte, G., & Raymond, V. (2007). A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research*, *177*(2), 750–762. http://dx.doi.org/10.1016/j.ejor.2005.12.026.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blacker, P., Bridges, C. P., & Hadfield, S. (2019). Rapid prototyping of deep learning models on radiation hardened CPUs. In *2019 NASA/ESA conference on adaptive hardware and systems (AHS)* (pp. 25–32). http://dx.doi.org/10.1109/AHS.2019.000-4.

Bosanac, N., Cox, A. D., Howell, K. C., & Folta, D. C. (2018). Trajectory design for a cislunar CubeSat leveraging dynamical systems techniques: The Lunar IceCube mission. *Acta Astronautica*, *144*, 283–296. http://dx.doi.org/10.1016/j.actaastro.2017.12.025.

Braylan, A., Hollenbeck, M., Meyerson, E., & Miikkulainen, R. (2015). Frame skip is a powerful parameter for learning to play Atari. In *Proceedings of the workshops at the twenty-ninth AAAI conference on artificial intelligence*.

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172. http://dx.doi.org/10.1109/TSMCC.2007.913919.

Busoniu, L., de Bruin, T., Tolic, D., Kober, J., & Palunko, I. (2018). Reinforcement learning for control: performance, stability, and deep approximators. *Annual Reviews in Control*, *46*, 8–28. http://dx.doi.org/10.1016/j.arcontrol.2018.09.005.

Cancro, G., Innanen, W., Turner, R., Monaco, C., & Trela, M. (2007). Uploadable executable specification concept for spacecraft autonomy systems. In *2007 IEEE aerospace conference* (pp. 1–12). http://dx.doi.org/10.1109/AERO.2007.352802.

Cerf, M. (2012). Multiple space debris collecting mission—Debris selection and trajectory optimization. *Journal of Optimization Theory and Applications*, *156*(3), 761–796. http://dx.doi.org/10.1007/s10957-012-0130-6.

Chan, D. M., & Agha-mohammadi, A.-a. (2019). Autonomous imaging and mapping of small bodies using deep reinforcement learning. In *2019 IEEE aerospace conference* (pp. 1–12). http://dx.doi.org/10.1109/AERO.2019.8742147.

Ciabatti, G., Daftry, S., & Capobianco, R. (2021). Autonomous planetary landing via deep reinforcement learning and transfer learning. In *2021 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)* (pp. 2031–2038). http://dx.doi.org/10.1109/CVPRW53098.2021.00231.

Clohessy, W. H., & Wiltshire, R. S. (1960). Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, *27*(9), 653–658.

Contini, C., Zamaro, M., Marinas, S. R., & Casasco, M. (2021). Space guidance optimisation in real time (GO-GREAT). In *8th international conference on astrodynamics tools and techniques (ICATT)*.

Curtis, H. (2010). *Orbital mechanics for engineering students*. Elsevier.

Di Mauro, G., Lawn, M., & Bevilacqua, R. (2018). Survey on guidance navigation and control requirements for spacecraft formation-flying missions. *Journal of Guidance, Control, and Dynamics*, *41*(3), 581–602. http://dx.doi.org/10.2514/1.G002868.

Dietze, C., Herrmann, F., Kuss, S., Lange, C., Scharringhausen, M., Witte, L., et al. (2010). Landing and mobility concept for the small asteroid lander MASCOT on asteroid 1999 JU3. In *International astronautical congress*.

Dong, Y., Tang, X., & Yuan, Y. (2020). Principled reward shaping for reinforcement learning via Lyapunov stability theory. *Neurocomputing*, *393*, 83–90. http://dx.doi.org/10.1016/j.neucom.2020.02.008.

Dong, H., Zhao, X., & Yang, H. (2021). Reinforcement learning-based approximate optimal control for attitude reorientation under state constraints. *IEEE Transactions on Control Systems Technology*, *29*(4), 1664–1673. http://dx.doi.org/10.1109/TCST.2020.3007401.

Donti, P. L., Roderick, M., Fazlyab, M., & Kolter, J. Z. (2020). Enforcing robust control guarantees within neural network policies. arXiv preprint arXiv:2011.08105.

(2018). *ECSS-E-ST-10-02C Rev.1 – Verification*. European Cooperation for Space Standardization.

Eickhoff, J. (2011). *Onboard computers, onboard software and satellite operations: An introduction*. Springer Science & Business Media.

Elkins, J. G., Sood, R., & Rumpf, C. (2020). Autonomous spacecraft attitude control using deep reinforcement learning. In *71st international astronautical congress (IAC)* (pp. 1–13).

Elliott, I., Bosanac, N., Ahmed, N. R., & McMahon, J. W. (2020). Apprenticeship learning for maneuver design in multi-body systems. In *AIAA scitech 2020 forum*. http://dx.doi.org/10.2514/6.2020-1912.

Emami, S. A., Castaldi, P., & Banazadeh, A. (2022). Neural network-based flight control systems: Present and future. *Annual Reviews in Control*, http://dx.doi.org/10.1016/j.arcontrol.2022.04.006.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., et al. (2020). Implementation matters in deep RL: A case study on PPO and TRPO. In *International conference on learning representations*.

Fazlyab, A. R., Fani Saberi, F., & Kabganian, M. (2016). Adaptive attitude controller for a satellite based on neural network in the presence of unknown external disturbances and actuator faults. *Advances in Space Research*, *57*(1), 367–377. http://dx.doi.org/10.1016/j.asr.2015.10.026.

Fazlyab, M., Morari, M., & Pappas, G. J. (2022). Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, *67*(1), 1–15. http://dx.doi.org/10.1109/TAC.2020.3046193.

Federici, L., Benedikter, B., & Zavoli, A. (2021a). Deep learning techniques for autonomous spacecraft guidance during proximity operations. *Journal of Spacecraft and Rockets*, *58*(6), 1774–1785. http://dx.doi.org/10.2514/1.A35076.

Federici, L., Benedikter, B., & Zavoli, A. (2021b). Machine learning techniques for autonomous spacecraft guidance during proximity operations. In *AIAA scitech 2021 forum*. http://dx.doi.org/10.2514/6.2021-0668.

Federici, L., Scorsoglio, A., Zavoli, A., & Furfaro, R. (2021). Autonomous guidance for cislunar orbit transfers via reinforcement learning. In *AAS/AIAA astrodynamics specialist conference, Virtual*.

Fehse, W. (2003). *Cambridge aerospace series*, *Automated rendezvous and docking of spacecraft*. Cambridge University Press, http://dx.doi.org/10.1017/CBO9780511543388.

Forootani, A., Iervolino, R., & Tipaldi, M. (2019). Applying unweighted least-squares based techniques to stochastic dynamic programming: Theory and application. *IET Control Theory & Applications*, *13*(15), 2387–2398. http://dx.doi.org/10.1049/iet-cta.2019.0289.

Forshaw, J. L., Aglietti, G. S., Fellowes, S., Salmon, T., Retat, I., Hall, A., et al. (2020). The active space debris removal mission RemoveDebris. Part 1: From concept to launch. *Acta Astronautica*, *168*, 293–309. http://dx.doi.org/10.1016/j.actaastro.2019.09.002.

Frost, C., Butt, A., & Silva, D. (2010). Challenges and opportunities for autonomous systems in space. In *Frontiers of engineering symposium*.

Furano, G., Meoni, G., Dunne, A., Moloney, D., Ferlet-Cavrois, V., Tavoularis, A., et al. (2020). Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. *IEEE Aerospace and Electronic Systems Magazine*, *35*(12), 44–56. http://dx.doi.org/10.1109/MAES.2020.3008468.

Furfaro, R., Scorsoglio, A., Linares, R., & Massari, M. (2020). Adaptive generalized ZEM-ZEV feedback guidance for planetary landing via a deep reinforcement learning approach. *Acta Astronautica*, *171*, 156–171. http://dx.doi.org/10.1016/j.actaastro.2020.02.051.

Gankidi, P. R., & Thangavelautham, J. (2017). FPGA architecture for deep learning and its application to planetary robotics. In *2017 IEEE aerospace conference* (pp. 1–9). http://dx.doi.org/10.1109/AERO.2017.7943929.

Gao, D., Zhang, H., Li, C., & Gao, X. (2020). Satellite attitude control with deep reinforcement learning. In *2020 Chinese automation congress (CAC)* (pp. 4095–4101). http://dx.doi.org/10.1109/CAC51589.2020.9326605.

Garcıa, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, *16*(1), 1437–1480. http://dx.doi.org/10.5555/2789272.2886795.

Gaskell, R., Barnouin-Jha, O., Scheeres, D. J., Konopliv, A., Mukai, T., Abe, S., et al. (2008). Characterizing and navigating small bodies with imaging data. *Meteoritics & Planetary Science*, *43*(6), 1049–1061. http://dx.doi.org/10.1111/j.1945-5100.2008.tb00692.x.

Gaudet, B., & Furfaro, R. (2014). Adaptive pinpoint and fuel efficient mars landing using reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, *1*(4), 397–411. http://dx.doi.org/10.1109/JAS.2014.7004667.

Gaudet, B., Linares, R., & Furfaro, R. (2018). Integrated guidance and control for pinpoint mars landing using reinforcement learning. In *AAS/AIAA astrodynamics specialist conference* (pp. 1–20).

Gaudet, B., Linares, R., & Furfaro, R. (2020a). Adaptive guidance and integrated navigation with reinforcement meta-learning. *Acta Astronautica*, *169*, 180–190. http://dx.doi.org/10.1016/j.actaastro.2020.01.007.

Gaudet, B., Linares, R., & Furfaro, R. (2020b). Deep reinforcement learning for six degree-of-freedom planetary landing. *Advances in Space Research*, *65*(7), 1723–1741. http://dx.doi.org/10.1016/j.asr.2019.12.030.

Gaudet, B., Linares, R., & Furfaro, R. (2020c). Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning. *Acta Astronautica*, *172*, 90–99. http://dx.doi.org/10.1016/j.actaastro.2020.03.026.

Gaudet, B., Linares, R., & Furfaro, R. (2020d). Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations. *Acta Astronautica*, *171*, 1–13. http://dx.doi.org/10.1016/j.actaastro.2020.02.036.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: Theory and practice*. Elsevier.

Glavic, M. (2019). (Deep) Reinforcement learning for electric power system control and related problems: a short review and perspectives. *Annual Reviews in Control*, *48*, 22–35. http://dx.doi.org/10.1016/j.arcontrol.2019.09.008.

Guo, Y., Hawkins, M., & Wie, B. (2013). Applications of generalized zero-effort-miss/zero-effort-velocity feedback guidance algorithm. *Journal of Guidance, Control, and Dynamics*, *36*(3), 810–820. http://dx.doi.org/10.2514/1.58099.

Harris, A., Teil, T., & Schaub, H. (2019). Spacecraft decision-making autonomy using deep reinforcement learning. In *29th AAS/AIAA space flight mechanics meeting, Hawaii*.

Hockman, B., & Pavone, M. (2019). Stochastic motion planning for hopping rovers on small solar system bodies. In *Springer proceedings in advanced robotics* (pp. 877–893). Springer International Publishing.

Holt, H., Armellin, R., Baresi, N., Hashida, Y., Turconi, A., Scorsoglio, A., et al. (2021). Optimal Q-laws via reinforcement learning with guaranteed stability. *Acta Astronautica*, *187*, 511–528. http://dx.doi.org/10.1016/j.actaastro.2021.07.010.

Holt, H., Armellin, R., Scorsoglio, A., & Furfaro, R. (2020). Low-thrust trajectory design using closed-loop feedback-driven control laws and state-dependent parameters. In *AIAA scitech 2020 forum*. http://dx.doi.org/10.2514/6.2020-1694.

Hovell, K., & Ulrich, S. (2021). Deep reinforcement learning for spacecraft proximity operations guidance. *Journal of Spacecraft and Rockets*, *58*(2), 254–264. http://dx.doi.org/10.2514/1.A34838.

Hu, Q., Yang, H., Dong, H., & Zhao, X. (2021). Learning-based 6-DOF control for autonomous proximity operations under motion constraints. *IEEE Transactions on Aerospace and Electronic Systems*, *57*(6), 4097–4109. http://dx.doi.org/10.1109/TAES.2021.3094628.

Huang, Y., Mu, Z., Wu, S., Cui, B., & Duan, Y. (2021). Revising the observation satellite scheduling problem based on deep reinforcement learning. *Remote Sensing*, *13*(12), http://dx.doi.org/10.3390/rs13122377.

Izzo, D., Märtens, M., & Pan, B. (2019). A survey on artificial intelligence trends in spacecraft guidance dynamics and control. *Astrodynamics*, *3*, 287–299. http://dx.doi.org/10.1007/s42064-018-0053-6.

Jiang, X., Li, S., & Furfaro, R. (2019). Integrated guidance for mars entry and powered descent using reinforcement learning and pseudospectral method. *Acta Astronautica*, *163*, 114–129. http://dx.doi.org/10.1016/j.actaastro.2018.12.033.

Jiang, J., Zeng, X., Guzzetti, D., & You, Y. (2020). Path planning for asteroid hopping rovers with pre-trained deep reinforcement learning architectures. *Acta Astronautica*, *171*, 265–279. http://dx.doi.org/10.1016/j.actaastro.2020.03.007.

Joshi, G., & Padhi, R. (2014). Robust satellite formation flying through online trajectory optimization using LQR and neural networks. *IFAC Proceedings Volumes*, *47*(1), 135–141. http://dx.doi.org/10.3182/20140313-3-IN-3024.00173.

Junkins, J., & Taheri, E. (2018). Exploration of alternative state vector choices for low-thrust trajectory optimization. *Journal of Guidance, Control, and Dynamics*, *42*, 47–64. http://dx.doi.org/10.2514/1.G003686.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-International conference on neural networks, Vol. 4* (pp. 1942–1948).

Kleywegt, A. J., & Papastavrou, J. D. (1998). The dynamic and stochastic knapsack problem. *Operations Research*, *46*(1), 17–35. http://dx.doi.org/10.1287/opre.46.1.17.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, *32*(11), 1238–1274. http://dx.doi.org/10.1177/0278364913495721.

Koon, W. S., Lo, M. W., Marsden, J. E., & Ross, S. D. (2006). *The three-body problem and space mission design*. Marsden Books.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

Labrèche, G., Evans, D., Marszk, D., Mladenov, T., Shiradhonkar, V., Soto, T., et al. (2022). OPSSAT spacecraft autonomy with TensorFlow lite, unsupervised learning, and online machine learning. In *2022 IEEE aerospace conference*.

LaFarge, N. B., Miller, D., Howell, K. C., & Linares, R. (2021). Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. *Acta Astronautica*, *186*, 1–23. http://dx.doi.org/10.1016/j.actaastro.2021.05.014.

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Lemaıtre, M., Verfaillie, G., Jouhaud, F., Lachiver, J.-M., & Bataille, N. (2002). Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, *6*(5), 367–381. http://dx.doi.org/10.1016/S1270-9638(02)01173-2.

Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for NAS. arXiv preprint arXiv:1912.06059.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Liou, J.-C., & Johnson, N. L. (2006). Risks in space from orbiting debris. *Science*, *311*(5759), 340–341. http://dx.doi.org/10.1126/science.1121337.

Liu, J., Zhao, B., Xin, Q., Su, J., & Ou, W. (2021). DRL-ER: An intelligent energy-aware routing protocol with guaranteed delay bounds in satellite mega-constellations. *IEEE Transactions on Network Science and Engineering*, *8*(4), 2872–2884. http://dx.doi.org/10.1109/TNSE.2020.3039499.

Locoche, S. (2021). Reducing operation cost with autonomous guidance for electrical orbit raising. In *8th international conference on astrodynamics tools and techniques (ICATT)*.

Massenio, P. R., Rizzello, G., Comitangelo, G., Naso, D., & Seelecke, S. (2021). Reinforcement learning-based minimum energy position control of dielectric elastomer actuators. *IEEE Transactions on Control Systems Technology*, *29*(4), 1674–1688. http://dx.doi.org/10.1109/tcst.2020.3022951.

McGovern, A., & Wagstaff, K. L. (2011). Machine learning in space: extending our reach. *Machine Learning*, *84*, 335–340. http://dx.doi.org/10.1007/s10994-011-5249-4.

Miller, D., Englander, J. A., & Linares, R. (2020). Interplanetary low-thrust design using proximal policy optimization. *Advances in the Astronautical Sciences*, *171*, 1575–1592.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Nardone, V., Santone, A., Tipaldi, M., Liuzza, D., & Glielmo, L. (2019). Model checking techniques applied to satellite operational mode management. *IEEE Systems Journal*, *13*(1), 1018–1029. http://dx.doi.org/10.1109/JSYST.2018.2793665.

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the seventeenth international conference on machine learning* (pp. 663–670).

Oche, P. A., Ewa, G. A., & Ibekwe, N. (2021). Applications and challenges of artificial intelligence in space missions. *IEEE Access*, http://dx.doi.org/10.1109/ACCESS.2021.3132500.

Oestreich, C. E., Linares, R., & Gondhalekar, R. (2021). Autonomous six-degree-of-freedom spacecraft docking with rotating targets via reinforcement learning. *Journal of Aerospace Information Systems*, *18*(7), 417–428. http://dx.doi.org/10.2514/1.I010914.

Petropoulos, A. (2005). Refinements to the Q-law for low-thrust orbit transfers. *Advances in the Astronautical Sciences*, *120*, 963–982.

Pflueger, M., Agha, A., & Sukhatme, G. S. (2019). Rover-IRL: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robotics and Automation Letters*, *4*(2), 1387–1394. http://dx.doi.org/10.1109/LRA.2019.2895892.

Quadrelli, M. B., Wood, L. J., Riedel, J. E., McHenry, M. C., Aung, M., Cangahuala, L. A., et al. (2015). Guidance, navigation, and control technology assessment for future planetary science missions. *Journal of Guidance, Control, and Dynamics*, *38*(7), 1165–1186. http://dx.doi.org/10.2514/1.G000525.

Rao, A. V., Benson, D. A., Darby, C., Patterson, M. A., Francolin, C., Sanders, I., et al. (2010). Algorithm 902: GPOPS, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Transactions on Mathematical Software*, *37*(2), 1–39. http://dx.doi.org/10.1145/1731022.1731032.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Scorsoglio, A., D'Ambrosio, A., Ghilardi, L., Gaudet, B., Curti, F., & Furfaro, R. (2021). Image-based deep reinforcement meta-learning for autonomous lunar landing. *Journal of Spacecraft and Rockets*, *59*, 1–13. http://dx.doi.org/10.2514/1.A35072.

Shi, X., Ren, P., & Du, Q. (2020). Heterogeneous satellite network routing algorithm based on reinforcement learning and mobile agent. In *2020 IEEE globecom workshops (GC Wkshps)* (pp. 1–6). http://dx.doi.org/10.1109/GCWkshps50303.2020.9367476.

Shirobokov, M., Trofimov, S., & Ovchinnikov, M. (2021). Survey of machine learning techniques in spacecraft control design. *Acta Astronautica*, *186*, 87–97. http://dx.doi.org/10.1016/j.actaastro.2021.05.018.

Shotwell, R. (2005). Phoenix the first Mars Scout mission. *Acta Astronautica*, *57*(2), 121–134. http://dx.doi.org/10.1016/j.actaastro.2005.03.038.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. http://dx.doi.org/10.1038/nature24270.

Silver, D., Singh, S., Precup, D., & Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, *299*, http://dx.doi.org/10.1016/j.artint.2021.103535.

Silvestrini, S., & Lavagna, M. (2021). Neural-based predictive control for safe autonomous spacecraft relative maneuvers. *Journal of Guidance, Control, and Dynamics*, *44*(12), 2303–2310. http://dx.doi.org/10.2514/1.G005481.

Sims, J., & Flanagan, S. (2000). Preliminary design of low-thrust interplanetary missions (AAS 99-338). *Advances in the Astronautical Sciences*, *103*(1), 583–592.

Singh, G., SanMartin, A. M., & Wong, E. C. (2007). Guidance and control design for powered descent and landing on mars. In *2007 IEEE aerospace conference* (pp. 1–8). http://dx.doi.org/10.1109/AERO.2007.352818.

Smith, B., Abay, R., Abbey, J., Balage, S., Brown, M., & Boyce, R. (2021). Propulsionless planar phasing of multiple satellites using deep reinforcement learning. *Advances in Space Research*, *67*(11), 3667–3682. http://dx.doi.org/10.1016/j.asr.2020.09.025.

Smith, B., Capon, C., & Brown, M. (2019). Ionospheric drag for satellite formation control. *Journal of Guidance, Control, and Dynamics*, *42*(12), 2590–2599. http://dx.doi.org/10.2514/1.G004404.

Su, R., Wu, F., & Zhao, J. (2019). Deep reinforcement learning method based on DDPG with simulated annealing for satellite attitude control system. In *2019 Chinese automation congress (CAC)* (pp. 390–395). http://dx.doi.org/10.1109/CAC48633.2019.8996860.

Sullivan, C. J., & Bosanac, N. (2020). Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system. In *AIAA scitech 2020 forum*. http://dx.doi.org/10.2514/6.2020-1914.

Sullivan, C. J., Bosanac, N., Anderson, R. L., Mashiku, A. K., & Stuart, J. R. (2021). Exploring transfers between earth-moon halo orbits via multi-objective reinforcement learning. In *2021 IEEE aerospace conference (50100)* (pp. 1–13). http://dx.doi.org/10.1109/AERO50100.2021.9438267.

Sullivan, B., Kelm, B., Roesler, G., & Henshaw, C. G. (2015). DARPA robotic space servicer: On-demand capabilities in GEO. In *AIAA SPACE 2015 conference and exposition*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Szebehely, V. (1967). *Theory of orbits: The restricted problem of three bodies*. Academic Press.

Tavallali, P., Karumanchi, S., Bowkett, J., Reid, W., & Kennedy, B. (2020). A reinforcement learning framework for space missions in unknown environments. In *2020 IEEE aerospace conference* (pp. 1–8). http://dx.doi.org/10.1109/AERO47225.2020.9172272.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, *10*(7).

Tipaldi, M., Feruglio, L., Denis, P., & D'Angelo, G. (2020). On applying AI-driven flight data analysis for operational spacecraft model-based diagnostics. *Annual Reviews in Control*, *49*, 197–211. http://dx.doi.org/10.1016/j.arcontrol.2020.04.012.

Tipaldi, M., & Glielmo, L. (2018). A survey on model-based mission planning and execution for autonomous spacecraft. *IEEE Systems Journal*, *12*(4), 3893–3905. http://dx.doi.org/10.1109/JSYST.2017.2720682.

Tipaldi, M., Legendre, C., Koopmann, O., Ferraguto, M., Wenker, R., & D'Angelo, G. (2018). Development strategies for the satellite flight software on-board Meteosat Third Generation. *Acta Astronautica*, *145*, 482–491. http://dx.doi.org/10.1016/j.actaastro.2018.02.020.

Vavrina, M. A., Skelton, C. E., Deweese, K. D., Naasz, B. J., Gaylor, D. E., & D'souza, C. (2019). Safe rendezvous trajectory design for the restore-l mission. In *Advances in the astronautical sciences, Vol. 168* (pp. 3649–3668).

Vedant, V., Allison, J. T., West, M., & Ghosh, A. (2019). Reinforcement learning for spacecraft attitude control. In *70th international astronautical congress (IAC)* (pp. 1–10).

Viavattene, G., Devereux, E., Snelling, D., Payne, N., Wokes, S., & Ceriotti, M. (2022). Design of multiple space debris removal missions using machine learning. *Acta Astronautica*, *193*, 277–286. http://dx.doi.org/10.1016/j.actaastro.2021.12.051.

Walker, R., Koschny, D., Bramanti, C., & Carnelli, I. ESA CDF Study Team. (2017). Miniaturised asteroid remote geophysical observer (M-ARGO):A stand-alone deep space CubeSat system for low-cost science and exploration missions. In *Proceedings of the 6th interplanetary CubeSat workshop*.

Wang, C., Li, J., Jing, N., Wang, J., & Chen, H. (2011). A distributed cooperative dynamic task planning algorithm for multiple satellites based on multi-agent hybrid learning. *Chinese Journal of Aeronautics*, *24*(4), 493–505. http://dx.doi.org/10.1016/S1000-9361(11)60057-5.

Wang, X., Wang, G., Chen, Y., & Xie, Y. (2020). Autonomous rendezvous guidance via deep reinforcement learning. In *2020 Chinese control and decision conference (CCDC)* (pp. 1848–1853). http://dx.doi.org/10.1109/CCDC49329.2020.9163988.

Wang, H., Yang, Z., Zhou, W., & Li, D. (2019). Online scheduling of image satellites based on neural networks and deep reinforcement learning. *Chinese Journal of Aeronautics*, *32*(4), 1011–1019. http://dx.doi.org/10.1016/j.cja.2018.12.018.

Wei, L., Chen, Y., Chen, M., & Chen, Y. (2021). Deep reinforcement learning and parameter transfer based approach for the multi-objective agile earth observation satellite scheduling problem. *Applied Soft Computing*, *110*, http://dx.doi.org/10.1016/j.asoc.2021.107607.

Wertz, J. R. (2012). *Spacecraft attitude determination and control, Vol. 73*. Springer Science & Business Media.

Wesel, P., & Goodloe, A. (2017). Challenges in the verification of reinforcement learning algorithms. *NASA Technical Reports*.

Whitley, R., & Martinez, R. (2016). Options for staging orbits in cislunar space. In *Proceedings of 2016 IEEE aerospace conference* (pp. 1–9). http://dx.doi.org/10.1109/AERO.2016.7500635.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256. http://dx.doi.org/10.1007/BF00992696.

Wilson, C., & Riccardi, A. (2021). Improving the efficiency of reinforcement learning for a spacecraft powered descent with Q-learning. *Optimization and Engineering*, http://dx.doi.org/10.1007/s11081-021-09687-z.

Wolfe, W. J., & Sorensen, S. E. (2000). Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, *46*(1), 148–166. http://dx.doi.org/10.1287/mnsc.46.1.148.15134.

Yadava, D., Hosangadi, R., Krishna, S., Paliwal, P., & Jain, A. (2018). Attitude control of a nanosatellite system using reinforcement learning and neural networks. In *2018 IEEE aerospace conference* (pp. 1–8). http://dx.doi.org/10.1109/AERO.2018.8396409.

Yan, P., Xiao, H., Guo, J., Bai, C., & Zheng, H. (2019). Adaptive cooperative detection method for unmanned planetary vehicles based on deep reinforcement learning. In *2019 IEEE international conference on unmanned systems (ICUS)* (pp. 714–719). http://dx.doi.org/10.1109/ICUS48101.2019.8996016.

Yang, J., Hou, X., Hu, Y. H., Liu, Y., & Pan, Q. (2020). A reinforcement learning scheme for active multi-debris removal mission planning with modified upper confidence bound tree search. *IEEE Access*, *8*, 108461–108473. http://dx.doi.org/10.1109/ACCESS.2020.3001311.

Yang, C., Zhang, H., & Gao, Y. (2021). Analysis of a neural-network-based adaptive controller for deep-space formation flying. *Advances in Space Research*, *68*(1), 54–70. http://dx.doi.org/10.1016/j.asr.2021.03.007.

Yang, Z., Zhang, A., & Sudjianto, A. (2021). Enhancing explainability of neural networks through architecture constraints. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(6), 2610–2621. http://dx.doi.org/10.1109/TNNLS.2020.3007259.

Yang, T., Zhao, L., Li, W., & Zomaya, A. Y. (2020). Reinforcement learning in sustainable energy and electric systems: a survey. *Annual Reviews in Control*, *49*, 145–163. http://dx.doi.org/10.1016/j.arcontrol.2020.03.001.

Yin, H., Seiler, P., & Arcak, M. (2022). Stability analysis using quadratic constraints for systems with neural network controllers. *IEEE Transactions on Automatic Control*, *67*(4), 1980–1987. http://dx.doi.org/10.1109/TAC.2021.3069388.

Yoo, H., Byun, H. E., Han, D., & Lee, J. H. (2021). Reinforcement learning for batch process control: Review and perspectives. *Annual Reviews in Control*, *52*, 108–119. http://dx.doi.org/10.1016/j.arcontrol.2021.10.006.

Yu, V. F., Redi, A. P., Hidayat, Y. A., & Wibowo, O. J. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing*, *53*, 119–132.

Zavoli, A., & Federici, L. (2021). Reinforcement learning for robust trajectory design of interplanetary missions. *Journal of Guidance, Control, and Dynamics*, *44*(8), 1440–1453. http://dx.doi.org/10.2514/1.G005794.

Zhao, X., Wang, Z., & Zheng, G. (2020). Two-phase neural combinatorial optimization with reinforcement learning for agile satellite scheduling. *Journal of Aerospace Information Systems*, *17*(7), 346–357. http://dx.doi.org/10.2514/1.I010754.

Zhifei, S., & Joo, E. (2012). A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, *5*, 293–311. http://dx.doi.org/10.1108/17563781211255862.

Zoppi, M., Tipaldi, M., & Di Cerbo, A. (2018). Cross-model verification of the electrical power subsystem in space projects. *Measurement*, *122*, 473–483. http://dx.doi.org/10.1016/j.measurement.2018.01.014.