AAS 25-159

FAULT RESILIENCE OF REINFORCEMENT-BASED SATELLITE AUTONOMOUS TASK SCHEDULING

Yumeka Nagano[*] and Hanspeter Schaub[†]

This paper investigates the resilience of a policy trained using Deep Reinforcement Learning (DRL) to address the Agile Earth Observing Satellite Scheduling Problem (AEOSSP), with a focus on the impact of reaction wheel (RW) faults during operations. Although DRL-based policies are designed to handle dynamic and unforeseen scenarios, their resilience under fault conditions is not well understood. This study evaluates the adaptability of such policies under various fault scenarios, identifying thresholds for performance and safety degradation. The scheduling problem is formulated as a Partially Observable Markov Decision Process (POMDP) and solved using a policy trained in a fault-free environment. The policy is then tested across various fault scenarios, including individual RW failures, power limitations, friction increases, encoder measurement errors, and reduced battery capacity. Results explore boundaries where significant performance and safety degradation occur as a RW becomes increasingly faulted. Power limitations, friction, and battery capacity faults cause gradual performance decline as fault severity increases, with clear safety thresholds. Encoder faults, however, have minimal performance impact due to the torque-based attitude control mechanism.

INTRODUCTION

Earth-observing satellites collect data using imaging instruments, playing a crucial role in disaster tracking, environmental monitoring, and resource exploration.[1] Agile satellites, capable of maneuvering both along and across their orbital paths, offer increased flexibility in data collection. However, this flexibility also adds complexity to the scheduling process. The Agile Earth Observation Satellite Scheduling Problem (AEOSSP) has garnered significant attention due to the rising number of target requests and the need for more efficient image scheduling.[2] The primary objective of the AEOSSP is to organize satellite operations in a way that maximizes the value of the captured images. Traditionally, planning and scheduling are conducted ground-based, where a plan is created using optimizer, this plan is then sequenced and uplinked to the spacecraft for execution open-loop.[3] While effective, this approach struggles to keep pace with the increasing volume of target requests, particularly when rapid rescheduling is required for newly added tasks or the spacecraft behavior differs from the modeled behavior. To address these challenges, researchers are investigating autonomous satellite scheduling, enabling satellites to make real-time decisions onboard.[4]

The problem formulation of the AEOSSP optimization is presented, and it is demonstrated that the problem remains NP-hard, even with simplifications.[5] A problem is classified as NP-hard if

[*]PhD Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80309.

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

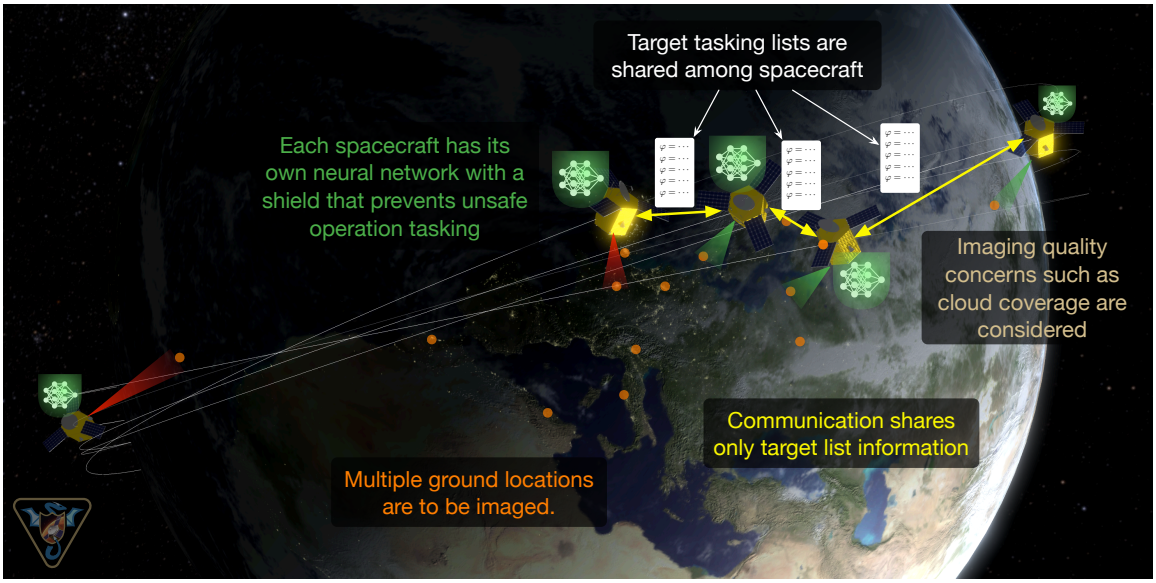

Figure 1. Concept Illustration of a Group of Satellites using Shielded Neural Networks to Achieve Autonomous Agile Earth Observing Satellite Scheduling

it cannot be solved in polynomial time in the worst case.[6] For potential onboard scheduling techniques, two exact methods are developed: Branch and Bound (B&B) and Mixed-Integer Linear Programming (MILP). The B&B method systematically explores and prunes potential scheduling solutions to efficiently identify the optimal one.[7] The MILP method formulates the AEOSSP as an optimization model with linear constraints and objective functions, providing solutions with optimality guarantees.[8] While these methods can solve the AEOSSP optimally, they are computationally intensive and are not suitable for real-time applications. Additionally, the MILP method simplifies the problem, which poses a limitation in real-world scenarios. To accelerate the scheduling process, heuristic methods are developed,[9] offering near-optimal solutions within a reasonable time frame. However, these methods struggle with the complexity of the AEOSSP, particularly when dealing with a large number of target requests and task additions, as they require an impractical number of rules to handle realistic scenarios. Additionally, other approaches, such as genetic algorithms and search methods, were proposed to address AEOSSP.[10,11] Still, they also face challenges in managing the complex constraints and large volume of target requests in real-time scenarios.

Machine Learning (ML) methods are proposed to enhance onboard scheduling by integrating offline learning with real-time onboard decision-making.[12] This combination keeps the onboard computational cost low, and the closed-loop policy allows the system to respond to newly added tasks or unexpected events. Although these learning-based approaches show promise for onboard real-time scheduling, they often simplify system dynamics and overlook critical safety factors, such as reaction wheel (RW) desaturation and energy constraints. To address these limitations, Deep Reinforcement Learning (DRL) is applied to the AEOSSP to optimize satellite scheduling, as illustrated in Figure 1. In this context, the AEOSSP is formulated and solved as a Markov Decision Process (MDP),[13,14] considering a single satellite scenario that incorporates more realistic elements such as charging, downlink, and desaturation actions.[3] The AEOSSP can also be formulated as a semi-MDP or Partially Observable Markov Decision Process (POMDP) to effectively represent the mission objectives.[15,16] This approach is further extended to the multi-agent scheduling problem by

incorporating communication capabilities between satellites after deployment.[17] To ensure satellite safety, which is often overlooked in Reinforcement Learning (RL) approaches, shields are integrated with the DRL algorithm.[18] Additionally, cloud coverage is considered for the single satellite case to enhance the selection of more effective imaging opportunities.[19] The RL-based planning and scheduling approach for small body science operations is evaluated in scenarios involving communication faults due to Deep Space Network outages.[20] This analysis highlights the advantages of the RL-based method in adapting to environmental changes. However, while these DRL-based policies are designed to handle unexpected events, its effectiveness under fault conditions remains insufficiently understood. These policies are typically trained across a range of nominal conditions but are not directly exposed to faults during training.

Traditionally, when a satellite experiences a measurement fault—defined as a measurement falling outside the nominal range—it is placed in safe mode, rendering it unable to perform any tasks until the fault is resolved. This approach, while ensuring satellite safety, is suboptimal for the AEOSSP as it can lead to missed data collection opportunities. Consequently, significant research is conducted on Fault Detection, Isolation, and Recovery (FDIR). Most of this research focuses on fault diagnosis using Fault Tree Analysis (FTA) with expert knowledge or machine learning (ML) methods.[21] Because relying on expert knowledge can miss unforeseen anomalies, implementing onboard AI to detect anomaly is investigated.[22] The Polaris project provides an open-source suite of ML tools designed to analyze spacecraft telemetry data, identify transitions between different spacecraft behaviors, and detect the most anomalous behavior changes.[23] However, these studies typically emphasize isolating the faulty system without addressing the optimal actions to take after a fault is detected.

Some research focuses on determining appropriate actions following fault occurrences. For example, attitude control is integrated with fault detection and isolation,[24] and fault-tolerant adaptive control of spacecraft attitude is investigated.[25] Satellite safety, particularly in collision avoidance scenarios, is also addressed in the presence of thruster faults.[26] Additionally, the trade-offs between mission completion and safe operation is explored by formulating spacecraft science mission control as a Markov Decision Process (MDP), solved using approximate dynamic programming, which simplifies both spacecraft dynamics and mission requirements.[27] Despite these advances, research on the resilience of RL-based approaches for satellite autonomy under fault conditions remains limited. Although some studies demonstrate the effectiveness of training policies under fault scenarios, these approaches often require specific training for each individual fault condition.[28] Therefore, this paper aims to investigate the resilience of a DRL-based policy performance in satellite autonomous tasking under fault conditions, with a focus on RW faults.

PROBLEM STATEMENT

This study considers a single satellite equipped with four RWs arranged in a configuration where three wheels align with the yaw, pitch, and roll axes, and a fourth redundant wheel is tilted at an equal angle relative to the others. The scenario involves the satellite scheduling observations of Earth targets, formulated as a POMDP. The policy is initially trained in an environment where no faults occur. Once trained, the policy is deployed in an environment where faults are introduced. The study first investigates the impact of each different RW fault out of four RW, and then tests different types of RW faults including power limit, friction, encoder measurement, and battery capacity faults. Simulations are conducted using Basilisk,* a high-fidelity and fast satellite simulation software

*https://avslab.github.io/basilisk

with flight-proven capabilities.[29] Basilisk is integrated with Gymnasium through BSK-RL,* which provides modular high-fidelity RL environments for spacecraft tasking.[30]

Problem Formulation

The satellite scheduling problem is formulated as POMDP defined by tuple $(\mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z)$, where:

- State Space $\mathcal{S}$: The state space comprises the satellite and environment states necessary for the Basilisk simulation, including the satellite's position, velocity, attitude, charge level, as well as the locations and priorities of image targets.
- 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. Both the charging and momentum management actions have a duration of 60 seconds.
- Transition Function $T(s'|s, a)$: The transition function is deterministic and generated from the simulator, ensuring $T(G(s, a)|s, a) = 1$, where $G(s, a) = s'$ represents a generative model provided by the environment.
- Reward Function $R(s, a, s')$: The reward is calculated as the sum of the priorities of the images captured during the episode. Each targets' reward is defined as:

$$R = \begin{cases} \rho_i, & \text{if target } i \text{ in } U \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

  where ρ_i is the priority of target i, and all the target requests are initially in an unfulfilled list U, then a target are moved to a fulfilled list F when the target is successfully imaged.
- Observation Space $\mathcal{O}$: The observation space is a subset of the state space which the agent can observe and use for decision-making in the problem. The observation space is normalized and listed in Table 1.
- Observation Function $Z(o|s, a)$: The observation function is deterministic, assuming that the satellite can observe the states within the observation space without uncertainty.

Three terminal conditions are defined for the episode: (1) the episode ends when it reaches the episode duration (three orbits), (2) the episode ends when the battery charge level reaches zero, and (3) the episode ends when one of the satellite's RW reaches the maximum angular speed. When the episode ends with case 2 or 3, the episode is considered as a failure.

SOLUTION APPROACH

Simulation Environment

BSK-RL The simulation environment is implemented using BSK-RL, an open-source Python package tailored for spacecraft tasking and planning. BSK-RL integrates the Gymnasium framework, widely used in reinforcement learning studies, with Basilisk, a high-fidelity astrodynamics

*https://avslab.github.io/bsk_rl

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.
ρ_i	-	Priority of upcoming target i.
t	Episode duration	Simulation time.

simulator capable of modeling spacecraft dynamics, systems, and subsystems with precision. In this setup, Basilisk serves as the generative model $G(s, a)$, facilitating the training and testing of policies under realistic scenarios that account for maneuver times and power consumption. Written in C and C++ with a Python interface, Basilisk ensures efficient interaction and rapid simulation, making it ideal for reinforcement learning applications.

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, its priority is added to the reward, 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.[5]

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 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.

Table 2. Satellite Parameters

Parameter	Value
Altitude	500 km
Mass	330 kg
Inertia	[121, 98, 82] kg m^2
Battery capacity	160 W
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

Satellite Configurations

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.[31]

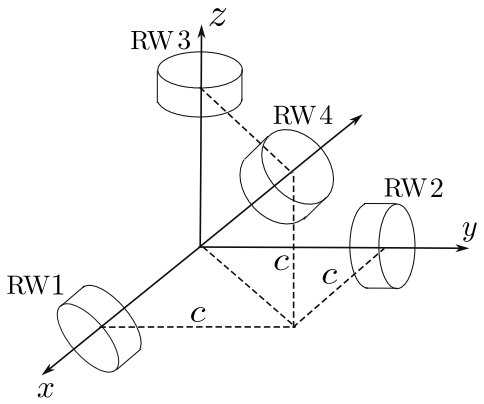

Figure 2. RW Configuration.

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 Opti-

mization (APPO) algorithm from RLlib, a library that provides scalable software primitives for RL, was used.[32] APPO is a variant of Proximal Policy Optimization (PPO).[33] The hyperparameters used for training the policy are listed in Table 3, with other parameters set to their default values in RLlib. A detailed hyperparameter analysis can be found in Reference 34. The training was conducted on the University of Colorado Boulder cluster (CURC), utilizing 32 cores and up to 20 million steps.

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
Number of SGD iterations	50
Neural Network	2 layers with 512 neurons each
Discount factor	1.0
Failure penalty	0

Faults

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. The investigation begins with scenarios where one of the four RWs becomes uncontrollable. In this case, the power supplied to the affected RW is limited to 1.0×10^{-12} W, therefore, allowing the wheel to rotate passively but rendering it incapable of contributing to attitude control. After analyzing the impact of faults on individual RWs, the study considers various types of RW-related faults, including power limitations, friction increases, encoder measurement errors, and battery capacity reductions. For each fault type, RW 1, which controls the satellite's cross-track direction, is selected for evaluation because the individual RW fault analysis revealed that faults in this maneuver had the greatest impact on overall performance. It is also noted that battery capacity faults are not specifically associated with RWs. All faults are introduced at the start of each episode. The fault scenarios are as follows:

1. **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. Power limits of 0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 5.0, and $10.0\,\mathrm{W}$ are tested.

2. **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 $0.5\,\mathrm{mNm}$, and it is scaled by factors of 10, 100, 200, 300, 400, and 1000.

3. **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.

4. **Battery Capacity Fault**: The battery capacity is reduced to 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of its nominal value, leading to faster depletion of the satellite's

charge.

Testing Environment

The trained policy is tested under both fault-free and fault-induced scenarios. The number of requests varies from 1,000 to 10,000, with tests conducted for 1,000, 2,000, 3,000, 4,000, 6,000, 8,000, and 10,000 targets. Each fault scenario is run 50 times (with different random seeds) for each target count. The same set of seed numbers is used across all fault cases. A shield is employed to prevent the agent from taking actions that could lead to failures during testing. The shield used in this study is based on the Handmade shield described in Reference 35. The shield forces the agent to take a charging action when the battery level falls below a specified threshold, and it also mandates a momentum management action if the reaction wheel (RW) angular speed exceeds its threshold. The threshold for the battery level is set to 25% of the maximum battery capacity, while the threshold for the RW angular speed is set to 70% of the maximum RW angular speed.

RESULTS

The performance of the policy 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. All fault scenarios are assessed in comparison to the fault-free case.

Individual RW Fault

First, the policy's performance is evaluated for each reaction wheel (RW) fault scenario. Figure 3 presents the following metrics relative to the fault-free case: rewards obtained, the percentage of episodes successfully completed without failures (e.g., battery depletion or RW speed exceeding the limit), and the image success rate (defined as the ratio of successfully imaged targets to the total number of imaging actions). Additionally, the figure shows the average battery level as a ratio of the maximum battery capacity. RW 1, 2, and 3 are aligned with the principal body frame axes, respectively, while RW 4 is tilted at an equal angle relative to the first three. Performance deteriorates significantly when any of the reaction wheels (RWs) becomes uncontrollable, as most actions fail to succeed under such conditions. In particular, when the across-track (RW 1) or along-track (RW 2) maneuver is uncontrollable, it not only leads to a more pronounced performance degradation but also negatively impacts safety. This is because a higher number of charging actions fail to succeed in these cases compared to faults in the other RWs.

Power Limit Fault

The performance of the policy is evaluated under a power limit fault scenario. Figure 4 illustrates the reward obtained and the image success rate relative to the fault-free case. It also shows the average angular speed of RW1 (the across-track maneuver where the fault occurs) and the average battery level. Performance gradually deteriorates as the power limit is further reduced, primarily because the image success rate decreases progressively due to the reaction wheel operating at slower speeds compared 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.

Figure 5 depicts the percentage of episodes successfully completed without failures and the breakdown of failure causes (battery depletion and RW angular speed exceeding the limit) for varying

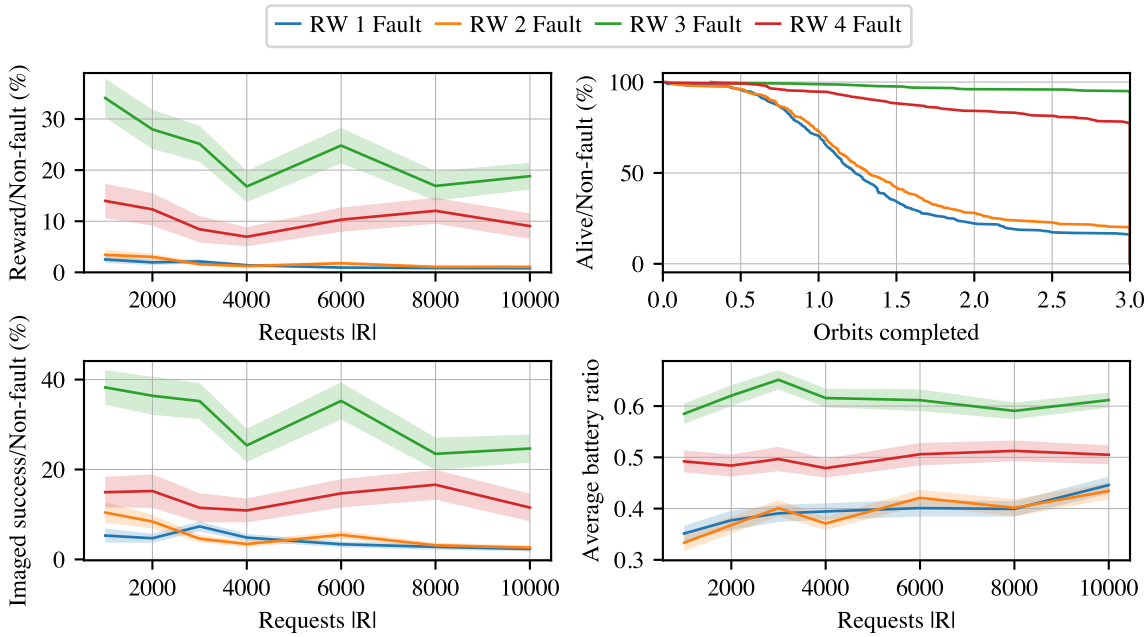

Figure 3. Each RW Fault's Reward, Alive, and Imaged Success Percentage Relative to Nominal Case and Average Battery.

numbers of requests relative to the fault-free case. Table 4 provides the ratio of time spent in eclipse at the episode's end due to failures, indicating that battery depletion does not strongly correlate with eclipse timing. 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.

Table 4. Ratio of Time in Eclipse for Power Limit Fault Scenarios

Power Limit	Ratio of Time in Eclipse
Without fault	0.67
0.001 W	0.34
0.01 W	0.35
0.1 W	0.65
0.5 W	0.43
1.0 W	0.50
2.0 W	0.44
3.0 W	0.25
5.0 W	0.50
10.0 W	0.33

Friction Fault

The performance of the policy is evaluated under the friction fault scenario. Figure 6 illustrates the reward obtained and the image success rate relative to the fault-free case. It also shows the average angular speed of RW1 and the average battery level. 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

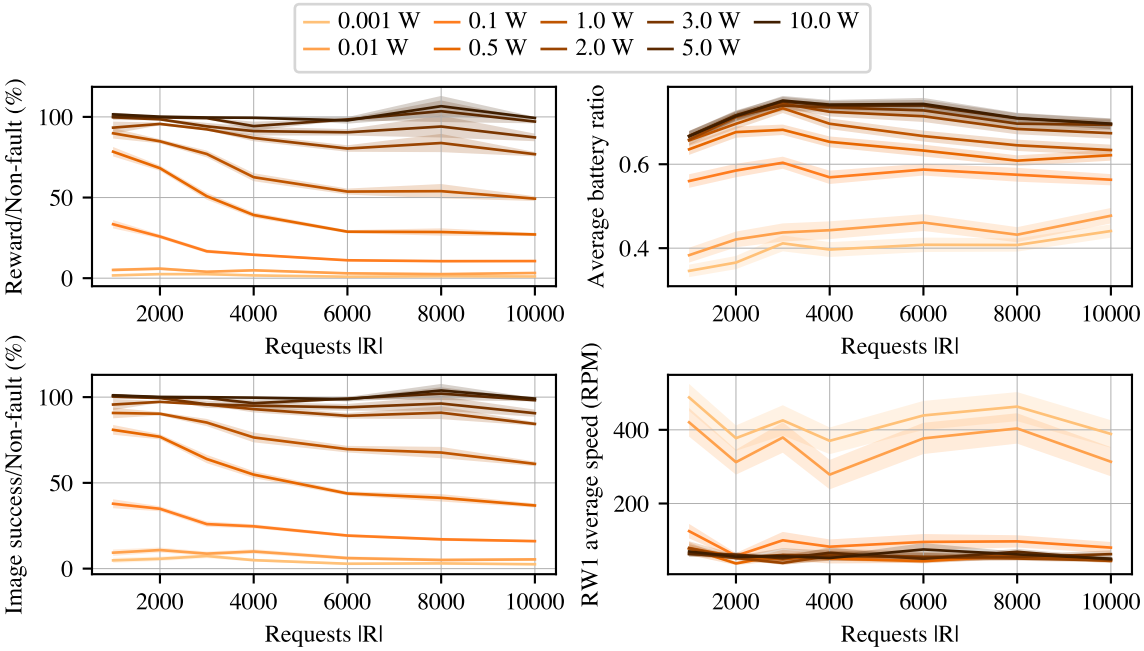

Figure 4. Reward and Imaged Success Percentage Relative to Nominal Case, RW1 Average Speed, and Average Battery for Power Limit Fault Scenario.

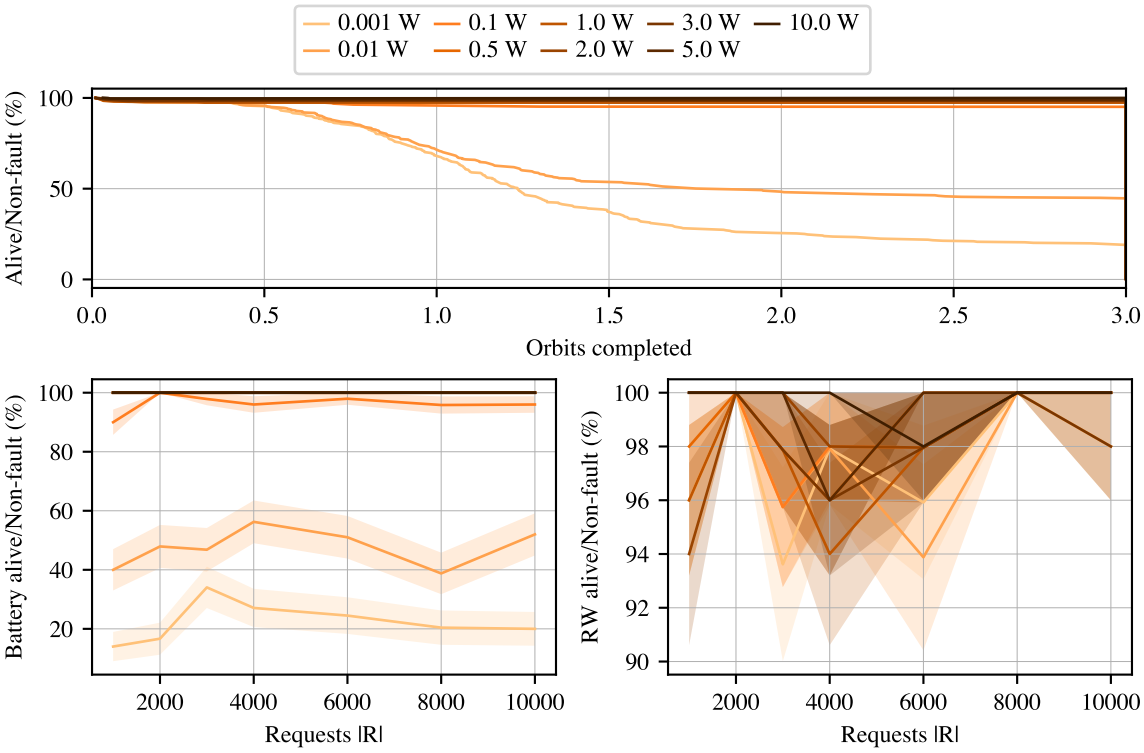

Figure 5. Percentage of Episodes Successfully Completed, Battery Depletion and RW Speed Exceeding the Limit Over the Number of Requests for Power Limit Scenario.

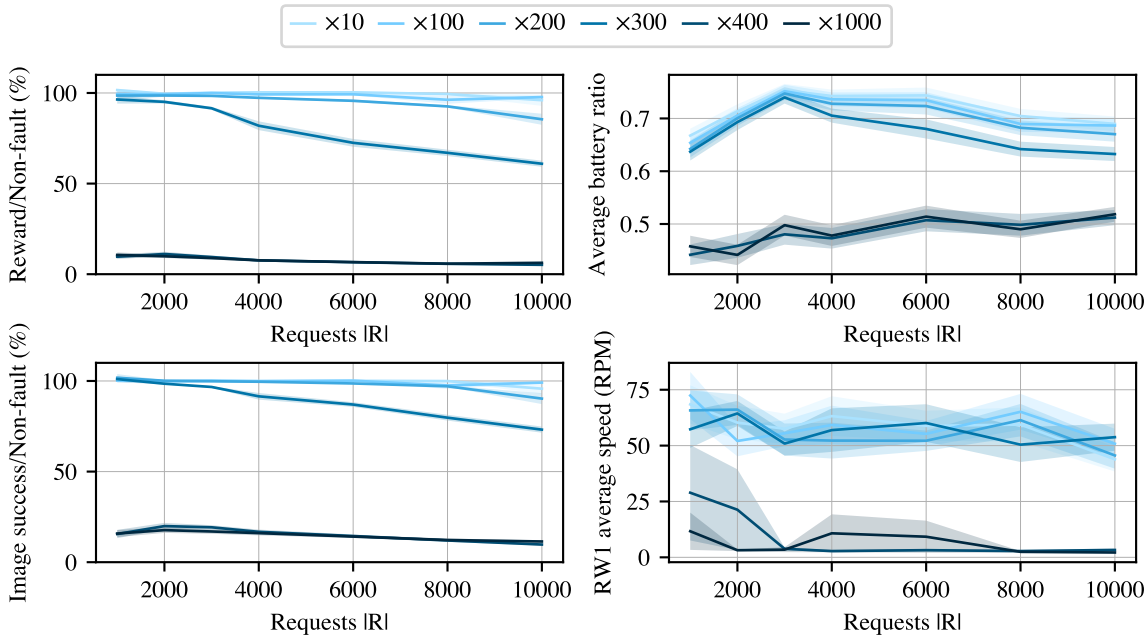

Figure 6. Reward and Imaged Success Percentage Relative to Nominal Case, RW1 Average Speed, and Average Battery for Friction Fault Scenario.

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.

Figure 7 shows the percentage of episodes successfully completed without failures, along with a breakdown of failure causes for varying numbers of requests relative to the fault-free case. Table 5 presents the ratio of time spent in eclipse at the episode's end due to failures. The safety aspect is significantly compromised when friction reaches 400 times the nominal value. As shown in Table 5, this is because charging becomes insufficient to support actions during eclipse conditions, given the higher power consumption caused by increased friction.

Table 5. Ratio of Time in Eclipse for Friction Fault Scenarios

Friction Multiplier	Ratio of Time in Eclipse
Without fault	0.67
10	0.67
100	0.67
200	0.75
300	0.57
400	0.81
1000	0.85

Encoder Measurement Fault

The performance of the policy is evaluated under the encoder measurement fault scenario. Figure 8 presents the reward obtained and the image success rate relative to the fault-free case. It also shows the average measured angular speed of RW1 alongside the actual angular speed and the average

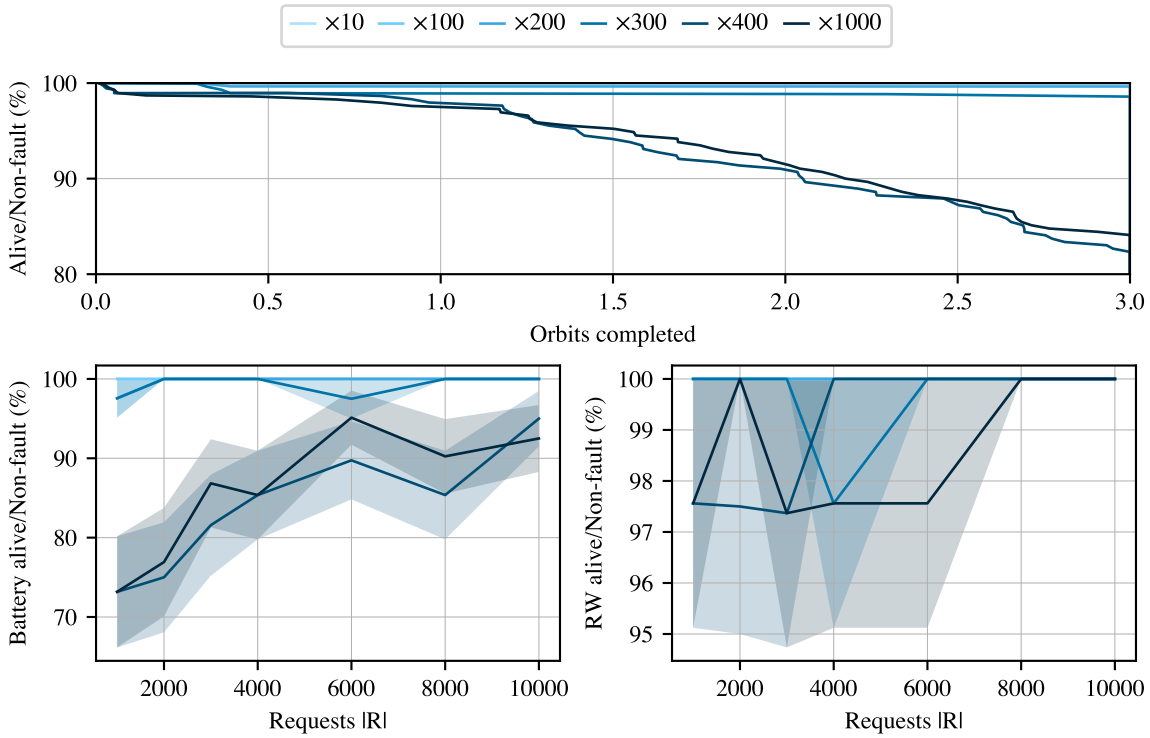

Figure 7. Percentage of Episodes Successfully Completed, Battery Depletion and RW Speed Exceeding the Limit Over the Number of Requests for Friction Scenario.

battery level. Figure 9 illustrates the percentage of episodes successfully completed without failures, along with a breakdown of failure causes for varying numbers of requests relative to the fault-free case. Overall, performance deteriorates slightly due to more frequent failures. However, the encoder measurement fault has minimal impact on performance because the attitude control system relies on torque-based control rather than speed-based control. The safety aspect is also slightly affected, as errors in RW angular speed measurements occasionally prevent successful momentum management actions when required.

Battery Capacity Fault

The performance of the policy is evaluated under the battery capacity fault scenario. Figure 10 illustrates the reward obtained and the image success rate relative to the fault-free case. It also shows the imaging action rate (the number of imaging actions relative to the total number of actions) and the average angular speed of RW1. Performance deteriorates gradually as the battery capacity decreases, as more charging actions are required when the battery capacity is lower.

Figure 11 shows the percentage of episodes successfully completed without failures, along with a breakdown of failure causes for varying numbers of requests relative to the fault-free case. Table 6 presents the ratio of time spent in eclipse at the episode's end due to failures. The safety aspect is significantly compromised when the battery capacity is 40% or less of the nominal value. As shown in Table 6, this is because charging becomes insufficient to support actions during eclipse conditions due to the reduced battery capacity.

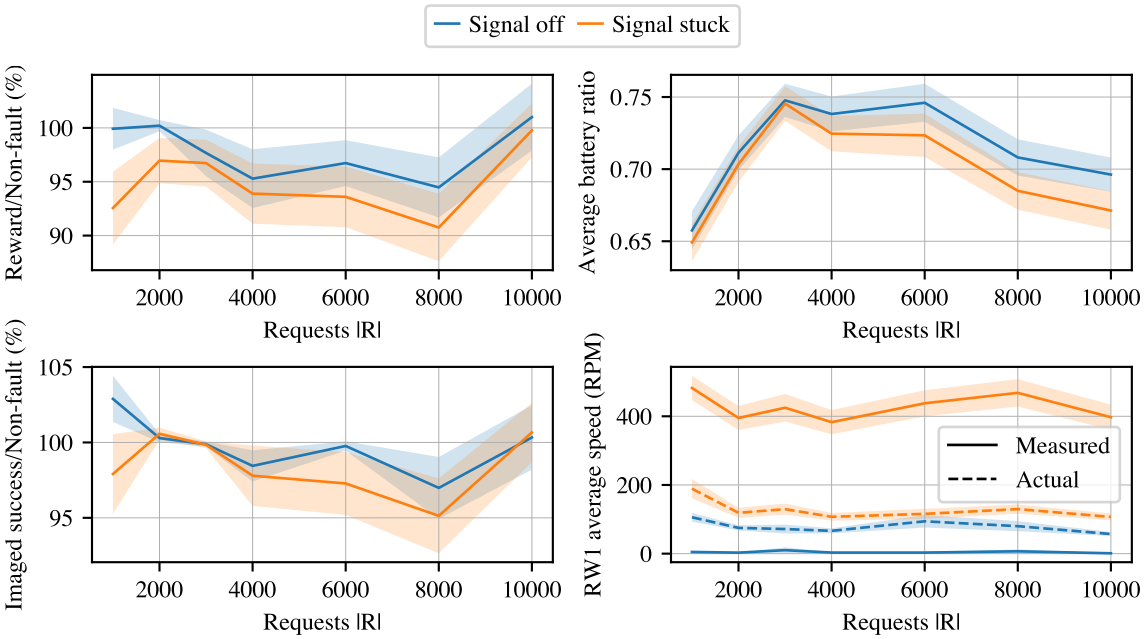

Figure 8. Reward and Imaged Success Percentage Relative to Nominal Case, RW1 Average Speed, and Average Battery for Encoder Fault Scenario.

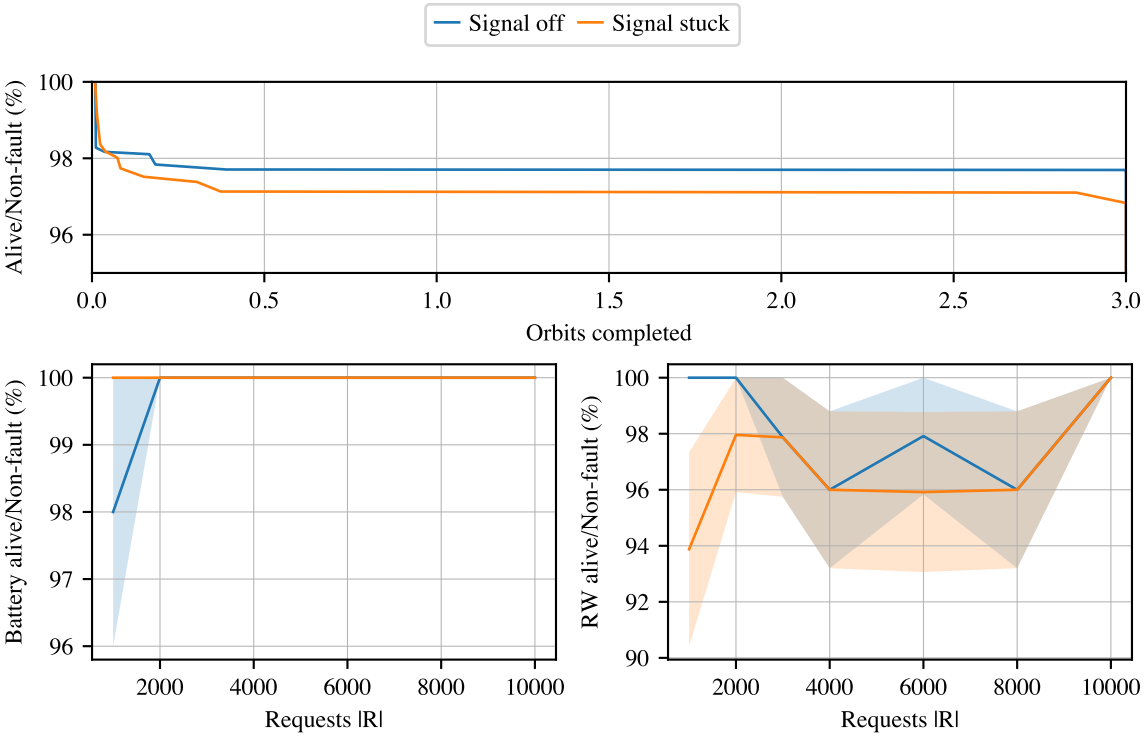

Figure 9. Percentage of Episodes Successfully Completed, Battery Depletion and RW Speed Exceeding the Limit Over the Number of Requests for Encoder Scenario.

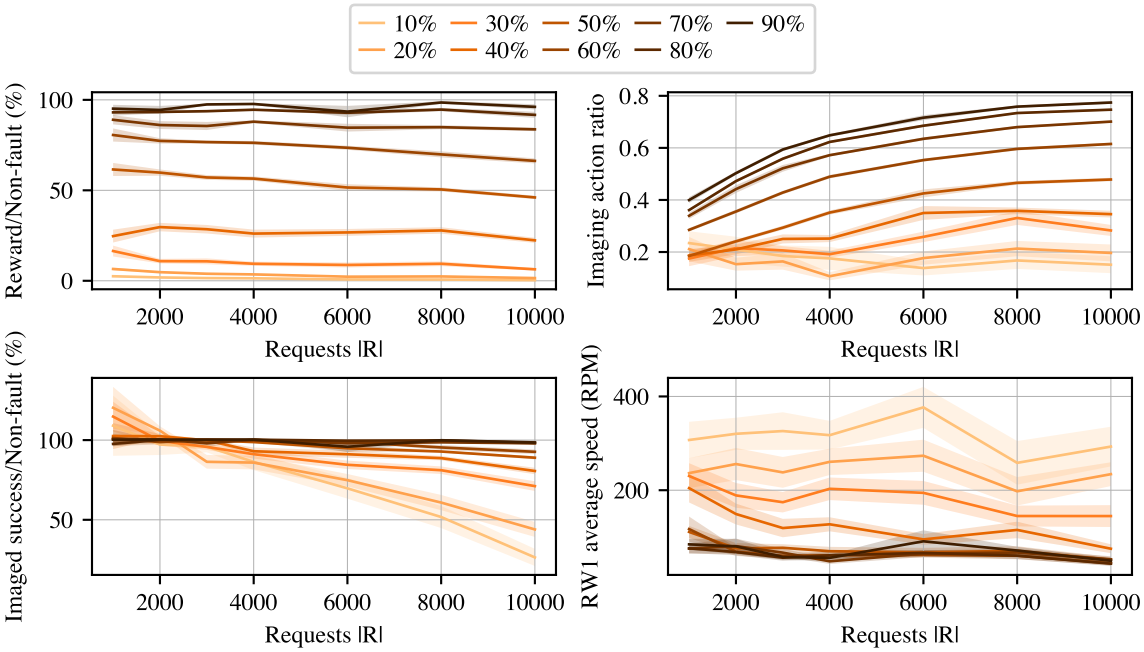

Figure 10. Reward and Imaged Success Percentage Relative to Nominal Case, RW1 Average Speed, and Average Battery for Battery Fault Scenario.

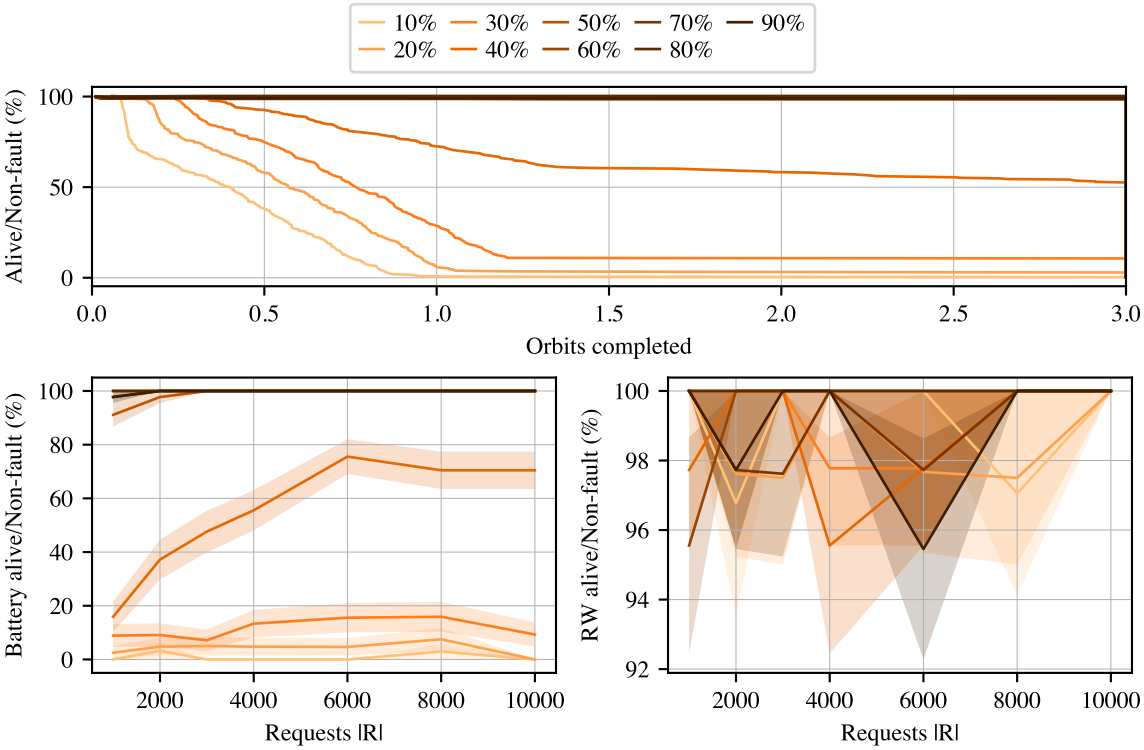

Figure 11. Percentage of Episodes Successfully Completed, Battery Depletion and RW Speed Exceeding the Limit Over the Number of Requests for Battery Scenario.

Table 6. Ratio of Time in Eclipse for Battery Fault Scenarios

Battery Capacity Reduction (% of Nominal)	Ratio of Time in Eclipse
Without fault	0.67
10	1.00
20	0.99
30	0.99
40	0.98
50	0.88
60	0.85
70	0.43
80	0.50
90	0.50

CONCLUSION

This study investigates the resilience of DRL-based policy performance in autonomous satellite tasking under fault conditions, with a focus on RW faults. The problem involves scheduling satellite observations of Earth targets, formulated as a POMDP. The policy is trained in a fault-free environment and subsequently tested under various fault scenarios. The analysis evaluates the impact of individual RW faults among the four wheels and examines different fault types, including power limitations, friction, encoder measurement errors, and battery capacity reductions.

The policy's performance is assessed based on the total reward accumulated during an episode and safety criteria, where failure is defined as either a depleted battery or an RW exceeding its maximum angular speed during the three-orbit episode. Results reveal at which level of a component failure that the policy's performance deteriorates significantly when any RW becomes uncontrollable, as most actions fail under these conditions. Safety is also compromised, with a higher incidence of failed charging actions. Furthermore, power limitations, friction, and reduced battery capacity result in a gradual decline in performance as fault severity increases, while safety thresholds are clearly defined. For less severe faults—such as power limits above 5.0 W, friction multipliers below 100 times the nominal value, and battery capacities exceeding 90% of the nominal value—the policy demonstrates resilience, with performance comparable to the fault-free case. In contrast, encoder faults have minimal impact on performance due to the torque-based attitude control mechanism. This study underscores the importance of addressing fault conditions in autonomous satellite tasking and highlights the need for resilient policies capable of adapting to such scenarios.

Future work should focus on enhancing the policy's resilience to faults. One potential direction is to train the policy in environments that include fault scenarios, allowing it to learn adaptive strategies for handling such conditions. Another approach is to integrate onboard fault detection capabilities and incorporate fault information into the observation space, enabling the agent to respond more effectively to anomalies. Additionally, the scope of this research could be expanded to include other fault types, such as thruster malfunctions, sensor failures, and solar array deployment issues, to provide a more comprehensive evaluation of DRL-based policy performance under fault conditions.

ACKNOWLEDGMENTS

This work is partially supported by the Air Force Research Lab grant FA9453-22-2-0050. Also, 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).

REFERENCES

[1] N. Bianchessi and G. Righini, "Planning and Scheduling Algorithms for the COSMO-SkyMed Constellation," *Aerospace Science and Technology*, Vol. 12, No. 7, 2008, pp. 535–544.

[2] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions," *IEEE Systems Journal*, Vol. 15, No. 3, 2021, pp. 3881–3892, 10.1109/JSYST.2020.2997050.

[3] A. Herrmann and H. Schaub, "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, 10.1109/TAES.2023.3251307.

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

[5] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "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.

[6] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. MIT Press, 2022.

[7] 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," *Annals of Operations Research*, Vol. 69, No. 0, 1997, pp. 115–134.

[8] 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," *European Journal of Operational Research*, Vol. 275, No. 2, 2019, pp. 431–445.

[9] S.-H. Mok, S. Jo, H. Bang, and H. Leeghim, "Heuristic-Based Mission Planning for an Agile Earth Observation Satellite," *International Journal of Aeronautical and Space Sciences*, Vol. 20, 2019, pp. 781–791.

[10] Y. Li, M. Xu, and R. Wang, "Scheduling Observations of Agile Satellites with Combined Genetic Algorithm," *Third International Conference on Natural Computation (ICNC 2007)*, Vol. 3, IEEE, 2007, pp. 29–33.

[11] J.-F. Cordeau and G. Laporte, "Maximizing the Value of an Earth Observation Satellite Orbit," *Journal of the Operational Research Society*, Vol. 56, No. 8, 2005, pp. 962–968.

[12] J. Lu, Y. Chen, and R. He, "A Learning-Based Approach for Agile Satellite Onboard Scheduling," *IEEE Access*, Vol. 8, 2020, pp. 16941–16952.

[13] A. Harris, T. Valade, T. Teil, and H. Schaub, "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2022, pp. 611–626.

[14] A. Hadj-Salah, R. Verdier, C. Caron, M. Picard, and M. Capelle, "Schedule Earth Observation Satellites with Deep Reinforcement Learning," *arXiv preprint arXiv:1911.05696*, 2019.

[15] 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," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 4, 2018, pp. 1259–1268.

[16] M. Stephenson, L. Mantovani, S. Phillips, and H. Schaub, "Using Enhanced Simulation Environments to Accelerate Reinforcement Learning for Long-Duration Satellite Autonomy," *AIAA SCITECH 2024 Forum*, 2024, p. 0990.

[17] M. Stephenson, L. Mantovani, and H. Schaub, "Intent Sharing For Emergent Collaboration In Autonomous Earth Observing Constellations," *AAS Guidance and Control Conference, Breckenridge, CO*, 2024, pp. 24–192.

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

[19] L. Q. Mantovani, Y. Nagano, and H. Schaub, "Reinforcement Learning For Satellite Autonomy Under Different Cloud Coverage Probability Observations," *AAS Astrodynamics Specialist Conference*, Broomfield, CO, August 2024.

[20] A. Herrmann and H. Schaub, "Autonomous Small Body Science Operations Using Reinforcement Learning," *Journal of Aerospace Information Systems*, Vol. 21, No. 10, 2024, pp. 865–884.

[21] S. K. Ibrahim, A. Ahmed, M. A. E. Zeidan, and I. E. Ziedan, "Machine Learning Techniques for Satellite Fault Diagnosis," *Ain Shams Engineering Journal*, Vol. 11, No. 1, 2020, pp. 45–56.

[22] L. Manovi, A. Leboffe, A. Marchioni, E. Mariotti, M. Mangia, F. Corallo, D. Di Ienno, I. Pinci, R. Rovatti, C. Ciancarelli, *et al.*, "Onboard AI for Enhanced FDIR: Revolutionizing Spacecraft Operations with Anomaly Detection," *Proceedings of SPAICE2024: The First Joint European Space Agency/IAA Conference on AI in and for Space*, 2024, pp. 116–120.

[23] R. Boumghar, A. Venkateswaran, H. Brown, and X. C. Alvarez, "Behaviour-Based Anomaly Detection in Spacecraft Using Deep Learning," 2021.

[24] 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," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 228, No. 8, 2014, pp. 1403–1424.

[25] G. Zhang, S. Qiu, and F. Wang, "Adaptive Fuzzy Fault-Tolerant Control of Flexible Spacecraft with Rotating Appendages," *International Journal of Fuzzy Systems*, Vol. 25, No. 1, 2023, pp. 326–337, 10.1007/s40815-022-01338-4.

[26] J. Ragan, B. Riviere, F. Y. Hadaegh, and S.-J. Chung, "Online Tree-Based Planning for Active Spacecraft Fault Estimation and Collision Avoidance," *Science Robotics*, Vol. 9, No. 93, 2024, p. eadn4722.

[27] A. Nasir, E. M. Atkins, and I. Kolmanovsky, "Robust Science-Optimal Spacecraft Control for Circular Orbit Missions," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 50, No. 3, 2020, pp. 923–934, 10.1109/TSMC.2017.2767077.

[28] M. Willoughby and H. Peng, "Satellite Reorientation Using Reinforcement Learning Under Unknown Attitude Failure: Reaction Wheel and Earth-Searching Implementation," *AAS Astrodynamics Specialist Conference*, Broomfield, CO, August 2024.

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

[30] 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.

[31] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA: AIAA, 4th ed., 2018.

[32] 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," *International Conference on Machine Learning*, PMLR, 2018, pp. 3053–3062.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[34] A. Herrmann and H. Schaub, "A Comparative Analysis of Reinforcement Learning Algorithms for Earth-Observing Satellite Scheduling," *Frontiers in Space Technologies*, Vol. 4, 2023, p. 1263489.

[35] 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.