JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 23, No. 3, March 2026

Improving Robustness of Autonomous Earth-Observing Spacecraft Using Training Environment Enhancements

Lorenzzo Quevedo Mantovani* and Hanspeter Schaub†
University of Colorado Boulder, Boulder, Colorado 80303

https://doi.org/10.2514/1.I011697

Recent research is developing autonomous policies for agile Earth-observing satellite (AEOS) tasking using deep reinforcement learning (DRL). Prior work uses a fixed training environment with enhanced safety challenges to allow DRL to train a policy over a short number of orbits. This paper investigates how training environment enhancements can improve these policies' robustness by varying the training environment and extending simulated mission times. DRL enables real-time, onboard decision-making while accounting for complex dynamics and constraints. However, policies often struggle to generalize to longer missions or conditions different from training, partly due to the short simulated mission times. Multi-spacecraft systems amplify these challenges, where battery capacity and solar panel efficiency diverge over time due to external factors. Although satellites might run copies of a single policy, its robustness is essential to accommodate agent variability and ensure mission success. This paper investigates training with two curricula, constantly degraded environments, and a domain randomization approach to improve AEOS policy robustness and performance. Policies are tested on episodes 125-times longer than training episodes under nominal and degraded conditions. The proposed training environment enhancements' applicability is demonstrated in a heterogeneous spacecraft system, achieving a 63% reduction in failures and a 2.8% increase in median cumulative reward.

I. Introduction

EARTH-OBSERVING satellites (EOSs) are equipped with instruments to acquire information about the Earth. EOSs are tasked with different activities, such as charging, downlinking data, momentum management, and science acquisition modes. The tasking of EOS has the goal of defining a sequence of actions to maximize the mission outcome while preserving the spacecraft's health and safety given by the system constraints. The advent of agile EOS (AEOS), capable of maneuvering not only along track as traditional EOS but also across track, increases the complexity of the solution space and problem. Traditionally, spacecraft tasking has been performed by humans or offline optimization algorithms. The resulting open-loop schedules are uploaded to the spacecraft but are brittle to initial condition changes and computationally expensive [1]. The use of deep reinforcement learning (DRL) has been proposed to mitigate these problems [2–6]. DRL can account for complex dynamics and constraints and provide solutions in real-time for onboard decision-making. Nevertheless, DRL-based techniques can bring concerns about the reliability of the policy and its robustness when deployed in the real world [7].

The AEOS scheduling problem has been shown to be NP-hard [8]. To address this, a variety of optimization-based methods have been proposed. Techniques such as greedy algorithms, dynamic programming, constraint programming, and local search are investigated to solve the scheduling problem [8] as well as heuristics [9]. In Ref. [10], the authors show the use of infeasibility-based graphs to solve problems with up to 10,000 requests and 24 satellites. Other works have tackled uncertainty by introducing the use of budgeted uncertainty to account for cloud coverage [11,12]. More recently, promising decentralized replanning methods have been explored to enable the observation of dynamic, time-varying targets such as floods and wildfires [13,14].

However, optimization-based methods often lack robustness to variations in initial conditions and typically require partial or total replanning when unexpected events occur. Many existing approaches also rely on simplifying assumptions, such as constant slew rates and simplified power generation and consumption models, which widen the gap between simulation and reality. This gap can decrease the performance of the obtained solutions when deployed in the real world. The use of DRL has been proposed in the context of spacecraft autonomy to mitigate these problems [15,16], offering the ability to account for complex dynamics and constraints [17]. Once trained, the resulting deep neural networks (DNNs) can be quickly evaluated, showing their potential for onboard, closed-loop decision-making.

DRL has demonstrated strong potential across a range of spacecraft autonomy tasks. For instance, Monte Carlo Tree Search (MCTS) combined with DNNs has been applied to the satellite tasking problem, yielding performance comparable to or better than genetic algorithms [17]. Comparisons among different DRL algorithms, such as advantage actor critic (A2C), deep Q-network (DQN), and proximal policy optimization (PPO), have been performed, as well as comparisons with optimization techniques such as mixed integer linear programming [1,18], highlighting DRL's potential to approach near-optimal solutions in onboard autonomy scenarios. The use of DRL has also been investigated in multi-satellite scenarios [6,19,20], with uncertainty in cloud coverage [21–23], and when accounting for image quality [24].

Despite promising results, deploying onboard planning without human supervision raises concerns about policy reliability. In particular, real-world deployment conditions may differ from the training environment or change over time due to hardware degradation. Battery capacity loss, reaction wheel wear, reduced solar panel performance, and other factors can affect the policy's performance. While DRL-based policies are typically trained using shorter episodes with randomized initial conditions to introduce the agent to different scenarios and promote generalization, these strategies do not always lead to robust performance over longer durations or under altered conditions. More specifically, long-term effects may not be adequately captured. For example, if environments with realistic atmospheric drag are used, the satellite may not accumulate enough angular momentum over a short three-orbit-long episode to require reaction wheel momentum safety considerations and learn from it. Thus, prior work used a constantly enhanced training

Received 19 May 2025; accepted for publication 15 November 2025; published online 2 January 2026.

*Ph.D. Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive; lorenzzo.quevedomantovani@colorado.edu. Student Member AIAA (Corresponding Author).

†Distinguished Professor and Department Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive. Fellow AIAA.

environment that increased the safety aspects over the training period and showed good pointing and safety management performance in the modified environment [6,17,18,25,26]. However, these prior works never consider the intensity of enhancements that should be used, nor how the policy would function in a nominal, unenhanced space environment.

These robustness challenges become even more pronounced in multi-spacecraft systems in formation or constellation configurations. Although satellites may begin with similar initial conditions and parameters, the agents may diverge over time due to partial failures or degradation. One approach is to train separate policies from scratch tailored to each spacecraft as they degrade, but it is impractical for large constellations due to scalability issues. A more feasible solution is having a single policy that can be used for all spacecraft, where the policy's robustness is essential to ensure the mission's success. Figure 1 illustrates this concept of identical spacecraft executing different actions based on their states, yet all operating under the same robust policy.

To address policy robustness and the impact of the training environment and simulation length, in Ref. [27] the authors investigate training of DRL agents in degraded environments characterized by reduced battery capacity, increased external torque, and limited onboard storage. These agents are evaluated in both nominal and degraded environments using episodes significantly longer than those seen during training in order to simulate long-term deployment conditions. The results indicate that agents trained in the degraded environment achieve superior performance and experience fewer failures across both nominal and degraded scenarios compared to agents trained in the nominal environment. However, the analysis was limited to only a few levels of environmental degradation, leaving open questions about how structured training variations might influence policy robustness more broadly.

Variable degradation in training can also be explored in addition to the fixed degradation seen in Ref. [27]. The authors in Ref. [28] introduce the concept of curriculum learning (CL) in machine learning, where the training environment is shaped to help the agent learn more complex tasks, analogous to how humans learn, in a gradual fashion. This strategy, which can be viewed as a continuation method, was shown to accelerate convergence speed and obtain better local minima. Since then, many CL strategies have been developed, including predefined progressions, automated curriculum generation, and student–teacher frameworks, which have demonstrated improvements in training efficiency and task accuracy [29].

Reverse and end-game-first curricula are shown to accelerate training [30] and to be effective in robotics and control applications [31,32], helping in unlearnable tasks when structured progression is not used [33,34]. CL also is shown to help with generalization to unseen scenarios [35] and safety-critical training scenarios [36]. In spacecraft domains, CL has shown promise in improving constraint handling in fuel-optimal guidance and control, planetary landing, and rendezvous via reinforcement learning [37–39]. In Ref. [40], the authors present a comprehensive overview of CL strategies and their role in reinforcement learning.

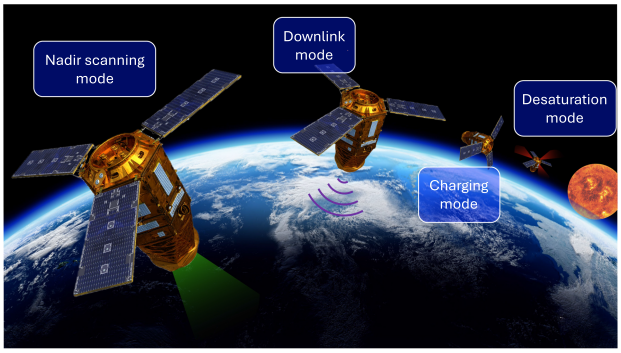


Fig. 1 Constellation of spacecraft being tasked with different actions. The spacecraft can have different parameters due to partial failures or degradation.

Curriculum is also shown to be an effective tool to improve the robustness of DRL policies. In Ref. [41], the authors generate curriculum tasks via a genetic algorithm that constructs evaluation environments, adding failed cases to the training set. This approach yields policies with up to eight-times fewer failures and comparable rewards. Similarly, in Ref. [42] the authors use genetic algorithms to generate curriculum tasks and obtain more robust policies that tend to outperform the baseline in terms of rewards both with and without external perturbations. In parallel, domain randomization (DR) has been used to improve policy generalization and reduce the simulation-to-real gap by randomizing agent and environment parameters during training [43–45]. Unlike CL, DR does not impose a structured progression of difficulty; instead, it exposes the agent to a wide variety of randomized scenarios from the beginning of training.

Although prior work has demonstrated the potential of DRL for spacecraft tasking, few studies have systematically examined how and to what extent the training environment should be modified to improve policy robustness while maintaining high performance across varying conditions. This paper addresses this gap through a comprehensive analysis of how different training environment enhancements (training structures, episode lengths, and randomization intervals) affect the robustness of DRL policies for spacecraft tasking in both nominal and degraded environments. Specifically, four training environment structures with varying intensity levels are evaluated, alongside the effects of different episode lengths and randomization intervals. Policies are evaluated under both nominal and degraded conditions on episodes up to 125-times longer than those used for training. Finally, the applicability of the proposed methods is demonstrated in a heterogeneous spacecraft system, where satellites operate under different parameters, showcasing the practical advantages and consistent performance of the resulting policies.

The remainder of the paper is organized as follows: Sec. II introduces the problem formulation and simulation environment; Sec. III presents the training environment enhancements; Sec. IV discusses the experimental results; and Sec. V concludes the paper.

II. Problem Formulation and Simulation Environment

This paper considers a satellite with an imaging instrument to collect data when imaging nadir. The spacecraft is equipped with a power and data storage subsystem and three reaction wheels to control its attitude. The spacecraft must have available storage space to store the collected data. To free up storage space, the spacecraft can downlink data to a ground station. The mission goal is to maximize the amount of data collected (time spent imaging nadir) while ensuring the spacecraft's safety by keeping the battery charge and reaction wheel speeds within operational limits.

The spacecraft is constrained to a circular orbit with a 500 km altitude and a 45 deg inclination. The spacecraft has four discrete actions available at each decision step, corresponding to four flight modes: charge, downlink, momentum management, and nadir scanning. The problem can be formulated as a sequential decision-making problem, where a policy is learned to map the spacecraft's observations to actions at every decision step to maximize the expected cumulative reward over the mission duration.

A. Partially Observable Markov Decision Process Formulation

The sequential decision-making problem can be formulated using a Markov decision process (MDP), where the next state s' depends only on the current state s and action a (Markov property [46]). When only partial information about the state space is available to the agent, the problem can be formulated as a partially observable MDP (POMDP). A POMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, Z, \gamma)$ [47]:

1) $\mathcal{S}$ is the state space. It includes states such as the satellite's battery charge, position, orientation, and states internal to the simulator (such as internal flight software states).

2) $\mathcal{A}$ is the set of actions that the agent can take. In this problem, the agent can choose between four discrete actions: charge (a_{charge}), downlink (a_{dlink}), reaction wheel momentum management (a_{desat}), and nadir scanning (a_{scan}).

3) $\mathcal{O}$ is observation space, which is a subset of the dimensions in $\mathcal{S}$, and includes information available for the agent and useful for the decision-making process, such as battery charge fraction, sensing instrument orientation, and time to next eclipse. Table 1 summarizes the observations made available to the agent and their normalization.

4) $T(s'|s,a)$ is the transition function that defines the probability of transitioning to the next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$. In this problem, the transition function is deterministic, and a generative model G is used, provided by a simulation engine, such that $s' = G(s,a)$.

5) $R(s,a,s')$ is the reward function that defines the reward obtained by the agent at every step. Rewards are awarded for the time spent in nadir scanning mode when within the relative angle and angular velocity limits during each time step k, given by $t_{\text{nadir}}^{(k)}$. The reward function is defined as

$$R(s,a,s') = \begin{cases} \frac{t_{\text{nadir}}^{(k)}}{T_{\text{episode}}} & \text{if } a = a_{\text{scan}}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

where T_{episode} is the total simulation time of the episode. Therefore, cumulative rewards are between 0 and 1, where 1 is the maximum reward obtained when the agent scans the nadir for the entire episode.

6) $Z(o|s,s',a)$ is the observation function that defines the probability of observing $o \in \mathcal{O}$ given s, s', and a. In this problem, the observation function is deterministic without uncertainty about its observations.

7) Note that γ is the discount factor that defines the importance of future rewards; $\gamma \in [0,1)$ ensures finite returns by discounting future rewards in infinite-horizon episodes.

The agent interacts with the environment by taking actions, receiving observations, and obtaining rewards. The agent's policy π is a mapping from observations to actions $a_k = \pi(o_k)$, and the goal is to find a policy that maximizes the expected sum of discounted rewards such that

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k, s_{k+1}) \middle| a_k = \pi(o_k), s_{k+1} = G(s_k, a_k)\right] \quad \forall\, s \in \mathcal{S} \tag{2}$$

where V is the value function. When using DRL to solve the problem, the policy is parameterized by a DNN.

The described POMDP formulation captures the essential elements of the spacecraft tasking problem and can be used in different instances of the problem, as discussed in [1,6], which consider the AEOS case with discrete point targets and safety aspects such as power consumption. The nadir scanning problem was selected instead to focus the analysis on the robustness and safety aspects of the problem. Different spacecraft configurations would be captured by modifying the generative model G and the observation space $\mathcal{O}$. Different mission goals could be represented by modifying the reward function R.

Table 1 Observation space

Parameter	Normalization
Angle between satellite's sensing instrument and nadir	π
Battery charge	Battery capacity
Buffer storage space used	Buffer capacity
Reaction wheel (RW) speeds	Maximum RW speed
Angle between solar panels and sun	π
Time until the next eclipse	Orbital period
Time until the end of the next eclipse	Orbital period
Time until next ground station pass	Orbital period
End of next ground station pass	Orbital period

B. Simulation Environment

The simulation was implemented in BSK-RL,[‡] which is a Python package for spacecraft tasking studies using RL [48]. BSK-RL combines the Basilisk[§] software with Gymnasium, a widely adopted RL library. Basilisk was used as the generative model for the simulation environment, which is a spacecraft simulation software written in C and C++ with a Python interface, making it fast for training and testing and easy to interact with [49]. The simulation accounted for the spacecraft dynamics, reaction wheel, power generation, and storage subsystems.

The goal of the agent was to maximize the time spent imaging nadir. Therefore, the scanning action tracked the nadir, pointing the camera to the Earth's surface. The scanning action obtained information from Earth when the relative attitude and angular velocity criteria were met and added it to the storage subsystem. If the storage unit was full, no more data could be added, and no more rewards could be awarded. To account for this, action downlink was available, which pointed the satellite's antenna to a ground station to downlink data stored in the satellite, accounting for transmission baud rate, amount of stored data, and access to a ground station.

In addition to a constant baseline power consumption representing essential subsystems, reaction wheels followed a nonlinear power consumption law that was a function of their angular velocity. The scanning instrument also consumed energy when in use. When tasking with the action charge, the satellite maneuvered to align its solar panels to maximize its power generation. Power generation was calculated by the underlying simulation in Basilisk, which accounted for eclipse and incidence angle. Therefore, the satellite could passively charge even when taking other actions. If the agent tasked with the charging action during an eclipse, it would not charge the battery.

Satellite attitude was controlled by a modified Rodrigues parameter [50] steering law using three reaction wheels positioned orthogonal to each other [51]. As the reaction wheels accumulated momentum due to external torque, leading to higher angular velocities and higher power consumption, a momentum management action was also available to the agent. The momentum management action aligned the spacecraft with the nadir and fired thrusters to reduce the momentum accumulated by the reaction wheels. The agent reached a terminal state when the battery level was below or equal to zero or any of the three reaction wheels exceeded their maximum angular speeds. Each action had a fixed duration of 180 s, which corresponded to the step duration.

Table 2 shows the satellite parameters used in the simulation. Parameters not listed here were set to the default values in BSK-RL. Parameters such as initial battery charge fraction and storage space fraction were randomized every episode and had two different ranges: a nominal range, where the bounds are within their nominal operation conditions, and a wide range, where the bounds are set to the maximum and minimum values of the parameters. A degraded version of the environment was also used to test the policy's robustness. The degraded environment was obtained by reducing the battery capacity by 50%, the solar panel efficiency by 25%, and setting the external torque as three times the nominal value.

An example script with a nadir scanning spacecraft demonstrating how the curriculum learning was implemented and applied is available at the BSK-RL website.[¶]

III. Training with Mission Environment Enhancements

A. Training Setup

The policy was obtained using the asynchronous PPO (APPO) algorithm [52] provided by the RLlib package [53]. The training

[‡]https://avslab.github.io/bsk_rl/.

[§]https://avslab.github.io/basilisk/.

[¶]https://avslab.github.io/bsk_rl/examples/curriculum_learning.html.

Table 2 Nominal satellite parameters

Parameter	Value
Altitude	500 km
Inclination	45 deg
Battery capacity	400 W · h
Base power consumption	10 W
Data storage capacity	5 GB
Relative angle limit for imaging	22.8 deg
Relative angular rate limit for imaging	0.1 rad/s
Reaction wheel maximum speed	6000 RPM
External torque magnitude	0.2 mN · m
Thruster power draw	80 W
Instrument baud rate	0.5 MB/s
Instrument power draw	30 W
Transmitter baud rate	112 MB/s
Transmitter power draw	25 W
Solar panel efficiency	20%
Initial storage space fraction	[0.0, 1.0]
Initial battery charge fraction (nominal)	[0.375, 0.625]
Initial battery charge fraction (wide)	[0.0, 1.0]
Initial reaction wheel speeds (nominal)	[−4000, 4000] RPM
Initial reaction wheel speeds (wide)	[−6000, 6000] RPM

Table 3 Training parameters

Parameter	Value
Number of workers	32
Number of training steps	$5 \cdot 10^6$
Learning rate	$3 \cdot 10^{-5}$
Training batch size	10,000
Minibatch size	250
Epochs	50
PPO clipping parameter	0.2
Gradient clipping parameter	0.5
Generalized advantage estimation parameter (λ)	0.95
Neural network	Two 512-neuron layers
Discount factor	0.999

was performed at the University of Colorado Research Computing (CURC) using 32 cores and limited to 5 million steps. Table 3 provides more information about the training hyperparameters (parameters not listed here were set as the default values in RLlib in version 2.6.3). For a comprehensive hyperparameter analysis on DRL algorithms for spacecraft tasking, refer to Ref. [18].

A failure penalty of −1 was used, being added to Eq. (1) when the satellite reached a terminal state. The value of −1 was used to limit the cumulative reward range to [−1, 1], similar to Ref. [27].

B. Training on Enhanced Environments

Different modifications to the training environment are explored to improve the robustness of the learned policies. More specifically, constantly degraded environments (as in Ref. [27]), two CL structures, and DR are investigated. These structures are combined with different training episode lengths and randomization intervals. Throughout the paper, these modifications are referred to as training environment enhancements.

Curriculum learning provides a methodology to shape the training environment to help the agent learn more complex tasks [28]. It starts with simple tasks and gradually increases the difficulty of the environment. In this work, two different curricula were tested: direct and inverse. The direct curriculum starts with a nominal environment and gradually increases the difficulty toward a harder environment. The inverse curriculum starts with a harder environment and gradually decreases the difficulty toward the nominal case.

Defining the difficulty of an environment can be subjective and usually requires expert knowledge of the problem. In this work, difficulty is associated with the satellite's battery capacity and external torque, which affect survivability. A nominal difficulty corresponds to the nominal environment; a higher difficulty corresponds to reduced battery capacity and, when applicable, increased external torque. Therefore, tasks can be seen as different POMDPs containing distinct transition probabilities because of the different battery capacities and external torques.

Because direct curriculum initially presents the agent with the nominal environment, in which it is easier to survive, it is expected that the agent will first learn how to obtain rewards through scanning actions. Later, as the environment gets more difficult, the agent will improve its survivability to maximize cumulative rewards. On the other hand, the inverse curriculum presents the agent with a harder environment at the beginning, where the agent has a stronger incentive to avoid terminal states and learn how to survive. As the number of training steps increases and the environment gets easier, the agent will learn how to optimize rewards through imaging actions. Inverse curriculum can be seen similarly to the reverse curriculum [30] or the end-game-first approach [54], where terminal states are presented first, making the agent learn how to achieve or avoid them.

A constant training structure, where the difficulty is constantly higher than nominal (a constantly degraded environment) throughout the training process, is also tested, inspired by the work of [27]. A DR training method was used in addition to the three other training structures. When using DR, the agent is trained in a variety of environments with different parameters, such as battery capacity and external torque, which are uniformly sampled at each environment reset. DR differs from CL mainly because of its lack of structured progress of difficulty. While the structured sequence of CL can help the agent learn more complex tasks, tasks learned in the early stages of training may be forgotten as the environment gets more complex. Therefore, the agent may not be able to generalize well. DR, on the other hand, can help the agent learn a policy that generalizes better to different environments and will be used for comparison.

The three intensity levels of curriculum applied to battery capacity correspond to decreases of 20% (B-Low), 40% (B-Med), and 60% (B-High) in battery capacity. When the battery capacity change was combined with an increase in external torque, the three intensity levels corresponded to a decrease of 20% in battery and an increase of 4 times in external torque (BT-Low), a decrease of 40% in battery capacity and a 6-times increase in external torque (BT-Med), and a 60% decrease in battery capacity and an 8-times increase in external torque (BT-High). The curriculum cases varying in only the battery capacity (B cases) were trained exclusively with 90-step episodes. Table 4 summarizes the different intensities. The battery and torque (BT) cases were trained with both 90- and 450-step episodes to identify the impact of longer training episodes.

Training structures are defined by their initial difficulty (at the beginning of training) and final difficulty (at the end of training). Each training run is limited to 5 million steps (total number of environment interactions, where the policy needs to make a decision and the environment is propagated for a step). Therefore, training progress p is defined as the fraction of steps completed, k, such that $p = k/(5 \cdot 10^6)$. In practice, each training run consists of multiple episodes (each with a maximum length), and the battery capacity and external torque are updated at each environment reset according to the current training progress and training structure, and initial conditions are randomly sampled. For the

Table 4 Definition of low, medium, and high intensity levels for battery-only (B) and battery and torque (BT) curriculum cases

Case	Battery variation	Torque variation
B-Low	20% decrease	— —
B-Med	40% decrease	— —
B-High	60% decrease	— —
BT-Low	20% decrease	4× increase
BT-Med	40% decrease	6× increase
BT-High	60% decrease	8× increase

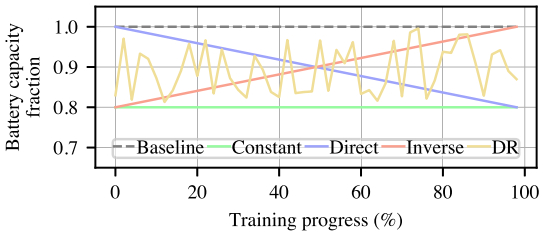


Fig. 2 Illustration of the four investigated training approaches with battery capacity variation during training (B-low).

Direct case with B-Low intensity, for example, the battery starts at the nominal capacity and linearly decreases to 80% of the nominal capacity at the end of training (increasing the difficulty), such that $\text{capacity}(k) = (1 - 0.2p)\text{capacity}_{\text{nominal}}$. The Inverse B-Low case linearly increases battery capacity toward the nominal value, such that $\text{capacity}(k) = (0.8 + 0.2p)\text{capacity}_{\text{nominal}}$. The DR B-Low method samples battery capacity uniformly between 80 and 100% of the nominal capacity at each environment reset. The battery capacity remains at 80% for the Constant B-Low case. When torque variation is included, the same linear schedule is applied to the magnitude of the external torque. Figure 2 illustrates the three different curriculum approaches and the DR method as a function of training progress for the B-Low case.

Table 5 summarizes the different training structures and intensities used for training. Each approach was combined with increased episode lengths and nominal and wide parameter initialization ranges. Baseline policies (without any training structures) were also trained. Each combination of structure, intensity, initialization range, and episode length was trained three times to account for the randomness in the training process, resulting in 204 trained policies. All policies were trained from scratch for 5 million steps, without any pretraining or fine-tuning. The training structure, intensity, episode length, and initialization range were kept constant throughout the training of each policy.

Table 5 Summary of all trained policies configurations

Structure	Init. range	Intensity	Episode length, steps 90	180	270	360	450	540
Baseline	Nominal	— —	✓	— —	— —	— —	✓	— —
	Wide	— —	✓	— —	— —	— —	✓	— —
Direct	Nominal	B-Low	✓	— —	— —	— —	— —	— —
		B-Med	✓	— —	— —	— —	— —	— —
		B-High	✓	— —	— —	— —	— —	— —
		BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	✓	✓	✓	✓	✓
	Wide	BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —
Inverse	Nominal	B-Low	✓	— —	— —	— —	— —	— —
		B-Med	✓	— —	— —	— —	— —	— —
		B-High	✓	— —	— —	— —	— —	— —
		BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —
	Wide	BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —
Constant	Nominal	B-Low	✓	— —	— —	— —	— —	— —
		B-Med	✓	— —	— —	— —	— —	— —
		B-High	✓	— —	— —	— —	— —	— —
		BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —
	Wide	BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —
DR	Nominal	B-Low	✓	— —	— —	— —	— —	— —
		B-Med	✓	— —	— —	— —	— —	— —
		B-High	✓	— —	— —	— —	— —	— —
		BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —
	Wide	BT-Low	✓	— —	— —	— —	✓	— —
		BT-Med	✓	— —	— —	— —	✓	— —
		BT-High	✓	— —	— —	— —	✓	— —

The intensities of training structures were manually set based on previous EOS studies that used enhanced environments to improve policy robustness and facilitate learning of the safety aspects [6,17,25,27]. Different environments have different levels of difficulty for the agent to survive and likely need specific tuning of the intensity levels.

IV. Effect of Training Environment Enhancements on Policy Robustness and Performance

The following section includes comparisons between policies trained with the different training environment enhancements. The policies were tested in the nominal and degraded versions of the environment. The degraded environment has 50% reduced battery capacity, 25% reduced solar panel efficiency, and a three-times increase in external torque compared to the nominal environment, as discussed in Sec. II.B. Policies were also tested with different episodes up to 125-times longer than training episodes to evaluate the robustness and safety aspects of long-term deployment. Initially, Sec. IV.A presents results comparing the different enhancements. Later, Sec. IV.B discusses the training time for the different cases. Lastly, Sec. IV.C illustrates the robustness of the policies obtained with the training environment enhancements when tested in a heterogeneous spacecraft system.

A. Training Structure Comparison

To analyze the effect of training environment enhancements on policy robustness and performance, a sequence of comparisons is presented. First, it is explored if policies trained with some structured training (constantly degraded, CL, or DR) exhibit higher robustness compared to those trained without these techniques, as measured by lower failure rates and higher cumulative rewards in both nominal and degraded environments (Sec. IV.A.1). Then, it is investigated if policies trained with battery and torque variations (BT cases) outperform their counterparts trained with only battery variations (B cases) in terms of robustness and performance (Sec. IV.A.2). Later, it is analyzed if policies trained with longer episodes (450 steps) outperform their counterparts trained with shorter episodes (90 steps) in terms of robustness and performance (Sec. IV.A.3). Lastly, it is examined if policies trained with wider initialization ranges outperform their counterparts trained with nominal initialization ranges in terms of robustness and performance (Sec. IV.A.4).

The 204 trained policies were tested in environments with 90, 450, 2250, and 11,250 three-minute steps (corresponding to 2.84, 14.2, 71.05, and 355.26 orbits, respectively), in both nominal and degraded versions of the environment. Each combination of policy, environment, and episode length was tested in 100 simulation runs with different random initial conditions (resulting in 800 test runs per trained policy). The results are shown in whisker plots. Each box contains the 25th and 75th percentiles, while the median is shown in an orange line and the mean in a black diamond. The whiskers extend to the points within 1.5 times the interquartile range from the 25th and 75th percentiles; points outside this range are shown as individual gray points. Each whisker contains 300 test runs (100 for each of the 3 policies with the same training structure, intensity, episode training length, and initialization range). The combined testing results correspond to 3001 years of simulated time.

Statistical analyses were conducted to compare policy performance. Because the performance data were non-normally distributed and exhibited unequal variances across policy configurations, nonparametric tests were applied. Group differences were evaluated

using the Kruskal–Wallis test, and pairwise comparisons were performed using Dunn's post hoc test with Holm correction to control the familywise error rate. Pairwise comparisons were limited to policies evaluated under identical conditions (i.e., same episode length, environment, and initialization range) to ensure fair and consistent evaluation. Results report the p-values (p) obtained from these tests, the median cumulative reward (M), and the median difference between policies (δM).

1. *Impact of Training Structure*

Initially, the impact of the four different training structures was investigated when considering only variation in battery. For this case, the policies were trained with 90-step-long episodes. While very few differences are seen when testing these policies in environments with 90, 450, and 2250 steps, these differences become more visible in the longest test case with 11,250 steps. For clarity, the reported (undiscounted) cumulative reward values exclude the -1 failure penalty applied during training, as including it can obscure differences among policies with distinct behaviors (e.g., conservative with no failures vs aggressive with failures). Excluding the penalty allows for a direct interpretation of the reward as the fraction of time spent in nadir scanning. Failures—which incur a -1 penalty during training—and failure type are instead reported separately. Thus, a cumulative reward of 0 corresponds to 0% nadir scanning time, while 1 indicates 100%.

Figure 3 reports the cumulative rewards obtained by the policies tested in the longer nominal environment, with 11,250 steps, the number of orbits completed per case, and the percent of battery and reaction wheel (RW) related failure across all 300 test runs. Whereas the white background indicates the baseline case, green represents the constant difficulty cases, blue is the direct curriculum, red is the inverse curriculum, and yellow is the DR cases. The results in Fig. 3 indicate that the failure rate of the baseline policies was higher than those trained with enhancements (except for the Direct B-High case). Additionally, the cumulative reward obtained by policies trained with enhanced training environments was higher than the baseline. All pairwise comparisons with the baseline were statistically significant ($p < 0.001$) and showed higher median cumulative rewards ($\delta M = 0.023$ on average), except for the Direct and Constant B-High cases, which did not differ significantly from the baseline. This result indicates that, although policies were trained in enhanced environments, they are outperforming the baseline case in the nominal environment without decreases in cumulative reward.

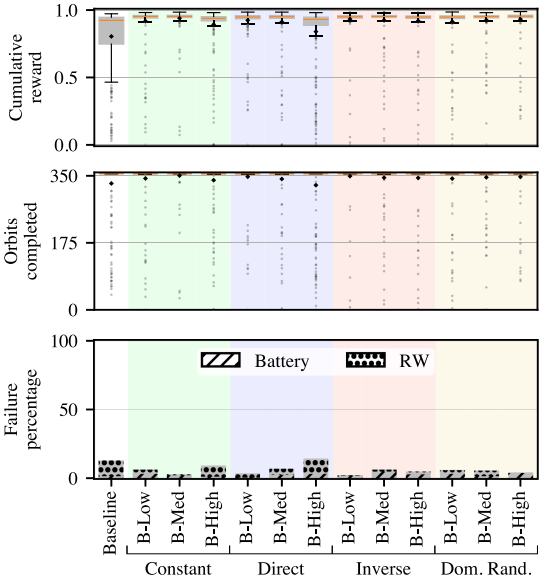


Fig. 3 Policies trained with battery variation in 90-step (2.84 orbits) episodes and tested in the nominal environment with 11250-step (355.25 orbits) episodes.

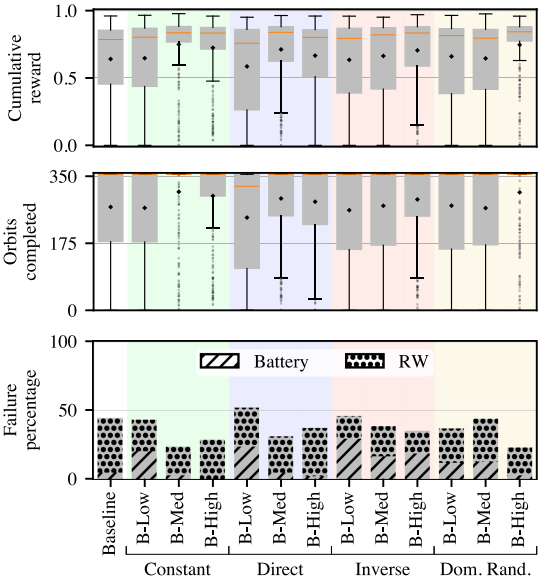


Fig. 4 Policies trained with battery variation in 90-step (2.84 orbits) episodes and tested in the degraded environment with 11250-step (355.25 orbits) episodes.

Different from Fig. 3, Fig. 4 shows the results for the degraded version of the environment with 11,250-step episodes. In this case, the baseline showed 44% failure percentage. The three best-performing cases were Constant B-Med ($p < 0.001$, $\delta M = 0.049$), Direct B-Med ($p < 0.01$, $\delta M = 0.053$), and DR B-High ($p < 0.001$, $\delta M = 0.057$), which showed statistically significant improvements over the baseline and 20%, 13%, and 21% difference in failure rates, respectively. The other cases did not differ significantly from the baseline in terms of rewards ($p > 0.05$). Overall, these results indicate the benefits of structured training approaches in both nominal and degraded environments, without loss of performance.

Whereas the cases trained with inverse curriculum showed a balanced number of battery and torque failures, the cases trained with constant difficulty, direct curriculums, and DR showed mostly reaction-wheel-related failures. This indicates that there is room to add torque variation to the training structure to help the agent learn how to avoid unsafe states related to maximum wheel speeds.

2. *Battery vs Battery and Torque Variation*

Varying the battery capacity with the difficulty of the environment is expected to enhance the policy robustness with respect to degraded battery capacities. When also adding external torque variation to the training structure, it is expected that the policy will be more robust to external disturbances too, leading to fewer reaction-wheel-related failures. To investigate this, the policies trained with only battery variation (discussed in Sec. IV.A.1) were compared to the cases trained with both variations. Figure 5 shows the results for policies trained with 90-step episodes with and without torque variation. The trained policies were tested in the degraded environment with 11,250-step episodes. The Direct BT-High ($p < 0.05$, $\delta M = 0.035$) and DR BT-Med ($p < 0.01$, $\delta M = 0.054$) showed statistically significant improvements over their battery-only counterparts in terms of increased rewards and reduced failure rate (17 and 19% difference, respectively). All other pairwise comparisons between B and BT cases did not show statistically significant differences in terms of rewards ($p > 0.05$). The constant case showed minor improvements (2% difference on average), whereas the inverse curriculum case showed an increase in failure rates (3.7% difference on average), especially for the low-intensity case. DR showed mixed results, with the low- and high-intensity cases showing an increase in failure rates (11 and 7% difference, respectively), while the medium-intensity case showed a decrease of 45% in failure rates (19% difference) accompanied by a 6.5% increase in median rewards. Interestingly, cases that benefitted the most from

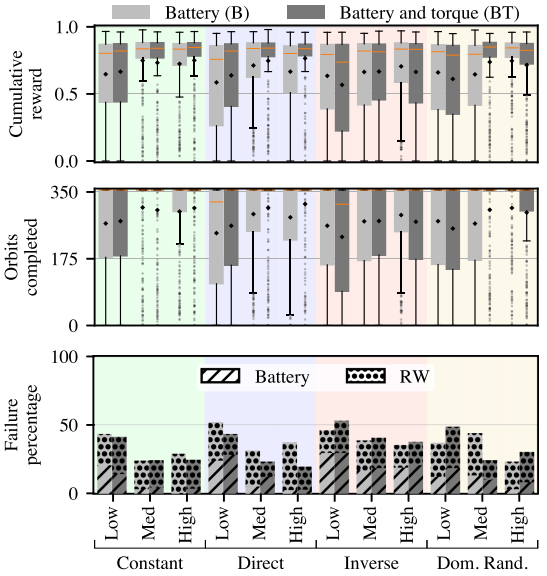


Fig. 5 Policies trained with only battery (B) and battery and torque (BT) variations. Tested in the degraded environment with 11,250-step (355.25 orbits) episodes.

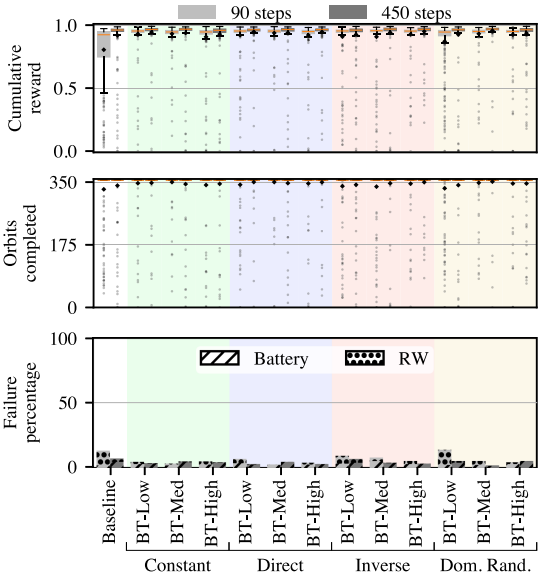


Fig. 6 Policies trained with 90-step (2.84 orbits) and 450-step (14.2 orbits) episodes and tested in the nominal environment with 11,250-step (355.25 orbits) episodes.

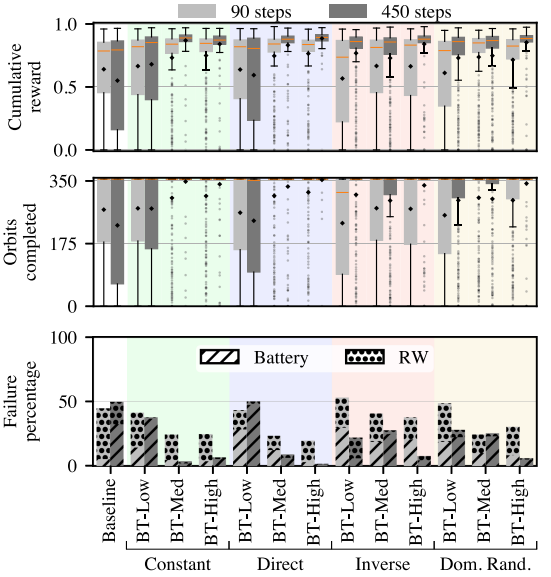


Fig. 7 Policies trained with 90-step (2.84 orbits) and 450-step (14.2 orbits) episodes and tested in the degraded environment with 11,250-step (355.25 orbits) episodes.

the addition of torque variation showed a decrease in reaction wheel-related failure, as intended.

The results presented in Fig. 5 suggest that adding torque variation to the training structure can help the agent learn how to avoid unsafe states related to reaction wheel speeds, reducing the number of related failures. Nevertheless, this outcome is linked to the training structure. Increasing the external torque over training, as in the direct curriculum case, allows the agent to learn other basic aspects of the simulation first. The constant presence of increased external torque in the constant case shows few improvements. The remaining analyses focus exclusively on cases that include both battery and torque variations, as the results indicate either improved or equivalent performance compared to the corresponding battery-only cases, with no evidence of performance degradation.

3. *Episode Length in Training*

While agents are usually trained for a few orbits, they are deployed for many more orbits, as satellite missions usually range from months to years. Still, it is impractical to train with such long episodes and randomize the environment enough to ensure generalization due to computational costs. While the training algorithm should use the estimate of the value function to bootstrap the return in shorter (truncated) environments, training with longer episodes should provide the agent with more experience in the region of the state space closer to its nominal operation. To investigate the impact of training with longer episodes, policies trained with battery and torque variations were trained with 90- and 450-step episodes. Figure 6 shows the obtained results when deploying in the nominal environment with 11,250-step episodes. The results indicate that all policies trained with 450-step episodes achieved statistically significant ($p < 0.001$) differences compared to their counterparts trained with 90-step episodes in terms of cumulative rewards, with $\delta M = 0.015$ increase on average. All cases were statistically significantly different from the 90-step baseline ($p < 0.05$) with $\delta M = 0.034$ increase on average, whereas the Constant BT-Low and BT-Med, Inverse BT-Low and BT-High, and DR BT-Med outperformed the 450-step baseline ($p < 0.01$).

More significant differences are seen when comparing those policies in the degraded environment, as shown in Fig. 7. Extending the training episode length from 90 to 450 steps leads to a 44% decrease in failure rates on average across all cases. Cases trained with 450-step episodes showed statistically significant ($p < 0.01$) differences compared to their 90-step counterparts with an average increase in the median cumulative reward of $\delta M = 0.057$, except for the baseline, Constant BT-Low, Direct BT-Low, and DR BT-Med (no statistically significant differences). Cases trained with 450-step episodes and enhancements showed statistically significant differences when compared to the 90-step and 450-step baselines ($p < 0.001$), except for Direct BT-High. The best-performing policies trained with 450-step episodes were trained with Constant BT-Med ($M = 0.888$) and BT-High ($M = 0.871$), Direct BT-Med ($M = 0.879$) and BT-High ($M = 0.893$), Inverse BT-High ($M = 0.879$), and DR-High ($M = 0.884$) structures. Higher cumulative rewards (higher median value and less variance, as seen in the whiskers being more concentrated at the top) were accompanied by a decrease in failures. For example, Direct BT-High (450 steps) showed only a 1.6% failure percentage, 96% lower than the 90-step baseline and 91% lower than its 90-step counterpart, with respective 13.7% and 6.7% gains in median cumulative reward.

Despite the improvements seen in cases trained in enhanced environments, the baseline policy showed an increased number of failures when trained with longer episodes. Interestingly, the cases

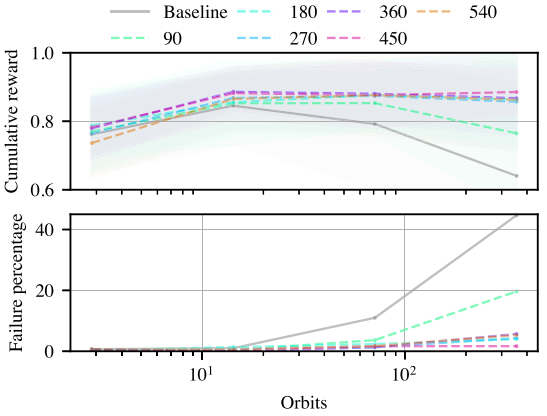


Fig. 8 Different training episode lengths using direct BT-High curriculum and tested in the degraded environment with different episode lengths.

trained with longer episodes showed fewer reaction wheel-related failures, which is expected as there is more time for the buildup of momentum due to external torque during training and the need for momentum management actions, leading to more learning opportunities. These results demonstrate the benefits of extending the training episode length, with no case showing a statistically significant decrease in performance compared to their shorter episode counterparts in both nominal and degraded environments.

Figure 8 illustrates the results for the case with the Direct BT-High curriculum under different episode lengths during training and testing. Policies were trained with 90-, 180-, 270-, 360-, 450-, and 540-step episodes. These resulting policies were tested in the degraded environment with different numbers of orbits. The baseline case showed consistently worse performance than cases trained with curriculum. Policies trained with 180, 270, 360, and 450 steps presented similar performance across the number of orbits. Whereas the case with 90 steps showed a decrease in performance in longer testing episodes, the case with 540 steps showed a more conservative behavior in 90-step testing episodes. The number of failures for the baseline case was significantly higher than for all curriculum cases. These results highlight the effect of extending training episode length on the performance of the policies, but also an insensitivity to the exact length of the training episode. Moreover, it indicates that continuously increasing the training episode does not lead to continuous improvements, especially when it comes at the expense of less randomization during training while preserving maximum training time or maximum number of training steps.

4. *Nominal vs Wide Initialization Range*

While extending the length of training episodes can return a better experience in terms of the learning perspective, initializing the environment over a wider range of initial conditions can help to better estimate the value function across the state space. To investigate the impact of the initialization range in the training process, policies were trained with the nominal and wide initialization ranges, as indicated in Table 2. Figure 9 shows the results for policies trained with 90-step episodes with both nominal and wide initialization. The wide initialization benefited the constant, direct, and DR cases the most in terms of reduced failures (83% decrease on average). Nevertheless, the Direct BT-High case presented an associated decrease in reward, indicating a more conservative behavior. Overall, the cases with wide initialization differed from their counterparts in terms of cumulative reward with $p < 0.01$ except for Constant BT-High, Inverse BT-Med, and DR BT-Med. Nevertheless, the changes in cumulative reward were not consistent across cases, with some showing increases and others decreases.

Figure 10 shows the results when combining longer training episodes with a wide initialization range. While some cases benefited from the wide initialization range in terms of failure rate, the cumulative reward was affected. Interestingly, policies that suffered the largest decrease in cumulative reward tended to be policies with

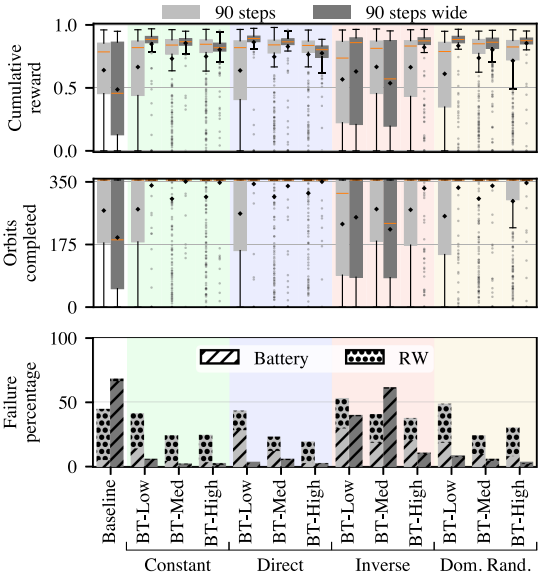


Fig. 9 Policies trained with 90-step (2.84 orbits) episodes and nominal and wide initialization and tested in the degraded environment with 11,250-step (355.25 orbits) episodes.

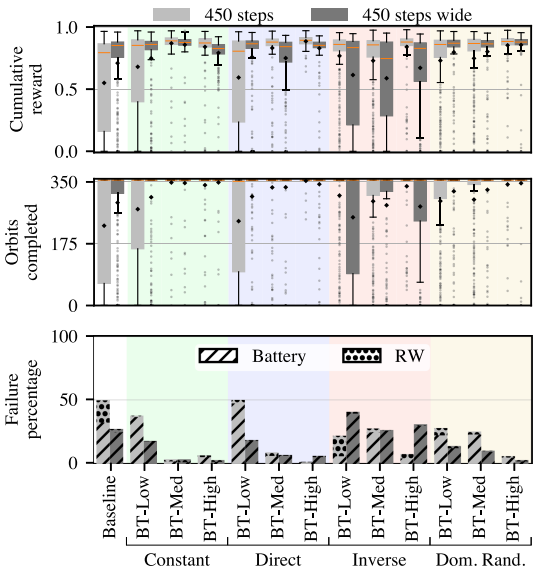


Fig. 10 Policies trained with 450-step (14.2 orbits) episodes and nominal and wide initialization and tested in the degraded environment with 11,250-step (355.25 orbits) episodes.

more intense curriculums. This outcome indicates that the training environments were too hard for the agents to generalize well. On the other hand, the baseline case significantly benefited from the wide initialization, showing fewer failures and higher median cumulative rewards ($\delta M = 0.057$) when compared to the nominal case ($p < 0.001$). No statistically significant differences were observed in the DR and Constant BT-Low or BT-Med cases.

While using the wide initialization helped to expose the agent to more diverse parts of the state space, it also led to more failures. Some agents ended up being overexposed to terminal states, resulting in a more conservative policy—showing great robustness but also a decrease in cumulative reward. Although finding an adequate initialization range can be challenging, it can also improve the policy's robustness and generalization. Cases with low intensity and shorter training episodes benefitted the most from the wider initialization.

Despite the results indicating that the use of training environment enhancements leads to more robust policies, all cases showed at

least one failure when deployed in 11,250-step episodes. In a real deployment, the selected policy could be deployed with shields, a technique that monitors the actions selected by the agent and can provide safety guarantees [55]. Notably, [56] indicates that even when using shields in spacecraft tasking, policies with better safety management obtain better performance, indicating the importance of a robust training process.

B. Training Time

Although different training environment enhancements showed different performance in terms of rewards and failures, it is important to consider the training time required for each policy. If a given enhancement takes much longer to finish training, the baseline case could be trained for more steps instead. Figure 11 shows the training time for the policies grouped by training structure, intensity, episode length, and initialization range. The results indicate that policies trained with longer episodes took more time to finish training. Training with a wider initialization range takes longer than with a nominal initialization, which is linked to more satellites starting with initial conditions close to terminal states, which is more likely to lead to a failure and a reset of the environment. Still, the relation between the time to propagate a step and perform a reset is heavily dependent on the simulation environment.

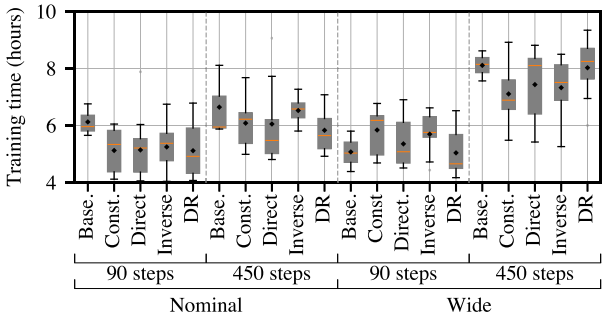
C. Policy Robustness in a Heterogeneous Spacecraft System

Robust policies are useful in scenarios where satellite parameters vary over time due to unexpected events or natural degradation. This effect is pronounced when considering a system of multiple agents with identical spacecraft. In reality, each spacecraft will have slightly different parameters, with differences increasing over time. While tailoring a policy for each agent is ideal, it is often impractical due to the number of agents and the time required to train each policy in a constantly evolving environment. Instead, the agents could use identical, robust policies.

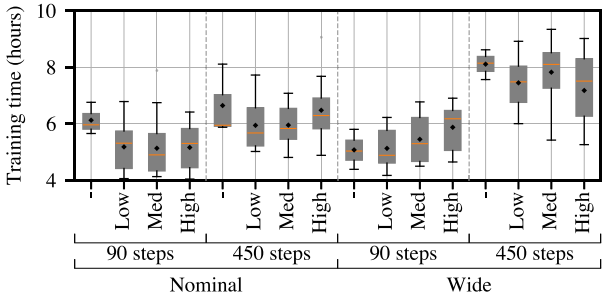
To demonstrate the usefulness of such robust policies in a heterogeneous spacecraft system, five different satellites were considered, each with a unique combination of battery capacity and external torque. Whereas the first four satellites had fixed parameters, the fifth satellite had time-varying battery capacity and external torque to represent progressive degradation. Table 6 shows the parameters used for each satellite with respect to the nominal values shown in Table 2.

The best baseline policy was selected, as well as the best policy trained with a training structure. These policies were selected based on cumulative rewards and failure percentage in the degraded environment with 11,250 steps. The best-performing baseline policy was the second training run with 450-step episodes and wide initialization ranges (0.75 mean reward and 312.87 mean orbits alive). The best policy with a curriculum was the third training run of the case trained with 450-step episodes and the high-intensity direct curriculum (0.88 mean reward and 351.71 mean orbits alive) with the nominal initialization range.

Figure 12 shows the results obtained for the heterogeneous system in 100 test runs. If a satellite failed, the episode was still propagated forward until the failure of all satellites or the maximum episode duration was reached. The policy in blue represents the case trained with high-intensity direct curriculum, while the policy in gray represents the baseline. The case trained with curriculum showed 63.8% fewer failures and $\delta M = 0.026$ (2.8% increase) on average. Besides, none of its failures occurred in the first hundred orbits, as in some cases using the baseline policy. Despite presenting more failures than the baseline case in the EO-0 satellite, the policy trained with curriculum obtained higher median cumulative rewards. The largest difference was seen in the EO-4 satellite with time-varying parameters, with the policy trained with curriculum showing significantly fewer failures (98% reduction) and higher median cumulative rewards ($\delta M = 0.045$). All differences were statistically significant ($p < 0.001$) when using a Mann–Whitney U test for each satellite configuration with Holm correction. This result highlights the usefulness of a robust policy capable of handling different and time-varying parameters in spacecraft.



a) Policies grouped by training structure, episode length, and initialization range



b) Policies grouped by intensity, episode length, and initialization range

Fig. 11 Policies training time.

Table 6 Satellite parameters for the heterogeneous system

	Satellite				
Parameter	EO-0	EO-1	EO-2	EO-3	EO-4
Battery capacity	100%	100%	50%	80%	100% → 40%
External torque	100%	400%	100%	200%	100% → 1,000%

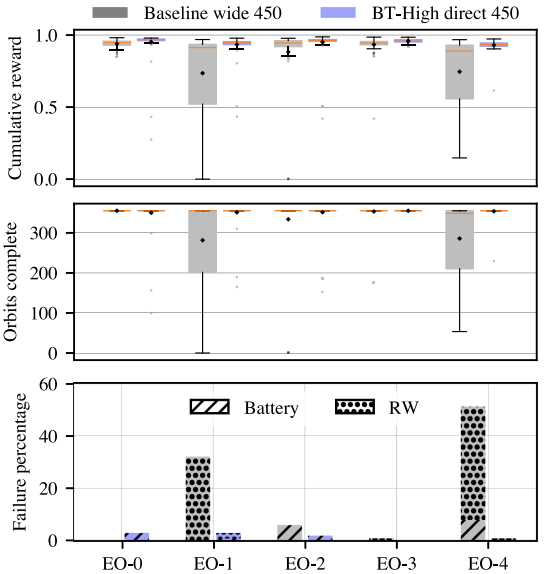


Fig. 12 Results for a heterogeneous spacecraft system. Gray represents the best baseline policy, while blue represents the best policy trained with curriculum.

V. Conclusions

This work demonstrates that enhanced training environments can significantly improve the robustness of deep reinforcement learning (DRL) policies for autonomous Earth-observing satellite tasking. Policies trained with enhanced environments not only withstand battery degradation and increased external torque but also maintain high performance when deployed for up to 125-times longer than their training episodes—a crucial factor for real-world, long-term missions.

Among the strategies tested, direct CL and constant difficulty, as well as DR, yielded the most robust results, with the Direct BT-High case showing up to a 96% decrease in failures and a 13.7% increase in median return compared to the baseline in the degraded environment. Training structures varying in both battery and torque did not yield statistically significant overall performance differences, though some cases showed up to a 6.5% increase in median cumulative reward and a 19% reduction in failure percentage compared to their battery-only counterparts. Longer training episodes showed a 44% decrease in failures on average, and wider initialization ranges improved generalization but sometimes led to conservative behaviors, especially with more intense training structures. The value of robust policies was further demonstrated in a heterogeneous spacecraft scenario, where a single policy successfully adapted to agents with varying and time-evolving parameters, showing up to a 63.8% reduction in failures and a 2.8% increase in median rewards on average.

While this work presents a systematic analysis of training environment enhancement strategies for robust policy training, some limitations remain. In particular, the training structures and parameter ranges were manually defined, and their optimal configuration is likely to be environment-specific. As such, structure, intensity, episode length, and randomization ranges should be treated as tunable hyperparameters to be optimized based on mission-specific constraints and goals.

Acknowledgments

This work is partially supported by the Air Force Research Lab grant FA9453-22-2-0050. This work utilized the Alpine high-performance computing resource at the University of Colorado Boulder. Alpine is jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, Colorado State University, and the National Science Foundation (award 2201538). Artificial intelligence tools were used to help improve the grammar and clarity of the paper. Final editing and approval were performed by the authors.

References

[1] Stephenson, M. A., Mantovani, L. Q., and Schaub, H., "Learning Policies for Autonomous Earth-Observing Satellite Scheduling over Semi-Markov Decision Processes," *Journal of Aerospace Information Systems*, Vol. 22, No. 9, 2025, pp. 789–799. https://doi.org/10.2514/1.I011649

[2] Zhao, X., Wang, Z., and Zheng, G., "Two-Phase Neural Combinatorial Optimization with Reinforcement Learning for Agile Satellite Scheduling," *Journal of Aerospace Information Systems*, Vol. 17, No. 7, 2020, pp. 346–357. https://doi.org/10.2514/1.I010754

[3] Harris, A., Valade, T., Teil, T., and Schaub, H., "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2022, pp. 611–626. https://doi.org/10.2514/1.A35169

[4] Herrmann, A., and Schaub, H., "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 59, No. 5, 2023, pp. 5235–5247. https://doi.org/10.1109/TAES.2023.3251307

[5] Chun, J., Yang, W., Liu, X., Wu, G., He, L., and Xing, L., "Deep Reinforcement Learning for the Agile Earth Observation Satellite Scheduling Problem," *Mathematics*, Vol. 11, No. 19, 2023, p. 4059. https://doi.org/10.3390/math11194059

[6] Herrmann, A., Stephenson, M. A., and Schaub, H., "Single-Agent Reinforcement Learning for Scalable Earth-Observing Satellite Constellation Operations," *Journal of Spacecraft and Rockets*, Vol. 61, No. 1, 2023, pp. 114–132. https://doi.org/10.2514/1.A35736

[7] Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., and Peters, J., "Robust Reinforcement Learning: A Review of Foundations and Recent Advances," *Machine Learning and Knowledge Extraction*, Vol. 4, No. 1, 2022, pp. 276–315. https://doi.org/10.3390/make4010013

[8] Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.-M., and Bataille, N., "Selecting and Scheduling Observations of Agile Satellites," *Aerospace Science and Technology*, Vol. 6, No. 5, 2002, pp. 367–381. https://doi.org/10.1016/S1270-9638(02)01173-2

[9] Bianchessi, N., Cordeau, J.-F., Desrosiers, J., Laporte, G., and Raymond, V., "A Heuristic for the Multi-Satellite, Multi-Orbit and Multi-User Management of Earth Observation Satellites," *European Journal of Operational Research*, Vol. 177, No. 2, 2007, pp. 750–762. https://doi.org/10.1016/j.ejor.2005.12.026

[10] Eddy, D., and Kochenderfer, M. J., "A Maximum Independent Set Method for Scheduling Earth-Observing Satellite Constellations," *Journal of Spacecraft and Rockets*, Vol. 58, No. 5, 2021, pp. 1416–1429. https://doi.org/10.2514/1.A34931

[11] Wang, X., Song, G., Leus, R., and Han, C., "Robust Earth Observation Satellite Scheduling with Uncertainty of Cloud Coverage," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 3, 2020, pp. 2450–2461. https://doi.org/10.1109/TAES.2019.2947978

[12] Wang, X., Gu, Y., Wu, G., and Woodward, J. R., "Robust Scheduling for Multiple Agile Earth Observation Satellites Under Cloud Coverage Uncertainty," *Computers & Industrial Engineering*, Vol. 156, June 2021, Paper 107292. https://doi.org/10.1016/j.cie.2021.107292

[13] Gorr, B., Jaramillo, A. A., Gao, H., Selva, D., Mehta, A., Sun, Y., Ravindra, V., David, C. H., and Allen, G. H., "Decentralized Satellite Constellation Replanning for Event Observation," *Journal of Spacecraft and Rockets*, Vol. 62, No. 4, 2025, pp. 1–19. https://doi.org/10.2514/1.A36143

[14] Aguilar Jaramillo, A., Gorr, B. J., Gao, H., Mehta, A., Sun, Y., Ravindra, V., David, C., Allen, G., and Selva, D., "Decentralized Consensus-Based Algorithms for Satellite Observation Reactive Planning With Complex Dependencies," *AIAA SCITECH 2025 Forum*, AIAA Paper 2025-1148, 2025. https://doi.org/10.2514/6.2025-1148

[15] Harris, A., Teil, T., and Schaub, H., "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *AAS Spaceflight Mechanics Meeting*, AAS Paper 19-447, Springfield, VA, 2019.

[16] Harris, A., and Schaub, H., "Deep On-Board Scheduling for Autonomous Attitude Guidance Operations," *AAS Guidance, Navigation and Control Conference*, AAS Paper 12-117, Springfield, VA, 2020.

[17] Herrmann, A. P., and Schaub, H., "Monte Carlo Tree Search Methods for the Earth-Observing Satellite Scheduling Problem," *Journal of Aerospace Information Systems*, Vol. 19, No. 1, 2022, pp. 70–82. https://doi.org/10.2514/1.I010992

[18] Herrmann, A., and Schaub, H., "A Comparative Analysis of Reinforcement Learning Algorithms for Earth-Observing Satellite Scheduling," *Frontiers in Space Technologies*, Vol. 4, Nov. 2023, Paper 1263489. https://doi.org/10.3389/frspt.2023.1263489

[19] Wang, X., Zhao, F., Shi, Z., and Jin, Z., "Deep Reinforcement Learning-Based Periodic Earth Observation Scheduling for Agile Satellite Constellation," *Journal of Aerospace Information Systems*, Vol. 20, No. 8, 2023, pp. 508–519. https://doi.org/10.2514/1.I011209

[20] Stephenson, M., Mantovani, L. Q., and Schaub, H., "Intent Sharing For Emergent Collaboration In Autonomous Earth Observing Constellations," *AAS Guidance and Control Conference*, AAS Paper 24-192, Springfield, VA, 2024.

[21] Hadj-Salah, A., Verdier, R., Caron, C., Picard, M., and Capelle, M., "Schedule Earth Observation Satellites with Deep Reinforcement Learning," *Proceedings of the 11th International Workshop on Planning and Scheduling for Space (IWPSS), co-located with the International Conference on Automated Planning and Scheduling*, edited by S. Chien, Berkeley, CA, 2019, pp. 61–64.

[22] Naik, K., Chang, O., and Kotulak, C., "Deep Reinforcement Learning for Autonomous Satellite Responsiveness to Observed Events," *2024 IEEE Aerospace Conference*, Inst. of Electrical and Electronics Engineers, New York, 2024, pp. 1–10. https://doi.org/10.1109/AERO58975.2024.10521008

[23] Mantovani, L. Q., Nagano, Y., and Schaub, H., "Reinforcement Learning for Satellite Autonomy Under Different Cloud Coverage

Probability Observations," *AAS Astrodynamics Specialist Conference*, AAS Paper 24-189, Springfield, VA, 2024.

[24] Wei, L., Cui, Y., Chen, M., Wan, Q., and Xing, L., "Multi-Objective Neural Policy Approach for Agile Earth Satellite Scheduling Problem Considering Image Quality," *Swarm and Evolutionary Computation*, Vol. 94, April 2025, Paper 101857. https://doi.org/10.1016/j.swevo.2025.101857

[25] Nazmy, I., Harris, A., Lahijanian, M., and Schaub, H., "Shielded Deep Reinforcement Learning for Multi-Sensor Spacecraft Imaging," *2022 American Control Conference (ACC)*, IEEE Publ., Piscataway, NJ, 2022, pp. 1808–1813. https://doi.org/10.23919/ACC53348.2022.9867762

[26] Herrmann, A., Carneiro, J. V., and Schaub, H., "Reinforcement Learning For the Multi-Satellite Earth-Observing Scheduling Problem," *Proceedings of the 44th Annual American Astronautical Society Guidance, Navigation, and Control Conference, 2022*, edited by M. Sandnas, and D. B. Spencer, Springer International Publ., Cham, 2024, pp. 1351–1368.

[27] Stephenson, M. A., Mantovani, L., Phillips, S., and Schaub, H., "Using Enhanced Simulation Environments to Improve Reinforcement Learning for Long-Duration Satellite Autonomy," *AIAA Science and Technology Forum and Exposition (SciTech)*, AIAA Paper 2024-0990, 2024. https://doi.org/10.2514/6.2024-0990

[28] Bengio, Y., Louradour, J., Collobert, R., and Weston, J., "Curriculum Learning," *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, Montreal Quebec Canada, 2009, pp. 41–48. https://doi.org/10.1145/1553374.1553380

[29] Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N., "Curriculum Learning: A Survey," *International Journal of Computer Vision*, Vol. 130, No. 6, 2022, pp. 1526–1565. https://doi.org/10.1007/s11263-022-01611-x

[30] Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P., "Reverse Curriculum Generation for Reinforcement Learning," *Proceedings of the 1st Annual Conference on Robot Learning, Proceedings of Machine Learning Research*, edited by S. Levine, V. Vanhoucke, and K. Goldberg, Vol. 78, PMLR, Cambridge, MA, 2017, pp. 482–495.

[31] Hermann, L., Argus, M., Eitel, A., Amiranashvili, A., Burgard, W., and Brox, T., "Adaptive Curriculum Generation from Demonstrations for Sim-to-Real Visuomotor Control," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2020, pp. 6498–6505. https://doi.org/10.1109/ICRA40945.2020.9197108

[32] Rudin, N., Hoeller, D., Reist, P., and Hutter, M., "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," *Proceedings of the 5th Conference on Robot Learning, Proceedings of Machine Learning Research*, edited by A. Faust, D. Hsu, and G. Neumann, Vol. 164, PMLR, Cambridge, MA, 2022, pp. 91–100.

[33] Xiao, C., Lu, P., and He, Q., "Flying Through a Narrow Gap Using End-to-End Deep Reinforcement Learning Augmented With Curriculum Learning and Sim2Real," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 34, No. 5, 2023, pp. 2701–2708. https://doi.org/10.1109/TNNLS.2021.3107742

[34] Margolis, G. B., Yang, G., Paigwar, K., Chen, T., and Agrawal, P., "Rapid Locomotion via Reinforcement Learning," *International Journal of Robotics Research*, Vol. 43, No. 4, 2024, pp. 572–587. https://doi.org/10.1177/02783649231224053

[35] Wang, H.-C., Huang, S.-C., Huang, P.-J., Wang, K.-L., Teng, Y.-C., Ko, Y.-T., Jeon, D., and Wu, I.-C., "Curriculum Reinforcement Learning from Avoiding Collisions to Navigating Among Movable Obstacles in Diverse Environments," *IEEE Robotics and Automation Letters*, Vol. 8, No. 5, 2023, pp. 2740–2747. https://doi.org/10.1109/LRA.2023.3251193

[36] Turchetta, M., Kolobov, A., Shah, S., Krause, A., and Agarwal, A., "Safe Reinforcement Learning via Curriculum Induction," *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Curran Assoc., New York, 2020.

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

[38] Federici, L., and Zavoli, A., "A Curriculum Learning Approach for Improving Constraint Handling in Reinforcement Learning Applications to Spacecraft Guidance and Control," *SPAICE2024: The First Joint European Space Agency/IAA Conference on AI in and for Space*, Zenodo, Genova, Switzerland, 2024, pp. 482–486. https://doi.org/10.5281/ZENODO.13885667

[39] Federici, L., and Furfaro, R., "Improving Reinforcement Learning Performance in Spacecraft Guidance and Control Through Meta-Learning: A Comparison on Planetary Landing," *Neural Computing and Applications*, Vol. 37, No. 22, 2024, pp. 17,249–17,271. https://doi.org/10.1007/s00521-024-10520-8

[40] Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., and Stone, P., "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey," *Journal of Machine Learning Research*, Vol. 21, No. 1, 2020, pp. 7382–7431.

[41] Song, Y., and Schneider, J., "Robust Reinforcement Learning via Genetic Curriculum," *2022 International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2022, pp. 5560–5566. https://doi.org/10.1109/ICRA46639.2022.9812420

[42] Li, Y., Tian, Y., Tong, E., Niu, W., and Liu, J., "Robust Reinforcement Learning via Progressive Task Sequence," *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization*, 2023, pp. 455–463. https://doi.org/10.24963/ijcai.2023/51

[43] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P., "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 3803–3810. https://doi.org/10.1109/ICRA.2018.8460528

[44] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D., "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," *2019 International Conference on Robotics and Automation (ICRA)*, Inst. of Electrical and Electronics Engineers, New York, 2019, pp. 8973–8979. https://doi.org/10.1109/ICRA.2019.8793789

[45] Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D., "Deep Drone Racing: From Simulation to Reality with Domain Randomization," *IEEE Transactions on Robotics*, Vol. 36, No. 1, 2020, pp. 1–14. https://doi.org/10.1109/TRO.2019.2942989

[46] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., Adaptive Computation and Machine Learning Series, MIT Press, Cambridge, MA, 2018.

[47] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R., "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, Vol. 101, Nos. 1–2, 1998, pp. 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

[48] Stephenson, M., and Schaub, H., "BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking," *International Astronautical Congress*, International Astronautical Federation, Paris, France, 2024.

[49] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 496–507. https://doi.org/10.2514/1.I010762

[50] Schaub, H., and Junkins, J. L., "Stereographic Orientation Parameters for Attitude Dynamics: A Generalization of the Rodrigues Parameters," *Journal of the Astronautical Sciences*, Vol. 44, No. 1, 1996, pp. 1–19.

[51] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 4th ed., AIAA Education Series, AIAA, Reston, VA, 2003, pp. 387–518. https://doi.org/10.2514/4.861550

[52] Luo, M., Yao, J., Liaw, R., Liang, E., and Stoica, I., "IMPACT: Importance Weighted Asynchronous Architectures with Clipped Target Networks," Jan. 2020. https://doi.org/10.48550/arXiv.1912.00167

[53] Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I., "RLlib: Abstractions for Distributed Reinforcement Learning," June 2018. https://doi.org/10.48550/arXiv.1712.09381

[54] West, J., Maire, F., Browne, C., and Denman, S., "Improved Reinforcement Learning with Curriculum," *Expert Systems with Applications*, Vol. 158, Nov. 2020, Paper 113515. https://doi.org/10.1016/j.eswa.2020.113515

[55] Reed, R., Schaub, H., and Lahijanian, M., "Shielded Deep Reinforcement Learning for Complex Spacecraft Tasking," *2024 American Control Conference (ACC)*, Inst. of Electrical and

Electronics Engineers, New York, 2024, pp. 2331–2337. https://doi.org/10.23919/ACC60939.2024.10644855

[56] Mantovani, L. Q., and Schaub, H., "Performance Evaluation of Shielded Neural Networks for Autonomous Agile Earth Observing Satellites in Long Term Scenarios," *Proceedings of the 14th International Workshop on Planning and Scheduling for Space (IWPSS)*, edited by C. Artigues, J. Jaubert, C. Pralet, and S. Chien, Toulouse, France, 2025, pp. 112–121.

E. Atkins
Editor-in-Chief