(Preprint) AAS 25-637

ENHANCING FAULT RESILIENCE IN RL-BASED SATELLITE AUTONOMOUS TASK SCHEDULING

Yumeka Nagano[*] and Hanspeter Schaub[†]

This paper investigates methods to enhance the resilience of policies trained using deep reinforcement learning (DRL) for agile Earth-observing satellite scheduling, with a focus on reaction wheel (RW) faults during operation. The problem is formulated as a partially observable Markov decision process (POMDP), reflecting real-world limitations in fault observability. Four common and critical RW fault types are considered: complete loss of control, power draw limitations, increased Coulomb friction, and faulty encoder measurements. These faults frequently cause satellites to enter safe mode, leading to missed imaging opportunities. Previous studies have shown that policies trained solely under nominal conditions exhibit limited fault tolerance. To address this, the study evaluates three approaches: training policies in fault-injected environments, augmenting observation spaces with fault information, and leveraging recurrent neural networks (RNNs) to capture temporal patterns indicative of faults. Policies are trained using the Asynchronous Proximal Policy Optimization (APPO) algorithm within Basilisk, a high-fidelity spacecraft simulation framework. A curriculum learning (CL) policy is used as the baseline for fault resilience. Results show that for severe faults such as uncontrollable RWs, fault-adaptive training substantially improves performance. For milder faults, incorporating fault information during training further enhances resilience without compromising nominal performance. However, limitations remain: modeling all real-world faults is infeasible, and when exposed to multiple fault types and severities during training, policies may overly adapt to more frequent faults, potentially neglecting rare but critical ones.

INTRODUCTION

Earth-observing satellites play a vital role in monitoring the environment, tracking natural disasters, and managing resources.[1] These satellites collect data through imaging instruments, which are essential for a wide range of applications. The increasing number of target requests and the need for efficient image scheduling have led to the development of agile Earth-observing satellite scheduling, which can slew both along and across the orbital path.[2] The primary goal of the scheduling problem is to organize satellite operations to maximize the value of collected images—by increasing the number of imaged targets and prioritizing higher-value ones—while maintaining system safety, such as battery charging and reaction wheel (RW) momentum management.

Traditionally, planning and scheduling are performed on the ground using optimization algorithms, with the resulting plans sequenced and uplinked to the spacecraft for open-loop execution.[3] Optimization-based scheduling techniques, such as branch-and-bound and mixed-integer linear programming (MILP), have been developed to solve the problem with optimality guarantees, but they

[*]PhD Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80309.
[†]Distinguished Professor and Department Chair, Schaden Leadership Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80309.

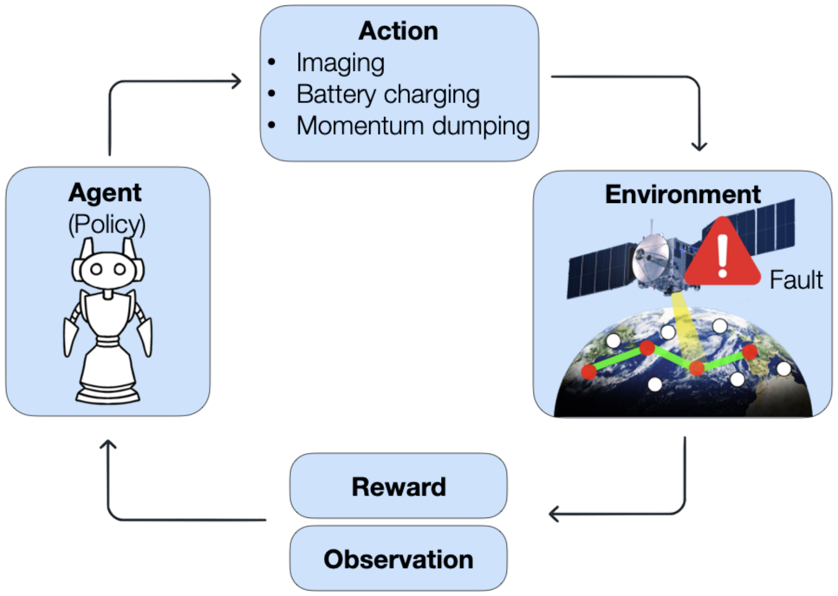

Figure 1: Concept Illustration of RL for satellite task scheduling in fault environment.

are computationally intensive and not suitable for real-time or onboard applications.[4,5] Furthermore, MILP-based formulations often rely on simplified models that may not adequately capture the complexity of real-world scenarios. Heuristic approaches, including greedy algorithms and local search methods, offer near-optimal solutions within reasonable time frames,[6] but struggle to handle the high dimensionality and dynamic nature of real-time scheduling, particularly with frequent task updates and a large volume of target requests. Stochastic methods such as genetic algorithms and simulated annealing have also been explored,[7] but these techniques face similar challenges in managing complex constraints under operational time constraints.

To overcome these limitations, recent research has explored autonomous satellite scheduling using machine learning (ML) techniques, enabling onboard real-time decision-making.[8] In this paradigm, the learning process is conducted on the ground, and the trained model is deployed onboard, minimizing computational overhead while enabling closed-loop responses to dynamically added tasks or unexpected events.[9] While promising, many learning-based methods simplify system dynamics and often neglect critical safety considerations such as RW momentum dumping and energy constraints. To address these limitations, deep reinforcement learning (DRL) has been applied to satellite scheduling to optimize decision-making in complex and dynamic environments.[10] DRL uses reinforcement learning (RL) algorithms to train policies parameterized by deep neural networks, as illustrated in Figure 1. These problems are typically formulated as Markov decision processes (MDPs),[11,12] with recent work extending the formulation to include more realistic operational factors such as charging, downlink windows, and RW momentum management.[3] Alternative formulations to effectively represent the mission objectives include semi-Markov decision processes (semi-MDPs), which accommodate variable time steps,[13] and partially observable MDPs (POMDPs),[14] which model limited observability of the system state.

Sequential decision-making models, such as incorporating long short-term memory (LSTM) networks, have been used to enhance the temporal understanding of DRL-based scheduling policies.[15] To improve safety, formal shielding methods have been integrated with DRL frameworks,[16] and curriculum learning approaches have been shown to increase the robustness of policies to varying initial conditions.[17] While DRL-based scheduling policies are often designed to be robust to unexpected events, their resilience under fault conditions remains insufficiently explored, as these policies are typically trained under nominal conditions and are not explicitly exposed to faults during training. The need for resilient policies capable of adapting to faults has been recognized.[18]

In conventional satellite operations, measurement faults—where sensor outputs deviate from expected ranges—typically trigger a safe mode response, halting mission tasks until the fault is resolved. Although this approach prioritizes safety, it can result in significant data loss. Existing research on fault detection, isolation, and recovery (FDIR) has primarily focused on fault diagnosis using expert systems or machine learning techniques,[19–21] with limited attention to optimal recovery actions. Some studies have addressed this by integrating fault diagnosis with spacecraft control systems,[22] exploring adaptive control for attitude regulation and collision avoidance,[23,24] and analyzing trade-offs between mission success and operational safety.[25] However, investigations into the resilience of RL-based satellite autonomy under fault conditions remain scarce. While some studies have shown that training under specific fault scenarios can improve policy performance,[26] these approaches often require tailored training for each fault type. Other efforts, such as those addressing communication faults during small body science missions, demonstrate the adaptability of RL-based planning and scheduling under constrained conditions like Deep Space Network outages.[27]

In the field of robotics, fault-tolerant control methods have been developed using RL to enhance the resilience of robotic systems under fault conditions. Fault scenarios involving joint failures in manipulators and locomotion systems have been studied under the assumption that the system can detect faults through self-diagnosis.[28] Additionally, meta-RL—where algorithms quickly adapt to new tasks by leveraging training data from related scenarios[29]—has been applied to locomotion fault scenarios.[30] These approaches demonstrate adaptability to faults, with meta-RL in particular showing strong performance in handling unseen situations and sudden environmental changes. Moreover, RL-based fault-tolerant control strategies for unmanned aerial vehicles (UAVs) under actuator faults have been explored, demonstrating effective fault-handling capabilities.[31] However, this line of research primarily addresses actuator degradation rather than severe failure scenarios that require switching to an alternative controller.

Despite recent advances in DRL for agile satellite task scheduling, enhancing policy resilience under actuator fault conditions remains a largely unexplored area. While prior work such as Reference 17 has demonstrated that CL can improve robustness against environmental variations, its effectiveness in the presence of actuator faults—particularly RW failures—has not been evaluated. Previous research assessed fault resilience by testing policies trained in nominal environments under faulted conditions, without using CL or incorporating fault-specific training.[18] Building on that foundation, this study focuses on evaluating DRL-based policies with CL training approach under RW faults, including four representative fault types: complete RW failure (uncontrollable), power draw limitation, increased Coulomb friction, and encoder measurement fault. The CL framework is adapted to the satellite imaging task, and its performance is evaluated using two key metrics: reward, which reflects imaging task success, and aliveness, defined as the satellite's ability to complete three full orbits without battery depletion or RW overspeed.

To explore ways of improving policy resilience under RW fault conditions, three training approaches are evaluated. The first approach involves training the policy in environments that include fault scenarios, without assuming the agent has access to explicit fault information. For this case, policies are trained both with and without an LSTM architecture to examine whether leveraging temporal dependencies in the observation history enables the agent to infer latent fault states and make more informed, context-aware decisions. The second approach augments the observation space with explicit fault information, assuming that onboard systems can accurately detect and identify faults. Two types of fault observability are considered: a binary indicator specifying the presence or absence of a fault, and a RW identifier indicating which RW is faulty. These policies are tested under RW uncontrollable scenarios and compared with a strategy that disables the faulty RW upon detection. The third approach introduces varied fault types and severities during training. In this setting, the agent is informed of which RW is faulted but not the specific fault type or severity—reflecting more realistic operational uncertainty. This scenario assesses whether the policy can find effective actions when the RW is partially degraded but not completely failed. Together, these approaches provide a broad assessment of DRL-based strategies aimed at enhancing fault resilience in satellite task scheduling under representative actuator fault conditions.

The remainder of the paper is organized as follows. The Problem Statement section introduces the formulation of the satellite scheduling problem. The Solution Approach section details the curriculum learning methodology and the various training strategies employed to improve policy resilience. The Results section presents an in-depth analysis of policy performance under multiple fault conditions. Finally, the paper concludes with a summary of findings.

PROBLEM STATEMENT

This paper investigates methods to enhance the fault resilience of policies trained using DRL algorithms for autonomous task scheduling in satellites. The focus is specifically on faults in RWs, which are critical components for attitude control and precise target pointing. The study considers a single satellite equipped with four reaction wheels: three aligned with the principal axes (yaw, pitch, and roll), and a fourth redundant wheel tilted at equal angles relative to the others. While three RWs are sufficient for full attitude controllability under nominal conditions, the inclusion of a fourth RW provides redundancy in the presence of faults.

Problem Formulation

To account for limited state observability, the satellite scheduling problem is formulated as a POMDP defined by tuple $(\mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z)$,[32] where:

- **State Space** $\mathcal{S}$: The state space comprises the satellite and environmental states required to propagate the simulation at each discrete time step k, denoted $s_k \in \mathcal{S}$.
- **Action Space** $\mathcal{A}$: The action space consists of 34 distinct actions, including a charging action, a momentum management action for desaturating the reaction wheels, and 32 imaging actions corresponding to the upcoming 32 targets. At time step k, the agent selects an action $a_k \in \mathcal{A}$. Both the charging and momentum management actions have a fixed duration of 60 seconds.
- **Transition Function** $T(s_{k+1} \mid s_k, a_k)$: The transition function is deterministic and generated by the simulator, such that

$$T(G(s_k, a_k) \mid s_k, a_k) = 1 \tag{1}$$

  where $G(s_k, a_k) = s_{k+1}$ represents the generative model provided by the environment.
- **Reward Function** $R(s_k, a_k, s_{k+1})$: At each time step k, the reward for each target i is defined

as:

$$R_i = \begin{cases} \rho_i, & \text{if target } i \in U_k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where ρ_i is the priority of target i, and U_k is the unfulfilled target list at time k. A target is moved from U_k to the fulfilled list F_k upon successful imaging.

- **Observation Space** $\mathcal{O}$: At each time step k, the agent receives an observation $o_k \in \mathcal{O}$, which is a subset of the full state s_k and is normalized as described in Table 1.
- **Observation Function** $Z(o_k \mid s_k, a_k)$: The observation function is deterministic, assuming that the satellite can observe the states within the observation space without uncertainty.
- **Discount Factor** γ: The discount factor $\gamma = 0.999$ balances immediate and future rewards.

An episode terminates when a failure condition occurs: the battery charge level drops to zero, or any of the satellite's reaction wheels reaches its maximum angular speed. Otherwise, the episode is truncated when the duration reaches three orbits.

The objective is to find a policy π that maps observations to actions and maximizes the expected discounted return starting from state s_0:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, a_k, s_{k+1}) \,\middle|\, a_k = \pi(o_k),\ s_{k+1} = G(s_k, a_k)\right] \forall s \in \mathcal{S} \tag{3}$$

Table 1: Observation Space of the Satellite

Parameter	Normalization	Description
${}^{\mathcal{P}}\hat{\boldsymbol{r}}_{\mathcal{B}/\mathcal{N}}$	Earth's radius	Position of the satellite with respect to the inertial frame $\mathcal{N}$, expressed in the planet frame $\mathcal{P}$.
${}^{\mathcal{P}}\hat{\boldsymbol{v}}_{\mathcal{B}/\mathcal{P}}$	$\lvert \boldsymbol{v}_{\mathcal{B}/\mathcal{N}} \rvert$	Velocity of the satellite with respect to the planet frame, expressed in the planet frame.
$\hat{\boldsymbol{c}}_{\mathcal{B}/\mathcal{P}}$	π	Orientation of the satellite's sensing instrument with respect to the planet frame.
${}^{\mathcal{P}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{P}}$	0.03	Angular velocity of the satellite with respect to the planet frame, expressed in the planet frame.
P_f	Maximum battery capacity	Battery charge level.
Φ_s	π	Angle between solar panels and the sun vector.
Ω_i	Maximum RW angular speed	RW angular speed of each RW i.
e_{start}	Orbital period	Next eclipse start time.
e_{end}	Orbital period	Next eclipse end time.
${}^{\mathcal{P}}\hat{\boldsymbol{r}}_{R_i/\mathcal{B}}$	Earth's radius	Position of the target i in the planet frame
θ_i	π	Angle between target i and the current pointing direction of the satellite.
ρ_i	-	Priority of upcoming target i.
t	Episode duration	Simulation time.

SOLUTION APPROACH

In order to improve the fault resilience of RL-based scheduling policies, several approaches are explored:

1. **Evaluating Curriculum Learning (CL) Policy in Fault Environments**: This approach involves training a policy using CL method described in Reference 17. This CL approach

shows the effectiveness for the satellite task scheduling problem to make the policy more robust to conditions differ from nominal conditions. Therefore, the CL-based policy is used as a baseline to evaluate its performance in fault environments. The policy is not exposed to faults during training, while it experiences wide range of battery capacity, external torque, and initial conditions.

2. **Training in Fault Environments**: This approach involves training the policy in environments that include fault scenarios. The policy is trained to maximize the reward while being exposed to various fault conditions, focusing on reaction wheel failures. This method allows the policy to learn how to handle faults during training, potentially improving its performance in fault scenarios. That algorithm includes incorporating observation history through recurrent architectures. The policy is trained using a recurrent neural network (RNN) architecture, such as LSTM, to capture temporal dependencies in the observation history. This allows the policy to infer latent information—such as undetected faults—from patterns in past observations and take context-aware actions. In this approach, the agent cannot observe the fault information.
3. **Augmenting Observation Space with Fault Information**: This approach assumes that faults are detected and accurately identified onboard. The corresponding fault information is appended to the observation space in addition to the features listed in Table 1. By providing explicit knowledge of the fault state, the policy can learn to take corrective or compensatory actions.

Simulation Environment

All simulations are conducted using BSK-RL,[*] an open-source Python package that provides modular, high-fidelity reinforcement learning environments for spacecraft tasking and planning.[33] BSK-RL integrates the Gymnasium framework, widely used in reinforcement learning research, with Basilisk,[†] a high-fidelity astrodynamics simulation software developed for spacecraft flight dynamics and operations.[34]

Basilisk, written in C and C++ with a Python interface, serves as the generative model $G(s, a)$ in this framework, enabling realistic simulation of spacecraft dynamics, subsystems, maneuvering times, and power constraints. Its fast runtime and flight-proven capabilities make it particularly well-suited for reinforcement learning applications that require both physical accuracy and computational efficiency.

Actions The simulation environment models the spacecraft dynamics, RWs, battery subsystem, and momentum management capabilities. When an imaging action is chosen, the simulator calculates the control torque to align the satellite with the target. During the maneuver, the simulator checks the imaging constraints, the relative angle between the target and the satellite, and the relative angular rate. If the constraints are satisfied, the target is successfully imaged, the reward is awarded proportional to the targets priority, and the target is moved to the fulfilled list. If the constraints are not met, the imaging action fails, and the target remains in the unfulfilled list. This study exclusively considers point-by-point imaging; however, the approach can be extended to support strip imaging.[6]

The simulator also accounts for the battery charge level, incorporating baseline power consumption to represent essential subsystems, maneuver power consumption for providing torques to the

[*]`https://avslab.github.io/bsk_rl`
[†]`https://avslab.github.io/basilisk`

Table 2: Satellite Parameters

Parameter	**Value**
Altitude	500 km
Mass	330 kg
Inertia	[121, 98, 82] kg m^2
Battery capacity	160 Wh
Base power consumption	20 W
Initial battery charge fraction	[0.4, 1.0]
Relative angle limit for imaging	28°
Relative angular rate limit for imaging	0.01 rad/s
RW maximum torque	0.2 Nm
RW maximum speed	1500 RPM
Initial RW speed	[-900, 900] RPM
Number of requests	[1000, 10000]
Target priority ρ_i	[0, 1]
External torque	0.1 mNm

RWs, and imaging power consumption for operating the imaging instrument. The satellite can passively charge during the simulation whenever its solar panels face the Sun and are not in eclipse, even while performing other actions. The charging rate depends on the solar incidence angle. When the charging action is selected, the satellite maneuvers to position its solar panels perpendicular to the Sun, maximizing energy generation.

The momentum management action is designed to desaturate the RWs, which are used to control the spacecraft's attitude. Over time, RWs accumulate momentum due to external torques, necessitating periodic desaturation. This action aligns the spacecraft with an inertial reference frame and then fires thrusters to remove the excess momentum from the RWs.

Targets The targets are uniformly distributed across the Earth's surface, with the number of imaging requests (targets to be imaged) ranging from 1,000 to 10,000. This variation directly influences the target density. The target type can be extended to city targets.[32]

Satellite Configuration

The simulation employs satellite parameters, along with initialization and target parameters, as detailed in Table 2. The satellite follows a circular orbit with a period of 95 minutes. The initialization and target parameters are randomly selected within the predefined ranges for each episode, and the direction of external torque is also chosen randomly for each episode. Parameters not explicitly listed are set to their default values as provided by BSK-RL.

The RW configuration used in this simulation is shown in Figure 2. The RWs are arranged such that three wheels align with the yaw, pitch, and roll axes, while a fourth redundant wheel is tilted at an equal angle relative to the others. Due to the redundant RW, there are infinitely many solutions for mapping the required body control torque to the individual RW torques. Therefore, once the torque to be mapped is calculated, the RW null space is used to decelerate the wheels without exerting additional torque on the spacecraft.[35] In this setup, RW 1 is aligned with the across-track direction and RW 2 with the along-track direction, so they are used more frequently than the others during nominal operations such as imaging and charging maneuvers. This explains why the results are more sensitive to some RW failing than others For RW speed measurements, the simulator uses a simple encoder model that provides the angular velocity of each RW, with a resolution of 1024 counts per revolution.

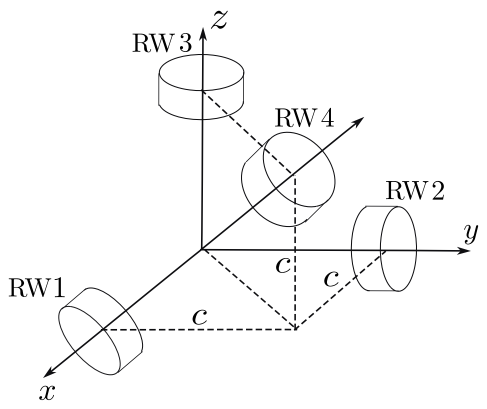

Figure 2: RW Configuration.

Fault Scenarios

This study focuses on RW faults, which are common in satellite systems. RWs are critical for satellite attitude control, and faults in these components can significantly affect the satellite's ability to maneuver and complete imaging tasks. All faults are introduced at the beginning of each episode. The fault scenarios are as follows:

1. **Uncontrollable RW Fault**: One of the RWs becomes uncontrollable, with its power draw limited to 1.0×10^{-12} W, therefore, allowing the wheel to rotate passively but rendering it incapable of contributing to attitude control.
2. **Power Limit Fault**: The RW's power draw is constrained, causing it to operate at reduced speed compared to nominal conditions. By default, while there is a torque limit, no restrictions on power draw are imposed.
3. **Friction Fault**: The RW's Coulomb friction, which remains constant regardless of changes in RW speed, is increased, resulting in slower operation and higher power consumption compared to nominal conditions. The default Coulomb friction is $0.5\,\mathrm{mNm}$.
4. **Encoder Measurement Fault**: Two types of encoder faults are considered: (1) the RW encoder measurement is turned off, and (2) the RW encoder measurement becomes stuck at its initial value.

An example script of a fault modeling for spacecraft tasking is available at the BSK-RL website*.

Training Environment

Once the problem was formulated, a training environment was established to train a policy using DRL. The training process assumed no fault scenarios. The Asynchronous Proximal Policy Optimization (APPO) algorithm from RLlib,[36] a library that provides scalable software primitives for RL, was used.[37] The hyperparameters used for training the policy are listed in Table 3, while all other parameters, including those related to the LSTM architecture, are set to their default values in RLlib. A detailed hyperparameter analysis can be found in Reference 38. Training was performed on the University of Colorado Boulder's computing cluster (CURC), using 32 CPU cores and up to 30 million environment steps per policy.

*`https://avslab.github.io/bsk_rl/examples/fault_environment.html`

The training environments are summarized in Table 4, with detailed descriptions provided in the following sections.

Table 3: Training Parameters

Parameter	**Value**
Number of workers	32
Number of training steps	$3 \cdot 10^7$
Learning rate	$3 \cdot 10^{-5}$
Training batch size	10,000
Minibatch size	250
Number of SGD iterations	50
Neural Network	2 layers with 512 neurons each
Discount factor	0.999
Failure penalty	0

Table 4: Training Environment Configurations for Each Policy

Policy Name	**Fault Observation**	**Training Scenario**
CL	None	Curriculum learning: battery capacity gradually decreases and external torque increases over time.
Fault	None	Nominal environment (50%) and RW uncontrollable fault (50%).
Fault-LSTM	None	Nominal environment (50%) and RW uncontrollable fault (50%) using LSTM.
Fault-bool	Fault presence	Nominal environment (50%) and RW uncontrollable fault (50%).
Fault-num	Faulty RW	Nominal environment (50%) and RW uncontrollable fault (50%).
Fault-random	Faulty RW	Nominal environment (25%) and random RW faults: power limit (25%), increased friction (25%), or encoder failure (25%).

Curriculum Learning Curriculum learning provides a framework for structuring the training environment to help the agent progressively acquire more complex skills.[39] Training begins with simpler tasks and gradually increases in difficulty. In this study, the curriculum selected is the one that demonstrated the best performance in the experiments presented in Reference 17. This curriculum uses the parameters listed in Table 5, in which battery capacity is linearly decreased and external torque magnitude is linearly increased over the course of training. Initial conditions are randomly sampled from wider ranges than those used in the nominal environment.

Table 5: Training Parameters

Parameter	**Nominal Value**	**CL Progression**
Battery Capacity	160 W	Decreased from 160 to 64 W
External Torque	0.1 mNm	Increased from 0.01 to 0.4 mNm
Initial Battery Charge Level	[40, 100]%	Sampled from [0, 100]%
Initial RW Speeds	[–900, 900] RPM	Sampled from [–1500, 1500] RPM
Episode Horizon	3 orbits	15 orbits

Training with Reaction Wheel Faults To account for fault scenarios, the training environment was extended to include conditions where one of the four RWs becomes uncontrollable, as described in the Fault Scenarios section. The environment was configured such that each episode has a 50% chance of being fault-free (nominal) and a 50% chance of including a fault. In the fault case, one of the four single-RW faults is selected uniformly at random. As a result, the training set includes five scenarios: the nominal case and four distinct single-RW fault cases.

Policies were trained both with and without the use of a LSTM architecture, referred to as the Fault and Fault-LSTM policies, respectively. The LSTM enables the agent to capture temporal dependencies and infer latent information—such as undetected faults—from sequences of past observations, thereby facilitating context-aware decision-making. The implementation leverages RLlib's native support for recurrent policies.

Augmenting Observation Space with Fault Information In an alternative training strategy, fault information is explicitly included in the observation space. The environment setup remains identical to the fault-inclusive case above, but the agent receives additional input describing the fault status. Two types of augmented observations are considered:

1. A binary indicator specifying the presence or absence of a fault (Fault)
2. A categorical identifier indicating the specific fault status - 0: no fault, 1: RW1 fault, 2: RW2 fault, 3: RW3 fault, 4: RW4 fault, (Fault-num policy).

Various Fault Conditions In practice, RW faults may be less severe and still allow partial functionality. To reflect this, the training environment is further extended to encompass a variety of RW fault types, including power draw anomalies, increased Coulomb friction, and encoder malfunctions. The environment includes four equally probable scenarios: nominal, power fault, friction fault, and encoder fault. If a fault is selected, one of the four RWs is randomly chosen to exhibit the fault. Fault severity is varied within the following predefined ranges:

1. **Power Limit Fault:** 0.001 W to 10 W
2. **Friction Fault:** 10 to 1000 times the nominal Coulomb friction
3. **Encoder Fault:** either the encoder is disabled or its output is stuck at the initial value

In this setup, the agent receives information indicating which RW is faulted, but no details about the fault type or severity are provided. This setting encourages the agent to generalize across fault types and develop resilient fault-adaptive control strategies. This configuration is referred to as the Fault-random policy.

Testing Environment

The trained policies are evaluated under both fault-free (nominal) and fault-induced scenarios. For each target count—1,000, 2,000, 3,000, 4,000, 6,000, 8,000, and 10,000—the policy is tested under nominal conditions and each defined fault type. All testing is conducted using the nominal parameter settings listed in Table 5, unless a specific fault is introduced. For each configuration, 50 trials are run using different random seeds, with the same seed set applied across all cases to ensure consistency in evaluation.

To improve robustness during evaluation, a safety mechanism—referred to as a shield—is employed for all the testing cases to prevent the agent from executing potentially unsafe actions. The shield used in this study is adapted from the Handmade shield described in Reference 40. It enforces two safety constraints:

- The agent is forced to take a charging action when the battery level drops below 25% of its maximum capacity.

- The agent is required to perform a momentum management action when any reaction wheel (RW) angular speed exceeds 70% of its maximum allowable value.

Fault scenarios are evaluated with specific configurations for each fault type, as detailed below:

1. **Uncontrollable RW Fault:** One of the four reaction wheels (RW 1–4) becomes completely unresponsive.
2. **Power Limit Fault:** The RW power draw is constrained to one of the following limits: 0.001, 0.01, 0.1, 1.0, or 10.0,W.
3. **Friction Fault:** The nominal Coulomb friction of 0.5 mNm is scaled by factors of 10, 100, 200, 300, 400, and 1000.
4. **Encoder Measurement Fault:** Two encoder failure modes are considered: (1) the RW encoder measurement is disabled, and (2) the encoder becomes stuck at its initial value.

The testing environments are summarized in Table 6. All refers to testing under every fault type considered in this study: uncontrollable RW, power limit, increased friction, and encoder faults.

Table 6: Testing Environment Configurations

Policy Name	**Tested Faults**
CL	All
Fault	Uncontrollable RW
Fault-LSTM	Uncontrollable RW
Fault-bool	Uncontrollable RW
Fault-num	Uncontrollable RW
Fault-random	Power limit, friction decrease, and encoder failure

In addition to testing each policy, an additional scenario is considered under the assumption that onboard fault detection and identification are available. Specifically, in the case of a severe RW fault where one wheel becomes fully unresponsive, this evaluation investigates whether deactivating the faulty RW and operating the spacecraft with the remaining three wheels is effective when using the CL policy, which was not exposed to faults during training. This is assessed by disabling the identified faulty RW upon detection and controlling the spacecraft attitude using only the remaining three RWs.

RESULTS

The performance of policies is evaluated based on the total reward obtained during the episode and the safety aspect, where failure is defined as either the battery running out or the angular speed of one of the reaction wheels exceeding the maximum limit during the three-orbit episode.

CL Policy Baseline

To quantify the performance gap between fault and nominal environments, the change in reward is measured as the normalized difference relative to the nominal environment cases:

$$\Delta R = \frac{R_{\text{fault},i} - R_{\text{nominal},i}}{R_{\text{nominal,mean}}} \tag{4}$$

Here, $R_{\text{fault},i}$ is the reward under the policy trained in fault environments for seed i, $R_{\text{nominal},i}$ is the corresponding reward under the nominal policy, and $R_{\text{nominal,mean}}$ is the average nominal policy

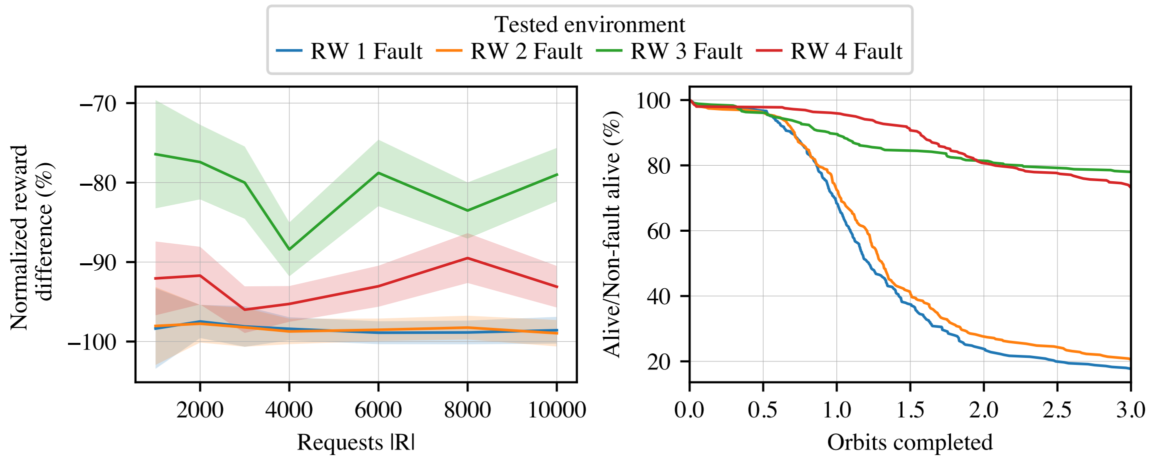

Figure 3: Reward and alive ratio under RW uncontrollable fault conditions, relative to the nominal environment, for the CL baseline policy.

reward for the number of requests. A positive value indicates improved performance over the nominal policy, and a negative value indicates a decline. Additionally, the alive rate is reported relative to the nominal case, defined as the proportion of runs that do not end in failure—where failure is determined by either the satellite's battery depleting or one of the reaction wheels exceeding its maximum angular speed. If the relative alive rate is above 100%, the policy has better survivability than the nominal baseline; if below 100%, it performs worse in maintaining satellite health.

Uncontrollable RW Fault The performance of the CL policy is first evaluated under the uncontrollable RW fault scenario. All fault cases are assessed relative to the fault-free (nominal) case. In the nominal environment, 100% of episodes were completed without failure—defined as battery depletion or RW speed limit violations—indicating that the combination of the CL policy and the shield yields a highly robust approach. To isolate the shield's effectiveness, nominal environment tests were also conducted without it. In this unshielded setting, one episode failed out of 350 trials. Based on this result, the shield is applied in all subsequent tests and comparisons.

Figure 3 presents the reward and alive ratio for each RW fault, expressed relative to the nominal case. RW 1, 2, and 3 are aligned with the principal body axes, while RW 4 is tilted at an equal angle relative to the others. Performance degrades substantially when any RW becomes uncontrollable, as most actions fail to execute successfully. In particular, faults in reaction wheels aligned with the across-track (RW 1) and along track directions (RW 2) result in more pronounced performance degradation and increased safety violations. These cases exhibit a higher number of failed charging attempts compared to faults in the other RWs.

For the remaining fault types, RW 1—responsible for control in the cross-track direction—is selected for evaluation. This choice is based on the results of the individual RW fault analysis, which indicated that faults affecting this axis have the greatest impact on overall performance. Although all RWs were tested, the observed trends were consistent across cases.

Power Limit Fault The performance of the policy is next evaluated under a power limit fault scenario. Figure 4 illustrates the reward obtained and alive rate relative to the fault-free case. Performance gradually deteriorates as the power limit is further reduced, primarily because the image action attempts failed progressively due to the reaction wheel operating at slower speeds compared

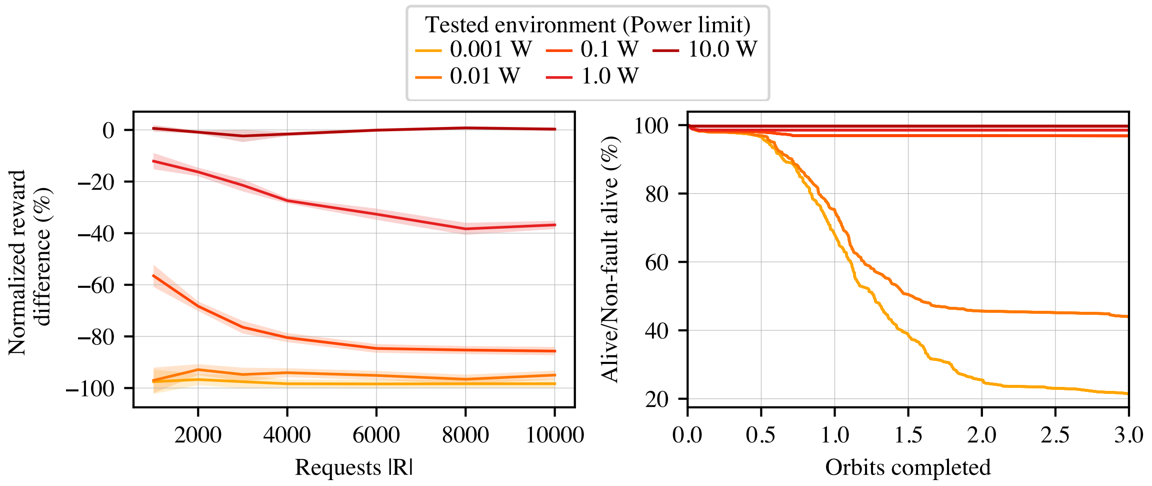

Figure 4: Reward and alive ratio under power limit fault conditions, relative to the nominal environment, for the CL baseline policy.

to nominal conditions. Additionally, performance declines further as the number of requests increases, since a higher number of targets requires quicker maneuvers, exacerbating the impact of the power limit. The safety aspect is significantly compromised when the power limit drops to 0.01 W or lower, as the maneuver speed becomes insufficient even for charging actions, which have a longer time window compared to imaging actions.

Friction Fault The performance of the policy is also evaluated under the friction fault scenario. Figure 5 illustrates the reward obtained and the alive rate relative to the fault-free case. Performance deteriorates significantly when friction increases to 400 times the nominal value, with a gradual decline observed at lower friction levels. The severe performance degradation at high friction levels is due to the reaction wheel operating at excessively slow speeds, rendering most actions unsuccessful. Additionally, as with the power limit fault scenario, performance declines further as the number of requests increases, since more targets demand quicker maneuvers. The safety aspect is significantly compromised when friction reaches 400 times the nominal value. This is because charging becomes insufficient to support actions during eclipse conditions, given the higher power consumption caused by increased friction.

Encoder Measurement Fault The policy's performance is finally evaluated under the encoder measurement fault scenario. Figure 6 presents the reward and alive ratio relative to the fault-free case. Overall, the performance remains comparable to the nominal environment, with similar reward levels and no observed failures across all cases. This robustness is attributed to the attitude control system's reliance on torque-based control rather than direct dependence on RW angular speed measurements

Training in Fault Environments

In this section, the policy is trained in an environment that includes both nominal and fault conditions, where the fault type is limited to the uncontrollable RW scenario. The agent, however, does not receive explicit information about the fault state; instead, it must infer fault conditions from observation.

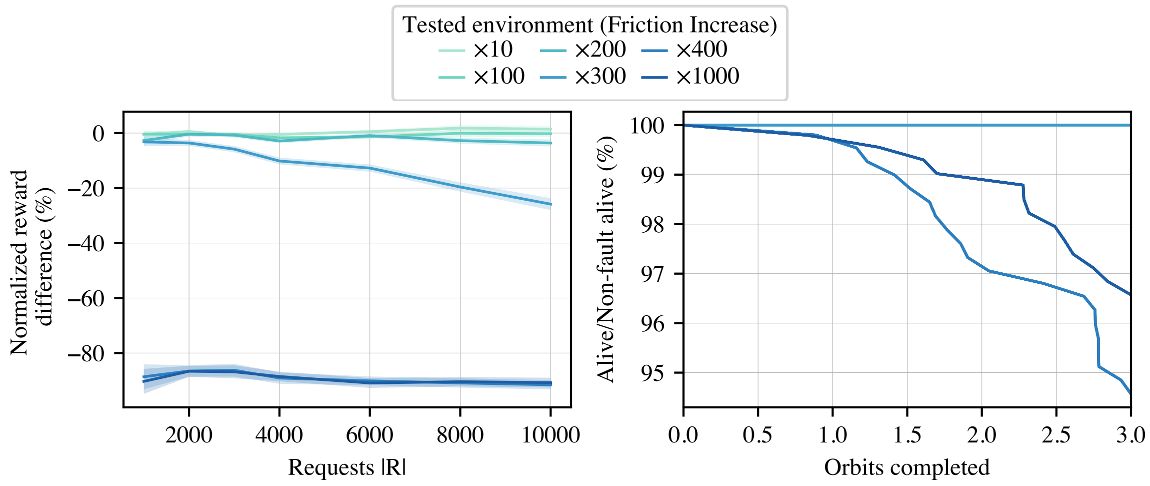

Figure 5: Reward and alive ratio under friction fault conditions, relative to the nominal environment, for the CL baseline policy.

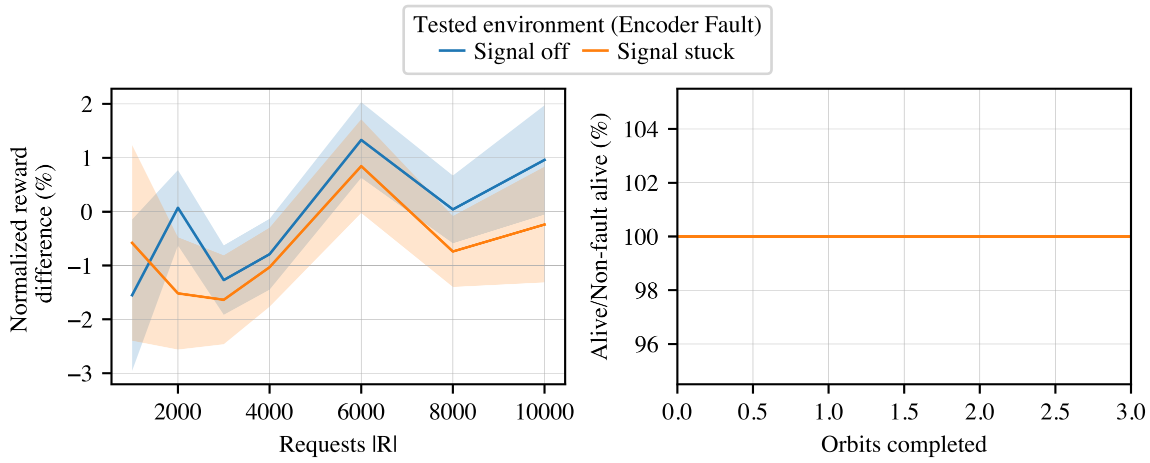

Figure 6: Reward and alive ratio under encoder fault conditions, relative to the nominal environment, for the CL baseline policy.

To quantify performance changes, the improvement in reward is measured as a normalized difference relative to the CL policy like in Equation 4:

$$\Delta R = \frac{R_{\text{fault},i} - R_{\text{CL},i}}{R_{\text{CL,mean}}} \tag{5}$$

Here, $R_{\text{fault},i}$ is the reward under the policy trained in fault environments for seed i, $R_{\text{CL},i}$ is the corresponding reward under the CL policy, and $R_{\text{CL,mean}}$ is the average CL policy reward for the number of requests. In addition to reward comparison, the alive rate is reported relative to the CL case, where alive rate is defined as the proportion of episodes that do not end in failure.

Figure 7 illustrates the performance of the Fault policy, relative to the CL baseline policy, using the reward and alive ratio metrics. The results show that while the fault-trained policy improves the alive ratio in fault scenarios, it generally sacrifices reward performance—particularly in the nominal

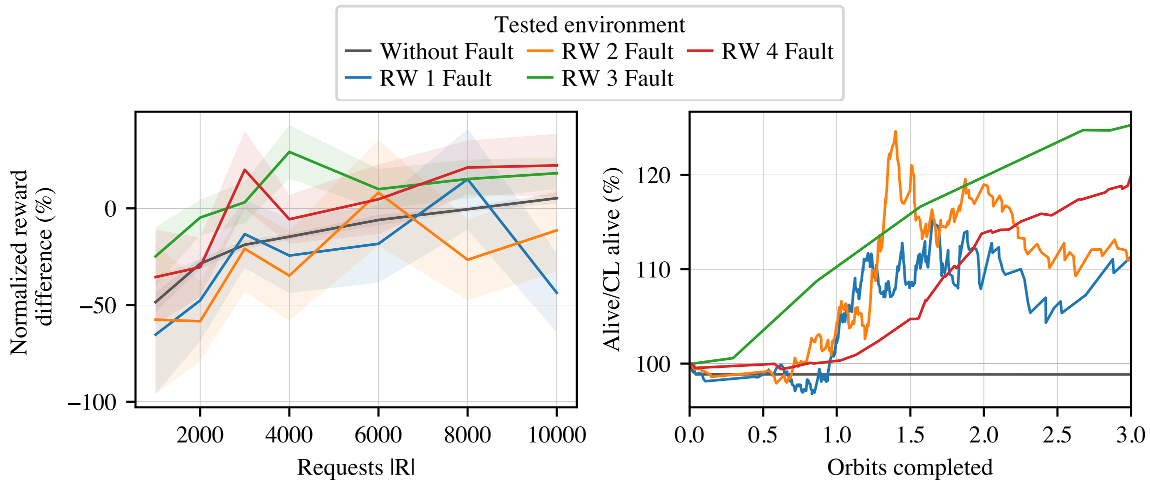

Figure 7: Reward and alive ratio of the Fault policy, relative to the CL baseline policy.

environment and under lower target request counts. This trade-off arises from the policy's action decisions. By encountering RW faults during training, the policy learns that desaturation actions offer limited benefit under such conditions and instead favors more frequent charging actions. While this behavior enhances survivability in fault cases, it compromises efficiency in fault-free scenarios. This highlights a key challenge: designing a policy that is resilient to faults while preserving performance in nominal conditions.

To address the lack of explicit fault observability, the policy is also trained using an LSTM architecture, which incorporates temporal information to help infer latent states such as faults (Fault-LSTM policy). Figure 8 presents the training curves for both the Fault and Fault-LSTM policies. The curves show reward progression and alive ratio over training steps. The Fault-LSTM policy, however, demonstrates limited learning within the training horizon. It fails to adopt effective charging behavior until around 30 million steps and continues to prioritize imaging actions throughout most of training. Although longer training and hyperparameter search may improve its performance, the rate of convergence remains significantly slower than that of the Fault policy. Furthermore, extended training time would benefit all policy architectures, not only LSTM, and may not resolve the underlying inefficiency.

Augmenting Observation Space with Fault Information

Building on the previous results, the next strategy incorporates fault information into the observation space.

RW Uncontrollable Fault Conditions First, the RW uncontrollable fault scenario is revisited, using the same evaluation metric defined in Equation 5 to assess policy performance. Figure 9 presents the reward and alive rate relative to the CL baseline policy. Overall, when the agent is provided with fault information, it can adapt its actions to mitigate the impact of the fault, leading to improvements in both reward and alive rate. The Fault-num policy, which receives detailed information identifying the specific faulty RW, consistently outperforms the Fault-bool policy—which only receives a binary fault indicator—in terms of reward, while both policies achieve comparable alive rates.

For the Fault-bool policy, the policy tends to favor charging actions over desaturation actions to

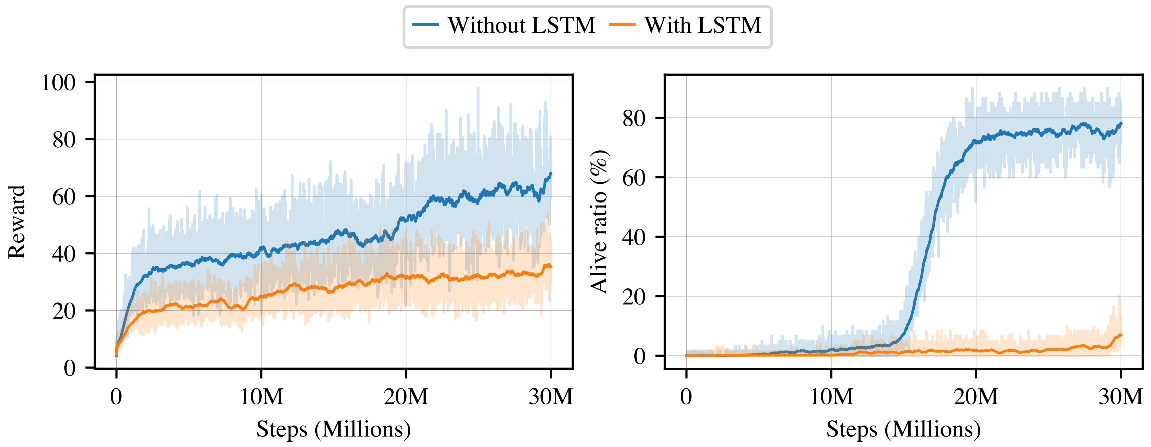

Figure 8: Training curve for policies with and without LSTM.

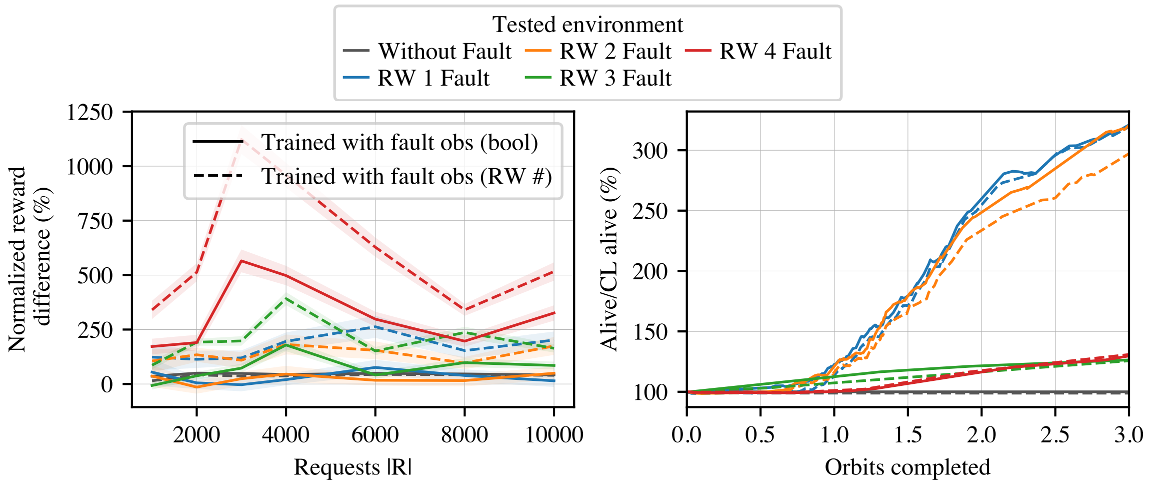

Figure 9: Reward and alive ratio of the Fault-bool and Fault-num policy, relative to the CL baseline policy.

avoid failures. In contrast, the Fault-num policy—where the exact faulty RW is identified—enables the agent to strategically prioritize imaging actions over desaturation, thereby improving performance while managing operational risk. The policies learn that, in the presence of a faulty RW, momentum dumping via desaturation action is ineffective, as the faulty wheel cannot offload the accumulated angular momentum.

As discussed in the Solution Approach section, an important question is whether it is more effective to shut down the faulty RW in the case of severe faults, such as RW uncontrollable faults, rather than training a policy to handle them. To address this, the CL policy was evaluated under the RW uncontrollable fault scenario, with the faulty RW being shut down immediately upon fault detection. The results, presented in Figure 10, compare the reward and alive rate achieved by the shutdown strategy and by the policy trained with fault information, both relative to the original CL policy.

In terms of reward, the Fault-num policy outperforms the shutdown strategy when RW 3 or RW 4

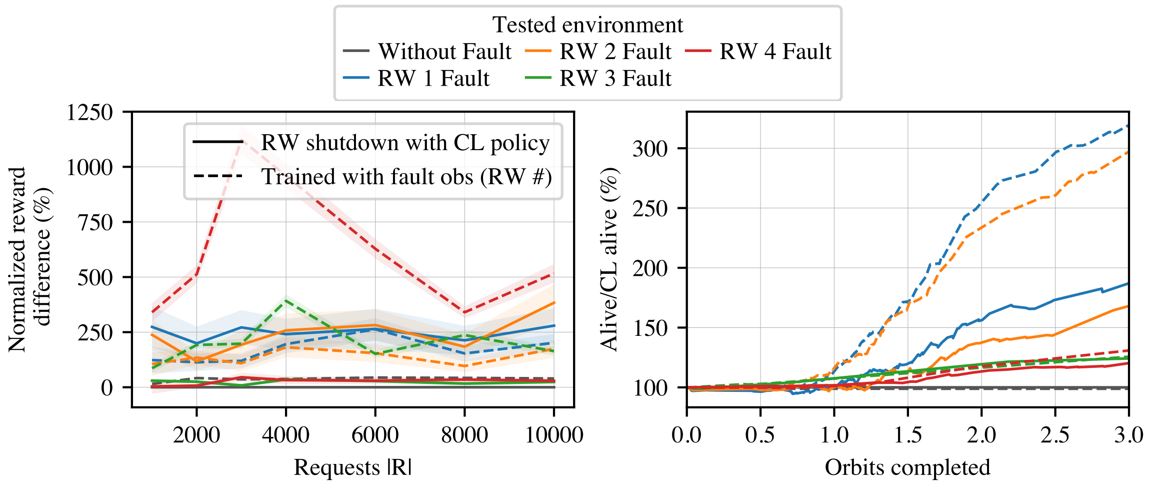

Figure 10: Reward and alive ratio of the CL policy with shutdown strategy, relative to the CL baseline policy.

is faulted, while the rewards are comparable in the other cases. Regarding the alive rate, the Fault-num policy achieves higher survivability than the shutdown strategy in the RW 1 and RW 2 fault cases, with similar performance observed in the remaining cases.Although the shutdown strategy might be effective if specifically trained for such faults, it is highly tailored and difficult to extend to other types of faults. Therefore, the next section shifts focus from severe faults to less critical ones, where the RW retains partial functionality with certain limitations.

Random Fault Conditions The Fault-random policy is trained in an environment with randomized RW fault types, where the agent is informed of which RW is faulted but receives no information about the specific fault type or its severity. Figures 11, 12, and 13 show the performance of this policy under power limit, increased friction, and encoder measurement faults, respectively. Evaluation is conducted using RW 1 faults, as faults in the other RWs exhibit similar performance trends.

Under the power limit fault, the policy trained with fault information generally outperforms the CL policy in both reward and alive rate across all fault scenarios. In particular, when the power limitation is mild, the fault-trained policy shows a substantial performance gain, indicating that it effectively adapts to slightly degraded conditions.

Under the friction fault scenario, the Fault-random policy generally achieves higher rewards than the CL policy, particularly when the fault severity is mild. However, its performance deteriorates at severe fault levels (multipliers of 400 and 1000), especially under high target request conditions, where the number of failed episodes increases. This outcome appears to result from the CL policy adopting more conservative action strategies, which are advantageous in severe fault conditions but suboptimal for milder faults.

In the encoder measurement fault scenario, the Fault-random policy significantly outperforms the CL policy in terms of reward. This improvement is attributed to the CL policy's overly conservative behavior—even in nominal conditions—stemming from its exposure to high external torques during training. While the Fault-random policy exhibits a slightly lower alive rate, the difference is minimal, with only 3 failures observed out of 1050 test cases.

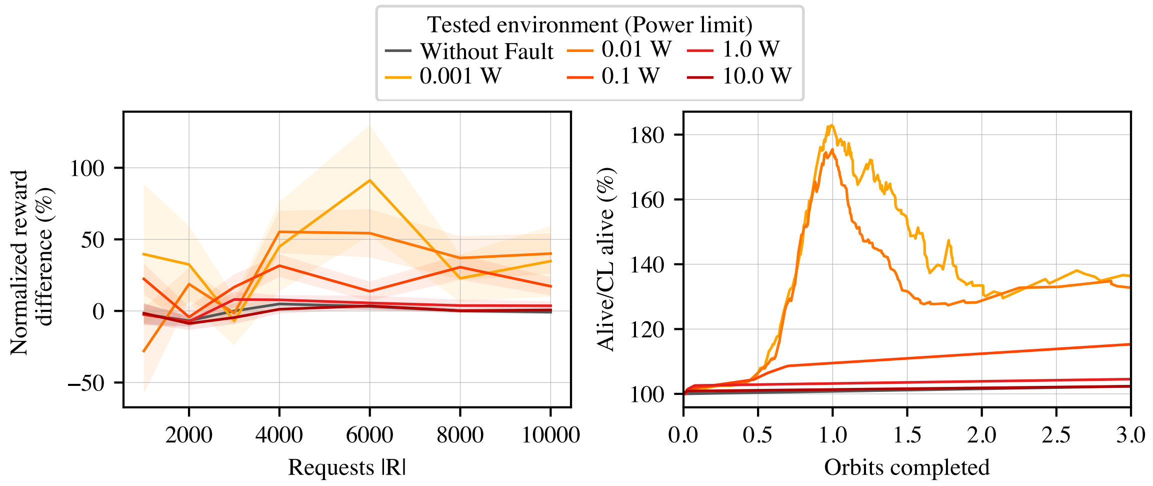

Figure 11: Reward and alive ratio of the Fault-random policy under power limit fault environment, relative to the CL baseline policy.

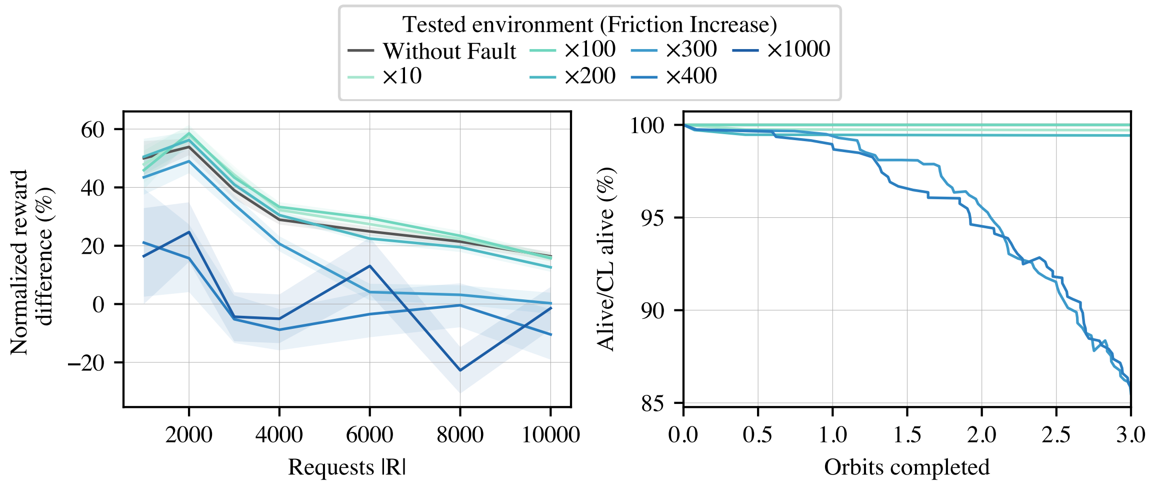

Figure 12: Reward and alive ratio of the Fault-random policy under friction fault environment, relative to the CL baseline policy.

CONCLUSION

This study investigates the impact of reaction wheel (RW) faults on a satellite tasked with imaging targets on Earth. Four types of RW faults are considered: power limit, friction, encoder measurement, and uncontrollable RW faults. The investigation begins by identifying the performance degradation points for each fault type using a curriculum learning (CL)-based policy, which is trained to be robust under conditions that deviate from the nominal case. The study then employs a reinforcement learning (RL) approach to train a fault-adaptive policy that maintains performance under both nominal and faulted conditions.

The results demonstrate that in the case of severe RW faults—such as an uncontrollable RW—training a fault-adaptive policy yields strong performance, as the agent learns to take more effective actions when desaturation is no longer viable. For less severe faults, training under those fault conditions

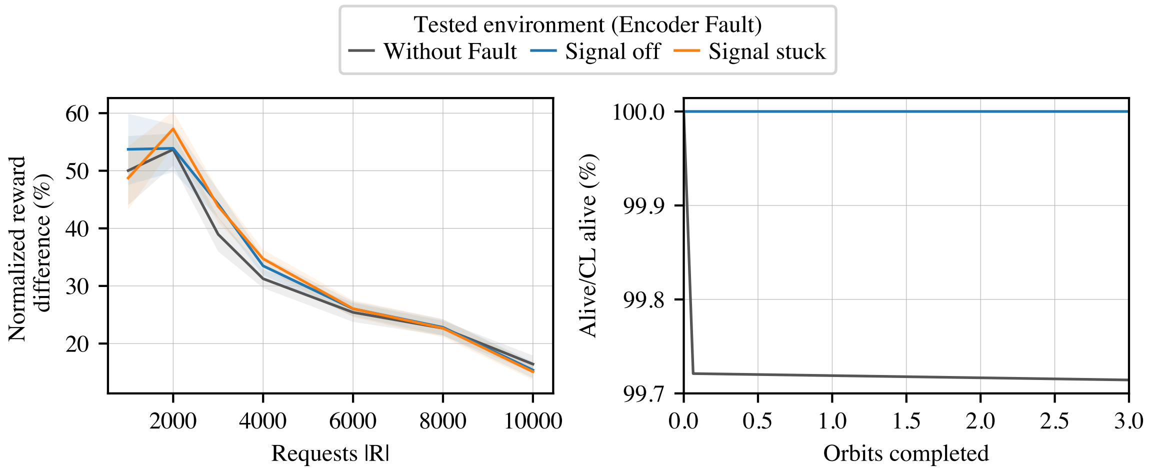

Figure 13: Reward and alive ratio of the Fault-random policy under encoder measurement fault environment, relative to the CL baseline policy.

also enhances policy resilience and overall performance, as the agent benefits from exposure to fault scenarios and information about which RW is affected.

Despite these promising results, the approach has limitations. The training environment cannot realistically represent all possible real-world fault conditions, as modeling every fault type is impractical. When exposed to a mix of fault types and severities during training, the policy may disproportionately adapt to the more frequently encountered faults. This tendency can limit the policy's ability to effectively generalize to rarer or more severe fault scenarios, potentially reducing its robustness in critical situations.

ACKNOWLEDGMENTS

This work utilized the Alpine high-performance computing resource, jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, Colorado State University, and the National Science Foundation (Award No. 2201538). Artificial intelligence tools were used to help improve the grammar and clarity of the manuscript. Final editing and approval were performed by the authors.

REFERENCES

[1] N. Bianchessi and G. Righini, "Planning and Scheduling Algorithms for the COSMO-SkyMed Constellation," Vol. 12, No. 7, pp. 535–544, 10.1016/j.ast.2008.01.001.

[2] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions," Vol. 15, No. 3, pp. 3881–3892, 10.1109/JSYST.2020.2997050.

[3] A. Herrmann and H. Schaub, "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," pp. 1–13, 10.1109/TAES.2023.3251307.

[4] V. Gabrel, A. Moulet, C. Murat, and V. T. Paschos, "A New Single Model and Derived Algorithms for the Satellite Shot Planning Problem Using Graph Theory Concepts," Vol. 69, No. 0, pp. 115–134, 10.1023/A:1018920709696.

[5] C. G. Valicka, D. Garcia, A. Staid, J.-P. Watson, G. Hackebeil, S. Rathinam, and L. Ntaimo, "Mixed-Integer Programming Models for Optimal Constellation Scheduling given Cloud Cover Uncertainty," Vol. 275, No. 2, pp. 431–445, 10.1016/j.ejor.2018.11.043.

[6] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, “Selecting and Scheduling Observations of Agile Satellites,” Vol. 6, No. 5, pp. 367–381, 10.1016/S1270-9638(02)01173-2.

[7] Y. Li, M. Xu, and R. Wang, “Scheduling Observations of Agile Satellites with Combined Genetic Algorithm,” *Third International Conference on Natural Computation (ICNC 2007)*, IEEE, pp. 29–33, 10.1109/ICNC.2007.652.

[8] A. Fukunaga, G. Rabideau, S. Chien, and D. Yan, “Towards an Application Framework for Automated Planning and Scheduling,” *1997 IEEE Aerospace Conference*, IEEE, pp. 375–386 vol.1, 10.1109/AERO.1997.574426.

[9] J. Lu, Y. Chen, and R. He, “A Learning-Based Approach for Agile Satellite Onboard Scheduling,” Vol. 8, pp. 16941–16952, 10.1109/ACCESS.2020.2968051.

[10] A. Hadj-Salah, R. Verdier, C. Caron, M. Picard, and M. Capelle, “Schedule Earth Observation Satellites with Deep Reinforcement Learning,” , 10.48550/ARXIV.1911.05696

[11] A. Harris, T. Valade, T. Teil, and H. Schaub, “Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning,” Vol. 59, No. 2, pp. 611–626, 10.2514/1.A35169.

[12] A. Hadj-Salah, R. Verdier, C. Caron, M. Picard, and M. Capelle, “Schedule Earth Observation Satellites with Deep Reinforcement Learning,” , 10.48550/ARXIV.1911.05696

[13] K. Menda, Y.-C. Chen, J. Grana, J. W. Bono, B. D. Tracey, M. J. Kochenderfer, and D. Wolpert, “Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes,” Vol. 20, No. 4, pp. 1259–1268, 10.1109/TITS.2018.2848264.

[14] M. Stephenson, L. Q. Mantovani, S. Phillips, and H. Schaub, “Using Enhanced Simulation Environments to Improve Reinforcement Learning for Long-Duration Satellite Autonomy,” *AIAA Science and Technology Forum and Exposition (SciTech)*, Orlando, FL, Jan. 8–12 2024, 10.2514/6.2024-0990.

[15] S. Peng, H. Chen, C. Du, J. Li, and N. Jing, “Onboard Observation Task Planning for an Autonomous Earth Observation Satellite Using Long Short-Term Memory,” Vol. 6, pp. 65118–65129, 10.1109/ACCESS.2018.2877687.

[16] R. Reed, H. Schaub, and M. Lahijanian, “Shielded Deep Reinforcement Learning for Complex Spacecraft Specifications,” *American Control Conference*, Toronto, Canada, July 8–12 2024.

[17] L. Quevedo Mantovani and H. Schaub, “Improving Robustness of Autonomous Spacecraft Scheduling Using Curriculum Learning,” *AAS Guidance, Navigation and Control Conference*, Breckenridge, CO, Jan. 31 – Feb. 5 2025. Paper No. AAS 25-055.

[18] Y. Nagano and H. Schaub, “Fault Resilience of Reinforcement-Based Satellite Autonomous Task Scheduling,” *AAS Spaceflight Mechanics Meeting*, Kauai, Hawaii, Jan. 19–23 2025. Paper No. AAS 25-159.

[19] S. K. Ibrahim, A. Ahmed, M. A. E. Zeidan, and I. E. Ziedan, “Machine Learning Techniques for Satellite Fault Diagnosis,” Vol. 11, No. 1, pp. 45–56, 10.1016/j.asej.2019.08.006.

[20] B. Xiao and S. Yin, “A Deep Learning Based Data-Driven Thruster Fault Diagnosis Approach for Satellite Attitude Control System,” Vol. 68, No. 10, pp. 10162–10170, 10.1109/TIE.2020.3026272.

[21] L. Manovi, A. Leboffe, A. Marchioni, E. Mariotti, M. Mangia, F. Corallo, D. Di Ienno, I. Pinci, R. Rovatti, C. Ciancarelli, and G. Furano, “Onboard AI for Enhanced FDIR: Revolutionizing Spacecraft Operations with Anomaly Detection,” Zenodo, 10.5281/ZENODO.13885531.

[22] H. Bolandi, M. Abedi, and M. Haghparast, “Fault Detection, Isolation and Accommodation for Attitude Control System of a Three-Axis Satellite Using Interval Linear Parametric Varying Observers and Fault Tree Analysis,” Vol. 228, No. 8, pp. 1403–1424, 10.1177/0954410013493230.

[23] G. Zhang, S. Qiu, and F. Wang, “Adaptive Fuzzy Fault-Tolerant Control of Flexible Spacecraft with Rotating Appendages,” Vol. 25, No. 1, pp. 326–337, 10.1007/s40815-022-01338-4.

[24] J. Ragan, B. Riviere, F. Y. Hadaegh, and S.-J. Chung, “Online Tree-Based Planning for Active Spacecraft Fault Estimation and Collision Avoidance,” Vol. 9, No. 93, p. eadn4722, 10.1126/scirobotics.adn4722.

[25] A. Nasir, E. M. Atkins, and I. Kolmanovsky, “Robust Science-Optimal Spacecraft Control for Circular Orbit Missions,” Vol. 50, No. 3, pp. 923–934, 10.1109/TSMC.2017.2767077.

[26] M. Willoughby and H. Peng, “Satellite Reorientation Using Reinforcement Learning Under Unknown Attitude Failure: Integrating Antenna and Solar Panel Controls,” *AAS/AIAA Space Flight Mechanics Meeting*, Kaua’i, Hawaii, January 2025.

[27] A. Herrmann and H. Schaub, “Autonomous Small Body Science Operations Using Reinforcement Learning,” Vol. 21, No. 10, pp. 865–884, 10.2514/1.I011376.

[28] F. Yang, C. Yang, D. Guo, H. Liu, and F. Sun, “Fault-Aware Robust Control via Adversarial Reinforcement Learning,” *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, IEEE, pp. 109–115, 10.1109/CYBER53097.2021.9588329.

[29] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-Learning in Neural Networks: A Survey," pp. 1–1, 10.1109/TPAMI.2021.3079209.

[30] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning," , 10.48550/arXiv.1803.11347

[31] X. Liu, Z. Yuan, Z. Gao, and W. Zhang, "Reinforcement Learning-Based Fault-Tolerant Control for Quadrotor UAVs Under Actuator Fault," Vol. 20, No. 12, pp. 13926–13935, 10.1109/TII.2024.3438241.

[32] M. A. Stephenson, L. Q. Mantovani, and H. Schaub, "Learning Policies for Autonomous Earth-Observing Satellite Scheduling over Semi-Markov Decision Processes," pp. 1–11, 10.2514/1.I011649.

[33] M. A. Stephenson and H. Schaub, "BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking," *75th International Astronautical Congress*, Milan, Italy, IAF, Oct. 2024.

[34] P. W. Kenneally, S. Piggott, and H. Schaub, "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," Vol. 17, No. 9, pp. 496–507, 10.2514/1.I010762.

[35] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc, fourth edition ed.

[36] M. Luo, J. Yao, R. Liaw, E. Liang, and I. Stoica, "IMPACT: Importance Weighted Asynchronous Architectures with Clipped Target Networks," , 10.48550/ARXIV.1912.00167

[37] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, 10–15 Jul 2018, pp. 3053–3062.

[38] A. Herrmann and H. Schaub, "A Comparative Analysis of Reinforcement Learning Algorithms for Earth-Observing Satellite Scheduling," Vol. 4, p. 1263489, 10.3389/frspt.2023.1263489.

[39] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, pp. 41–48, 10.1145/1553374.1553380.

[40] L. Q. Mantovani and H. Schaub, "Performance Evaluation of Shielded Neural Networks for Autonomous Agile Earth Observing Satellites in Long Term Deployment," *American Control Conference*, Denver, CO, July 8–10 2025. Submitted.