Journal of Spacecraft and Rockets
Vol. 59, No. 2, March–April 2022

Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning

Andrew Harris,* Trace Valade,† Thibaud Teil,* and Hanspeter Schaub‡
University of Colorado Boulder, Boulder, Colorado 80309

https://doi.org/10.2514/1.A35169

The high cost of space mission operations has motivated several space agencies to prioritize the development of autonomous spacecraft command and control technologies. Deep reinforcement learning (DRL) techniques present one promising domain for the creation of autonomous agents for complex, multifaceted operations problems. This work examines the feasibility of adapting DRL-driven policy generation algorithms to problems in spacecraft decision-making, including strategies for framing spacecraft decision-making problems such as Markov decision processes, avenues for dimensionality reduction, and simplification using expert domain knowledge, sensitivity to hyperparameters, and robustness in the face of mismodeled environmental dynamics. In addition, consideration is given to ensuring the safety of these approaches by hybridizing them with correct-by-construction control techniques in a novel adaptation of shielded deep reinforcement learning. These strategies are demonstrated against a prototypical low-fidelity stationkeeping scenario and a high-fidelity attitude mode management scenario involving flight heritage attitude control and momentum management algorithms. DRL techniques are found to compare favorably to other black-box optimization tools or heuristic solutions for these problems and to require similar network sizes and training durations as widely used testing datasets in the deep learning community.

I. Introduction

TECHNOLOGICAL trends including miniaturization and the consistent decline of launch costs have dramatically reshaped the space landscape by enabling space missions that involve greater levels of operational complexity than previously feasible. New mission architectures that rely on coordination between multiple spacecraft have been proposed for domains ranging from communications to heliophysics [1]. At the same time, there is growing interest in the application of artificial intelligence and machine learning techniques to the space domain by both government [2,3] and commercial actors. This work proposes the use of deep reinforcement learning (DRL) for the generation of operations procedures for space missions of arbitrary complexity.

This work considers spacecraft operations consisting of the implementation of a mission design through the command and control of a spacecraft. As a core component of the space mission life cycle, a variety of techniques has been used to generate and implement operational plans and concepts of operations for space missions. Several early spacecraft, including Explorer 1, performed their missions with virtually no ground input after launch as a form of extremely minimal autonomy. For relatively simple demonstration missions (such as in Ref. [4]) or those that need to conduct precise maneuvers under the presence of light-speed delay (such as in Ref. [5]), a common workflow involves the generation of detailed operational plans or schedules on the ground using human experts while relying on autonomous closed-loop execution on board. Owing to the complexity of the operations planning problem, a variety of tools have been developed or discussed to aid this process. The Jet Propulsion Laboratory's Automated Scheduling and Planning ENvironment (ASPEN) tool and its related developments [6–8] use a constraint-driven job-shop scheduling approach that is amicable to onboard use, and it has been demonstrated in flight for science observation tasking on board the Earth Observing-1 (EO-1) mission. A variety of constraint-driven optimization approaches has been applied to various subproblems in the spacecraft operations domain; for example, a variety of works deal with the scheduling of image collection events [9], communication links [10], or combinations of the two. Although these approaches often produce acceptable results for small numbers of tasks or spacecraft, many have difficulties scaling as either the number of possible events, states, or spacecraft increases: especially those that rely on discrete representations of spacecraft states.

For spacecraft that are expected to conduct repetitive behavior, such as the nadir-staring CubeSats of the Planet Labs constellation [11], state-driven operations procedures can be generated that conceptualize the spacecraft as a hybrid system transitioning between discrete dynamical or operational conditions rather than a set of discrete tasks to be scheduled. These approaches are attractive from an implementation perspective because they require relatively little computational power to execute on board and can be rigorously verified and validated on the ground. However, developing state-driven rulesets that adequately meet mission criteria presents a major engineering challenge, especially when realistic models of spacecraft hardware and software behavior are considered. These challenges are amplified by changes to hardware and mission parameters that occur over a mission's lifespan, which can force the re-design of hard-coded procedures.

At the same time, deep reinforcement learning approaches have been broadly studied in the context of autonomous decision making and planning for large-scale domains, especially those that incorporate complex dynamics with few analytical models. Unlike other techniques that aim to solve Markov decision processes (MDPs), DRL techniques do not require explicit models of the environments that they are intended to solve and can instead learn from preexisting numerical simulators alone; in addition, their usage of deep neural networks for value and policy representation allows them to generalize from discrete- to real-valued representations of state, greatly enhancing their usability and avoiding one aspect of the "curse of dimensionality." This flexibility has enabled DRL-based techniques to demonstrate human or superhuman performance in tasks ranging from real-time strategy games [12–14], to the command and control of autonomous vehicles [15,16], to data-center power management [17]. Notably, DRL solutions to these

Presented as Paper 2020-0386 at the AIAA Scitech 2020 Forum, Orlando, FL, January 6–10, 2020; received 20 May 2021; revision received 20 August 2021; accepted for publication 22 August 2021; published online 13 December 2021. Copyright © 2021 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 1533-6794 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Research Assistant, Ann and H. J. Smead Department of Aerospace Engineering Sciences.

†Discovery Learning Assistant, Ann and H. J. Smead Department of Aerospace Engineering Sciences.

‡Glenn L. Murphy Chair of Engineering, Ann and H. J. Smead Department of Aerospace Engineering Sciences, 431 UCB, Colorado Center for Astrodynamics Research.

challenges not only learn to reproduce behaviors that arise from human planners but also identify novel behaviors with greater performance than human-designed approaches. These merits prompt the investigation in DRL algorithms for addressing spacecraft mission planning tasks.

Space missions present several additional challenges as compared to Earth-based applications of DRL. The existing reinforcement learning literature is primarily concerned with sample efficiency, which drives computational costs, and therefore implementation difficulty; these constraints are doubly present for space missions, which face constraints on telemetry bandwidth and may operate only a single spacecraft of a given configuration, greatly limiting the amount of data or training time available. Multisatellite missions may partially but not completely alleviate these concerns, especially if said missions consist of identically constructed spacecraft in similar orbits, and can therefore share training data; however, data used for agent training must still be traded data used for the primary mission. The safety and robustness of these machine learning (ML)-driven systems represents an additional constraint. Although many real systems share common constraints on the learning process [18], unique features of the space mission life cycle create additional factors and constraints that motivate this work. Unlike many other reinforcement learning domains, fairly accurate a priori models of system behavior are well known for space systems [19], or are at least bounded by mission requirements. At the same time, statuses that would cause a mission to fail, such as a low-power condition, must be avoided at all costs during agent execution and are desirable to avoid in training for similar reasons. As a result, safety for space mission operations must be treated and enforced as a constraint for DRL-driven operations problems.

A small collection of other works in the application of machine learning techniques to spacecraft problems exists in the recent literature, mostly focusing on the application of learning approaches to control problems in uncertain environments. Several works such as Refs. [20,21] consider reinforcement learning in the context of autonomous aerobraking planners, demonstrating the benefits of deep neural network architectures versus conventional tabular reinforcement learning for astrodynamics problems. Others explore machine learning techniques for asteroid proximity operations [22] or autonomous lunar landing [23]. Additional recent works have investigated the utility of DRL techniques for providing inputs to traditional control techniques in difficult problem domains, such as proximity operations [24]. Importantly, these approaches have focused on low-level control with reinforcement learning: an area that has been traditionally addressed by conventional estimation and control techniques with great success. In contrast, this work explicitly examines applications of reinforcement learning to high-level spacecraft planning and decision-making problems that have traditionally been the domain of rigid expert-defined policies or optimization-focused strategies.

This work is organized as follows. First, a description of a general high-level spacecraft mission operations problem is presented and contextualized in the language of partially observable Markov decision processes (POMDPs), considering specific common attributes of these problems that can be exploited to improve the efficiency of learning techniques. Next, a brief description of deep reinforcement learning algorithms such as proximal policy optimization (PPO) and its potential use in designing operations procedures is described. Specific attention is paid toward identifying techniques in the literature for ensuring the satisfaction of safety properties using both reward-engineering and formal methods. Finally, the recommendations of this work are put into practice for two representative operational challenges, outlining the technique's adaptability and merits in comparison to heuristic or timeline-driven approaches.

II. Challenges in Spacecraft Operations Procedure Design

Traditional spacecraft operations planning and execution are complex, multistep processes with many stakeholders that rely heavily on expert knowledge. For reference, a generic version of this paradigm is presented here. First, mission stakeholders specify mission objectives and a reference mission trajectory. Given this trajectory and a set of desired tasks, a set of activities is defined and scheduled as spacecraft resources (power, fuel, compute time) and mission resources (observation/maneuver/communication windows) permit. Finally, these activities are converted into an action sequence, uplinked to a spacecraft, and executed by onboard software. In parallel to these planning activities, teams of human operators typically monitor mission execution and spacecraft health parameters, and they intervene when parameters fall outside of a defined specification: either directly by changing the current action sequence or indirectly by initiating a replanning sequence. Uhlig et al. [25] identified several key aspects of the mission operations life cycle:

1) The first key aspect is downlink/uplink scheduling. Communicating results and telemetry is almost always a critical aspect of space mission operations. Many operational design processes emphasize the design and management of communication opportunities.

2) The second key aspect is orbit and attitude maneuver design. Most missions will require regular attitude slews or stationkeeping maneuvers throughout their lifetime; the design of these maneuvers and the conditions that trigger them are core components of spacecraft operations.

3) The third key aspect is operations mode design. Owing to fundamental physical or electronic constraints, it is almost always necessary to specify multiple operating states for the spacecraft's hardware and software that can satisfy both mission goals and said constraints.

4) The fourth key aspect is mission-driven tasking. Some missions, such as those focused on surveillance or targeted ground observation, involve the active assignment of spacecraft tasks to mission-relevant domains.

5) The fifth key aspect is operational plan development and execution. The preceding actions must be combined at a high level to meet a diverse set of mission goals while satisfying hardware and software constraints.

The first three components are typically shared between missions, and as a result can leverage a large body of work describing ground access prediction, orbit determination and maneuvering, and attitude control. However, no comparable body of standardized, generalized approaches exists for the development of operational plans across a variety of mission types. Although important subproblems have been automated or assisted using various techniques, other important aspects of the spacecraft operations life cycle (such as spacecraft health management) are not typically considered or would render such techniques computationally infeasible.

A. Deep Reinforcement Learning

Deep reinforcement learning algorithms seek to optimize the behavior of decision-making agents as they interact with environments that are represented as Markov decision processes. MDPs are formally defined as tuples of states S, actions A, observations O, rewards R, and functions that map between states $[s' = T(s, a)]$; they may also include mappings between states and observations $[o = H(s)]$. As their name implies, MDPs are Markovian such that system trajectories can be predicted or inferred given the system state at a single time; however, partially observable MDPs can break the Markov property through partial observability, necessitating the use of belief or memory functions to infer the status of unobserved states. In general, this work focuses on the formulation and solution of time-sequential POMDPs, as shown in Fig. 1.

These behaviors are represented by policies π that map from states or state observations to actions or action probabilities. As a differentiator from classical reinforcement learning, these policies are parameterized by the weights and biases of one or more deep neural networks. Following with the description of Markov decision processes, the objective of virtually all DRL approaches is to maximize the expected discounted reward obtained by an agent interacting with an MDP:

$$V^{\pi}(s) = E\left[\sum_{k=0}^{k=\infty} \gamma^k r_t | s_t = s, \pi(\theta)\right] \tag{1}$$

where $V^\pi(s)$ represents the value of state s under policy π, γ represents the reward discounting factor (chosen to be between zero and one), r_t is the reward value at step t, s_t is the state at step t, and $\pi(\theta)$ is a policy parameterized by a vector of parameters θ. A wide variety of DRL techniques has been developed, ranging from straightforward extensions of reinforcement learning approaches using neural networks to modern model-free policy gradient methods. To meet the aim of having a broadly applicable approach with few limitations on the structure of the action or observation space, proximal policy optimization [26] (a recently developed model-free policy gradient algorithm) is used as a benchmark algorithm for the purposes of this work. Empirical results have shown that PPO provides a robust mix of performance, relative insensitivity to hyperparameter selection, and applicability to a variety of problems owing to its use of a probabilistic policy. At the same time, because the resulting policy is not deterministic but instead a conditional probability distribution over actions given an observation, the behavior of PPO-derived agents is nondeterministic, which has important implications for safety and verification. For the reader's convenience, a brief review of policy gradient methods and PPO specifically is provided.

PPO is a simplified version of trust-region policy optimization (TRPO) that obtains robustness by restricting the size of policy updates via a clipped surrogate reward function. First, the ratio of the new and old policies is taken:

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta\text{old}}(a|s)} \tag{2}$$

PPO enforces a step-size constraint on the size of gradient updates by clipping θ updates to remain within $1 \pm \epsilon$ of the previous policy. PPO therefore adjusts the TRPO objective function to include clipping to constrain the size of a given update:

$$J^{\text{CLIP}}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{3}$$

where $\hat{A}_t$ is an estimate of the advantage function $A_\pi = Q_\pi(s,a) - V(s)$ at time t, and ε is a tunable hyperparameter described as the clipping fraction. When implementing PPO with a single neural network for both the policy and value functions, the full objective function is typically augmented to include both a value target and entropy term:

$$J^{\text{Full}}(\theta) = \mathbb{E}[J^{\text{CLIP}}(\theta) - c_1(V_\theta(s) - V_{\text{target}})^2 + c_2 H(\pi_\theta(s))] \tag{4}$$

where $H(\pi_\theta(s))$ represents the entropy of the probabilistic policy π in the state s, $V_\theta(s)$ represents the current value prediction in the current state, and V_{target} represents the value target at the current state.

A representative implementation of PPO is described in Algorithm 1. First, a set of partial trajectories is sampled from the environment using the current policy. Next, advantage estimates $\hat{A}^{\pi_k}$ are calculated from those samples using an advantage estimation algorithm such as the generalized advantage estimation method presented in Ref. [27]. Finally, minibatch stochastic gradient descent is used with the loss function described in Eq. (4) to compute the policy improvement.

Algorithm 1: Proximal policy optimization algorithm [26]

```
1: Result: $\pi(\theta)$
2:     for $k < k_{max}$ do
3:         Collect sampled transitions $D_k^\pi$ using current policy $\pi_k$
4:         Compute advantage estimate $\hat{A}_k^\pi$ for each sampled transition
5:         Compute new policy parameters using minibatch SGD with Eq. (4)
6: end
7: $a$ = SampleDistribution($\pi(o)$) return $a$
```

SGD = Stochastic Gradient Descent.

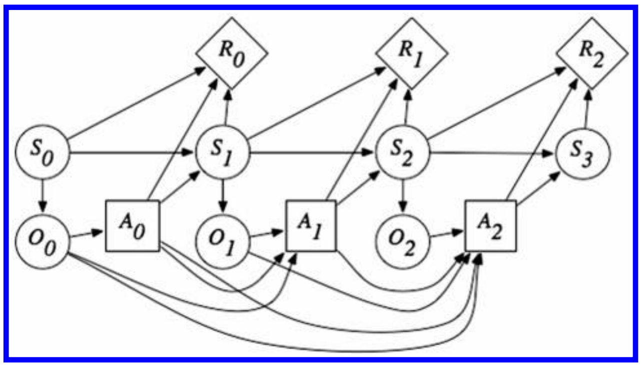


Fig. 1 Sequential partially observable Markov decision process framework for representing decision problems.

B. Safety Guarantees

Safety in the face of uncertain spacecraft performance, environmental parameters, and operating sequences is a critical requirement for future spacecraft autonomy architectures. Although some reinforcement learning techniques can bound their performance with respect to a reward function within an MDP, these weak guarantees often do not generalize, especially when moving from simulated data to real-world application. In practice, this is dealt with through reward engineering; unsafe action or state combinations are given large costs or reward penalties. This approach has several key disadvantages: many problems for which reinforcement learning is well suited have complex environment/reward interactions, which make manual reward engineering difficult. When reward engineering is feasible, it does not prevent the agent from taking unsafe actions in conditions outside the training set presented by its environment, especially when considering agents that use stochastic policies such as PPO. Finally, there is no quantifiable boundary or degree of safety provided through reward engineering. These shortcomings have motivated the search for alternative approaches to safety that can be combined with common DRL approaches.

Reactive synthesis is one category of techniques that can provide performance bounds and guarantees for controllers on specified systems. In general, reactive synthesis algorithms operate on discrete, known, finite systems and attempt to produce behavior on such systems that satisfies a specification written in a temporal logic language, such as linear temporal logic (LTL). Also described as "correct-by-construction" approaches, reactive synthesis algorithms only produce control policies that meet a given specification; if the specification cannot be met on the current system, no policy will be produced, allowing for designers to check feasibility before implementation. Although powerful for addressing systems with discrete, finite, known dynamics, reactive synthesis approaches scale poorly with system and specification complexity. These characteristics limit their applicability in solving general spacecraft planning problems, which are difficult to discretize to sufficient fidelity [20].

Shielded learning techniques [28] combine common DRL approaches with reactive synthesis-based shields to combine the power of black-box optimization with formal guarantees of safety. Shielded deep reinforcement learning (S-DRL) depends on the construction of a coarse finite-state safety MDP from the original MDP that the learning agent is intended to solve, which is conservative with respect to the original environment's dynamics and the safety specification yet limited enough that reactive synthesis can be applied to it. Next, a safety specification is created using linear temporal logic that encapsulates all desired safety conditions and is provided as an input to a reactive synthesis algorithm, such as a two-player game, which produces a discrete state-dependent strategy. Finally, this strategy is implemented alongside the learning agent as shown in Fig. 2; in this implementation, the shield accepts observations of the current system state and the action attempted by the learning agent, and it permits the action only if it aligns with the shield's strategy. This implementation architecture is applicable to both training and online use of the sequential decision agent, allowing it to provide safety boundaries during mission execution.

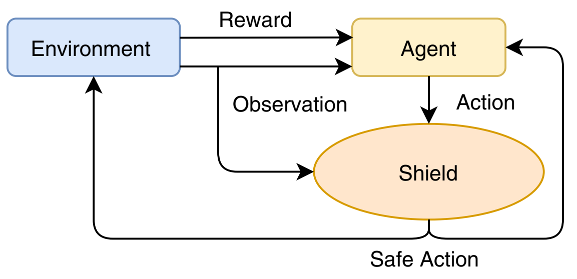


Fig. 2 Post-posed shielded reinforcement learning framework.

An example of this transformation in practice is shown for a system with two safety-critical dimensions in Fig. 3. Mission designers first identify state combinations that represent mission failure, such as depleting the spacecraft's battery or allowing reaction wheels to spin up beyond manufacturer's specifications. In addition to the hard safety constraints, operators and mission planners typically incorporate additional boundaries to act as margins of safety against actual failure; these are represented by the dashed lines labeled "operational boundary," which are used to define "warning states." While in this boundary, operators typically take immediate action to return the system to safe, nominal operating conditions. In this view, the system's behavior can be plotted on a phase plot, where individual samples of the system's true trajectory are represented as curves in the observation variable space. The continuous but bounded system creates a natural framework for the construction of a safety MDP, wherein each warning state becomes a discrete state, including products of warning states. It is important that the safety MDP contains all information necessary for the system to operate safely, which may require the inclusion of states that are not themselves safety risks but which affect the performance of actions necessary for the safety of the system. This process results in a discrete "safety" MDP that exists in parallel with the continuous POMDP.

C. Dimensionality of Objective Landscapes

Reference [29] identifies a random subspace growth methodology for identifying the intrinsic dimension of arbitrary machine learning objective problems, providing a scalar figure of merit to compare the difficulty of learning in different classification and reinforcement learning problems. This work is briefly summarized in the context of policy-based learning agents for the reader's convenience.

Given a learning agent parameterized by a set of parameters $\theta^D \in \mathbb{R}^D$, the intrinsic dimensionality d_{int} of a given problem is defined as the codimension of the solution set inside of $\mathbb{R}^D$:

$$D = d_{\mathrm{int}} + s \tag{5}$$

In general, the dimensionality of this space is nontrivial to determine analytically. To determine these dimensions empirically, an iterative process wherein the learning agent is trained with successively larger random subspaces drawn from the overall policy space is used by defining θ^D as

$$\theta^D = \theta_0^D + P\theta^d \tag{6}$$

where P is a randomly generated, orthonormalized $D \times d$ projection matrix; θ^d is a parameter vector in a subspace of D such that $d \leq D$; and θ_0^D is an initial vector suited to the problem at hand. Gradients are taken with respect to θ^d; the training process is repeated for a specified number of iterations or samples and, at some point, d is incremented; when $d < d_{\mathrm{int}}$, it is by definition not possible for the learning agent to adequately solve the problem at hand. As a result, sweeping across a range of values for d and identifying the value at which solutions appear provide an estimate for the real value of d_{int}. Due to the numerical challenge of obtaining "100%" solutions, the intrinsic dimension of an agent with 90% of the baseline solution $d_{\mathrm{int}90}$ is used as the figure of comparison for problems.

This technique is particularly attractive because it allows direct comparisons between the number of parameters required for a given neural network to sufficiently address a given problem; moreover, this technique allows for comparisons of difficulty across different problems and problem types in terms of network requirements. Given the explosion of Deep Neural Network (DNN)-driven techniques in other fields, it is desirable to understand exactly how problems in spacecraft tasking and planning relate in terms of difficulty and learnability.

III. Autonomous Tasking of Spacecraft Flight Modes

A. Spacecraft Operations as Control

This work is primarily concerned with mission operations that abstract collections of relevant low-level behaviors and states into operational modes that can be readily composed by operators as part of a general trend toward the formalization of such design practices. Mode-based operations planning is common in the small satellite domain for both Earth-oriented and deep-space missions; for example, dialects of the Colorado system test and operations language used for commanding spacecraft systems make use of mode specifications when sending spacecraft commands [30]. The use of operational modes as the basic primitives for mission planning greatly simplifies the overall learning problem and allows the use of existing tools and processes to address low-level problems, such as attitude determination and control. As with all abstractions, the application of operational modes also hides true subsystem behaviors that can impact missions on a high level. Specifically, this work conceptualizes spacecraft operations as a hybrid system consisting of discrete operational modes $q_i \in Q$ that affect the evolution of a constant set of continuous states $\boldsymbol{x} \in X$. The aim of the tasking process is to select discrete modes to maximize performance with respect to a mission

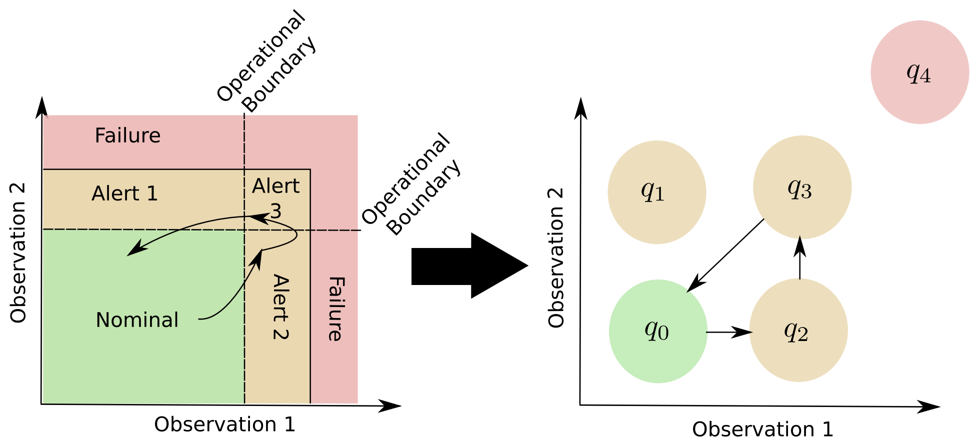


Fig. 3 Conversion from continuous states to a discrete safety MDP.

objective function $R = R(q_i, \boldsymbol{x})$ while satisfying a set of constraints corresponding to system hardware and software limitations.

A key benefit of this approach is the natural manner in which it can be translated into a partially observable Markov decision process, allowing the use of contemporary solution algorithms. For a spacecraft, the general high-level autonomy POMDP can be stated as follows. Given the constraints of orbital dynamics, onboard hardware, and predefined software behaviors, select the sequence of behaviors that best satisfies mission objectives. This framework situates the operational procedure as an "agent" that reacts to given circumstances in the state space using a set of predefined actions. Under this definition, there are multiple issues in translating from the real-world problems of spacecraft operations to the Markov framework that are common to other real-world examples [18]. In seeking an MDP formulation for spacecraft decision making, three major questions must be addressed:

1) How is time represented?

2) Which states/actions should we select? Should we consider discrete or continuous spaces?

3) How do we define reward functions, and therefore agent objectives?

Although the POMDP framework places no restrictions on the nature of any of the transition functions or states, the consideration of infinite-dimensional, continuous state, and action spaces can be extremely computationally intensive. Given the limited computational resources of both research and development efforts in aerospace, it is desirable to identify strategies for reducing the dimensionality of the state and action environment without losing representative information about the problem. Additionally, it is noted that POMDPs attempt to describe holistic system-level problems within a unified framework that is theoretically related to but practically divorced from traditional estimation and control approaches. For these reasons, POMDP-based approaches to autonomy are most frequently studied in cases where traditional estimation and controls approaches are not readily tractable, including human-assisted machine decision-making [31] or multivehicle coordination problems [32].

1. *State–Action Models*

State, action, and transition-space modeling is a critical method for encoding known information into the decision space for a learning agent. At present, it is common in the reinforcement learning space to include a wide variety of "raw" information from a system as the input to an agent (AtariNet, for example, attempts to map directly from pixels on a screen to button inputs). Shortcomings in this approach have spurned further research in the domain of world modeling and intermediate representation learning, wherein an agent learns a model of the world and intermediate representations of actions or observations in addition to policy-defining behaviors. Given the range of prior work in spacecraft state estimation and control, it is desirable instead to leverage existing state and action representations, such as the hybrid systems representation of spacecraft operations suggested in Sec. III.A, which provides a straightforward way to reduce the space of actions to a finite set of discrete operational modes and the observations to a subset of continuously or discrete-valued system states.

To further simplify the observation model, assumptions can be applied based on prior knowledge of other control strategies for hybrid systems. One well-known result to demonstrate stability of switching strategies for hybrid systems is the theory of multiple Lyapunov functions (MLFs) [33]. MLF theory demonstrates that, for a switched hybrid system, the stability of switching sequences on said system can be shown by constructing candidate Lyapunov functions for each subsystem V_i and demonstrating that said functions remain Lyapunov-like for each switching time:

$$V_i = \boldsymbol{s}_i^T P_i \boldsymbol{s}_i \qquad \text{for } \boldsymbol{s}_i \in \boldsymbol{s}(k);\, \dot{V}_i < 0 \quad \text{if } a(k) = a_i \tag{7}$$

where P_i is a positive-definite matrix associated with subsystem i.

Inspired by this approach, this work proposes "Lyapunov dimensionality reduction" (LDR) to simplify MDP construction for switched hybrid systems. Rather than reporting the entire system state to the agent, LDR proposes that it is sufficient to learn switching sequences by observing the value of candidate Lyapunov functions for subsets of the system state that are stabilized by each operational mode, alongside other information necessary to ensure proper subsystem functionality, which would otherwise break the hybrid system abstraction. LDR is specifically useful in reducing the dimensionality of planning operations that involve the management of continuous vector-valued states, such as representations of attitude state or orbital position errors, which are capable of being handled at a low-level by preexisting control techniques.

2. *Reward Functions*

A major issue in the application of MDP solution methods is the difficulty of specifying agent reward functions. In the space domain, several considerations are present. At a minimum, reward functions must be specified such that desirable results produce large rewards (or result in small penalties). When possible, it is desirable for rewards to be *shaped* such that agents can determine more- and less-desirable behaviors. For deep learning agents specifically, there is considerable debate surrounding the use of "reward engineering" to encourage exploration of alternate strategies by assigning smaller rewards to intermediate actions. Due to the complex relationship between problem dynamics, state representation, and reward functions, it can be challenging to design complex-shaped rewards that produce desired behavior. In complex environments with complex reward functions, it is common for agents to successfully optimize against a reward function that imperfectly represents the actual agent objective: a failure mode known as reward hacking.

1) Discrete events are a common mission archetype studied broadly in the literature that involve obtaining access to specific points on a planet under specific constraints (time, local solar time, etc.). Agents receive a reward for accomplishing specific mission events under the provided constraints.

2) Abstracted events, as referenced in Ref. [6], are mission-specific heuristic algorithms for scheduling science events that may already exist; in this case, these behaviors may be abstracted to a "mission mode" that provides the agent a reward for entering that mode (i.e., the learning objective is to maximize time available for mission operations).

For space missions, reward functions can be readily specified given mission-level success criteria to the degree that such criteria are known. For example, an Earth-observation mission might search for plans that maximize the amount of data downlinked to the ground, with no specifications for intermediate behavior.

B. Agent Implementation Frameworks

A major assumption in our formulation of the spacecraft control problem as a (PO)MDP shown in Eq. (16) is the discretization of time that, when combined with the mechanics of learning as described in Sec. II.A, results in decision-making agents that can only *react* to current observations, as shown in Fig. 4. Rather than using a specific plan or strategy, all relevant planning and strategy information is encoded in the deep network used by the agent. In practice, evaluating neural networks is nearly constant time and can be readily hardware

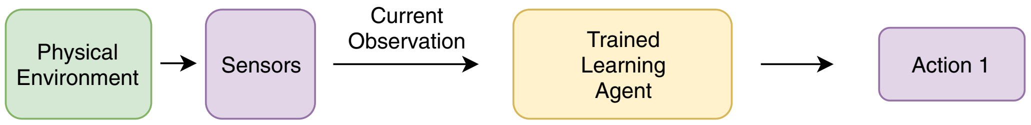


Fig. 4 Sequential decision-making agent architecture.

accelerated, making this implementation attractive for future onboard use where system information is readily available and humans are already out of the loop.

At the same time, many existing systems assume that discrete sets of actions will be periodically uplinked from the ground and lack the onboard processing power to evaluate a neural network. For these systems, an architecture that uses a ground-side simulator to propagate forward existing observations and actions is proposed as shown in Fig. 5. The incorporation of a simulator allows for the agent to make "future" decisions based on current knowledge and plan ahead. This architecture is also attractive for near-term implementation because it allows human operators to verify and validate action sequences in advance of execution.

Examination of the properties and benefits of planning versus reactive agents is left outside the scope of this work, which focuses on establishing training and safety properties for DRL-based sequential decision-making agents for spacecraft command and control.

C. Safety Guarantees

To apply the shielded learning technique to space mission operations, a simplified version of the mission POMDP is first constructed using a priori knowledge. Here, alert states are defined using the operational limits found in Table 1. These limits are applied to transform the continuous-time continuous-state system described by Eq. (16) into a simplified, discrete MDP in the observed variables, represented graphically in Fig. 6. This MDP is stated as P_{disc}:

$$P=\begin{cases} s=\{\boldsymbol{\omega}_{BN}\in\{\text{nominal},\text{high}\},|\boldsymbol{\omega}_{RW}|\in\{\text{nominal},\text{alert},\text{failure}\}, \\ \text{J}\in\{\text{nominal},\text{low},\text{failure}\} \\ o=\{q\in\{q_0,q_1,\ldots q_7,q_8\} \\ a=\{\text{Mission},\text{SunPointing},\text{Desaturation}\} \\ T=\{f_{\text{Mission}},f_{\text{SunPointing}},f_{\text{Desaturation}}\} \\ R=\{\varnothing\} \end{cases} \tag{8}$$

Although substantially smaller than the continuous-state POMDP, the safety MDP encodes important information; for example, desaturation events are only feasible when the spacecraft is not in a tumbling state, and tumbling states themselves do not lead to failure unless the battery charge or wheel speed are already near the failure criteria. In addition, the various state combinations that lead to failure are lumped into q_8 for brevity; this permits the use of the simple LTL specification

$$\varphi = G(\text{“fail”}) \tag{9}$$

which can be understood in English as "globally never allow the state to reach the failure state."

Table 1 Safety MDP labeling parameters[a]

Observed variable	Operational limit	Safety limit
$\lvert\boldsymbol{\omega}_{BN}\rvert$	0.05 rad/s	N/A
$\lvert\boldsymbol{\omega}_{RW}\rvert$, rpm	1000	1500
J_{stored}, W · h	5	0

[a]N/A denotes "not applicable."

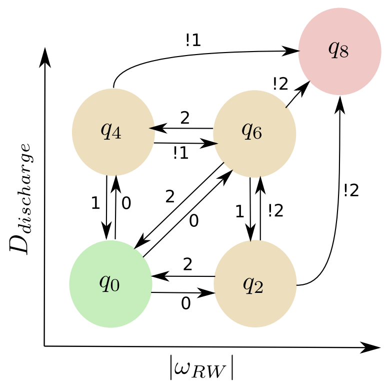


Fig. 6 Safety MDP constructed for LEO attitude mode planning simulator. $D_{\text{discharge}}$ represents depth of discharge (i.e., $1 - J$). Modes relating to "tumble" states with large body rates are omitted for clarity.

D. Safety Game Solutions

To solve this safety game, the game itself is implemented as a two-player stochastic Markov game within the PRISM-games solver. In this case, PRISM-games solves the safety game using value iteration [34]. PRISM-games then saves the shield strategy as a table which maps from shield states to safety-maximizing actions. For this work, the resulting strategy is memoryless and state based, making it especially amicable to online implementation.

IV. Reference Problems

To demonstrate how the formulation and guidelines presented earlier in this paper could be applied to practical missions, two baseline mission operations problems have been identified and implemented. These problems are briefly summarized here.

1) The first problem is the Mars science operations. This is a hybrid systems regulation problem in which the learning agent must choose between conducting orbit determination, maneuvering based on their orbit knowledge to a target orbit, or collecting science data while in the target orbit. These operational modes are implemented as a set of linear dynamical modes; the agent is rewarded based on their proximity to the target orbit while collecting science data.

2) The second problem is low-Earth-orbit (LEO) Earth observation. This scenario considers a spacecraft operating in LEO that

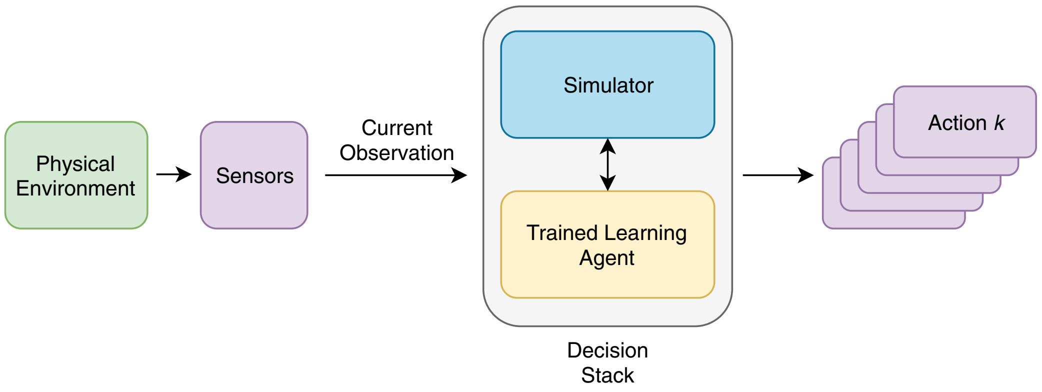


Fig. 5 Planning architecture using a sequential decision-making agent.

must maintain its health status (battery charge, wheel speed) while maximizing its time spent observing the Earth. This scenario incorporates safety constraints.

A. Mars Stationkeeping Task

1. Problem Description

Spacecraft conducting science operations typically need to maintain specific orbital parameters to achieve location-specific mission objectives. Although trajectory designers seek to minimize the impact of perturbations on such trajectories, mismodeling of these perturbations is inevitable and spacecraft typically conduct stationkeeping burns at regular intervals. Because stationkeeping performance is coupled to the spacecraft's navigation accuracy and navigation processes are not run or updated constantly, orbit determination activities must be considered when sequencing stationkeeping burns. This scenario simulates the high-level tradeoffs between estimation, stationkeeping burns, and science operations for a spacecraft that can only accomplish one mode at a time.

Due to hardware constraints, the spacecraft is capable of entering either estimation mode or control mode but not both at the same time; as a result, the spacecraft operations challenge is centered around managing the (unknown) true state error while maximizing observation time. To match the hybrid system assumption for dimensionality reduction described in Sec. III.A.1, the estimation and control modes are implemented using piecewise-Hurwitz matrices that are stable in their respective states (i.e., estimation error decays exponentially in the estimation mode and control error decays exponentially in the control mode). To represent safety constraints, these modes fail to operate if the respective error state falls outside of a specified bound; this is representative of challenges presented by linear or linearized estimation and control approaches, which face challenges when operating outside of their linear regime.

The "true" nonlinear dynamics of a spacecraft with an inertial position vector $\boldsymbol{r}$ resulting from gravity interactions are taken to follow the two-body equations of motion in the presence of perturbing accelerations:

$$\ddot{\boldsymbol{r}} = \frac{-\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_p \tag{10}$$

At the same time, a predefined reference trajectory obeying two-body dynamics without perturbing accelerations is used to define the desired mission:

$$\ddot{\boldsymbol{r}}^* = f^*(\boldsymbol{r}^*) = \frac{-\mu}{r^{*3}}\boldsymbol{r}^* \tag{11}$$

The erroneous propagator in Eq. (11) is also used to propagate forward the spacecraft's current orbital state estimate, $\hat{\boldsymbol{x}}$. The resulting state, estimate, and control errors are defined as

$$\boldsymbol{e}_s = \boldsymbol{x} - \boldsymbol{x}^*, \quad \boldsymbol{e}_{\text{est}} = \boldsymbol{x} - \hat{\boldsymbol{x}}, \quad \boldsymbol{e}_c = \hat{\boldsymbol{x}} - \boldsymbol{x}^* \tag{12}$$

The asymptotically stabilizing Cartesian continuous feedback control law for orbits defined in Ref. [35] is used in the control mode to define control accelerations that will lead back to the reference trajectory:

$$\boldsymbol{u} = -(f^*(\hat{\boldsymbol{x}}) - f^*(\boldsymbol{x}^*)) - [K_1](\boldsymbol{e}_c) - [K_2](\dot{\boldsymbol{e}}_c) \tag{13}$$

where $\boldsymbol{u}$ is the control acceleration in the planet-centered inertial frame, f^* is the two-body equations of motion, and $[K_1], [K_2]$ are positive definite 3×3 matrices. This control law is chosen due to its amicable convergence properties, which allow $\hat{\boldsymbol{x}}$ to converge to $\boldsymbol{x}^*$ from arbitrary orbits (albeit at the cost of excessive fuel usage, which is not considered in this environment). The estimation process is modeled by approximating the dynamics of a well-tuned and robust Kalman filter by stable and unstable estimate error vector and covariance matrix dynamics. When not in the estimation mode, the estimated state $\hat{\boldsymbol{r}}$ and covariance matrix $[P]$ are propagated by

$$\ddot{\hat{\boldsymbol{r}}} = f^*(\hat{\boldsymbol{r}}), [P] = [P] + [Q] \tag{14}$$

which reflects the drift of the mean estimate due to mismodeled dynamics and the steady growth of the covariance matrix due to process noise. In the estimation mode, the error vector explicitly computed and propagated separately with exponentially decaying dynamics across all states, with some noise added to the estimate to represent additional sensor noise:

$$\boldsymbol{e}_{\text{est}} = \boldsymbol{x} - \hat{\boldsymbol{x}}, \quad \dot{\boldsymbol{e}}_{\text{est}} = [A_{\text{est}}]\boldsymbol{e}_{\text{est}} + \boldsymbol{q}, \quad \hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{e}_{\text{est}} \tag{15}$$

where $[A_{\text{est}}]$ is a diagonal Hurwitz matrix, and $\boldsymbol{q}$ is a normally distributed random vector. The full MDP statement for this problem is therefore

$$P = \begin{cases} s = \{\boldsymbol{r} \in \mathbb{R}^3, \dot{\boldsymbol{r}} \in \mathbb{R}^3, \boldsymbol{r}^* \in \mathbb{R}^3, \dot{\boldsymbol{r}}^* \in \mathbb{R}^3\} \\ o = \{\boldsymbol{e}_s \in \mathbb{R}^6, \boldsymbol{e_c} \in \mathbb{R}^6, \boldsymbol{\sigma} \in \mathbb{R}^6\} \\ a = \{\text{Mission, Orbit Determination, Orbit Control}\} \\ T = \{f_{\text{Mission}}, f_{\text{Orbit Determination}}, f_{\text{Orbit Control}}\} \\ R = \{R_s, -1 \text{if } \boldsymbol{e}_s > \boldsymbol{e}_{s,\text{crit}} | \boldsymbol{e}_c > \boldsymbol{e}_{c,\text{crit}}\} \end{cases} \tag{16}$$

Although simple in its dynamics and implementation of spacecraft estimation and control constraints, this problem reflects real-world challenges in managing couplings between state estimation and control for real spacecraft. Because the spacecraft can only control with respect to its current state estimate, failing to reduce its estimation error can cause divergence in the true state error as the spacecraft computes burns that inaccurately reflect the current state error. As a result, this problem tasks an agent with managing both mismodeled dynamics, coupling of estimation and control processes, and optimization of total mission science time given an orbit accuracy constraint while remaining computationally quick to execute.

2. Shield Construction

The primary challenge for safety properties in this environment is ensuring that the agent has enough knowledge of its true state (obtained by entering the estimation mode) to correctly understand which state it should enter. The discretized system used to represent the safety game is shown in Fig. 7, with the variables q_i representing enumerated discrete safety conditions and with "ctrl" and "est" representing the dominant transitions between those discrete states. If the agent's estimator covariance is low and its state error is close to the linearity constraint, the shield will force the agent to conduct a stationkeeping burn; if both errors are high, the graph is constructed to bias the agent toward conducting estimation modes to ensure that the estimated errors are actually as large as they believe.

B. LEO Attitude and Health Management Task

1. Problem Description

To represent the feasibility of applying DRL techniques to spacecraft health keeping as well as stationkeeping, a scenario reflect-

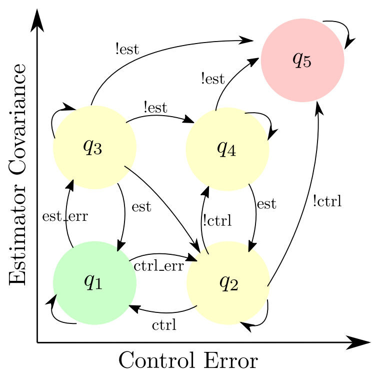


Fig. 7 Safety MDP constructed for the stationkeeping task.

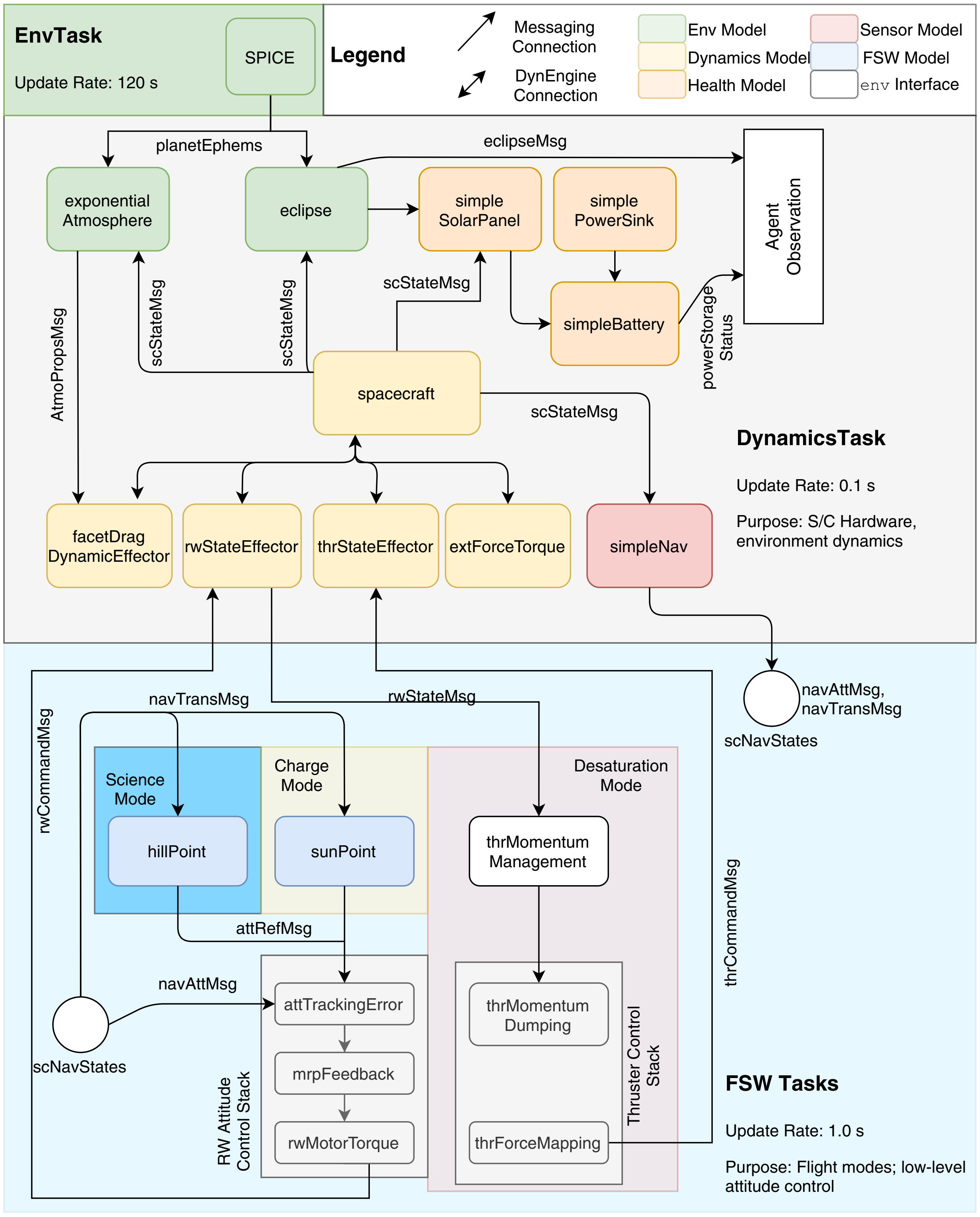


Fig. 8 Simulation block diagram for the LEO attitude and health management task (FSW = Flight Software; S/C = Spacecraft; SPICE = the JPL Spacecraft, Planet, Instrument, Orientation, and Events system).

ing the challenges of day-to-day operations in low Earth orbit is presented. A spacecraft on a predetermined trajectory around the Earth is tasked with maximizing its time spent conducting an Earth observation task (represented by a nadir pointing attitude) while maintaining onboard power and dumping excess reaction wheel momentum. This scenario is implemented in the Basilisk software framework, and it uses existing flight heritage control laws and hardware models. A representative block diagram of the simulation components is shown in Fig. 8; further documentation on these components and their functionality can be found in the Basilisk documentation.[§] As a result, this environment represents a more realistic challenge for prospective planning and scheduling approaches because agents interact directly with a simulation stack intended for Attitude Determination and Control System (ADCS)

[§]http://hanspeterschaub.info/basilisk/index.html [retrieved 16 November 2021].

design and verification rather than an approximation of those systems made more amicable to planning approaches.

Stated formally, the full-system POMDP under the assumption of full observation of system states on board is provided by Eq. (17):

$$P=\begin{cases} s=\{\boldsymbol{r}\in\mathbb{R}^3,\dot{\boldsymbol{r}}\in\mathbb{R}^3,\boldsymbol{\sigma}_{BN}\in\mathbb{O}^3,\boldsymbol{\omega}_{BN}\in\mathbb{R}^3,\boldsymbol{\omega}_{RW}\in\mathbb{R}^3,J\in\mathbb{R}^1\} \\ o=\{\sigma_{BN}\in\mathbb{O}^3,\boldsymbol{\omega}_{BN}\in\mathbb{R}^3,\boldsymbol{\omega}_{RW}\in\mathbb{R}^3,J\in\mathbb{R}^1\} \\ a=\{\text{Science},\text{Charge Mode},\text{Desaturation Mode}\} \\ T=\{f_{\text{NadirPointing}},f_{\text{SunPointing}},f_{\text{Desaturation}}\} \\ R=\{r_s,-1\,\text{if}\,J=0\,\text{or}\,|\boldsymbol{\omega}_R W|>250\,\text{rad/s}\} \end{cases} \tag{17}$$

In this case, it is assumed that an operations agent would observe all relevant onboard dynamic information as it is observed or estimated. The reward function is engineered to provide the agent with a positive reward that is inversely proportional to the attitude error $\boldsymbol{\sigma}_{\text{BR}}$ when in the science mode:

$$r_s=\frac{1}{\boldsymbol{\sigma}_{\text{BR}}^T\boldsymbol{\sigma}_{\text{BR}}+1} \tag{18}$$

In general, attitude transients may not settle out within one time step, even with proper time-step size selection. In addition, there is a tradeoff between maintaining the Markov property and the granularity of decision-making intervals that can be challenging to resolve; longer time steps are more likely to resolve transient behavior but result in fewer planning intervals over a set period of time, and therefore less flexibility for agent responses in other nonattitude system domains. By rewarding agents for entering science mode with small attitude errors, this reward function ensures that agents that must take multiple steps in the science mode to settle out transient behavior are properly rewarded. Rewards are scaled such that the maximum achievable reward over one environment run is one; this simplifies reward engineering for failure states, which are defined as providing a reward of -1 and ending the scenario, ensuring that the maximum reward for a failed run is zero. This strategy has several appealing properties: it simplifies analysis of agent performance because best- and worst-case scores are knowable; it clarifies cases wherein agents fail; it is unit norm, and therefore unlikely to cause numerical issues; and rewards are continuously shaped and convex around a desired objective but also simple to implement.

In addition to this structured reward, feature engineering inspired by the LDR approach described in Sec. III.A.1 is also applied to the base POMDP described by Eq. (17). Rather than observing the current attitude and reference states separately, the agent is instead provided with the magnitude of the error Modified Rodriguez Parameter (MRP) state, which compactly represents the overall attitude error. Similarly, the overall body to inertial angular velocity is reduced to the norm angular velocity, as is the reaction wheel speed vector. These modifications reflect the intended behavior of each mode and summaries relevant to the reward function. In this problem, applying LDR reduces the system dimensionality from 13 individual elements to five elements:

$$P=\begin{cases} s=\{\boldsymbol{r}\in\mathbb{R}^3,\dot{\boldsymbol{r}}\in\mathbb{R}^3,\boldsymbol{\sigma}_{BN}\in\mathbb{O}^3,\boldsymbol{\omega}_{BN}\in\mathbb{R}^3,\boldsymbol{\omega}_{RW}\in\mathbb{R}^3,J\in\mathbb{R}^1\} \\ o=\{|\sigma_{BN}|\in\mathbb{R}^1,|\boldsymbol{\omega}_{BN}|\in\mathbb{R}^1,|\boldsymbol{\omega}_{RW}|\in\mathbb{R}^1,J\in\mathbb{R}^1\} \\ a=\{\text{Science},\text{Charge Mode},\text{Desaturation Mode}\} \\ T=\{f_{\text{NadirPointing}},f_{\text{SunPointing}},f_{\text{Desaturation}}\} \\ R=\{r_s,-1\,\text{if}\,J=0\,\text{or}\,|\boldsymbol{\omega}_R W|>250\,\text{rad/s}\} \end{cases} \tag{19}$$

2. *Shield Design*

Owing to its higher complexity, the shield design problem for the LEO attitude mode selection problem is substantially more complex. In this case, the system is again discretized in accordance with safety-relevant states along expert-defined the operational thresholds listed in Table 1, resulting in the safety MDP shown in Fig. 6. However, the reaction wheel desaturation controller computes the momentum to be removed by thruster impulses under the assumption that the spacecraft is near stationary; as a result, triggering this mode (action 2 in Fig. 6) will destabilize the spacecraft, potentially *increasing* the momentum in the reaction wheels and triggering a failure state. To prevent this, additional states representing a combination of one or more of the safety conditions and tumbling above the body rate specified in Table 1 are added to the safety MDP; it is assumed that these states can be transitioned from by using the sun-pointing mode, which shares a reference attitude with the desaturation mode for simplicity.

V. Feasibility Analysis

This section aims to demonstrate the merits of DRL-based approaches for addressing spacecraft tasking and planning challenges using the representative environments described in Sec. IV. DRL-based approaches are evaluated against four key questions:

1) The first question relates to hyperparameter sensitivity. How sensitive is the performance of DRL agents to the selection of appropriate hyperparameters?

2) The second question relates to absolute performance. How does the performance of DRL agents compare against other approaches?

3) The third question relates to the sensitivity to environmental parameters. How robust is DRL performance to small changes in the environment?

4) The fourth question relates to network size. What network size is required to reach benchmark performance?

These questions aim to justify exploring the usage of DRL in the face of common criticisms of these approaches in the field, such as described by Refs. [18,36].

To provide points of comparison for absolute performance, both a heuristic method based on the shields constructed for the S-DRL techniques (described in Algorithm 2) and a timeline-optimizing genetic algorithm (GA) are evaluated on identical versions of the environments used to train and test the DRL-based optimizers.

The heuristic method described in Algorithm 2 is a straightforward extension of the shield approach described in Sec. III.C. Rather than releasing control to the DRL agent outside of warning states, the heuristic agent simply chooses an operational mode that maximizes a one-step reward return when the shield is not active. This algorithm is intended to represent the performance of a rule-based approach to autonomy designed with safety in mind; however, because it uses an identical shield to the S-DRL approaches, it also provides a reference for the additional benefit of implementing DRL behind the protection of a shield algorithm.

On the other end of the autonomy spectrum is timeline optimization, wherein a set of spacecraft activities IS sequenced over a time horizon while respecting resource constraints and mission objectives. To represent these approaches, a GA-based approach is used as a point of comparison. Rather than optimizing a policy, this algorithm uses a GA to optimize a timeline of modes with the same mode interval and total number of modes used to train and evaluate the DRL-based approaches. A vector of integers is used to represent the sequence of modes, using a discrete minimum planning interval to allow for mode switching. Parameters describing the timeline-optimization genetic algorithm are given in Table 2; Fig. 9 shows the

Algorithm 2: Heuristic greedy-safe action selection

```
1:  Result: $a$
2:  $q_k$ = ShieldDiscretizer($\boldsymbol{o}_k$);
3:  if $q_k \in Q_{nominal}$, then
4:      $a$ = Reward Mode;
5:  else
6:      $a$ = ShieldPolicy($q_k$)
7:  end
8:  return $a$
```

Table 2 Timeline-optimization genetic-algorithm parameters

Parameter	Value
Selection criteria	Tournament
Tournament Size	3
Crossover mechanism	None
Mutation mechanism	Uniform with probability of 0.01
Mutation probability	0.75
Generation size	24

Table 3 Parameters and parameter ranges used in hyperparameter search

Parameter	Baseline value	Range
Discount factor γ	0.99	(0.9, 1)
Batch size n_{steps}	64	(32, 240)
Clip range ϵ	0.3	(0.1, 0.3)
Entropy coefficient	0.1	(0, 0.3)
Network shape	(64, 64)	N/A

relative convergence of this approach on both environments using a log–log scale.

A. Hyperparameter Sensitivity

Most deep learning approaches are dependent on proper selection of hyperparameters such as learning rate, network size, or reward discount factors for good performance; indeed, on many classic DRL tasks, the selection of correct hyperparameters can be the distinction between successful agents and policies that are worse than random. It is expected that through the correct construction of spacecraft operations policy problems, this extreme sensitivity to hyperparameter selection can be avoided. To examine this sensitivity, DRL agents were trained over multiple random seeds at a grid set of hyperparameters listed in Table 3 in both the simple-science and LEO attitude management environments; after training, simple linear fits were performed on both sets of data to establish the sensitivity of agent returns to these hyperparameters.

The results of this survey are shown in Figs. 10 and 11. In general, we find weak correlations between both overall agent performance and specific hyperparameters, with the exception of the discount factor γ and batch size.

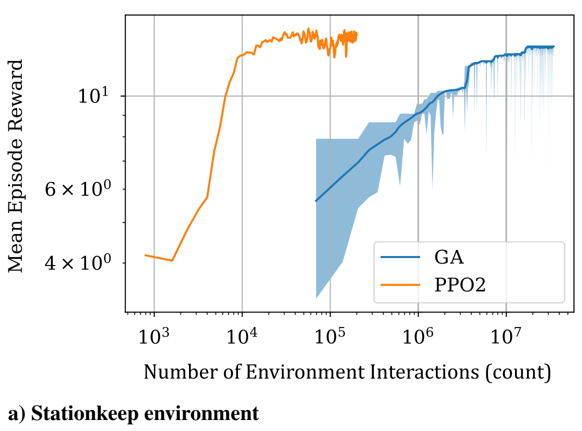
Both environments are specified as finite duration, and therefore do not require the use of discount factors for overall episode return convergence. High discount factors reflect long periods of viability for rewards in a specific environment, meaning that rewards (and penalties) should propagate backward farther during training. This result suggests that the LEO operations problem has a complex and long-lived dynamics, which must be accounted for in the planning process, whereas the stationkeeping environment performs best with a slightly smaller discount factor.

Batch size is regarded as an important factor for policy gradient methods; large batches may inadvertently average out desirable behaviors, whereas small ones may result in erroneous gradient updates if poor actions are over-represented in the batch. The batch sizes listed in Figs. 10a–11a represent the n_{steps} parameter in the stable baselines implementation of PPO for parallel task execution (referred to interchangeably as both PPO and PPO2); the true batch size used in gradient updates is defined by $n_{\text{steps}} \times n_{\text{cpus}}$, where n_{cpus} represents the number of CPU cores used during training and is set to 24 for all presented results. Both environments display good performance across a range of values but begin to suffer at extremely large or small values of the batch size parameter.

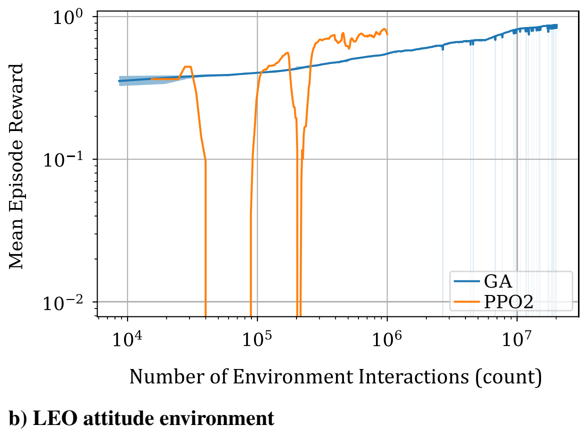
B. Performance Comparison

This analysis aims to compare the performance of DRL-derived agents against other solution methods, which are represented by the greedy-shield agent described in Algorithm 2 (representing safety-aware rule-driven autonomous tasking approaches) and the timeline-optimizing genetic algorithm described. To mitigate performance losses associated with brittleness, a "nominal" set of initial conditions was selected for both environments that is intended to represent realistic, feasible operational constraints a tasking agent may be faced with in flight; as a result, the GA-optimized timeline for that operational condition is treated as an upper bound on the optimal reward achievable in that environment.

Given the high-level nature of the designed reference environments, a genetic-algorithm-based scheduler was implemented to ground the results of the DRL-driven responsive tasking approach. The genetic algorithm, built using the Distributed Evolutionary Algorithms in Python (DEAP) evolutionary computing toolbox, encodes an action sequence for a given agent as a list of integers reflecting operational modes with a length corresponding to the maximum number of steps available in a given environment, which is identical to the maximum number of steps allowed during DRL training. The parameters and selection mechanism for this GA are listed in Table 2. A comparison of the final evaluated reward between the heuristic agent, the GA-based scheduler, and the DRL-based agents is shown in Table 4. Although the GA-driven approaches generally perform 3–5% better on the reward metrics in the nominal environment, they struggle to find operational timelines that are well suited to a variety of initial conditions. On the other hand, the heuristic agents using the shield policy find better-than-random performance but generally do not match the performance of either the DRL or S-DRL agents. This result suggests that the addition of DRL to the mode optimization problem provides benefits to mission performance beyond heuristic policies alone.



a) Stationkeep environment



b) LEO attitude environment

Fig. 9 Comparison of training curves versus environment interaction count for both PPO and a timeline-driven GA.

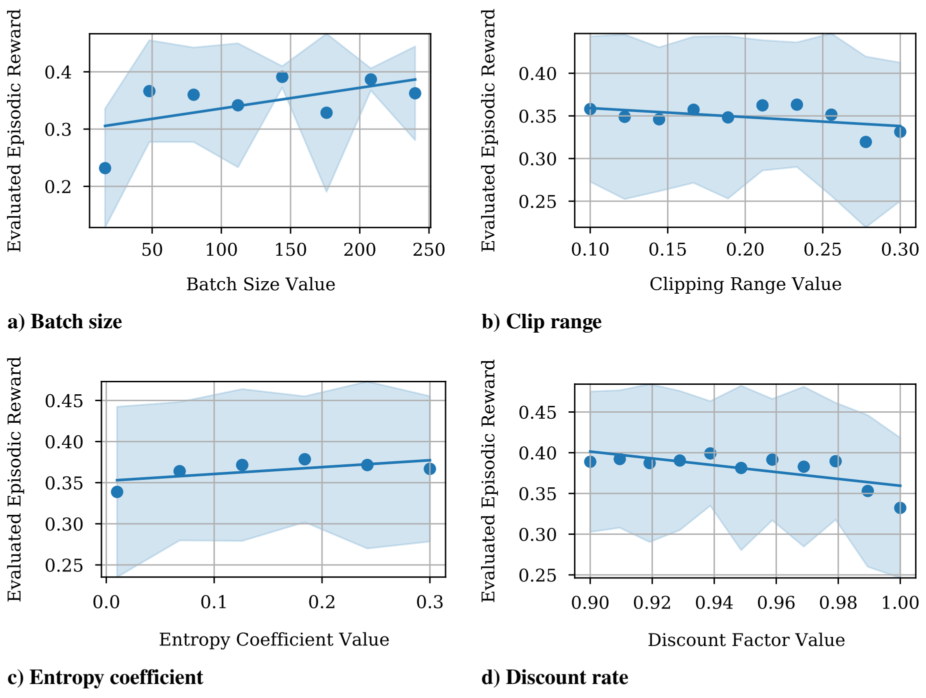


Fig. 10 DRL performance on station keeping task across selected hyperparameters. Dots represent mean performance, shaded regions indicate 1-σ covariance bounds, and lines represent linear fits to show overall trends.

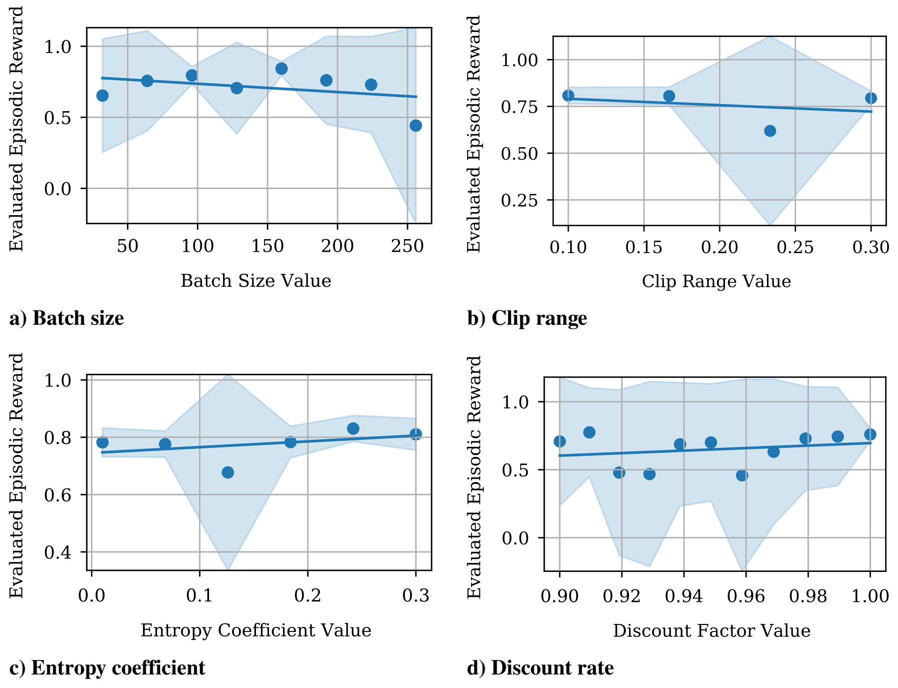


Fig. 11 DRL performance on LEO health management task across selected hyperparameters.

Table 4 Summary of performance for PPO, shielded, timeline, and heuristic agents

	Stationkeep		LEO attitude	
Algorithm	Demonstration	Random	Demonstration	Random
Heuristic	$0.0001779 \pm 4.524 \times 10^{-7}$	$0.002402 \pm 2.5299 \times 10^{-7}$	0.8500 ± 0.000	0.7372 ± 0.09812
GA	0.1513 ± 0.0001408	0.1445 ± 0.000557	0.8700 ± 0.00	-0.5576 ± 0.03426
PPO	$0.2983 \pm 7.4785 \times 10^{-5}$	0.3001 ± 0.000152	$0.8024 \pm 1.381 \times 10^{-4}$	0.7624 ± 0.05000
Shielded PPO	$0.2955 \pm 9.489 \times 10^{-5}$	0.2919 ± 0.01	$0.8406 \pm 5.4521 \times 10^{-5}$	0.8038 ± 0.0001516

One notable area of difference between both approaches is the computational complexity induced by the timeline-driven approach. In this case, evaluating one gene requires a full run through the simulation environment; given that large populations are typically required for good convergence properties, this approach requires the dedication of substantial computational resources to simulating these action trajectories. Although DRL is frequently described as data intensive, Fig. 9 shows that the DRL agent approaches similar mean episodic rewards to the maximum produced by the genetic-algorithm approach while requiring 10–100 times fewer environment evaluations. This can be attributed to the fact that the GA-driven optimizer does not use information about the environment dynamics outside of the reward associated with a specific action sequence, whereas the DRL approaches by definition operate on and learn the relationships between observed states, actions, and rewards.

To demonstrate the relative merits of DRL-based policies for general operations, the genetic-algorithm scheduler, heuristic policy agent, and best-performing PPO and S-PPO approaches were evaluated on 100 initializations of both the demonstration and training environments for both scenarios; the resulting mean reward and 1-σ bounds are listed in Table 4. On the demonstration environment used to construct the point solution produced by the GA, DRL-based approaches produce comparable mean rewards, falling 2–5% short on the LEO attitude management environment and actually exceeding the performance of the GA on the stationkeeping environment. When considering environments with randomly sampled initial conditions, the DRL approaches outperform the point solution produced by the GA in all circumstances. Importantly, both the default PPO implementation and the S-PPO extension outperform the greedy heuristic agent on both tasks, demonstrating both the increased performance possible with the adaptation of DRL versus hand-tuned policies and the benefit of combining correct-by-construction approaches with DRL techniques via shielding.

C. Sensitivity to Environment Parameters

The viability of this work hinges on the ability to train data-intensive DRL agents in simulation before applying them on board. A natural shortcoming of this approach is the fact that simulations may not reflect the exact environment in which an agent might be deployed, resulting in degraded behavior; this phenomenon is described as the "simulation gap" in deep learning literature. Modeling errors are classified as either parametric errors (wherein dynamics are modeled correctly but constants that govern those dynamics are mismatched) or systemic errors (wherein additional dynamics are not included in the training model). During nominal operations, mission analysts are typically able to account for systemic errors to an extremely high degree of precision. Prior work in astrodynamics problems has shown that perturbation methods, which assume small variations from a prescribed dominant dynamical regime, work very well for spacecraft trajectory design and navigation problems. For these reasons, it is desirable to evaluate the robustness of each agent architecture to parametric uncertainties, such as mismodeled spacecraft inertias or environmental perturbation strengths.

To demonstrate the empirical robustness of trained DRL algorithms in the reference scenarios described in Sec. IV, agents were evaluated against environments with parametric differences in their dynamics from the training environments; varied parameters and their ranges are listed in Table 5. Once again, the best-performing agents analyzed in Sec. V.B are used as benchmark agents for each approach. Each agent is run in the demonstration initial condition for each environment while taking three samples at each parameter combination; to evaluate mean performance, a Gaussian process regressor was fit to these samples to predict the mean episodic reward of each algorithm, on each environment, at each set of varied parameters.

The resulting reward contours are shown in Figs. 12 and 13. The LEO attitude management task shows a wide domain near the training condition in which the DRL-derived policies provide good performance, with degrading performance as power consumption

Table 5 Parameters varied in stationkeeping and attitude management problems

Stationkeeping parameter	Range	Attitude management parameter	Range
J_2	$0.1 \times J_2 - 100 \times J_2$	m	200–400
$[Q]$	$0.1 \times [Q] - 100 \times [Q]$	P_{out}	3–7 W

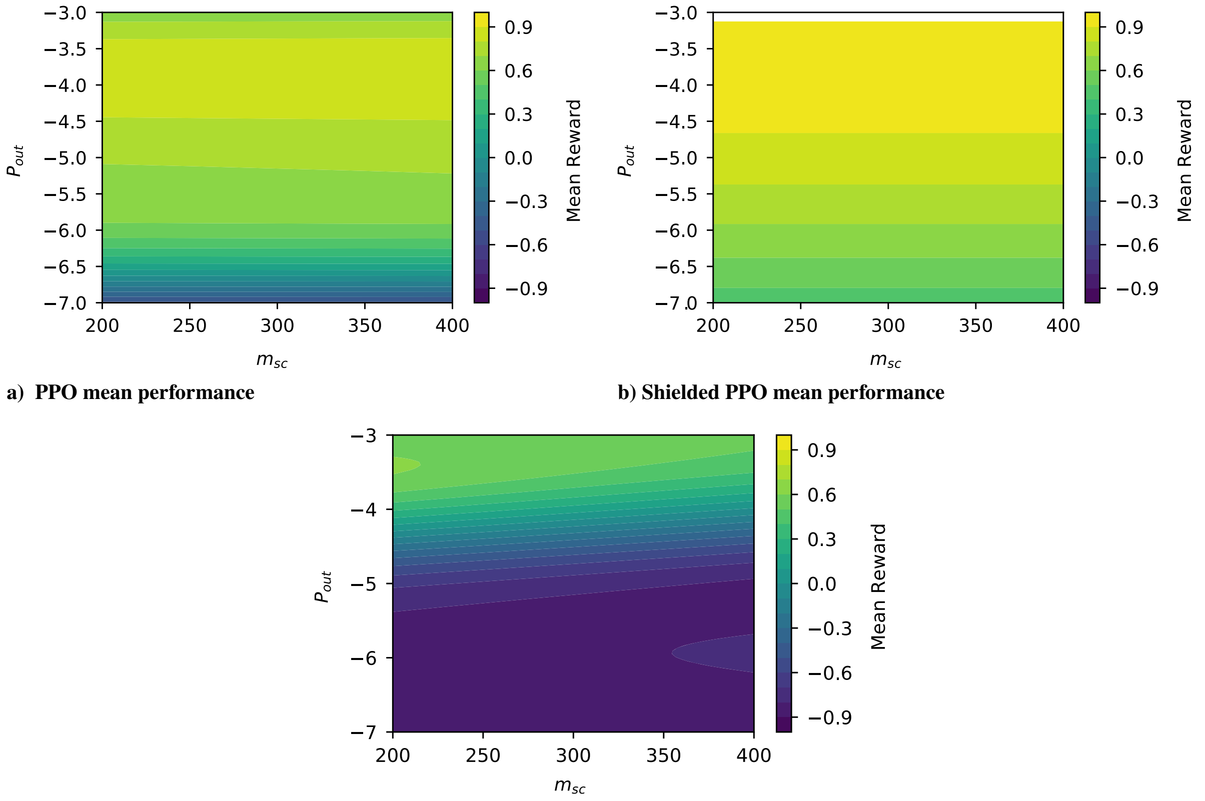
increases and marginal differences in performance as the spacecraft mass, and therefore inertia is varied. On the other hand, the GA-optimized timeline provides decent performance at the specific combination of mass and power consumption in the optimization environment but degrades rapidly as either parameter is varied away from the reference condition. In the stationkeeping task, the performance of the DRL-based agent tends to improve as both the J_2 parameter and magnitude of optimization noise are shrunk, whereas the shielded PPO implementation shows broader areas of high performance and a higher overall floor on performance with parametric variation. In a reflection of the partially observable nature of this environment, the GA-optimized timeline does substantially well across a range of observation noise variables (because the GA optimizer does not take into account observations when identifying action sequences).

In addition, these plots can also be considered as a reflection of the challenges presented by each environment; for example, small increases in power consumption in the LEO attitude management problem rapidly cause agents to fail. This is ultimately a result of the balanced power generation and consumption models used in the environment; the agent can generate, at most, 20 W of power in the sun-pointing mode. If an eclipse lasts 30% of an orbit (a common figure for the LEO environment during unfavorable beta angles), the agent can rapidly burn through its power reserve even before accounting for increases in power consumption; on the other hand, changes in mass, and therefore inertia (and therefore the settling time and accuracy of pointing modes), have a more modest effect on overall system performance.

D. Intrinsic Dimension Survey

Finally, it is desirable to understand where the spacecraft operations procedure problem is situated with other common deep learning problems, such as image classification or game playing. The intrinsic dimension approach described in Sec. II.C was applied to both the trained stationkeeping agent and the trained attitude manager to evaluate the intrinsic dimension of this solution approach. In addition, the stationkeep environment was evaluated with multiple observation models, ranging from full observations of the agent's estimated and reference state, to observations of the estimated error vector and covariance, to simply the norm of the error vector and covariance (referred to as "full," "error vector," and "LDR" observation models, respectively), reflecting increasing application of the LDF hypothesis described in Sec. III.A.1. The resulting plots of reward vs network dimensionality are shown in Fig. 14, with shaded regions around each evaluated reward line representing $\pm 1\sigma$ bounds for each approach over three random seeds and 100 evaluations per seed. Table 6 presents the intrinsic dimension of several common reference problems and datasets in the broader machine learning community alongside these benchmark values.

These results demonstrate several notable findings. First, it is apparent that the reference problems presented herein are comparable in complexity to other classic deep learning and deep reinforcement learning benchmarks, falling between the image classification task CIFAR-10 and the deep reinforcement learning task Atari Pong in terms of intrinsic dimension. Second, these results suggest that acceptable performance can be obtained with substantially smaller neural networks than was originally used for training: a feature that is extremely important in the context of

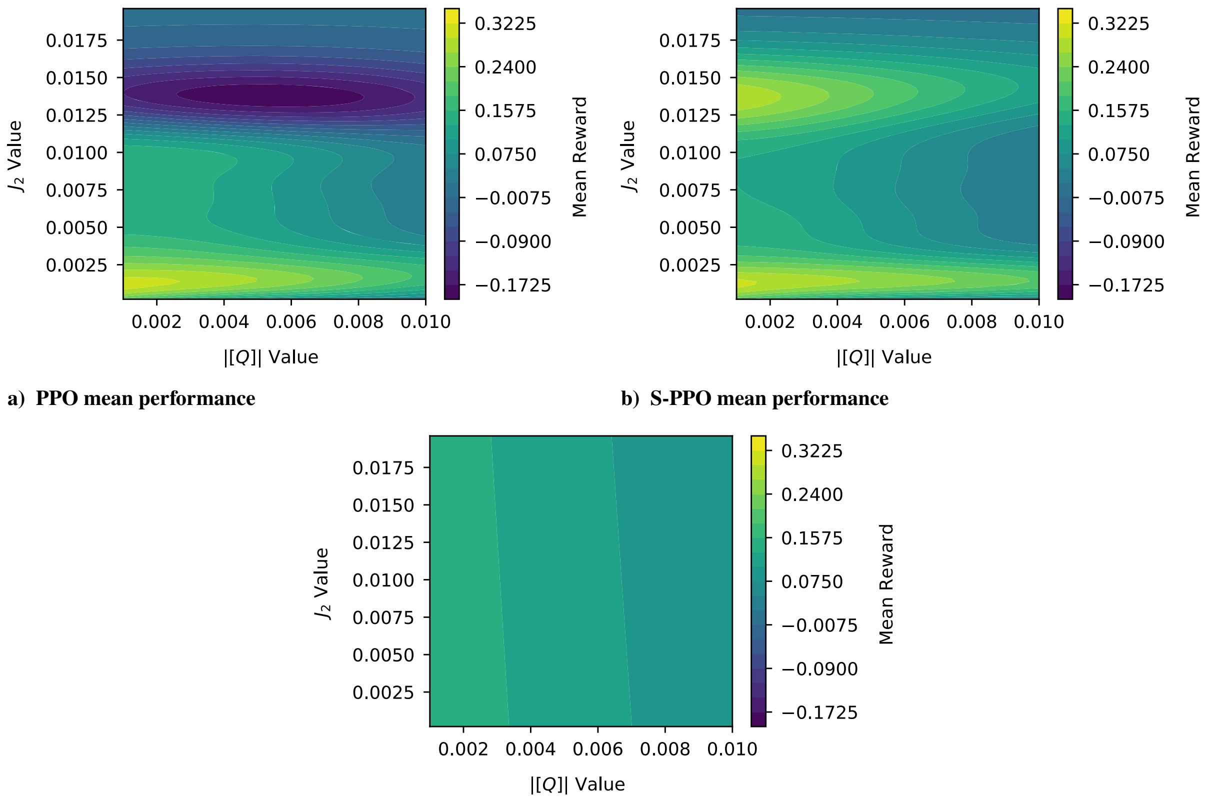


a) PPO mean performance

b) Shielded PPO mean performance

c) GA-Optimized timeline mean performance

Fig. 12 Comparison of GA, PPO, and Shielded PPO approaches versus parametric variation on the LEO management task.

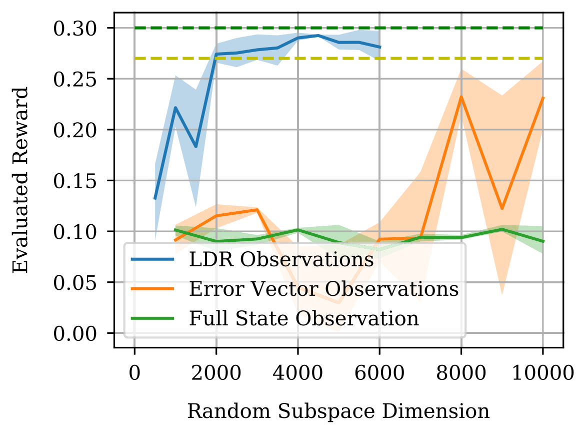


a) PPO mean performance

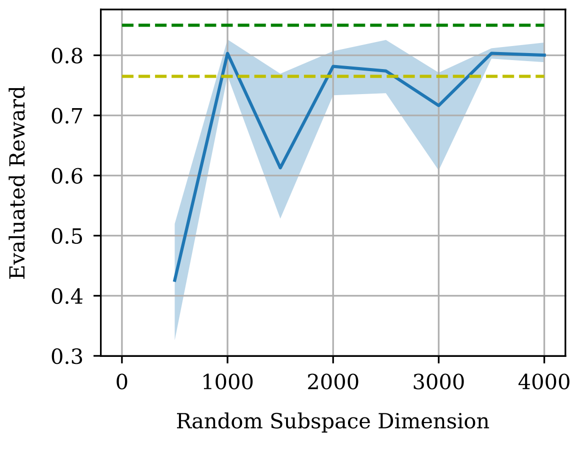
b) S-PPO mean performance

c) GA-Optimized timeline mean performance

Fig. 13 Comparison of GA-optimized timeline and PPO in stationkeep environment with demonstration initial conditions showing performance variation with J_2 and estimation noise changes in environment.



a) Stationkeep



b) LEO attitude guidance

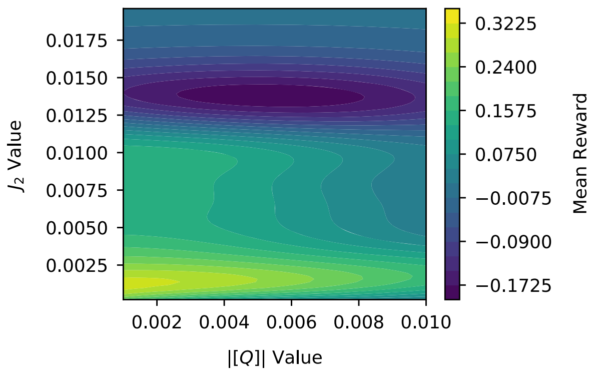
Fig. 14 Evaluated performance vs intrinsic dimension for PPO on LEO attitude guidance environment. Green dashed lines represent benchmark performance, and yellow dashed lines represent 90% of benchmark performance.

Table 6 Intrinsic dimension of PPO and S-PPO solutions for stationkeep and LEO attitude mode versus other problems in machine learning

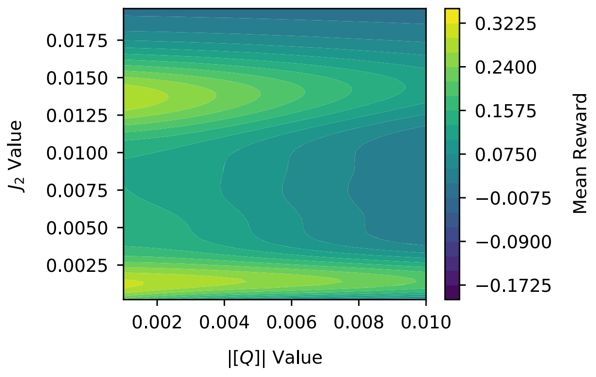
Problem	d_{int90}
Stationkeep: simple obs	2,000
Stationkeep: semi-obs	>10,000
Stationkeep: full obs	≫10,000
LEO attitude management	≈1,000–3,500
CIFAR-10	2,900–9,000
Humanoid	700
Atari Pong	6,000

obs = observations.

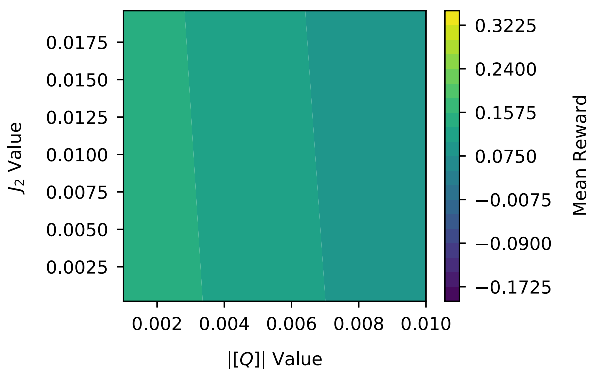
compute-constrained onboard decision making. Finally, Fig. 14a demonstrates the relative advantage of the LDF hypothesis in terms of training complexity for systems that resemble switched hybrid systems in practice, demonstrating that solutions to the “simplified” stationkeep environment can be obtained with smaller parameter sets than larger ones with no loss in performance. Despite the substantially increased problem complexity in terms of simulated hardware and software dynamics, the LEO attitude health management task requires similarly sized networks to the simpler Mars stationkeeping task, suggesting that substantial increases in dynamical complexity may not increase the complexity of a mode-switching task. Taken together, these results suggest that the spacecraft operations problem can be solved with reasonably sized neural networks that fall well within the current state of the art for



a) PPO mean performance

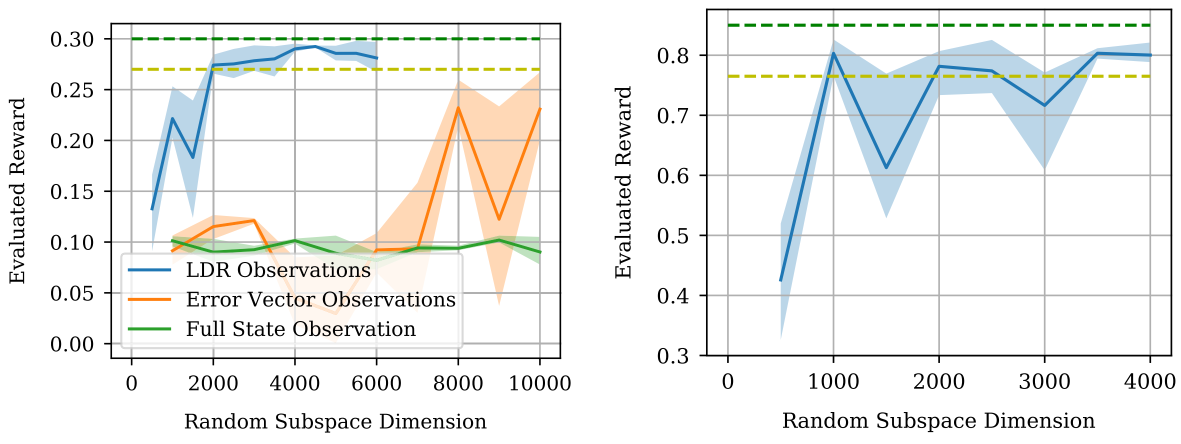


b) S-PPO mean performance



c) GA-Optimized timeline mean performance

Fig. 15 Comparison of GA-optimized timeline and PPO in stationkeep environment with demonstration initial conditions showing performance variation with J_2 and estimation noise changes in environment.



a) Stationkeep **b) LEO attitude guidance**

Fig. 16 Evaluated performance vs intrinsic dimension for PPO on LEO attitude guidance environment. Green dashed lines represent benchmark performance, and yellow dashed lines represent 90% of benchmark performance.

the field of deep learning, especially when prior knowledge is leveraged in problem construction.

VI. Conclusions

This work has established the viability of deep reinforcement learning for generating operations procedures for next-generation space mission autonomy. Mode-based operations design, wherein specific software and hardware status combinations are represented by higher-level modes, provides both a common entry point for modeling day-to-day spacecraft operations problems as a mode selection task that can be readily transformed into Markov decision processes. In addition, challenges inherent to the use of reinforcement learning for operations policy creation, such as safety and the use of existing information, have been addressed through the application of shielded reinforcement learning. In comparison with both heuristic and timeline-optimization approaches, DRL-driven procedures provide comparable or improved performance with respect to mission objective satisfaction while generalizing to a wider range of initial conditions and parametric uncertainties, providing additional robustness.

Acknowledgments

This work was funded in part by the National Defense Science and Engineering Graduate Fellowship. The authors would also like to thank Morteza Lahijanian of the University of Colorado Boulder for his contributions and insight in reactive synthesis.

References

[1] Alibay, F., Lazio, J. W., Kasper, J. C., and Neilsen, T., "Sun Radio Interferometer Space Experiment (SunRISE) Proposal: Status Update," *31st Annual AIAA/USU Conference on Small Satellites*, Utah State Univ. Digital Commons, Paper # SSC17-IX-04, 2017, https://digitalcommons.usu.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=3662&context=smallsat [accessed 17 Nov. 2021].

[2] "NASA Technology Taxonomy 2020," NASA TR, July 2015, 2020.

[3] "DARPA 2019 Strategic Framework," Defense Advanced Research Project Agency TR, Aug. 2019, https://www.darpa.mil/about-us/about-darpa [accessed 17 Nov. 2021].

[4] Rumford, T. E., "Demonstration of Autonomous Rendezvous Technology (DART) Project Summary," *Space Systems Technology and Operations*, Vol. 5088, edited by P. T. Shoemaker, Jr., International Soc. for Optics and Photonics, Bellingham, WA, 2003, pp. 10–19. https://doi.org/10.1117/12.498811

[5] Kubitschek, D. G., "Impactor Spacecraft Encounter Sequence Design for the Deep Impact Mission," *Jet Propulsion*, Georgia Tech Digital Repository, Paper # Gt-SSEC.C.3 #15, 2005, pp. 1–14, https://smartech.gatech.edu/handle/1853/8031 [accessed 17 Nov. 2021].

[6] Rabideau, G., et al., "Mission Operations of Earth Observing-1 with Onboard Autonomy," *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, 2006, pp. 7, 373. https://doi.org/10.1109/SMC-IT.2006.48

[7] Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castano, R., Davis, A., Mandl, D., Frye, S., Trout, B., and Shulman, S., "Using Autonomy Flight Software to Improve Science Return on Earth Observing One," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 4, April 2005, pp. 196–216. https://doi.org/10.2514/1.12923

[8] Chien, S. A., Tran, D., Rabideau, G., Schaffer, S. R., Mandl, D., and Frye, S., "Timeline-Based Space Operations Scheduling with External Constraints," *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, ICAPS, 2010, pp. 34–41.

[9] Eddy, D., and Kochenderfer, M., "Markov Decision Processes for Multi-Objective Satellite Task Planning," *2020 IEEE Aerospace Conference*, IEEE, New York, 2020, pp. 1–12. https://doi.org/10.1109/AERO47225.2020.9172258

[10] Spangelo, S., Cutler, J., Gilson, K., and Cohn, A., "Optimization-Based Scheduling for the Single-Satellite, Multi-Ground Station Communication Problem," *Computers and Operations Research*, Vol. 57, May 2015, pp. 1–0. https://doi.org/10.1016/j.cor.2014.11.004

[11] Foster, C., Hallam, H., and Mason, J., "Orbit Determination and Differential-Drag Control of Planet Labs Cubesat Constellations," *Advances in the Astronautical Sciences*, Vol. 156, 2016, pp. 645–657.

[12] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., and Tsing, R., "StarCraft II: A New Challenge for Reinforcement Learning," Preprint, submitted 16 Aug. 2017, https://arxiv.org/abs/1708.04782.

[13] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D., "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning," *Nature*, Vol. 575, No. 7782, 2019, pp. 350–354. https://doi.org/10.1038/s41586-019-1724-z

[14] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S., "Dota 2 with Large Scale Deep Reinforcement Learning," Preprint, submitted 13 Dec. 2019, https://arxiv.org/abs/1912.06680.

[15] Nageshrao, S., Tseng, H. E., and Filev, D. P., "Autonomous Highway Driving Using Deep Reinforcement Learning," Preprint, submitted 29 March 2019, https://arxiv.org/abs/1904.00035.

[16] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A., Yogamani, S., and Perez, P., "Deep Reinforcement Learning for Autonomous

Driving: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, advance online publication, 9 Feb. 2021. https://doi.org/10.1109/TITS.2021.3054625

[17] Evans, R. D., and Gao, J. D., "DeepMind AI Reduces Google Data Centre Cooling Bill by 40%," *DeepMind Blog* (online database), 20 July 2016, https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40.

[18] Dulac-Arnold, G., Mankowitz, D. J., and Hester, T., "Challenges of Real-World Reinforcement Learning," Preprint, submitted 29 April 2019, https://arxiv.org/abs/1904.12901.

[19] Harris, A., Teil, T., and Schaub, H., "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *29th AAS/AIAA Space Flight Mechanics Meeting*, AAS Paper 19-447, HI, 2019, pp. 1–19.

[20] Harris, A., and Schaub, H., "Towards Reinforcement Learning Techniques for Spacecraft Autonomy," *AAS Guidance, Navigation and Control Meeting*, Vol. 164, Univelt Publishers, San Diego, CA, 2018, pp. 467–476.

[21] Cianciolo, A. D., Maddock, R. W., Prince, J. L., Bowes, A., Powell, R. W., White, J. P., Tolson, R., Shaughnessy, O., and Carrelli, D., "Autonomous Aerobraking Development Software: Phase 2 Summary," AAS Paper 13-736, 2013, pp. 1–0.

[22] Gaudet, B., and Furfaro, R., "Robust Spacecraft Hovering Near Small Bodies in Environments with Unknown Dynamics Using Reinforcement Learning," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2012-5072, 2012. https://doi.org/10.2514/6.2012-5072

[23] Furfaro, R., Bloise, I., Orlandelli, M., Di Lizia, P., Topputo, F., and Linares, R., "Deep Learning for Autonomous Lunar Landing," *Proceedings of the 2018 AAS/AIAA Astrodynamics Specialist Conference*, Vol. 167, Univelt., Snowbird, UT, 2018, pp. 3285–3306.

[24] Hovell, K., and Ulrich, S., "Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance," *Journal of Spacecraft and Rockets*, Vol. 58, No. 2, 2021, pp. 254–264. https://doi.org/10.2514/1.A34838

[25] Uhlig, T., Sellmaier, F., and Schmidhuber, M., *Spacecraft Operations*, Springer, New York, 2015, pp. 167–211. https://doi.org/10.1007/978-3-7091-1803-0

[26] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," Preprint, submitted 20 July 2017, https://arxiv.org/abs/1707.06347.

[27] Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *4th International Conference on Learning Representations, ICLR 2016—Conference Track Proceedings*, 2016, pp. 1–14, https://arxiv.org/abs/1506.02438 [accessed 17 Nov. 2021].

[28] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U., "Safe Reinforcement Learning via Shielding," Preprint, submitted 29 Aug. 2017, https://arxiv.org/abs/1708.08611.

[29] Li, X., Warier, R. R., Sanyal, A. K., and Qiao, D., "Trajectory Tracking Near Small Bodies Using Only Attitude Control," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 1, 2018, pp. 109–122. https://doi.org/10.2514/1.G003653

[30] Labonde, C. J., Flynn, S., Muszynski, M., Ryan, S., McCabe, D., and Pilinski, E., "Ground Autonomy for an Aging Spacecraft," *15th International Conference on Space Operations*, AIAA Paper 2018-2619, June 2018, pp. 1–10. https://doi.org/10.2514/6.2018-2619

[31] Julian, K. D., and Kochenderfer, M. J., "Autonomous Distributed Wildfire Surveillance Using Deep Reinforcement Learning," *2018 AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2018-1589, 2018. https://doi.org/10.2514/6.2018-1589

[32] Sample, E., Ahmed, N., and Campbell, M., "An Experimental Evaluation of Bayesian Soft Human Sensor Fusion in Robotic Systems," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2012-4542, Aug. 2012. https://doi.org/10.2514/6.2012-4542

[33] Branicky, M. S., "Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 4, 1998, pp. 475–482. https://doi.org/10.1109/9.664150

[34] Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., and Simaitis, A., "PRISM-Games: A Model Checker for Stochastic Multi-Player Games," Vol. 7795, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, New York, 2013, pp. 185–191. https://doi.org/10.1007/978-3-642-36742-7_13

[35] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 3rd ed., AIAA, Reston, VA, 2014, pp. 778–782. https://doi.org/10.2514/4.102400

[36] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D., "Deep Reinforcement Learning that Matters," Preprint, submitted 19 Sept. 2017, https://arxiv.org/abs/1709.06560.

O. Abdelkhalik
Associate Editor