JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 21, No. 10, October 2024

Autonomous Small Body Science Operations Using Reinforcement Learning

Adam Herrmann* and Hanspeter Schaub†
University of Colorado, Boulder, Colorado 80303

https://doi.org/10.2514/1.I011376

This work studies the feasibility of using reinforcement learning for small body science operations subject to resource constraints. Two mission scenarios are considered. In the first scenario, a spacecraft autonomously maneuvers between waypoints about a small body while performing science activities, such as mapping and imaging, and periodically downlinking data and managing on-board resources like battery charge, data buffer storage, and fuel usage. In the second scenario, the spacecraft periodically performs navigation updates to improve its state estimate, ensuring that the collected science is within the specified requirements. A Markov decision process formulation of the mission scenarios is formulated, and reinforcement learning is applied to solve the problem. A range of noisy observation types are tested, demonstrating that a fully observable formulation of the problem trained on direct observations of the state is robust to noisy measurements or a filtered state estimate. A decision-making agent is then trained to manage the state estimate by choosing when to take measurements, demonstrating that near-equivalent policies, in comparison to nominal problem formulation, can be trained with an optional navigation update. Finally, a demonstration is performed in which a ground station outage is simulated. The decision-making agent is shown to be robust to this outage, rapidly adjusting its plan to continue nominal operations.

Nomenclature

Problem Formulation

j	=	map index
k	=	map point index
M	=	set of map points
$M_{j,k}$	=	kth map point of map j
r_w	=	radius of waypoint w
T	=	set of surface imaging targets
W	=	set of waypoints
ΔV	=	total change in velocity
θ	=	azimuth angle of waypoint w
λ	=	set of local solar times of maps
ψ	=	polar angle of waypoint w

Dynamics

A	=	semimajor axis of asteroid orbit
A_{sc}	=	surface area of spacecraft
$\boldsymbol{a}_{\text{srp}}$	=	acceleration due to solar radiation pressure
$\hat{\boldsymbol{d}}$	=	direction of the sun
E	=	eccentricity of asteroid orbit
$\dot{F}$	=	first time derivative of true anomaly
$\ddot{F}$	=	second time derivative of true anomaly
$[K_1]$	=	proportional gain matrix
$[K_2]$	=	derivative gain matrix
M_{sc}	=	mass of spacecraft
${}^{\mathcal{O}}\boldsymbol{r}$	=	position of spacecraft
$\mathcal{O}$	=	sun–asteroid Hill frame
${}^{\mathcal{O}}\dot{\boldsymbol{r}}$	=	velocity of spacecraft
P_0	=	solar flux at 1 AU
U_g	=	gravitational potential of asteroid
U_s	=	gravitational potential of sun
$\boldsymbol{u}$	=	control acceleration
$\mathcal{E}$	=	asteroid body frame
μ_{sun}	=	gravitational parameter of the sun
μ_{ast}	=	gravitational parameter of the asteroid
ρ	=	surface reflectivity

State Estimation

H_i	=	measurement Jacobian at time t_i
$\boldsymbol{h}_i$	=	measurement function at time t_i
K_i	=	Kalman gain at time t_i
P_i	=	state error covariance at time t_i
Q	=	process noise covariance
R	=	measurement noise covariance
$\boldsymbol{r}_i$	=	measurement residual vector at time t_i
$\boldsymbol{y}_i$	=	measurement vector at time t_i
$\hat{\boldsymbol{x}}_i$	=	state estimate at time t_i
$\Phi(t, t_{i-1})$	=	state transition matrix

Markov Decision Processes

$\mathcal{A}$	=	action space
a_i	=	action at step i
$G(s, a)$	=	generative transition function
$Q^{\pi}(s, a)$	=	state-action value function following policy π
$\mathcal{R}(s, a)$	=	reward function
r_i	=	reward at step i
$\mathcal{S}$	=	state space
s_i	=	state at step i
$V^{\pi}(s)$	=	value function following policy π
$\pi(a\|s)$	=	policy
$\mathcal{T}(s'\|s, a)$	=	transition probability

Received 5 October 2023; accepted for publication 30 May 2024; published online 2 July 2024.

*Graduate Research Assistant, Ann & H. J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive, AERO 446; adam.herrmann@colorado.edu. Member AIAA.

†Professor, Schaden Leadership Chair, Ann & H. J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive, AERO 446; hanspeter.schaub@colorado.edu. Associate Fellow AIAA.

I. Introduction

MISSIONS to small bodies such as asteroids and comets present several challenges for planning and scheduling, which is the process by which the sequence of activities a spacecraft must perform to fulfill its mission objectives is computed. Epistemic uncertainty regarding the operational environment, aleatory uncertainty regarding the state of the environment, and

the round-trip light-time delay are three challenges that planning and scheduling capabilities for small body operations must address. Epistemic uncertainty is uncertainty regarding the environment that can be reduced with additional information. For example, the shape of a small body is not known with great fidelity until the spacecraft arrives at the body and constructs a shape model. Aleatory uncertainty is uncertainty regarding the state of the environment that cannot be reduced with additional information. For example, the state of the spacecraft is not known with absolute certainty due to inherent noise in measurements, which can be used to estimate the state of the spacecraft. The round-trip light-time delay is the amount of time it takes for a signal to leave the Earth, reach a spacecraft, and return to Earth. In deep space, the round-trip light-time delay is tens of minutes, depending on the distance from the Earth to the spacecraft in question. Spacecraft cannot rely on input from Earth for unexpected events that occur during this time, such as unexpected science opportunities. To make matters even more challenging, a NASA Inspector General report found that the deep space network (DSN), which is responsible for communication and navigation for 60 NASA and international space missions, is currently oversubscribed and will remain oversubscribed as NASA's Artemis Program, Perseverance Rover, and James Webb Space Telescope compete for DSN access [1]. Therefore, it is desirable to have spacecraft autonomously continue nominal operations if an unexpected outage occurs.

To address these challenges, further development of advanced autonomous capabilities that can handle uncertainty and operate with minimal input from the ground is required. Much work has been done to achieve this goal for several missions that have flown. In particular, on-board planning and scheduling capabilities have advanced greatly in the past few decades. The first example of on-board planning and scheduling is the Remote Agent, which was deployed on board NASA's Deep Space One spacecraft to demonstrate goal-based commanding, on-board planning, robust execution, and fault protection [2–4]. Other examples include the ASPEN [5] and CASPER [6] systems, which have flight heritage on a number of missions, demonstrating autonomous ground-based planning and on-board replanning to enable opportunistic science. The Earth-Observing 1 [7,8] and IPEX [9] missions are two examples. Another example of such a tool is MEXEC, an on-board planning and execution tool originally developed by NASA for the Europa Clipper mission [10]. An on-board scheduler is also planned for the Perseverance Rover to adjust activities to account for variations in resources or task execution [11]. The development of such tools demonstrates the need for on-board planning and scheduling capabilities for future missions. This work investigates fully autonomous planning and scheduling capabilities, which do not require ground-based plans that are generated beforehand and continually monitored by an on-board scheduler or executive. Autonomous decision-making agents are instead trained on the ground and uplinked to the spacecraft before the initiation of proximity operations.

Recently, deep reinforcement learning (DRL) has emerged as a promising class of solution methods and problem formulations for a variety of spacecraft decision-making problems. The goal of reinforcement learning is to solve for a policy that maps states to actions to maximize a numerical reward function [12]. In contrast to optimization-based formulations and solution methods like genetic algorithms [13] or mixed integer programs [14–19], reinforcement learning is interested in solving for a policy that generalizes across the state space such that the optimal action is known for every state. Trained policies are executed in a closed-loop manner, responding to the observed states of the environment. The domain randomization in training also allows reinforcement learning to cope with model uncertainty [20]. This is extremely advantageous for small body exploration where epistemic and aleatory uncertainty are present. Several authors have investigated using reinforcement learning for a variety of small body operation problems. Chan and Agha-Mohammadi formulate a small body mapping problem as a partially observable Markov decision process (POMDP), where the objective is to improve the quality of a map assembled using stereophotoclinometry (SPC) [21]. The authors apply the REINFORCE (Reward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility) algorithm to generate policies over the belief space, showing that the trained policies perform better than heuristic policies. Piccinin et al. formulate a global mapping problem for SPC as a Markov decision process (MDP) [22]. In this problem, the spacecraft enters an orbit about the body, and the decision-making agent determines whether or not to take an image. The authors compare deep Q-learning and neural fitted Q (NFQ) learning, showing that these two algorithms outperform random and heuristic policies. Takahashi and Scheeres formulate a surface mapping problem about a small body as an MDP where the output of the policy is a change in elevation and a transfer time, which is fed into a two-point boundary value solver that generates a fuel-optimal control solution [23]. An extended Kalman filter (EKF) is implemented to provide a state estimate to the two-point boundary value problem solver and decision-making agent. The authors apply proximal policy optimization to train decision-making agents, showing how autonomous guidance, navigation, and control (GNC) technologies may be combined with reinforcement learning for surface imaging.

The small body operations work presented thus far contains some elements of planning and scheduling (i.e., science collection while managing resource constraints). However, much of the literature investigating reinforcement learning for spacecraft decision-making problems falls within the domain of the GNC subsystem. GNC-related reinforcement learning (RL) work has been used for planetary landing [24–27], small body proximity operations [28–31], and spacecraft rendezvous, proximity operations, and docking (RPOD) [32–37]. Many of these works focus on using reinforcement learning algorithms that can adapt to off-nominal conditions on the fly, such as thruster failures, and still successfully complete the mission and/or use reinforcement learning to provide end-to-end solutions for some or all aspects of the guidance, navigation, and control subsystem. A popular approach is the use of recurrent policies trained with reinforcement learning algorithms that maintain an internal belief state to handle the partial observability of the associated problem. The authors of Refs. [25,27–29,31] integrate the full guidance, navigation, and control subsystem into a reinforcement learning-only framework (i.e., they map observations to controls) using this approach. The authors of Refs. [24,26,34–36] assume full observability over the state, while the authors of Ref. [23] assume that a state estimate from a navigation subsystem is provided. While there are many interesting GNC problems that reinforcement learning can solve, this work focuses on planning and scheduling. The aforementioned works may incorporate some aspects of planning and scheduling (e.g., maneuver sequencing, maneuvering for science purposes, etc.), but are not planning and scheduling problems due to the lack of focus on science objectives and the lack of resource constraints and management. This work treats the GNC subsystem as an input into the problem formulation as opposed to being the primary focus of the problem formulation.

Past work has demonstrated how various proximity operations problems about small bodies may be formulated as (PO)MDPs and solved with reinforcement learning algorithms. However, these problem formulations typically do not account for resource constraints such as on-board storage and power. Because on-board storage is not modeled, communication with the ground is typically left out of the problem formulations as well. Attitude guidance and control and its relation to the aforementioned resource constraints, particularly power, is not considered. The addition of these aspects of the problem is important because they have strong implications for the learned policies. Furthermore, while many of these problem formulations add partial observability, the impact of partial observability on performance, particularly the quality of scientific observations, is not explored. It should also not be assumed that the navigation architecture supports continuous measurement updates. Instead, one should assume that the measurement update either requires communication with the ground or dedicated imaging

for optical navigation, which means that the estimation error covariance should grow between navigation updates. To address these gaps in the literature, this work makes the following novel contributions to the field of planning and scheduling for small body operations:

1) The formulation of a novel small body science operations problem with power consumption and generation, on-board data storage, data downlink, attitude guidance and control, and translational guidance and control.

2) The training of RL policies that are robust to noisy or filtered measurements of the state.

3) The addition of navigation updates to the problem formulation and the training of RL policies that can manage the state estimate of the spacecraft to ensure science is collected within requirements.

This paper is organized as follows: The small body science operations problem with the aforementioned features is first formulated and presented. An overview of past small body missions and their associated mission phases is presented in Sec. II.A to ground the problem formulation in a degree of realism. The general formulation of the problem is described in Sec. II.B. The dynamics, translational guidance and control solution, and relative navigation solution are then described in detail in Secs. II.C–II.E. The MDP formulation of the problem is presented in Sec. II.F, and the high-fidelity astrodynamics simulation used to represent the generative transition function of the MDP is given in Sec. II.K. Then, the methods used to solve the problem are presented in Sec. III. More precisely, proximal policy optimization (PPO), the selected solution method, is presented alongside the training pipeline. Finally, the results are presented and discussed in Sec. IV. Conclusions are drawn and recommendations for future work are provided in Sec. V.

II. Problem Formulation

A. Small Body Proximity Operations Phases

Small body proximity operations may be described in several phases, each with its own objectives and data products. Each of these phases may be thought of as separate operations problems where the science and data products from one phase are utilized in the next. Past work in spacecraft autonomy for small body exploration has defined these mission phases in various ways [38,39]. This section will provide its own summary for clarity. Because these phases are defined using concepts of operations from several different missions with target asteroids of different sizes and shapes, the boundaries between them are fluid. Ashman et al. provide a detailed summary of the Rosetta operations phases [40], and Lauretta et al. provide a summary of the OSIRIS-REx operations phases [41]. The phases this work defines are A) approach, B) characterization, C) science operations, and D) landing. The characteristics of each phase and the corresponding mission phases for OSIRIS-REx and Rosetta are summarized in the rows of Table 1.

The first phase is the approach phase. During the approach phase, the spacecraft performs trajectory correction maneuvers to rendezvous with the asteroid. During this phase, a low-fidelity shape model is constructed, a refined estimate of the spin state is gathered, and the ephemeris of the body is improved [39]. This phase is analogous to Rosetta's far approach trajectory (FAT) phase and OSIRIS-REx's approach phase. The second phase is typically a characterization phase. During this phase, the spacecraft enters the body's sphere of influence, performing hyperbolic flybys about the body. The shape model is improved, preliminary science data is gathered, and an estimate of the body's gravitational parameter is generated. This phase is analogous to Rosetta's close approach trajectory (CAT) and characterization phase and OSIRIS-REx's preliminary survey phase. Finally, the spacecraft enters the science operations phase, which may be decomposed further into more specific operations phases depending on the mission. This is when the detailed science campaign about the body begins, which is highly dependent on the mission. During this phase, the spacecraft either enters into a stable orbit about the body, transfers between or holds a position at an inertial waypoint(s), or performs low-altitude fly-bys about the body. This also marks the transition from centroid-based optical navigation to feature-tracking optical navigation due to the spacecraft's proximity to the body. This phase typically includes some sort of mapping to build temperature maps, reflectance maps, and identify candidate landing sites. In the case of Rosetta, the global mapping and close observation phases fall into this category. In the case of OSIRIS-REx, the detailed survey, orbital B, and reconnaissance phases fall into this category. The final phase of proximity operations is often some sort of landing phase. In the case of Rosetta, this includes the landing of the Philae lander, and in the case of OSIRIS-REx, this includes the touch-and-go phase.

This work focuses on formulating and solving a science operations phase for the exploration of a small near-Earth asteroid. It is assumed that the spacecraft has a GNC solution for state estimation and maneuvering relative to the body. It is also assumed that the gravity about the body is weak enough to allow for waypoint-to-waypoint locomotion of the spacecraft without expending an inordinate amount of fuel or time to complete the mission. Furthermore, because the approach and characterization phases have passed, low epistemic uncertainty (i.e., shape model and gravity model) is assumed. As a result, the primary objective of this operational phase is the collection of scientific data. This work focuses on two science objectives: spectroscopy mapping and high-resolution imagery of candidate landing sites.

B. Problem Description

In this work, a spacecraft operates in the vicinity of a small body to maximize the amount of spectroscopy map and surface imaging targets collected and downlinked while avoiding collision with the body and managing on-board resources such as power, on-board data storage, and ΔV. The spacecraft maneuvers between a set of waypoints W defined in the sun–asteroid Hill frame. The set of 10 surface imaging targets is referred to as T, and the set of all spectroscopy maps is referred to as M. The spacecraft has an

Table 1 Small body mission phases

	Approach	Characterization	Science operations	Landing
		Families of proximity operations phases		
Data products	Body ephemeris, spin state, preliminary shape model	Preliminary science, gravity estimate, improved shape model	Science maps, landing site images, detailed shape model	Surface science
Optical navigation	Centroid based	Centroid based	Feature tracking	Feature tracking
Dynamics	Approach trajectory	Hyperbolic fly-bys	Orbital motion, inertial waypoints, low-altitude fly-bys	Descent and ascent trajectory
		Associated phases for Rosetta and OSIRIS-REx		
Rosetta	Far approach trajectory	Close approach trajectory and characterization	Global mapping, close observation	Philae
OSIRIS-REx	Approach	Preliminary survey	Detailed survey, orbital B, reconnaissance	Touch-and-go

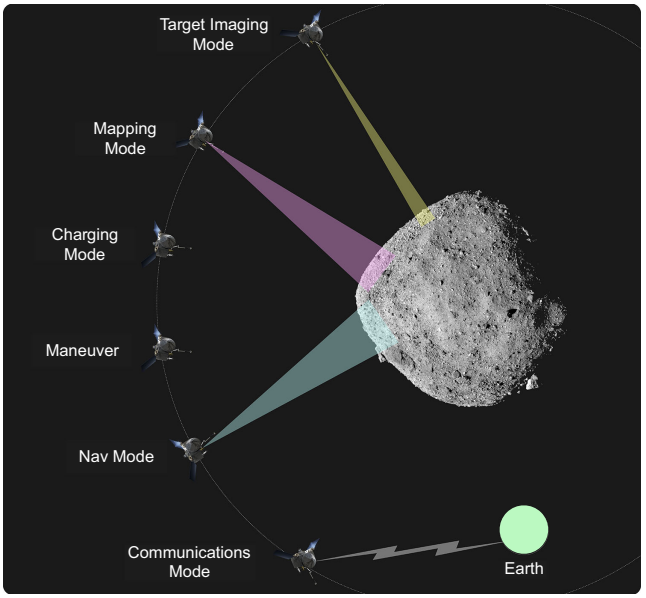


Fig. 1 Concept of operations for the small body science operations problem.

attitude control system, a power system, and a data system on board. The spacecraft completes its mission by entering into a series of operational modes. Each operational mode creates high-level abstractions of the low-level behavior of the spacecraft. The low-level behavior of the spacecraft is dictated by the attitude reference, the current waypoint reference of the guidance and control subsystem, whether the navigation system is collecting measurements, and the ON/OFF states of various instruments and transmitters. A concept of operations, including each of the operational modes, is included in Fig. 1. The problem is simulated using the Basilisk astrodynamics software architecture,[‡] which provides high-fidelity astrodynamics simulation capabilities for dynamics, resources, and flight software modeling [42].

The spacecraft has various resources on board, such as power, data storage capacity, and limited fuel. In the charging mode, the spacecraft holds its position at the targeted waypoint and uses its attitude control system to point its solar panels at the sun, which charges its batteries. Both the instrument and transmitter are turned off in this mode. While this mode is a dedicated charging mode, the spacecraft may end up charging its batteries in the other operational modes as well, depending on the orientation of the spacecraft's solar panels with respect to the sun. In the mapping mode, the attitude control system points the mapping instrument in the direction of the small body, and the spacecraft turns on its mapping instrument to begin collecting map data. The map data is stored within the data buffer but is only counted as collected if the data meets the specified requirements. Similarly, in the imaging mode, the spacecraft points its imaging instrument at the nearest surface imaging target and collects an image if the target meets the requirements. In the downlink mode, the spacecraft turns off all instruments and points an antenna towards the Earth, sending data to the DSN and emptying its data buffer if the temporal and physical access constraints of the DSN are met.

The remaining operational modes deal with navigation updates and waypoint maneuvers. In the navigation update mode, the spacecraft points its imager at the body and begins collecting simulated measurements of the state to improve its estimate of the state. This is done with an EKF, which is configured to operate with or without measurements, with the latter configuration resulting in a decrease of the quality of the state estimate and an increase in the state error covariance. The last set of operational modes deals with waypoint maneuvers. The waypoints are defined in the sun–asteroid Hill frame, $\{\mathcal{O}: \hat{\boldsymbol{o}}_1, \hat{\boldsymbol{o}}_2, \hat{\boldsymbol{o}}_3\}$. The sun–asteroid Hill frame is centered at

[‡]Data available online at https://hanspeterschaub.info/basilisk/index.html.

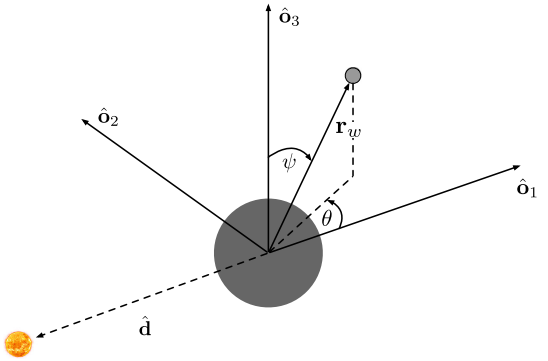


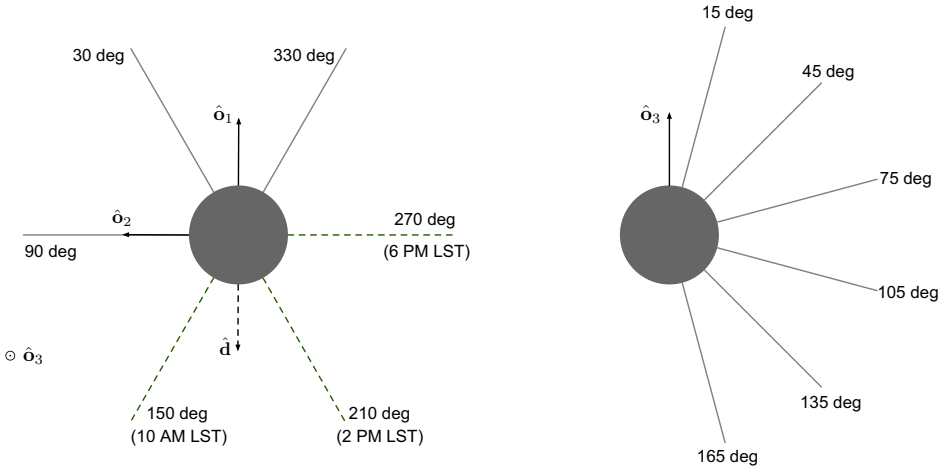
Fig. 2 Sun–asteroid Hill frame.

the origin of the small body. The first component, $\hat{\boldsymbol{o}}_1$, is in the radial direction from the sun to the small body. The third component, $\hat{\boldsymbol{o}}_3$, is in the orbit normal direction. Finally, the second component completes the right-handed coordinate system. A diagram of the Hill frame is provided in Fig. 2. The $\hat{\boldsymbol{d}}$ vector denotes the direction of the sun. Each waypoint $w \in W$ is defined with a set of spherical coordinates $w = (r_w, \theta, \psi)$. The waypoints are evenly distributed across six azimuth and polar angles, as shown in Fig. 3, numbering 36 in total. The waypoint radii r_w are set to three body radii, i.e., $r_w = 3R_{\text{ast}}$. In waypoint maneuvering mode, the spacecraft points its solar panels at the sun to remain power positive, and control forces are computed to take the spacecraft from its current position to the desired position. The spacecraft can only target neighboring waypoints, which results in eight different waypoint maneuvering modes at each decision interval. Thrusters are not simulated, and it is assumed that the requested control forces can be fulfilled while pointing the solar panels at the sun.

For the spectroscopy mapping, there are $j = 3$ separate maps that must be collected, one at each of the following local solar times: $\lambda = \{6{:}\,00\ \text{PM}, 2{:}\,00\ \text{PM}, 10{:}\,00\ \text{AM}\}$. Each map is represented by a set of $k = 500$ points, M_j, evenly distributed on the surface of the body, where j is the map number. These points are generated using a Fibonacci lattice to ensure equal coverage of the body. In Fig. 3a, the dotted lines represent the local solar times where spectroscopy mapping may take place. In Fig. 3b, the various polar angles of the waypoints are displayed. Mapping may occur at any of these polar angles if the spacecraft is at an azimuth within one degree of the azimuth associated with the local solar time for a particular map. During the Detailed Survey Phase, OSIRIS-REx had seven equatorial stations, each at its own local solar time. At each equatorial station, data was collected to construct maps of the surface of Bennu, such as global mineral and chemical maps. Furthermore, the mapping took place at a relatively narrow range of polar angles. This work selects three specific local solar times for mapping and removes the narrow polar angle requirement to maintain minimal simulation time. A single map is collected at each of these three local solar times. This work assumes that the small body is spherical. This work could also be applied to oblate spheroids, but more complicated shapes would require more sophisticated mapping point generation and representation. This work also assumes that the rotation pole of the asteroid is aligned with the orbit normal of the asteroid.

C. Dynamics

The position and velocity of the spacecraft relative to the small body, ${}^{\mathcal{O}}\boldsymbol{r}$ and ${}^{\mathcal{O}}\dot{\boldsymbol{r}}$, are expressed in the sun–asteroid Hill frame, which is a convenient coordinate frame for this problem due to the illumination requirements for mapping. The derivation of the relative dynamics may be found in work from Takahashi and Scheeres [23] and Scheeres [43]. The dynamics described in this section are utilized within the EKF and continuous feedback control law used for translational control about the body. The dynamics treat the gravity of the small body as a point mass and the gravity of the sun as a third-body perturbation. A cannonball solar radiation pressure (SRP) model is utilized to model SRP. The equations of motion for the spacecraft in



a) Azimuth angles for waypoints (θ) **b) Polar angles for waypoints (ψ)**

Fig. 3 Spherical coordinates of waypoints. Dotted lines represent the local solar time of the three maps.

proximity to the small body are given in Eq. (1), where $\tilde{\hat{o}}_3$ is the skew-symmetric cross-product matrix of $\hat{\boldsymbol{o}}_3$.

$$\ddot{\boldsymbol{r}} = -\ddot{F}\tilde{\hat{o}}_3\boldsymbol{r} - 2\dot{F}\tilde{\hat{o}}_3\dot{\boldsymbol{r}} - \dot{F}^2\tilde{\hat{o}}_3\tilde{\hat{o}}_3\boldsymbol{r} + \frac{\partial U_g}{\partial \boldsymbol{r}} + \frac{\partial U_s}{\partial \boldsymbol{r}} + \boldsymbol{a}_{\text{srp}} \tag{1}$$

The first time derivative of the true anomaly is provided in Eq. (2), and the second time derivative of the true anomaly is provided in Eq. (3).

$$\dot{F} = \sqrt{\mu_{\text{sun}}/[A(1-E)^2]^3}(1 + E\cos F)^2 \tag{2}$$

$$\ddot{F} = -2E\sqrt{\mu_{\text{sun}}/[A(1-E)^2]^3}\sin F(1 + E\cos F)\dot{F} \tag{3}$$

The gravitational parameter of the sun is given by μ_{sun}, the semimajor axis of the asteroid is given by A, and the eccentricity of the asteroid's orbit is E. A point-mass gravity model is utilized for the asteroid, with the derivative of the gravitational potential given as

$$\frac{\partial U_g}{\partial \boldsymbol{r}} = -\frac{\mu_{\text{ast}}}{r^3}\boldsymbol{r} \tag{4}$$

where μ_{ast} is the gravitational parameter of the asteroid. The gravity of the sun is modeled as a third-body perturbation, with the derivative of the gravitational potential given as

$$\frac{\partial U_s}{\partial \boldsymbol{r}} = \frac{\mu_{\text{sun}}(3\hat{\boldsymbol{d}}\hat{\boldsymbol{d}}^T - [I_{3\times 3}])}{d^3}\boldsymbol{r} \tag{5}$$

where $\hat{\boldsymbol{d}}$ is the direction of the sun. The acceleration due to SRP is given as

$$\boldsymbol{a}_{\text{srp}} = -\frac{P_0(1+\rho)A_{\text{sc}}(1AU)^2}{M_{\text{sc}}d^2}\hat{\boldsymbol{d}} \tag{6}$$

where ρ is the surface reflectivity, A_{sc} is the surface area of the spacecraft, M_{sc} is the mass of the spacecraft, and P_0 is the solar flux at 1 AU. The values for ρ (0.4) and P_0 (4.56×10^{-6} N·m^{-2}) are taken from Takahashi and Scheeres [23].

D. Waypoint Maneuvering

A continuous feedback control law is implemented to maneuver the spacecraft between the various waypoints using the methodology described in Chapter 14 of Schaub and Junkins [44]. The control acceleration is computed with the following equation:

$$\boldsymbol{u} = -(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}_{\text{ref}})) - [K_1]\Delta\boldsymbol{x}_1 - [K_2]\Delta\boldsymbol{x}_2 \tag{7}$$

where $\boldsymbol{f}(\boldsymbol{x})$ is given in Eq. (1), $\Delta\boldsymbol{x}_1 = {}^{\mathcal{O}}\boldsymbol{r} - {}^{\mathcal{O}}\boldsymbol{r}_{\text{ref}}$ and $\Delta\boldsymbol{x}_2 = {}^{\mathcal{O}}\dot{\boldsymbol{r}} - {}^{\mathcal{O}}\dot{\boldsymbol{r}}_{\text{ref}}$. In the case where noisy measurements of the spacecraft position and velocity are utilized, a dead band is placed around the state vectors such that the control law does not expend all of its ΔV budget, correcting for small errors due to noise. The dead band is not required in the event that the state is known perfectly or if the measurements are filtered. The controller gains are manually tuned to ensure the spacecraft can maneuver to the waypoints in about 2 h. The gains are given in Eq. (8).

$$[K_1] = (5 \times 10^{-4})[I_{3x3}]\ \text{s}^{-2}, \quad [K_2] = [I_{3x3}]\ \text{s}^{-1} \tag{8}$$

E. State Estimation

Next, the mission simulation fidelity is enhanced with the inclusion of an EKF to estimate the state of the spacecraft $\boldsymbol{x} = [{}^{\mathcal{O}}\boldsymbol{r};\ {}^{\mathcal{O}}\dot{\boldsymbol{r}}]$. An EKF produces a state estimate for a dynamic system by predicting the state of the system through integrating equations of motion and updating this prediction with measurements from the environment [45]. Navigation solutions for small body missions typically use some combination of radiometric ground tracking measurements from Earth-based sensors like the DSN and optical measurements in proximity about the small body that are matched with landmark maps of the surface of the body, which are used to provide the relative navigation solution. A batch filtering process is utilized, which is the state-of-the-art for small body missions [46–49]. An iterative process between the orbit determination (OD) and optical navigation (OPNAV) teams occurs where the OD team updates the spacecraft trajectory and the OPNAV team updates the landmark maps. This is an intensive ground-based process, and the navigation solution for the next epoch is sent back up to the spacecraft for the next epoch of operations. Filters like extended or unscented Kalman filters are popular in the literature for autonomous spacecraft operations as they can continually produce a state estimate on board the spacecraft [30,50,51]. A benefit of such filters is that they do not require multiple iterations to converge to a solution, which a batch filter does.

The algorithm for the EKF is provided in Algorithm 1. The initial state of the EKF is initialized with a small amount of error added to the truth state sampled from uniform distributions of $\mathcal{U}[-5, 5]$ m and $\mathcal{U}[-0.01, 0.01]$ m/s for position and velocity, respectively. The first step of the algorithm is to propagate the dynamics of the system forward in time. The dynamics are propagated using a fourth-order

Algorithm 1: Extended Kalman filter for small body navigation

Initialize $i = 1$, $t_{i-1} = t_0$, $\hat{\boldsymbol{x}}_{i-1}^{+} = \hat{\boldsymbol{x}}_0$, $P_{i-1}^{-} = P_0$
for iteration $1:N$
  Propagate dynamics:
    $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\hat{\boldsymbol{x}}_{i-1}^{+}, \boldsymbol{u}_{i-1}, t)$
    $A(t) = \left.\dfrac{d\boldsymbol{f}}{d\boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}}$
    $\dot{\Phi}(t, t_{i-1}) = A(t)\Phi(t, t_{i-1})$
    Compute $\hat{\boldsymbol{x}}_i^{-}$ and $\Phi(t_i, t_{i-1})$ using RK4 integration
  Update covariance:
    $P_i^{-} = \Phi(t_i, t_{i-1})P_{i-1}^{+}\Phi^T(t_i, t_{i-1}) + Q(t_i, t_{-1})$
  if new measurements
    Read measurements, $\boldsymbol{y}_i$
    Compute measurement residuals and Kalman gain
      $\boldsymbol{r}_i = \boldsymbol{y}_i - \boldsymbol{h}(\hat{\boldsymbol{x}}_i^{-}, t_i)$
      $K_i = P_i^{-}H_i^T(H_iP_i^{-}H_i^T + R)^{-1}$, where $H_i = \left.\dfrac{d\boldsymbol{h}}{d\boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_i^{-}}$
    Perform measurement update
      $\hat{\boldsymbol{x}}_i^{+} = \hat{\boldsymbol{x}}_i^{-} + K_i\boldsymbol{r}_i$
      $P_i^{+} = (I - K_iH_i)P_i^{-}(I - K_iH_i)^T + K_iRK_i^T$
  else
    $\hat{\boldsymbol{x}}_i^{+} = \hat{\boldsymbol{x}}_i^{-}$
    $P_i^{+} = P_i^{-}$
    $i = i + 1$

Runge-Kutta (RK4) integrator. The state transition matrix, $\Phi(t, t_{i-1})$, is computed by integrating the linearized dynamics of the system, which are computed using the Jacobian of the dynamics, $A(t) = \left.\dfrac{d\boldsymbol{f}}{d\boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}}$. The state transition matrix is then used to propagate the covariance forward in time. The covariance is propagated using the following equation:

$$P_i^{-} = \Phi(t_i, t_{i-1})P_{i-1}^{+}\Phi^T(t_i, t_{i-1}) + Q(t_i, t_{i-1}) \tag{9}$$

where $Q(t_i, t_{i-1})$ is the process noise covariance. The process noise covariance is computed using the state noise compensation (SNC) algorithm [52]. The process noise covariance at any time t_i is given as

$$Q(t_i, t_{i-1}) = \sigma_u^2 \begin{bmatrix} \dfrac{\Delta t^3}{3}[I_{3\times3}] & \dfrac{\Delta t^2}{2}[I_{3\times3}] \\ \dfrac{\Delta t^2}{2}[I_{3\times3}] & \Delta t[I_{3\times3}] \end{bmatrix} \tag{10}$$

The diffusion coefficient σ_u^2 was experimentally tuned and is set to 10^{-11} m/s^2.

After the propagation step, the algorithm checks to see if there are any new measurements. If there are new measurements, the measurement update step is performed. The measurement residuals are computed as

$$\boldsymbol{r}_i = \boldsymbol{y}_i - \boldsymbol{h}(\hat{\boldsymbol{x}}_i^{-}, t_i) \tag{11}$$

where $\boldsymbol{y}_i$ is the measurement vector and $\boldsymbol{h}(\hat{\boldsymbol{x}}_i^{-}, t_i)$ is the measurement model. Basilisk's `simpleNav` module is used to provide measurements to the EKF. This module utilizes a second-order Gauss–Markov error model to provide realistic measurement error. While `simpleNav` does not capture several of the intricacies of small body navigation measurements (i.e., scale invariance or a dependency on lighting conditions), it provides a reasonable approximation of measurement error beyond additive white Gaussian noise. The standard deviations for the white noise component of the measurement error model and the walk bounds of the Gauss–Markov process are provided in Table 2. Because the measurement model is simply the states of the spacecraft with noise and random walk added, the Jacobian of the measurement model is simply the identity matrix. After the measurement residuals are computed, the Kalman gain is computed as

Table 2 SimpleNav parameters

Parameter	Position, m	Velocity, m/s
σ	5	0.001
Walk bounds	1	0.001

$$K_i = P_i^{-}H_i^T(H_iP_i^{-}H_i^T + R)^{-1} \tag{12}$$

The measurement update step is then completed with the following two equations, where the updated state is computed as

$$\hat{\boldsymbol{x}}_i^{+} = \hat{\boldsymbol{x}}_i^{-} + K_i\boldsymbol{r}_i \tag{13}$$

and the covariance is updated as

$$P_i^{+} = (I - K_iH_i)P_i^{-}(I - K_iH_i)^T + K_iRK_i^T \tag{14}$$

where R is the measurement noise covariance matrix.

The entire process repeats until the end of the simulation. If no measurements are provided after the propagation step, the algorithm simply propagates the state and covariance forward in time and writes these out as messages, which are read by the decision-making agent.

F. Markov Decision Process Formulation

The small body science operations problem is formulated as an MDP. This section describes each of the components of the MDP formulation for the small body science operations problem. An MDP is a sequential decision-making problem in which a decision-making agent takes an action $a_i \in \mathcal{A}$ in some state $s_i \in \mathcal{S}$ following a policy $\pi(s_i)$ that maps states to actions, $\pi: \mathcal{S} \to \mathcal{A}$ [12,53]. The decision-making interval is denoted by i, and the state and action spaces are referred to as $\mathcal{S}$ and $\mathcal{A}$, respectively. After taking the step in the environment with the selected action, the agent receives a reward $r_i = R(s_i, a_i)$ based on the reward function and transitions to a new state s_{i+1} according to the transition function $T(s_{i+1}|s_i, a_i)$. The transition function must adhere to the Markov assumption, which states that the next state is dependent only on the current state and action: $T(s_{i+1}|s_i, a_i) = T(s_{i+1}|s_i, a_i, ..., s_0, a_0)$. The goal of the agent is to maximize the expected sum of discounted rewards, $R = \sum_{i=0}^{\infty}\gamma^i r_i$, where $\gamma \in [0, 1)$ is the discount factor. The discount factor weighs the relative importance of short-term versus long-term rewards and ensures that the sum of returns does not grow to infinity.

G. State Space

The state space of the small body science operations problem is designed to capture all relevant information for the purposes of adhering to the Markov assumption. The state space of the small body science scheduling problem is given as

$$\mathcal{S} = \mathcal{S}_{\text{sc}} \times \mathcal{S}_{\text{asteroid}} \times \mathcal{S}_{\text{maps}} \times \mathcal{S}_{\text{targets}} \times \mathcal{S}_{\text{DSN}} \tag{15}$$

where $\mathcal{S}_{\text{sc}}$ is the state space of the spacecraft, $\mathcal{S}_{\text{asteroid}}$ is the state space of the asteroid, $\mathcal{S}_{\text{maps}}$ is the state space of the spectroscopy maps, $\mathcal{S}_{\text{targets}}$ is the state space of the imaging targets, and $\mathcal{S}_{\text{DSN}}$ is the state space of the DSN.

The state returned to the decision-making agent at decision interval i is defined as $s_i \in S$:

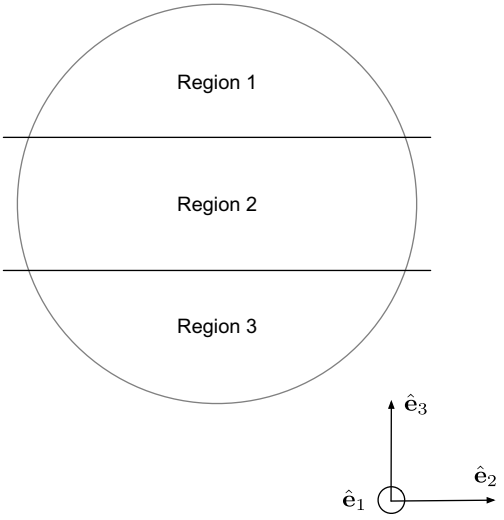


Fig. 4 Map regions.

of the position and velocity of the spacecraft in the sun–asteroid Hill frame are used instead of direct observations of the state. Additionally, the log of the diagonal of the covariance matrix divided by five is added and provided as an observation: $\log_{10}(\text{diag}(P))/5$. This provides information on the quality of the navigation solution that is normalized to a range of $[-1, 1]$ for $10^{-5} \leq P \leq 10^{5}$.

H. Action Space

A mode-based planning approach is taken in the action space. A spacecraft mode turns certain models on or off and sets the attitude reference for a prescribed amount of time, abstracting continuous low-level behavior into higher-level abstractions of spacecraft behavior. An action space $\mathcal{A}$ is constructed for the small body science scheduling problem that allows the decision-making agent to collect and downlink science data, manage its resources, and transition between waypoints:

$$\mathcal{A} = \{\text{Charge, Waypoint 1, ..., Waypoint 8, Map, Image, Downlink, Nav Update}\} \tag{17}$$

$$s_i = \left({}^{\mathcal{O}}\boldsymbol{r}_{\text{sc}},\ {}^{\mathcal{O}}\boldsymbol{v}_{\text{sc}},\ {}^{\mathcal{O}}\boldsymbol{r}_{t_{\text{nearest}}},\ {}^{\mathcal{O}}\boldsymbol{r}_{w_{\text{ref}}},\ {}^{\mathcal{O}}\boldsymbol{r}_{w_{\text{prev}}}, \ldots \right.$$
$$\text{num. imaged targets, num. downlinked targets, map regions, battery, } \ldots$$
$$\left. \text{eclipse, buffer, } \Delta V \text{ consumed, ground station indicator} \right) \tag{16}$$

Geometric information is included in the state space to capture the spatial relationship between the science objectives and the spacecraft. It can also provide information on resource management states and the risk of collision. These states include the spacecraft position and velocity (${}^{\mathcal{O}}\boldsymbol{r}_{\text{sc}}$ and ${}^{\mathcal{O}}\boldsymbol{v}_{\text{sc}}$), position of the nearest imaging target (${}^{\mathcal{O}}\boldsymbol{r}_{t_{\text{nearest}}}$), position of the current waypoint (${}^{\mathcal{O}}\boldsymbol{r}_{w_{\text{ref}}}$), and position of the previous waypoint (${}^{\mathcal{O}}\boldsymbol{r}_{w_{\text{prev}}}$). These states are all expressed in the sun–asteroid Hill frame, $\mathcal{O}$, a convenient expression for this problem given that one of the primary science objectives is mapping at specific local solar times.

Several observations are also included to provide a measure of science objective completion. The number of imaged and downlinked targets in T is included in the state space. For each map M_j, the mapping points are partitioned into three equally sized groups based on the value of the z-component of the body-fixed position of the mapping points. The body frame of the asteroid is defined as $\mathcal{E}: \{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{e}}_3\}$. The three regions are displayed in Fig. 4. This state provides the agent information on which regions still need to be mapped. For each region, the map state is equal to the total number of mapped points divided by the total number of points in the region. Because this work assumes the rotation pole of the body is aligned with the orbit normal, this state representation is sufficient. However, varying the rotation pole of the body may require a more sophisticated state representation.

Finally, several states are included to retain information on resource constraints and safety. The data stored in the buffer and ground station indicator provide state information for the on-board data system. The ground station indicator contains information on how close the next downlink opportunity is. If a DSN station is available, the indicator is equal to one. Otherwise, the indicator is computed by subtracting from 1 the amount of time to the next downlink window divided by 24 h. The battery charge and eclipse indicator are binary indicators that provide information for the purposes of power management. The available ΔV state indicates how much fuel the spacecraft has available to use.

Each state is normalized to a range of approximately $[-1, 1]$. The spacecraft position, position of the nearest imaging target, and position of the current and previous waypoint are all normalized by the radius of the body. The number of imaged and downlinked targets, the mapped regions, and resource states are all normalized by their respective max values such that they are within a range of $[0, 1]$.

In the event that the navigation mode is added to the action space and the agent acts using the belief state produced by the EKF, the estimate

Each mode lasts for 2000 s, with the exception of the mapping mode and optional navigation mode. The mapping mode lasts for 4000 s, which is approximately one quarter of a full revolution of the body about its rotation pole. The navigation mode only lasts for 1000 s. A detailed description of the action space is provided by the bulleted list below:

1) *Charge:* The spacecraft points its solar panels at the sun, turning off all instruments and transmitters to recharge the batteries.

2) *Waypoint actions:* The spacecraft targets one of the eight neighboring waypoints, turning off all instruments and transmitters during the duration of the maneuver mode. The eight neighboring waypoints are defined as follows:

a) $\psi_{\text{ref}} = \psi_{\text{ref}} + 30°, \theta_{\text{ref}} = \theta_{\text{ref}}$
b) $\psi_{\text{ref}} = \psi_{\text{ref}} + 30°, \theta_{\text{ref}} = \theta_{\text{ref}} + 60°$
c) $\psi_{\text{ref}} = \psi_{\text{ref}}, \theta_{\text{ref}} = \theta_{\text{ref}} + 60°$
d) $\psi_{\text{ref}} = \psi_{\text{ref}} - 30°, \theta_{\text{ref}} = \theta_{\text{ref}} + 60°$
e) $\psi_{\text{ref}} = \psi_{\text{ref}} - 30°, \theta_{\text{ref}} = \theta_{\text{ref}}$
f) $\psi_{\text{ref}} = \psi_{\text{ref}} - 30°, \theta_{\text{ref}} = \theta_{\text{ref}} - 60°$
g) $\psi_{\text{ref}} = \psi_{\text{ref}}, \theta_{\text{ref}} = \theta_{\text{ref}} - 60°$
h) $\psi_{\text{ref}} = \psi_{\text{ref}} + 30°, \theta_{\text{ref}} = \theta_{\text{ref}} - 60°$

3) *Map:* The spacecraft turns on its mapping instrument, collecting the map of whichever map region it is currently in.

4) *Image:* The spacecraft turns on its imaging instrument, collecting an image of the nearest target.

5) *Downlink:* The spacecraft turns on its transmitter, downlinking the data in the buffer to the DSN.

6) *Navigation update:* Only used in some experiments, the spacecraft begins collecting measurements to improve the state estimate.

In the charging mode, the spacecraft turns off all instruments and the transmitter and points the solar panels at the sun to charge the battery. The action space also includes eight separate waypoint reference change actions. When a waypoint reference change action is taken, the current waypoint reference $w_{\text{ref}} = \{\psi_{\text{ref}}, \theta_{\text{ref}}\}$ changes to the selected adjacent waypoint reference. If one of these modes is selected, the last time the waypoint was changed is checked to see if a new waypoint can be selected. The current waypoint does not change unless 8000 s have passed since the last switch to ensure convergence to the current waypoint. After each change, the new waypoint azimuth and polar angle are checked to ensure it is wrapped within the appropriate domains, $\theta \in [0, 360]$ deg and $\psi \in [0, 180]$ deg. An example of this is provided in Fig. 5. The nominal transitions are shown in the dotted green line. Wrapped transitions are shown with the solid red line.

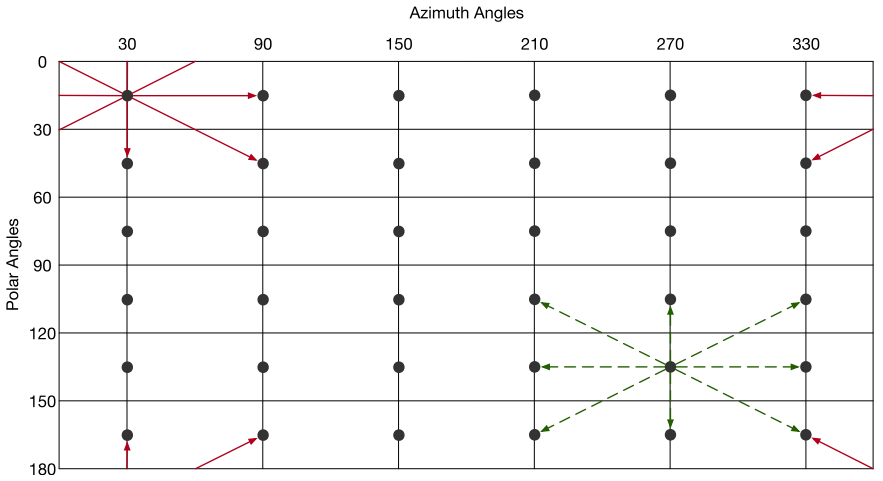


Fig. 5 Waypoint reference transitions.

Table 3 Science requirements

Parameter	Value
Imaging	
Minimum elevation	60°
Attitude error norm	0.1 rad
Mapping	
Minimum elevation	45°
Instrument half-FOV	22.5°
Azimuth tolerance	1°

In the mapping mode, the spacecraft points the mapping instrument at the asteroid. Data is collected in the on-board storage unit, and only the portion of the map collected within the requirements is considered mapped. Mapping requirements are provided in Table 3. Note that the requirement regarding the azimuth is 1 deg. At the nominal waypoint radius of 750 m, this translates to roughly 6.5 m of positional tolerance on either side of the nominal azimuth band. This is a very tight requirement. If the navigation mode is utilized in the action space, the spacecraft must periodically improve its state estimate with measurements to reduce the state error below this threshold. In the fully observable version of this problem, the decision-making agent has access to the map data and receives information on whether or not it has collected a mapping point. In the partially observable problem, the agent does not have access to the truth state directly, so the mapping states are not provided in the state space.

In the imaging mode, the spacecraft points the imager at the nearest target and attempts to take an image of the target. The image is collected if the spacecraft is within the elevation and range requirements of the target image. In the downlink mode, the spacecraft points the transmitter in the direction of the Earth. Data is downlinked once the spacecraft is within the elevation and range requirements of the DSN and the prescribed downlink time occurs.

Finally, the navigation mode is utilized in some experiments to provide a more realistic navigation update. In this mode, the spacecraft points an imager at the asteroid to simulate the use of feature-tracking optical navigation. The state estimate is improved, and the state-error covariance decreases.

I. Reward Function

The reward function $R(s_i, a_i, s_{i+1})$ is a piecewise function of the current state, action, and next state. The reward function builds off the reward functions designed for Earth-observing science scheduling problems [54,55], but adds mapping and additional failure conditions. The constant F scales the failure penalty, the constant A scales the imaging bonus, the constant B scales the mapping bonus, the constant C scales the image target downlink component, and the constant D scales the mapping downlink component. The reward at state i is given by

$$r_i = \begin{cases} -F & \text{if failure} \\ \dfrac{A}{|T|} H(c_j) & \text{if } \neg\,\text{failure} \wedge a_i \text{ is image} \\ \dfrac{B}{3|M|} \sum_j^3 \sum_k^{|\boldsymbol{M}_j|} H(m_{j,k}) & \text{if } \neg\,\text{failure} \wedge a_i \text{ is map} \\ \dfrac{C}{|T|} \sum_j^{|\boldsymbol{T}|} H(d_j) + \dfrac{D}{3|M|} \sum_j^3 \sum_k^{|\boldsymbol{M}_j|} H(f_{j,k}) & \text{if } \neg\,\text{failure} \wedge a_i \text{ is downlink} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

If the agent fails, a failure penalty of $-F$ is returned, and the episode terminates. The failure condition is true if the spacecraft expends all charge in the battery, overfills the data buffer, exceeds the ΔV budget, or collides with the body. Mathematically, this is represented as with Eq. (19), where z is the normalized charge of the battery and b is the normalized data buffer level.

$$\text{failure} = (z == 0 \vee b \geq 1 \vee \text{any}(||^{\mathcal{H}}\boldsymbol{r}_{s/c}|| \leq \boldsymbol{r}_{\text{ast}}) \vee \Delta V \geq \Delta V_{\text{budget}}) \tag{19}$$

The following function, $H(x_j)$, is used to check if the state variable x is false at step i and true at step $i+1$, returning 1 if these conditions are met.

$$H(x_j) = 1 \text{ if} \neg x_{j_i} \ \wedge \ x_{j_{i+1}} \tag{20}$$

The variable c_j represents whether target j has been imaged. If the imaging mode is initiated and a failure does not occur, target j is checked to determine if it was imaged for the first time. This reward component is normalized by the total number of targets and scaled by the constant A.

The variable $m_{j,k}$ represents whether mapping point k for map number j has been mapped. If the mapping mode is initiated and a failure does not occur, all map points are checked to determine if they were collected for the first time or not. The summation of this reward is normalized by $3|M|$ so the total possible reward for this component totals to the constant B.

The variable d_j represents whether or not target j has been downlinked, and the variable $f_{j,k}$ represents whether or not mapping point k for map number j has been downlinked. Both the set of targets and all map points are looped through to determine if they have been downlinked for the first time or not. Both the imaging and mapping components are multiplied by the constants C and D and divided by the total number of targets or mapping points.

J. Transition Function

Due to the continuous dynamics of the small body proximity operations science problem, it is difficult to construct a transition function with conditional probabilities that accurately captures state transitions. The transition function is instead represented by a generative model $G(s_i, a_i)$ given in Eq. (21). The generative model returns a new state s_{i+1} and reward r_i by integrating equations of motion forwards in time.

$$s_{i+1}, \ r_i = G(s_i, a_i) \tag{21}$$

The Basilisk astrodynamics software architecture is used to construct the simulation, which models the complex behavior of the spacecraft and environment. The Basilisk simulation is wrapped within a Gymnasium environment. The Gymnasium environment provides a standard interface for the agent to interact with the Basilisk simulation. The details of this simulation are provided in the next section.

K. Simulation Architecture

Basilisk models the dynamics of the spacecraft as fully coupled multibody dynamics, making no simplifying assumptions regarding the relative motion of the spacecraft, the asteroid, and the sun. However, both Basilisk and the dynamics described in this section utilize a cannonball SRP model. Furthermore, both model the gravity of the asteroid using a point-mass gravity model.

L. Basilisk Simulation Overview

A Basilisk simulation is implemented to serve as the generative transition function for the MDP. In Fig. 6, the task groupings and modules in the Basilisk simulation are provided. Each module contains either flight software or simulation code. The modules represent self-contained and modularized blocks of code that perform specific functions. For instance, the `simpleSolarPanel()` module contains the logic and the math to compute the amount of charge generated by a solar panel. The modules are grouped together as tasks, which organize the modules into similar blocks of code. Tasks are most useful when grouping flight software modules, as all modules on a task can be disabled, thus turning off the modules that belong to that task. Several flight software tasks are implemented. These include a sun-pointing task, an Earth-pointing task, a target-pointing task, a map-pointing task, an MRP control task, and a waypoint feedback control task. Depending on the flight mode, these tasks are turned on or off, primarily to determine which attitude reference should be used. A summary of each task's status in each flight mode is provided in Table 4. The sun-pointing, earth-pointing, target-pointing, and map-pointing tasks all use Basilisk's `locationPointing()`

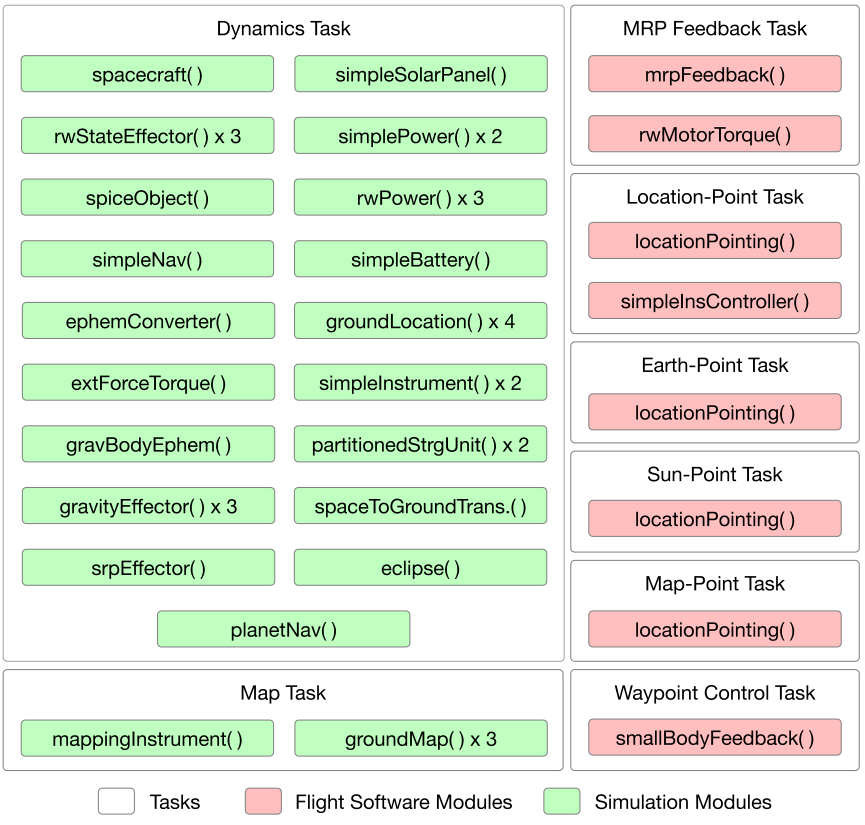


Fig. 6 Basilisk simulation diagram.

Table 4 Basilisk model and task status in different modes

Basilisk tasks and models	Modes				
	Charge	Waypoint change	Map	Image	Downlink
Sun-pointing task	Enabled	Enabled	Disabled	Disabled	Disabled
Earth-pointing task	Disabled	Disabled	Disable	Disabled	Enabled
Location-pointing task	Disabled	Disabled	Disabled	Enabled	Disabled
Map-pointing task	Disabled	Disabled	Enabled	Disabled	Disabled
MRP control task	Enabled	Enabled	Enabled	Enabled	Enabled
Waypoint control task	Enabled	Enabled	Enabled	Enabled	Enabled
Mapping task	Disabled	Disabled	Enabled	Disabled	Disabled
Imager power model	Off	Off	Off	On	Off
Imager data model	Off	Off	On	On	Off
Mapping power model	Off	On	Off	Off	Off
Mapping data model	Off	On	Off	Off	Off
Transmitter power model	Off	Off	Off	Off	On
Transmitter data model	Off	Off	Off	Off	On

module and output an attitude guidance message that includes the MRP attitude error $\boldsymbol{\sigma}_{B/R}$. The attitude guidance message is ingested by the `mrpFeedback()` module, which outputs a commanded torque. This commanded torque is utilized by the `rwMotorTorque()` module to compute reaction wheel motor torques and send a motor command message to the three reaction wheel state effectors in the dynamics task.

The waypoint feedback control task utilizes a feedback control law to regulate the state of the spacecraft to the desired Hill frame waypoint. The feedback control law outputs a force command, which the `externalForceTorque()` dynamics module utilizes to pass the commanded force to the spacecraft.

In addition to several flight software tasks, a dynamics task is also implemented, which holds the majority of the modules in the simulation. Gravity effectors for the asteroid, the sun, and the Earth are implemented. A `planetNav()` module is also implemented for the asteroid, which creates an ephemeris message utilized by the relevant flight software modules. Likewise, a `simpleNav()` module performs the same function, but for the spacecraft state. The `planetNav()` and the `simpleNav()` modules can optionally add noise to the states to imitate a navigation system.

Several dynamics modules are connected to the spacecraft. As previously stated, the commanded force is passed to the spacecraft with the `extForceTorque()` module. Additionally, a `solarRadiationPressure()` module is implemented. A cannonball SRP module is utilized. Finally, each reaction wheel state effector is connected to the spacecraft for the purposes of attitude control. Lastly, the `eclipse()` module utilizes the state of the asteroid and the spacecraft to indicate whether or not the spacecraft is in an eclipse.

A representative power system is modeled on board the spacecraft. At the center of the power system is a `simpleBattery()` module. The battery receives power generation and consumption messages from each other power module to compute the storage level at each time step. Solar panels are modeled using the `simpleSolarPanel()` module, which computes power generation based on the area of the panels, the efficiency of the panels, and the solar incidence angle. Instrument and transmitter power models are also implemented with the `simplePowerSink()` module.

An on-board data system is also modeled. This system is modeled using two tasks: the dynamics task and the mapping task. The dynamics task is always on, but the mapping pass is disabled for all modes except for the mapping mode. This is done to minimize required computation. In the mapping task, three `groundMap()` modules are connected to a `mappingInstrument()`. The `groundMap()` module loops through each mapping point and checks for three things: a) the spacecraft is within the elevation requirements of the point, b) the point is within the instrument's field-of-view, and c) the spacecraft is within the required azimuth angle band. A vector of access messages is then passed to the `mappingInstrument()`, which passes the data on to a `partitionedStorageUnit()`. This `partitionedStorageUnit()` in the mapping task keeps track of the points that have been imaged and those that have not.

In the dynamics task, two `simpleInstrument()` modules are implemented. One `simpleInstrument()` module is used in conjunction with the `simpleInstrumentController()` to image the ground targets if the imaging mode is entered. The other `simpleInstrument()` module is used to keep track of the amount of data generated by mapping. This module provides a scalar value for the data generated and does not keep track of the specific points. Both of these instruments pass the data to the `partitionedStorageUnit()` in the dynamics task.

Not shown in Fig. 6 is the addition of a `smallBodyNavEKF()` and an additional `simpleNav()` module. The `smallBodyNavEKF()` is a Basilisk module that implements an EKF for small body navigation. The `simpleNav()` module is the Basilisk module that implements a second-order Gauss–Markov error model for translational navigation measurements. The `smallBodyNavEKF()` is only utilized in some experiments.

M. Initial Conditions

The parameters of the spacecraft may be found in Table 5. The modeled spacecraft is a small spacecraft used in past work [54,55]. These parameters are balanced to create a scenario in which the spacecraft must make tradeoffs between resource constraints, science collection, and downlink. The initial conditions for the asteroid's orbit, size, and rotation may be found in Table 6. These parameters are based on those of Bennu [56,57]. During training, each of these values is kept constant, but the initial position of the spacecraft in the asteroid's Hill frame is randomized.

Table 5 Spacecraft parameters

Parameter	Value
General spacecraft parameter	
Mass	330 kg
Dimensions	1.38 m × 1.04 m × 1.58 m
Max ΔV	40 m/s
Power system	
Solar panel area	1.0 m^2
Solar panel efficiency	0.20
Instrument power draw	30 W
Transmitter power draw	15 W
Battery capacity	100 W·h
Data and communications system	
Data buffer storage capacity	125 GB
Transmitter Baud rate	120 Mbps
Instrument Baud rate	8 Mbps
Map instrument Baud rate	8 Mbps

Table 6 Asteroid parameters

Parameter	Value
Orbital parameters	
Semimajor axis, a	1.1259 AU
Eccentricity, e	0.016975
Inclination, i	0.0027666 deg
Longitude of ascending node, Ω	177.42 deg
Argument of periapsis, ω	284.26 deg
True anomaly, f	357.30 deg
Size and rotation	
Shape	Spherical
Rotation period	4.297461 h
Rotation pole	Orbit normal
Mean radius	250 m
Gravitational constant	$4.892\text{m}^3/\text{s}^2$

III. Methods

A. Reinforcement Learning

1. Introduction to Reinforcement Learning

The objective of reinforcement learning is to learn a policy that maps states to actions to maximize the expected value of all future rewards. The expected return is also referred to as the value function $V(s)$. The value function associated with some policy π when starting in some state s_i is given in Eq. (22), where $\mathbb{E}$ is the expected value operator. The value function can be thought of as a measure of how "good" a particular state is.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{i+k} \,\middle|\, s_i = s\right] \tag{22}$$

A state-action value function may also be defined, which is the expected return when starting in some state s_i, taking some action a_i, and following some policy π thereafter. This is given in Eq. (23). This is useful when evaluating the value associated with a particular state-action pair. Similar to the value function, the state-action value function can be thought of as a measure of how "good" a particular state-action pair is.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty}\gamma^k r_{i+k} \,\middle|\, s_i = s, a_i = a\right] \tag{23}$$

The optimal value function is the value function associated with following the optimal policy, and it is referred to as $V^*(s)$. The optimal value function can be defined recursively using the Bellman optimality equation, given in Eqs. (24) and (25).

$$V^*(s) = \max_a \sum_{s_{i+1}\in\mathcal{S}} T(s_{i+1}|s_i,a)[R(s_i,a) + \gamma V^*(s_{i+1})] \tag{24}$$

$$Q^*(s,a) = \sum_{s_{i+1}\in\mathcal{S}} T(s_{i+1}|s_i,a)\left[R(s_i,a) + \gamma \max_{a_{i+1}} Q^*(s_{i+1},a_{i+1})\right] \tag{25}$$

If the optimal value function has been solved for, the optimal policy can be extracted from the optimal value function by using the expression in Eq. (26).

$$\pi^*(s) = \arg\max_a \sum_{s_{i+1}\in\mathcal{S}} T(s_{i+1}|s_i,a)[R(s_i,a) + \gamma V^*(s_{i+1})] \tag{26}$$

Tabular solution methods solve for the optimal policy or value function for discrete state and action spaces or discretized continuous state and action spaces. A review of these methods may be found in Chapter 4 of Ref. [53]. If the transition probabilities and reward function are known, value or policy iteration can be used to iteratively solve for the optimal policy. These algorithms are known as dynamic programming algorithms. However, for many real-world problems, the use of these algorithms is not possible because the transition function cannot be represented using conditional probabilities. Another family of tabular RL methods are online RL methods, which restrict computation to states that are reachable from the current state and only require a generative transition function, which samples from some underlying distribution and/or integrates equations of motion. Many of these methods rely on constructing a search tree over the states and actions. Examples include a forward search or a Monte Carlo tree search. While these algorithms are useful for many problems, they can be very computationally expensive and must be rerun if the initial condition changes.

2. Deep Reinforcement Learning

Approximate solution methods remedy these issues with the use of a universal function approximation. Function approximation is used to generalize over the state space by learning from only a subset of the state space. Universal function approximators parameterize the value function or policy using a set of parameters θ. This work refers to parameterized value functions as $V_\theta(s)$ and parameterized policies as $\pi_\theta(s)$. The most popular type of universal function approximator in reinforcement learning is the artificial neural network (ANN). An ANN is a nonlinear function approximator made up of a series of interconnected layers that model the neurons in a nervous system. Each node in the ANN represents an activation function, and the lines represent the edges, or outputs, from one node to another. Each edge has its own weight and bias that determines the strength of the signal from the corresponding node.

Many algorithms exist that leverage deep neural networks to approximate the value function or policy for reinforcement learning. Deep Q-learning learns a parameterized state-action value function $Q_\theta(s,a)$ referred to as a deep Q-network (DQN) [58]. Deep Q-learning interacts with the environment, storing transitions in a replay buffer and periodically updating a target state-action value function. Policy gradient reinforcement learning algorithms learn a parameterized policy $\pi_\theta(a|s)$, which is a probability distribution conditioned on the state. For certain problems, it is easier to learn a policy directly than it is to learn a value function. It may be easier to learn to take an action as opposed to the precise numerical value associated with that action. For value-based methods, small errors in the learned value function may lead to large errors in the extracted policy. REINFORCE is an example of a policy gradient algorithm [12]. Finally, actor–critic methods learn both a parameterized policy and a parameterized value function. Actor–critic methods use the value function to estimate the value of the current state and the value of the next state in the update step, which reduces variance in the policy update. An example of such an algorithm is asynchronous advantage actor–critic (A3C) [59].

3. Proximal Policy Optimization

DQN, REINFORCE, A3C, and many other methods have advanced the state-of-the-art of reinforcement learning and demonstrated excellent performance on a number of tasks. However, these methods are not very data efficient. Each sample generated through interaction with the environment is used only once for training, with the exception of DQN, which uses a replay buffer. Furthermore, these algorithms are not particularly robust or stable. Each requires careful hyperparameter tuning for different tasks. PPO is an actor–critic reinforcement learning algorithm that addresses these issues [60]. To improve sample efficiency, PPO trains on the sampled data for multiple epochs. To improve stability, PPO uses a clipped objective function that ensures the size of the policy update isn't too large. The loss function for PPO is provided in Eq. (27).

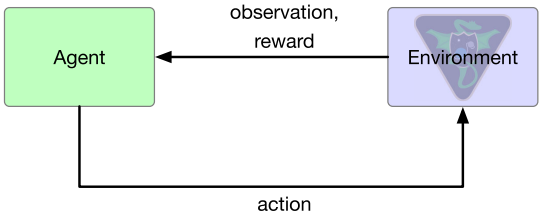


Fig. 7 Agent–environment interaction.

$$L_i^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_i[L_i^{CLIP}(\theta) - c_1 L_i^{VF}(\theta) + c_2 S[\pi_\theta](s_i)] \quad (27)$$

The loss components are provided in Eqs. (28) and (29).

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_i\Big[\min\Big(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\Big)\Big] \quad (28)$$

$$L^{VF}(\theta) = \hat{\mathbb{E}}_i[(V_\theta(s_i) - V)^2] \quad (29)$$

$S[\pi_\theta]$ is an entropy bonus, and $r_i(\theta)$ is the probability ratio:

$$r_i(\theta) = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta^-}(a_i|s_i)} \quad (30)$$

θ^- represents the parameters of the policy before the update.

The algorithm for PPO is provided in Algorithm 2. The parameters for the policy and value function are shared. The algorithm is run for N iterations. Each iteration consists of M actors, which each collect $|I|$ samples. I is the set of decision-making intervals for one full episode. The samples are stored, and the advantage estimates are then computed. The policy and value function are then updated for K epochs. The policy and value function are updated using a batch size of $\leq |I|$.

Algorithm 2: Proximal policy optimization algorithm

Initialize policy $\pi_\theta(s)$ and value function $V_\theta(s)$ with parameters θ
for iteration $1:N$
  for actor $1:M$
    for $i = 1:|I|$
      $a_i \sim \pi_\theta(a_i|s_i)$
      $s_{i+1}, r_i \sim G(s_i, a_i)$
      Store s_i, a_i, r_i
    compute advantage estimates $\hat{A}_1 \cdots \hat{A}_{|I|}$
  optimize $L(\theta)$ w.r.t. θ, with K epochs and batch size $\leq |I|$

B. Training Pipeline

This section describes the training pipeline for the small body science operations problem. The training pipeline consists of three important software packages. The Basilisk astrodynamics software architecture models the problem as described in Sec. II.K. This section describes the remaining two software packages: the Gymnasium package[§] used to wrap the simulation to provide a standard interface for reinforcement learning and the Stable-Baselines3 (SB3) software package[¶] that implements the PPO algorithm. The environment and training scripts may be found in the bsk_rl GitHub repository.[**]

The Gymnasium environment interface provides a standardized interface for interacting with reinforcement learning algorithms. The Gymnasium environment wraps the Basilisk simulation. It receives actions from the reinforcement learning agent, turning on or off the Basilisk tasks and models as necessary and running the simulation for the prescribed amount of time. Once the action has been taken and the simulation has been run in the desired flight mode for the specified amount of time, the Gymnasium environment constructs the observations and reward, which are returned to the decision-making agent. This process repeats until the end of the planning horizon or the episode terminates. A diagram of this interaction is provided in Fig. 7.

SB3's PPO implementation is utilized to train the decision-making agent. As described in Algorithm 2, for each iteration of the algorithm, PPO initializes M actors. Each worker spins up its own instantiation of the Gymnasium environment with its own set of initial conditions sampled from the distributions regarding initial battery charge, spacecraft attitude and attitude rate, spacecraft position relative to the body, etc. The actors step through the environment using the current learned policy. If one environment fails part way through due to a violation of the resource constraints or ends its episode early, the other actors pause until the terminated environment resets with a new set of initial conditions. After all actors have reached the required number of steps, the policy is updated with this experience, and the process repeats for the specified number of iterations.

IV. Results

This section presents several experiments for the small body science scheduling problem, using two versions of the environment: one without the optional navigation mode and one with it. The first experiment performed is a hyperparameter search over the batch size, number of epochs, network widths, and network depths without the optional navigation mode to determine the sensitivity of PPO and the environment to these hyperparameters. After the hyperparameters are optimized, a search over the reward function in Eq. (18) is performed to determine how image collection, map collection, image downlink, map downlink, and the failure penalty should be weighted. Some of these trained policies are also deployed on the various observation types to determine the impact of the observation type on the performance of the policy. The policies trained with the truth state are deployed using noisy measurements of the state and the EKF belief state to determine the impact of noise and state estimation error on the performance of the policy. Then, a demonstration is performed to show that the trained policies are robust to unexpected events like a DSN outage. Finally, a set of policies are trained with the dedicated navigation mode and associated state return. These policies are evaluated and compared to the policies trained without the dedicated navigation mode.

The experiments performed in this section are done using the Alpine high-performance computing resource at the University of Colorado, Boulder. Each experiment is performed with an x86_64 AMD Milan CPU with 64 cores and 3.75 GB of RAM allocated per core. No GPU is used in training. Alpine utilizes the RHEL version 8.4 operating system. However, the tools described thus far can be utilized on almost any Linux or MacOS operating system. Windows is not recommended.

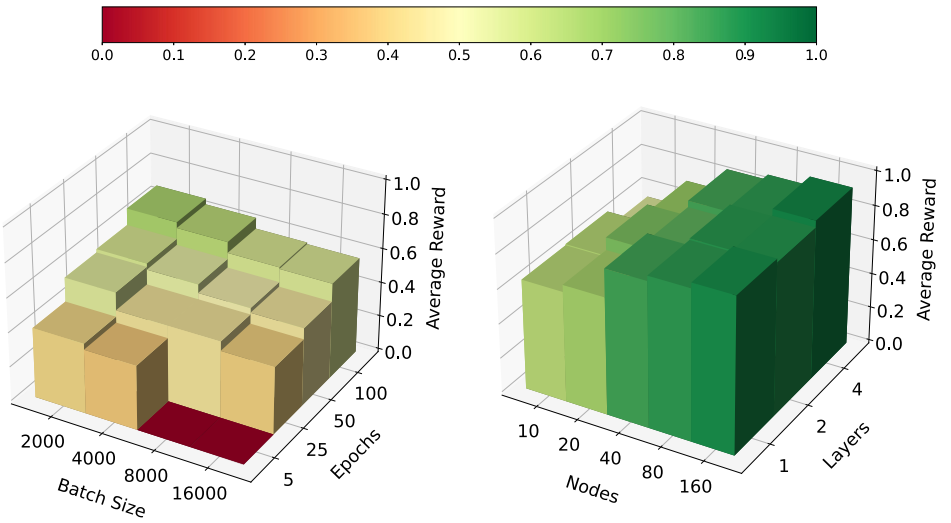
A. Hyperparameter Searches

To determine which hyperparameters should be selected for future experiments, two hyperparameter searches are performed for the small body science scheduling problem. For both experiments, reward is split evenly between collection and downlink for imaging and mapping, which are also split evenly. The constants of the reward function in Eq. (18) are $A = B = C = D = 0.25$, and the failure penalty is set to $F = 1$. In the first hyperparameter search, the batch size and number of epochs are varied. A small network size is selected for this hyperparameter search. The network width and depth are fixed at 20 and 4, respectively, to ensure that a relatively large network does not obfuscate issues with the batch size and number of epochs. The learning rate is fixed at 3e−5. The results of this hyperparameter search are shown in Fig. 8a. The

[§]Data available online at https://gymnasium.farama.org/index.html.

[¶]Data available online at https://stable-baselines3.readthedocs.io/en/master/.

[**]Data available online at https://github.com/AVSLab/bsk_rl.



a) Hyperparameter optimization over batch size and epochs. Hyperparameters: width = 20, depth = 4, LR = 3e-5

b) Hyperparameter optimization over policy width and depth. Hyperparameters: batch size = 2000, epochs = 100, LR = 3e-5

Fig. 8 PPO hyperparameter searches for the small body science environment. Each bar represents the average of five trials.

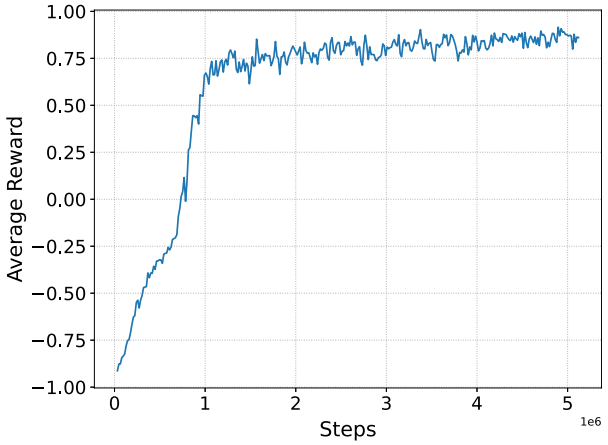


Fig. 9 Example curve of average reward vs steps.

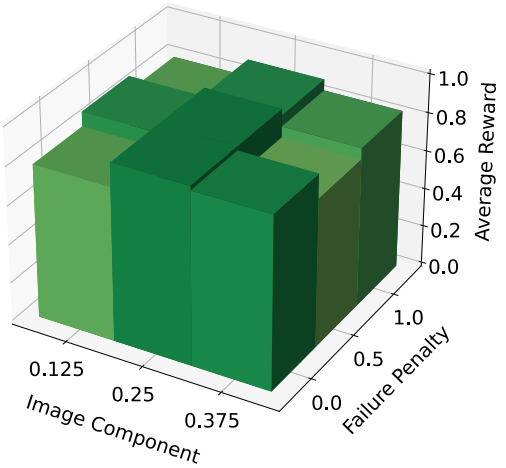


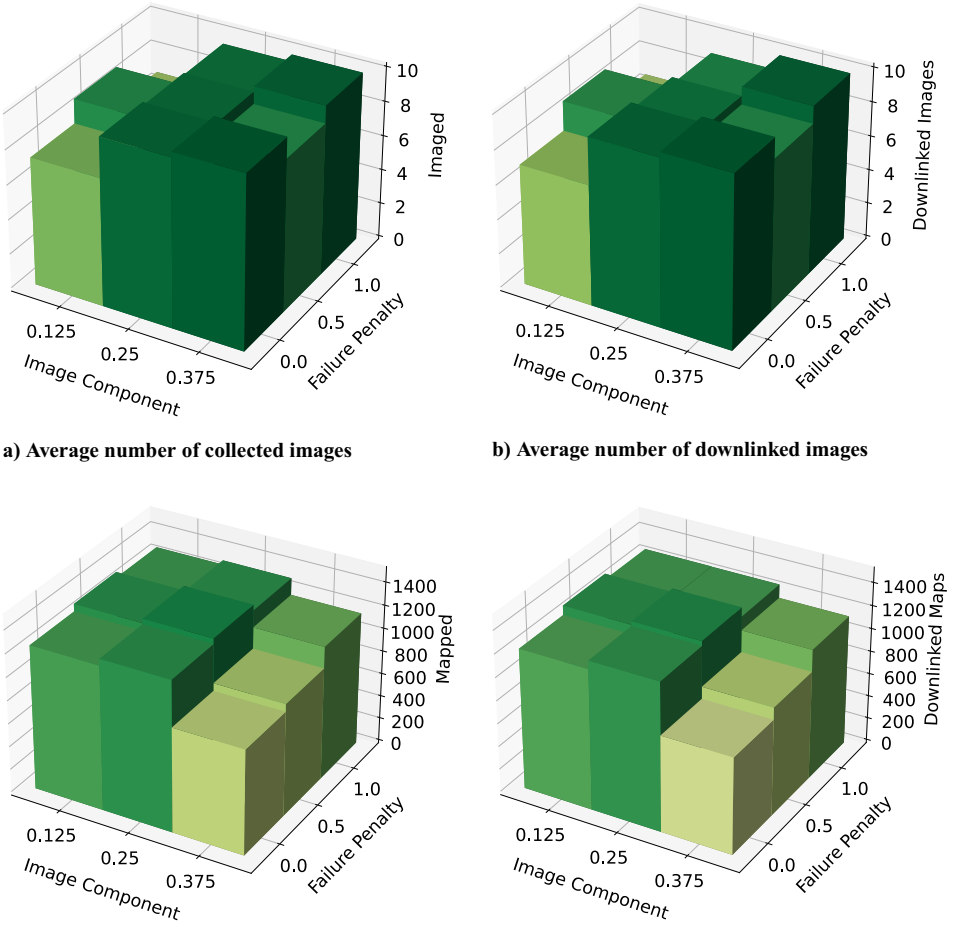
Fig. 10 Average reward across the reward function parameters. Hyperparameters: batch size = 2000, epochs = 100, LR = 3e−5, width = 320, depth = 1.

average reward increases as the batch size decreases and the number of training epochs increases. The best-performing policy is trained with a batch size of 2000 and 100 epochs. As the number of epochs increases, PPO is able to better fit the data in the batch. The second hyperparameter search varies the network width and depth. An example plot of reward versus the number of training steps is provided in Fig. 9. The batch size and number of epochs are fixed at 2000 and 100, respectively. The learning rate is fixed at 3e −5. The results of this hyperparameter search are shown in Fig. 8b. The best-performing policy is trained with a network width of 160 and a network depth of 1. The average reward increases with larger network widths and depths, with the optimized policies achieving reward close to 1, meaning that the decision-making agent is imaging and downlinking nearly all of the science data without violating a resource constraint. Larger numbers of nodes and layers can be advantageous because the selected RL algorithm can extract more complex features from the data. Larger numbers of layers and nodes also increase execution time, but execution time is still on the order of milliseconds for the selected hyperparameters. More layers and nodes could be explored, but this would only increase training time for minimal gains.

B. Reward Function Engineering

After the hyperparameters are tuned, a search over the reward function parameters in Eq. (18) is performed using the optimized hyperparameters (the network width is increased to 320, however). The reward is equally split between collection and downlink, but the weighting towards imaging and mapping is varied from 25 to 75%. The failure penalty is also varied from 0 up to a penalty of $F = 1$. The average reward across these hyperparameters after training is provided in Fig. 10. An imaging component of 0.125 means that $A = C = 0.125$, so the mapping components are $B = D = 0.375$. In terms of average reward, changing the relative weighting of imaging and mapping does not appear to have a large impact on the final average reward. However, because the failure penalty changes the range of reward, more metrics need to be collected to determine whether or not this is true.

In Fig. 11, the average number of imaged and downlinked surface targets and collected and downlinked maps are presented. A clear dependency on the relative weight of surface target imaging versus downlink is present in the results. When surface target imaging is weighted more, the average number of surface imaging targets collected and downlinked increases. When mapping is



a) Average number of collected images **b) Average number of downlinked images**

c) Average number of collected mapping points **d) Average number of downlinked mapping points**

Fig. 11 Average number of landing sites and map points collected and downlinked over the reward components.

weighted more, the average number of mapping points collected and downlinked increases. Equal weighting of surface target imaging and mapping results in decision-making agents that collect and downlink a high amount of both surface targets and mapping points. This result makes intuitive sense as both types of scientific data are prioritized equally. In general, the failure penalty does not appear to have a large impact on these numbers. If the decision-making agent fails, it is penalized by not being able to collect and downlink further science data.

The average length of the simulation and ΔV are also collected. For brevity, these plots are not shown here. The average length of the simulation is relatively constant over the reward parameters, with the average episode length around 7 days. However, a failure constant of 1 does result in the most stable average simulation length, which means the decision-making agent eliminates failures altogether. The ΔV is slightly more variable, with no obvious dependency on the reward parameters, but all decision-making agents keep the ΔV around 10 m/s. This is far lower than the maximum ΔV of 40 m/s. The agents display a tendency to use up all of the fuel early in training by doing too many unnecessary maneuvers. However, the agents relatively quickly learn to conserve fuel and only maneuver to the waypoints required for mapping.

C. Policy Evaluation

This section evaluates the trained policies using the following reward components: $A = B = C = D = 0.25$, $F = 1$. The science components of the reward function are equally weighted, and the failure penalty is set to the highest value. The first experiment performed in this section evaluates the policy trained with these reward parameters under various observation types. The nominal observation type observes the relative spacecraft state directly. The Basilisk `simpleNav` module observation type observes noisy measurements of the relative spacecraft state. If this observation type is utilized, a dead band equal to the 1σ values in Table 2 is added to the control law to ensure extra fuel is not spent trying to correct for measurement noise. The EKF observation type observes the EKF belief state, where the EKF continually ingests measurements to improve the state estimate. No controller dead band is required here. The results of these experiments for $N = 20$ trials are presented in Table 7. The nominal state observations result in the highest average reward, but the confidence intervals

Table 7 Trained policy deployed with different observation types: 7-day-long planning horizon

Metric	Nominal	SimpleNav	EKF
Average reward	0.91 ± 0.02	0.88 ± 0.04	0.86 ± 0.05
Average ΔV	9.4 ± 0.3	9.5 ± 0.4	9.4 ± 0.3
Collected images	9.5 ± 0.39	9.1 ± 0.5	8.5 ± 0.7
Downlinked images	9.4 ± 0.39	9.0 ± 0.5	8.5 ± 0.7
Collected map (6 PM LST)	451 ± 14	467 ± 9	468 ± 12
Collected map (2 PM LST)	426 ± 30	390 ± 46	392 ± 47
Collected map (10 AM LST)	443 ± 18	430 ± 30	455 ± 20
Downlinked map (6 PM LST)	451 ± 14	467 ± 9	468 ± 12
Downlinked map (2 PM LST)	408 ± 32	387 ± 45	381 ± 47
Downlinked map (10 AM LST)	439 ± 18	424 ± 29	454 ± 20

for each metric and each observation type overlap. There is no appreciable difference in performance between each observation type. This is a major feature of the problem formulation and training methodology, as a small amount of noise added to the state does not significantly impact the performance of the decision-making agent.

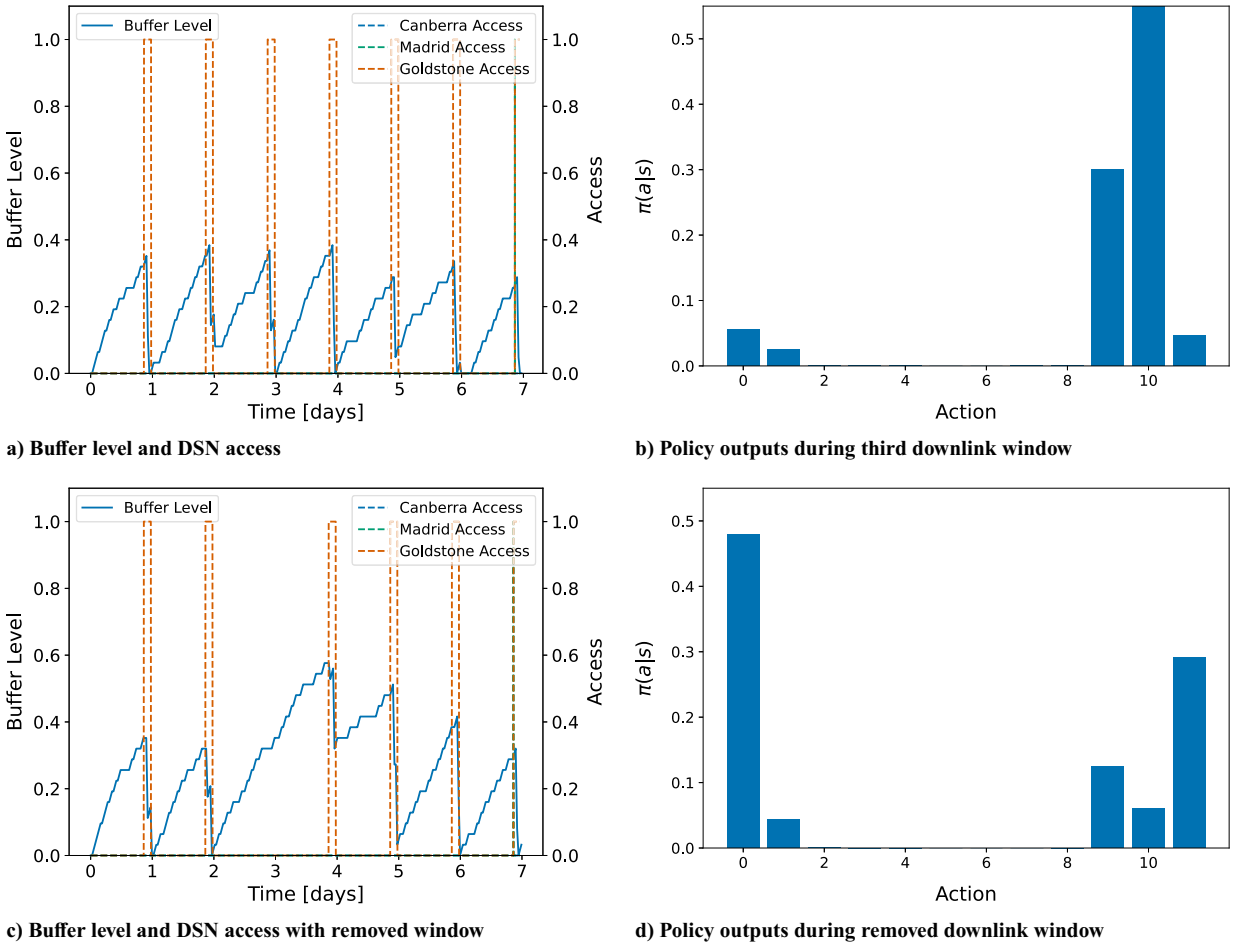
D. DSN Outage Experiment

A large benefit of using reinforcement learning for planning and scheduling is that it yields closed-loop planning solutions as opposed to open-loop solutions, which allows the decision-making agent to respond to opportunistic science events or ground station outages. While uncommon, DSN outages can occur and place mission timelines in jeopardy. On October 11th, 2019, a DSN outage occurred at the Madrid station before a "late update" (an update to the spacecraft's trajectory) for the OSIRIS-REx mission [61]. Engineers had to scramble and compress this update within a 4 h window. In addition to trajectory and navigation updates, missed downlink windows can also have implications on future collection and downlink of science data. In this experiment, a DSN outage is simulated by simply removing the third downlink opportunity. The nominal buffer level and access times, without this removal, are shown in Fig. 12a. The average policy outputs during the third downlink window are shown in Fig. 12b. The highest probability action is action 10, which is downlink.

The buffer level and policy outputs during the removed downlink window are shown in Figs. 12c and 12d, respectively. During the removed downlink window, the downlink state is replaced with zero, meaning that the downlink window is 24 h away. The decision-making agent responds by continuing the mapping and imaging campaign, requiring no input from the ground other than notification that the window has been removed. Theoretically, no notification from the ground is required at all if the spacecraft does not receive confirmation that a link has been achieved in the downlink mode. The downlink action goes from the highest probability action to one of the lowest probability actions. This is a major benefit of using reinforcement learning for planning and scheduling, as the decision-making agent can respond to changes in the environment with little or no human intervention. No computationally expensive replanning efforts on board the spacecraft are required either. The entire trajectory of actions does not need to be re-optimized using an integer program. Furthermore, this also highlights the utility of using the selected representation of the downlink state. Past work for Earth-observing satellite scheduling [54,55] relies on the spacecraft position to help the agent understand when downlink opportunities occur. However, relying on spacecraft position alone for ground station access doesn't easily allow for the use of temporal access constraints or the removal of a downlink window. Therefore, a temporal representation for upcoming downlink windows should be utilized.

E. Navigation Mode Policy Evaluation

In Table 8, the policy trained with the dedicated navigation mode is benchmarked over $N = 20$, 7-day planning horizons. When compared to Table 7, it is evident that PPO trained with the navigation mode is capable of producing policies equivalent (in terms of average reward) to those trained without the navigation mode and associated state uncertainty. The main differences regarding the trained policies include a slight increase in ΔV, a slight increase in the average number of collected and downlinked images, and a slight decrease in the amount of data collected and downlinked regarding the map associated with a 10 AM LST for



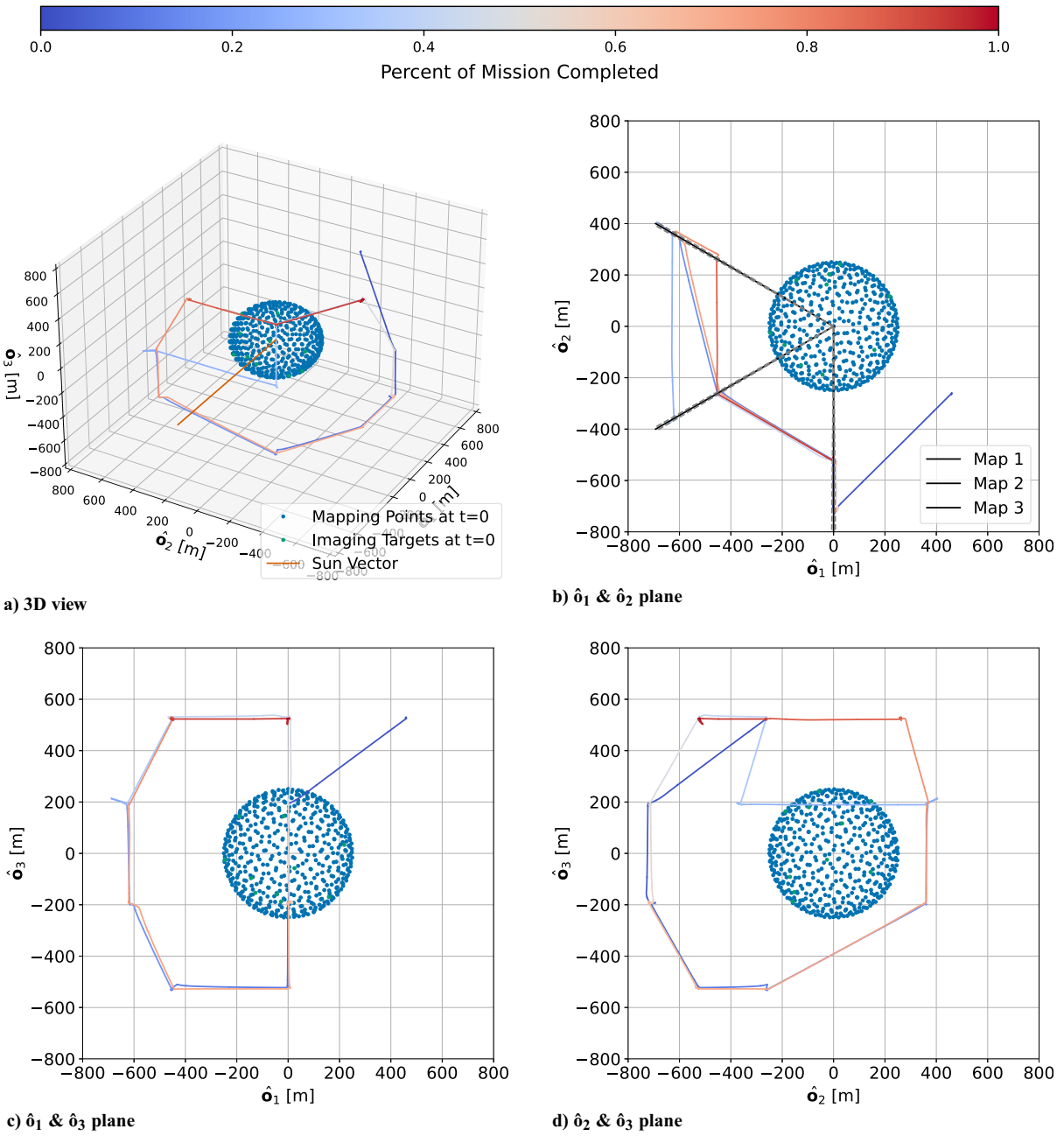
a) Buffer level and DSN access

b) Policy outputs during third downlink window

c) Buffer level and DSN access with removed window

d) Policy outputs during removed downlink window

Fig. 12 Buffer level and policy outputs during DSN schedule. Action 10 is the downlink action.



a) 3D view

b) $\hat{\mathbf{o}}_1$ & $\hat{\mathbf{o}}_2$ plane

c) $\hat{\mathbf{o}}_1$ & $\hat{\mathbf{o}}_3$ plane

d) $\hat{\mathbf{o}}_2$ & $\hat{\mathbf{o}}_3$ plane

Fig. 13 Hill frame trajectory with the navigation mode included in the action space.

Table 8 Benchmarking the policy trained for the MDP with the dedicated navigation mode: 7-day-long planning horizon

Metric	Value
Average reward	0.91 ± 0.02
Average ΔV	11.9 ± 0.5
Collected images	9.7 ± 0.3
Downlinked images	9.4 ± 0.4
Collected map (6 PM LST)	455 ± 12
Collected map (2 PM LST)	452 ± 20
Collected map (10 AM LST)	400 ± 17
Downlinked map (6 PM LST)	441 ± 20
Downlinked map (2 PM LST)	448 ± 20
Downlinked map (10 AM LST)	388 ± 23

the policies trained with the optional navigation mode. This is an impressive result, demonstrating that PPO can not only manage spacecraft resources, but state uncertainty as well, simply by monitoring the diagonals of the error covariance matrix. The 1 deg azimuth angle requirement for the mapping modes is very tight, and it only takes a few decision-making intervals for the estimation error to cause the control solution to drift out of this band. When this happens, the map points are no longer collected, and the agent ceases to receive rewards for mapping. In order to keep receiving rewards, the decision-making agent has learned to utilize navigation updates in between science modes to effectively collect the mapping data. If autonomous guidance and relative navigation capabilities mature enough for adoption, reinforcement learning is a viable method for on-board planning and scheduling small body science missions. Future work should investigate how state-of-the-art autonomous guidance and relative navigation methods impact the performance of the decision-making agent

and how the problem formulation may need to be adjusted to account for these impacts.

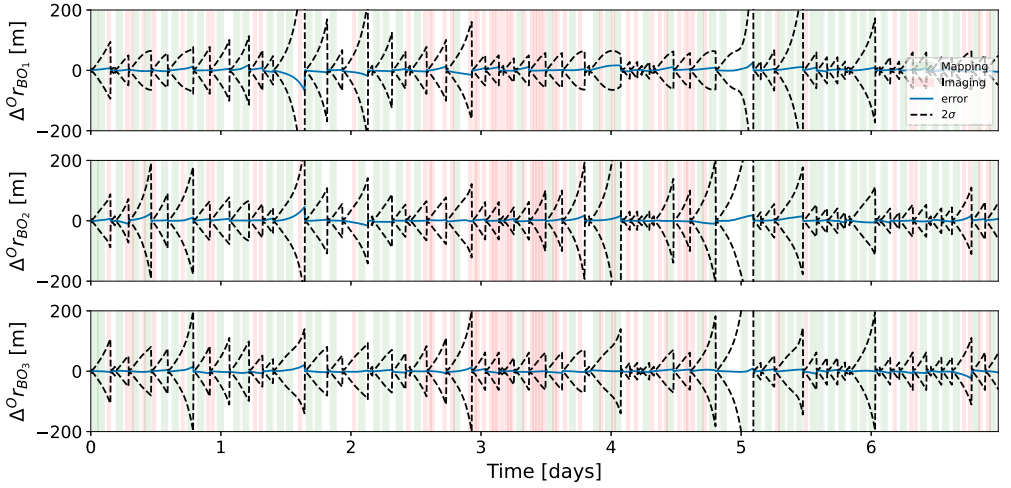
To provide a more complete understanding of the behavior of the trained decision-making agent, data regarding the actions, spacecraft position and velocity, and spacecraft resources is collected for a single run of the policy. The trajectory of the spacecraft in the sun–asteroid Hill frame is displayed in Fig. 13. The decision-making agent learns to make multiple passes through the mapping waypoints. This is an unintuitive result, but the decision-making agent has learned to map and then move to the next waypoint instead of waiting for the other side of the body to become visible. The position and velocity state error, along with the 2σ covariance bounds, are displayed in Fig. 14. The decision-making agent periodically updates its state estimate with new measurements, reducing the state error covariance so that mapping can be conducted. Imaging modes are sprinkled throughout the planning horizon. The data buffer level, stored power, and ΔV are presented in Fig. 15. The policy takes advantage of almost every downlink opportunity, keeping the data buffer well within the limits. The stored power also always stays above 20 W·h. Finally, the trained policy performs roughly 19 maneuvers over the course of the mission. The decision-making agent attempts to make more maneuvers, but these maneuvers are not actually performed by the spacecraft because the required amount of time before the next maneuver can be taken has not passed.

F. Management of the State Estimate

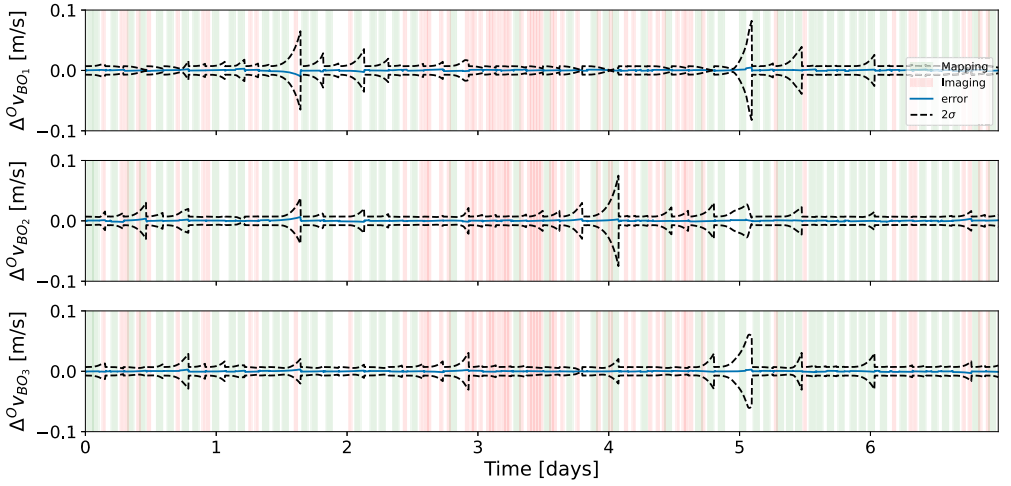
Based on the performance of the trained agents, it appears that the decision-making agent has learned to manage its state estimate. However, it is unclear if the agent has learned to assign a fixed probability to the navigation update mode or if the selection of the navigation update mode is dependent on the value of the state-error covariance. To determine this, the policy is run in the environment, the covariance matrix is collected, and the policy distribution is evaluated for different values of the diagonal of the covariance matrix. These results are presented in Fig. 16. For $10^1 \leq \|\mathrm{diag}(P)\| < 10^2$, when the covariance and state error are the lowest, the lowest probability action (outside of the maneuvers) is the navigation update. As the state error covariance grows, the navigation update becomes the most likely action, as evidenced by Fig. 16d, where $\pi(\text{nav update}|s) \approx 0.55$. The decision-making agent has learned to manage its state estimate based on the observations provided autonomously, monitoring the state-error covariance to determine when a measurement mode should be taken.

V. Conclusions

This work explores the application of DRL to a small body science operations problem. A representative problem is presented and formulated as an MDP. A high-fidelity simulation of the problem is developed, and PPO is applied to the problem. A hyperparameter search over the PPO hyperparameters and reward

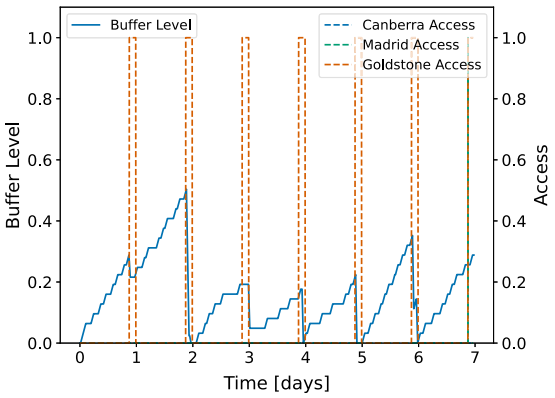


a) Position error and associated 2σ covariance bounds

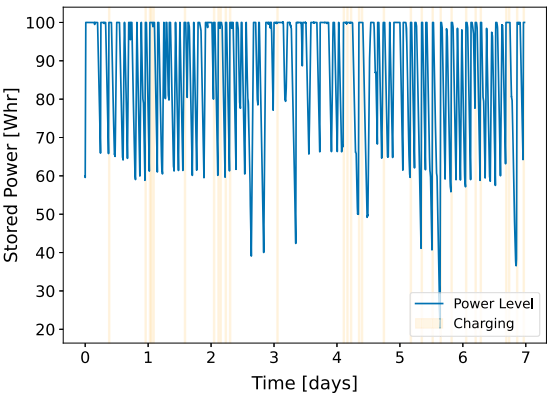


b) Velocity error and associated 2σ covariance bounds

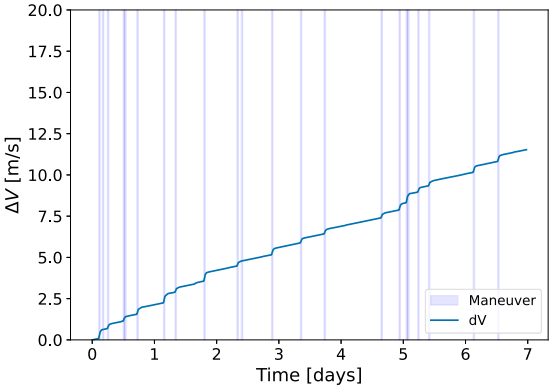
Fig. 14 EKF position and velocity error over 7 days of operations utilizing the optional navigation mode. Mapping modes are shown in green. Imaging modes are shown in red.



a) Data buffer level and DSN access

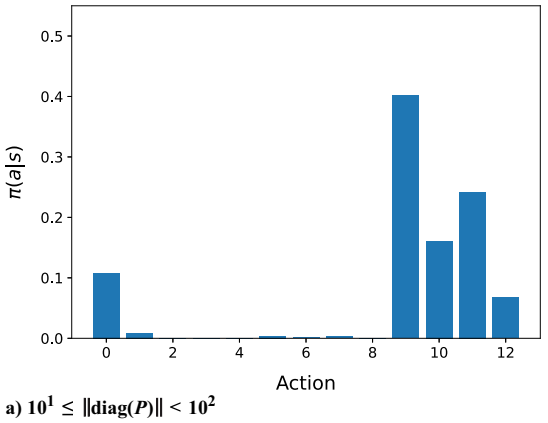


b) Stored power with charging modes included

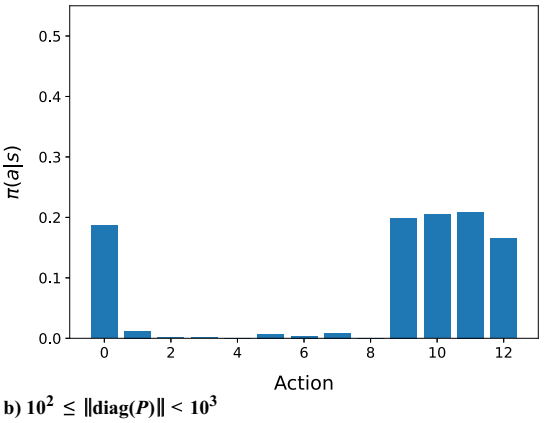


c) Total ΔV consumed with maneuver modes included

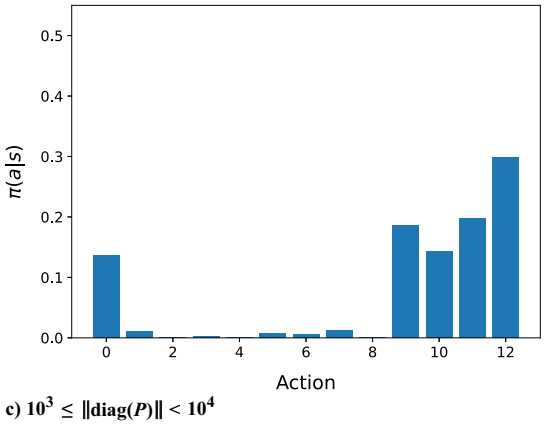
Fig. 15 Spacecraft resources over time using optional navigation mode.



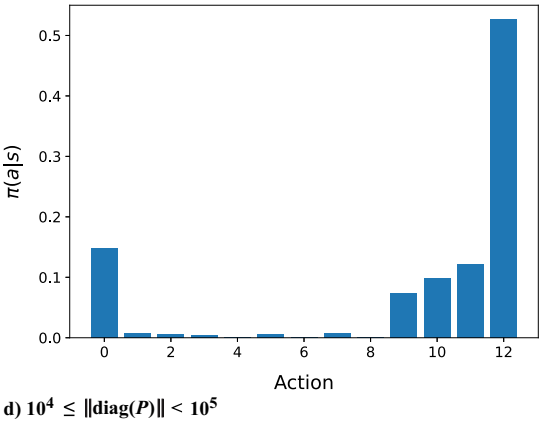
a) $10^1 \leq \|\text{diag}(P)\| < 10^2$



b) $10^2 \leq \|\text{diag}(P)\| < 10^3$



c) $10^3 \leq \|\text{diag}(P)\| < 10^4$



d) $10^4 \leq \|\text{diag}(P)\| < 10^5$

Fig. 16 Dependency of stochastic policy outputs on the magnitude of the diagonal of the state covariance matrix. Action 12 is the navigation mode.

function components is performed. Small batch sizes and larger networks are shown to produce the best-performing policies. In terms of the reward function experiments, equal weighting between surface target imaging and mapping produces the best policies. The performance of the trained policies is evaluated using direct observations of the state, noisy observations of the state, and filtered observations produced by an EKF. The decision-making agent is shown to be robust to noise in the state observations, even when only trained using direct observations of the state. The decision-making agent is also shown to be capable of responding to changes in the environment, such as a DSN outage, with no human intervention outside of notification that the window is no longer available. Finally, the decision-making agent is shown to be capable of managing its state estimate when an additional navigation mode is added to the action space. The decision-making agent is capable of producing policies that are equivalent to those trained without the navigation mode.

The aforementioned experiments demonstrate that reinforcement learning is a viable method for planning and scheduling small body science operations, capable of managing spacecraft resources, maneuvers, and navigation updates while achieving the science objectives of the formulated mission. These results are promising, and future work should investigate using state-of-the-art autonomous guidance and relative navigation methods within this planning and scheduling framework for a real mission study. Furthermore, problem formulations that either account for all mission phases under the umbrella of a single MDP or multiple phases as individual MDPs should be studied.

Acknowledgments

This work is supported by the NASA Space Technology Research Graduate Opportunity under Grant 80NSSC20K1162. This work utilized the Alpine high-performance computing resource at the University of Colorado Boulder. Alpine is jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, Colorado State University, and the National Science Foundation (award 2201538).

References

[1] "Audit of NASA's Deep Space Network," Rept. IG-23-016, NASA Office of Inspector General, Washington, D.C., 2023, https://oig.nasa.gov/docs/IG-23-016.pdf.

[2] Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C., "Remote Agent: To Boldly Go where No AI System Has Gone before," *Artificial Intelligence*, Vol. 103, Nos. 1–2, 1998, pp. 5–47. https://doi.org/10.1016/S0004-3702(98)00068-X

[3] Bernard, D., Dorais, G., Fry, C., Gamble, E. B., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Nayak, P., Pell, B., et al., "Design of the Remote Agent Experiment for Spacecraft Autonomy," *1998 IEEE Aerospace Conference Proceedings*, Vol. 2, Inst. of Electrical and Electronics Engineers, New York, 1998, pp. 281–259. https://doi.org/10.1109/AERO.1998.687914

[4] Bernard, D., Dorais, G., Gamble, E., Kanefsky, B., Kurien, J., Man, G. K., Millar, W., Muscettola, N., Nayak, P., Rajan, K., et al., "Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment," *Proceedings of the AIAA*, AIAA, Reston, VA, 1999, pp. 28–30; also AIAA Paper 1999-4512.

[5] Fukunaga, A., Rabideau, G., Chien, S., and Yan, D., "Towards an Application Framework for Automated Planning and Scheduling," *1997 IEEE Aerospace Conference*, Inst. of Electrical and Electronics Engineers, New York, 1997, pp. 375–386. https://doi.org/10.1109/AERO.1997.574426

[6] Knight, S., Rabideau, G., Chien, S., Engelhardt, B., and Sherwood, R., "CASPER: Space Exploration Through Continuous Planning," *IEEE Intelligent Systems*, Vol. 16, No. 5, 2001, pp. 70–75. https://doi.org/10.1109/MIS.2001.956084

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

[8] Chien, S., Tran, D., Rabideau, G., Schaffer, S., Mandl, D., and Frye, S., "Timeline-Based Space Operations Scheduling with External Constraints," *Twentieth International Conference on Automated Planning and Scheduling*, Vol. 20, 2010, pp. 34–41. https://doi.org/10.1609/icaps.v20i1.13410

[9] Chien, S., Doubleday, J., Thompson, D. R., Wagstaff, K., Bellardo, J., Francis, C., Baumgarten, E., Williams, A., Yee, E., Stanton, E., et al., "Onboard Autonomy on the Intelligent Payload EXperiment (IPEX) CubeSat Mission," *Journal of Aerospace Information Systems (JAIS)*, Vol. 14, No. 6, 2016, pp. 307–315. https://doi.org/10.2514/1.I010386

[10] Verma, V., Gaines, D., Rabideau, G., Schaffer, S., and Joshi, R., "Autonomous Science Restart for the Planned Europa Mission with Lightweight Planning and Execution," *International Workshop on Planning and Scheduling for Space (IWPSS 2017)*, 2017, https://ai.jpl.nasa.gov/public/papers/verma_iwpss2017_autonomous.pdf.

[11] Gaines, D., Chien, S., Rabideau, G., Kuhn, S., Wong, V., Yelamanchili, A., Towey, S., Agrawal, J., Chi, W., Connell, A., et al., "Onboard Planning for the Mars 2020 Perseverance Rover," *16th Symposium on Advanced Space Technologies in Robotics and Automation*, 2022, https://ai.jpl.nasa.gov/public/documents/papers/M2020-OBP-ASTRA-2022-Final.pdf.

[12] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 2018.

[13] Globus, A., Crawford, J., Lohn, J., Morris, R., and Clancy, D., "Scheduling Earth Observing Fleets Using Evolutionary Algorithms: Problem Description and Approach," *International Conference on Space Mission Challenges for Information Technology*, 2003.

[14] 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. 1–16. https://doi.org/10.1016/j.cor.2014.11.004

[15] Cho, D.-H., Kim, J.-H., Choi, H.-L., and Ahn, J., "Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation," *Journal of Aerospace Information Systems*, Vol. 15, No. 11, 2018, pp. 611–626. https://doi.org/10.2514/1.I010620

[16] Kim, J., Ahn, J., Choi, H.-L., and Cho, D.-H., "Task Scheduling of Agile Satellites with Transition Time and Stereoscopic Imaging Constraints," *Journal of Aerospace Information Systems*, Vol. 17, No. 6, 2020, pp. 285–293. https://doi.org/10.2514/1.I010775

[17] Chen, X., Reinelt, G., Dai, G., and Spitz, A., "A Mixed Integer Linear Programming Model for Multi-Satellite Scheduling," *European Journal of Operational Research*, Vol. 275, No. 2, 2019, pp. 694–707. https://doi.org/10.1016/j.ejor.2018.11.058

[18] Monmousseau, P., "Scheduling of a Constellation of Satellites: Creating a Mixed-Integer Linear Model," *Journal of Optimization Theory and Applications*, Vol. 191, Nos. 2–3, 2021, pp. 846–873. https://doi.org/10.1007/s10957-021-01875-2

[19] Kim, J., and Ahn, J., "Integrated Framework for Task Scheduling and Attitude Control of Multiple Agile Satellites," *Journal of Aerospace Information Systems*, Vol. 18, No. 8, 2021, pp. 539–552. https://doi.org/10.2514/1.I010910

[20] Song, Y., Romero, A., Müller, M., Koltun, V., and Scaramuzza, D., "Reaching the Limit in Autonomous Racing: Optimal Control Versus Reinforcement Learning," *Science Robotics*, Vol. 8, No. 82, 2023, Paper eadg1462. https://doi.org/10.1126/scirobotics.adg1462

[21] Chan, D. M., and Agha-mohammadi, A., "Autonomous Imaging and Mapping of Small Bodies Using Deep Reinforcement Learning," *2019 IEEE Aerospace Conference*, Inst. of Electrical and Electronics Engineers, New York, 2019, pp. 1–12. https://doi.org/10.1109/AERO.2019.8742147

[22] Piccinin, M., Lunghi, P., and Lavagna, M., "Deep Reinforcement Learning-Based Policy for Autonomous Imaging Planning of Small Celestial Bodies Mapping," *Aerospace Science and Technology*, Vol. 120, Jan. 2022, Paper 107224. https://doi.org/10.1016/j.ast.2021.107224

[23] Takahashi, S., and Scheeres, D. J., "Autonomous Reconnaissance Trajectory Guidance at Small Near-Earth Asteroids via Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, Vol. 46, No. 7, 2023, pp. 1–18. https://doi.org/10.2514/1.G007043

[24] Furfaro, R., Scorsoglio, A., Linares, R., and Massari, M., "Adaptive Generalized ZEM-ZEV Feedback Guidance for Planetary Landing via a Deep Reinforcement Learning Approach," *Acta Astronautica*, Vol. 171, June 2020, pp. 156–171. https://doi.org/10.1016/j.actaastro.2020.02.051

[25] Gaudet, B., Linares, R., and Furfaro, R., "Adaptive Guidance and Integrated Navigation with Reinforcement Meta-Learning," *Acta*

Astronautica, Vol. 169, April 2020, pp. 180–190. https://doi.org/10.1016/j.actaastro.2020.01.007

[26] Gaudet, B., Linares, R., and Furfaro, R., "Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Landing," *Advances in Space Research*, Vol. 65, No. 7, 2020, pp. 1723–1741. https://doi.org/10.1016/j.asr.2019.12.030

[27] Scorsoglio, A., D'Ambrosio, A., Ghilardi, L., Gaudet, B., Curti, F., and Furfaro, R., "Image-Based Deep Reinforcement Meta-Learning for Autonomous Lunar Landing," *Journal of Spacecraft and Rockets*, Vol. 59, No. 1, 2022, pp. 153–165. https://doi.org/10.2514/1.A35072

[28] Gaudet, B., Linares, R., and Furfaro, R., "Terminal Adaptive Guidance via Reinforcement Meta-Learning: Applications to Autonomous Asteroid Close-Proximity Operations," *Acta Astronautica*, Vol. 171, June 2020, pp. 1–13. https://doi.org/10.1016/j.actaastro.2020.02.036

[29] Gaudet, B., Linares, R., and Furfaro, R., "Six Degree-of-Freedom Body-Fixed Hovering over Unmapped Asteroids via LIDAR Altimetry and Reinforcement Meta-Learning," *Acta Astronautica*, Vol. 172, July 2020, pp. 90–99. https://doi.org/10.1016/j.actaastro.2020.03.026

[30] Takahashi, S., and Scheeres, D., "Autonomous Proximity Operations at Small Neas," *33rd International Symposium on Space Technology and Science (ISTS)*, Vol. 2, 2022.

[31] Federici, L., Scorsoglio, A., Ghilardi, L., D'Ambrosio, A., Benedikter, B., Zavoli, A., and Furfaro, R., "Image-Based Meta-Reinforcement Learning for Autonomous Guidance of an Asteroid Impactor," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 11, 2022, pp. 2013–2028. https://doi.org/10.2514/1.G006832

[32] 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

[33] Hovell, K., and Ulrich, S., "Laboratory Experimentation of Spacecraft Robotic Capture Using Deep-Reinforcement-Learning–Based Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 11, 2022, pp. 2138–2146. https://doi.org/10.2514/1.G006656

[34] Federici, L., Benedikter, B., and Zavoli, A., "Deep Learning Techniques for Autonomous Spacecraft Guidance During Proximity Operations," *Journal of Spacecraft and Rockets*, Vol. 58, No. 6, 2021, pp. 1774–1785. https://doi.org/10.2514/1.A35076

[35] Oestreich, C. E., Linares, R., and Gondhalekar, R., "Autonomous Six-Degree-of-Freedom Spacecraft Docking with Rotating Targets via Reinforcement Learning," *Journal of Aerospace Information Systems*, Vol. 18, No. 7, 2021, pp. 417–428. https://doi.org/10.2514/1.I010914

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

[37] Dunlap, K., Mote, M., Delsing, K., and Hobbs, K. L., "Run Time Assured Reinforcement Learning for Safe Satellite Docking," *Journal of Aerospace Information Systems*, Vol. 20, No. 1, 2023, pp. 25–36. https://doi.org/10.2514/1.I011126

[38] Pesce, V., Agha-mohammadi, A.-A., and Lavagna, M., "Autonomous Navigation & Mapping of Small Bodies," *IEEE Aerospace Conference*, Inst. of Electrical and Electronics Engineers, New York, 2018, pp. 1–10.

[39] Nesnas, I. A., Hockman, B. J., Bandopadhyay, S., Morrell, B. J., Lubey, D. P., Villa, J., Bayard, D. S., Osmundson, A., Jarvis, B., Bersani, M., et al., "Autonomous Exploration of Small Bodies Toward Greater Autonomy for Deep Space Missions," *Frontiers in Robotics and AI*, Vol. 8, Nov. 2021, Paper 650885. https://doi.org/10.3389/frobt.2021.650885

[40] Ashman, M., Barthélémy, M., Almeida, M., Altobelli, N., Sitjà, M. C., Beteta, J. J. G., Geiger, B., Grieger, B., Heather, D., Hoofs, R., et al., "Rosetta Science Operations in Support of the Philae Mission," *Acta Astronautica*, Vol. 125, Aug. 2016, pp. 41–64. https://doi.org/10.1016/j.actaastro.2016.02.007

[41] Lauretta, D., Balram-Knutson, S., Beshore, E., Boynton, W., d'Aubigny, C. D., DellaGiustina, D., Enos, H., Golish, D., Hergenrother, C., Howell, E., et al., "OSIRIS-REx: Sample Return from Asteroid (101955) Bennu," *Space Science Reviews*, Vol. 212, No. 1, 2017, pp. 925–984. https://doi.org/10.1007/s11214-017-0405-1

[42] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *7th International Conference on Astrodynamics Tools and Techniques (ICATT)*, DLR Oberpfaffenhofen, Germany, 2018.

[43] Scheeres, D. J., "Orbit Mechanics About Asteroids and Comets," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 3, 2012, pp. 987–997. https://doi.org/10.2514/1.57247

[44] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, AIAA Education Series, 4th ed., AIAA, Reston, VA, 2018. https://doi.org/10.2514/4.105210

[45] Simon, D., *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, Wiley, Hoboken, NJ, 2006.

[46] Bhaskaran, S., Desai, S., Dumont, P., Kennedy, B., Null, G., Owen, W., Jr., Riedel, S., Synnott, S., and Werner, R., "Orbit Determination Performance Evaluation of the Deep Space 1 Autonomous Navigation System," 1998.

[47] Gaskell, R., Barnouin-Jha, O., Scheeres, D. J., Konopliv, A., Mukai, T., Abe, S., Saito, J., Ishiguro, M., Kubota, T., Hashimoto, T., et al., "Characterizing and Navigating Small Bodies with Imaging Data," *Meteoritics & Planetary Science*, Vol. 43, No. 6, 2008, pp. 1049–1061. https://doi.org/10.1111/j.1945-5100.2008.tb00692.x

[48] Konopliv, A., Asmar, S., Park, R., Bills, B., Centinello, F., Chamberlin, A., Ermakov, A., Gaskell, R., Rambaux, N., Raymond, C., et al., "The Vesta Gravity Field, Spin Pole and Rotation Period, Landmark Positions, and Ephemeris from the Dawn Tracking and Optical Data," *Icarus*, Vol. 240, Sept. 2014, pp. 103–117. https://doi.org/10.1016/j.icarus.2013.09.005

[49] Williams, B., Antreasian, P., Carranza, E., Jackman, C., Leonard, J., Nelson, D., Page, B., Stanbridge, D., Wibben, D., Williams, K., et al., "OSIRIS-REx Flight Dynamics and Navigation Design," *Space Science Reviews*, Vol. 214, No. 4, 2018, pp. 1–43. https://doi.org/10.1007/s11214-018-0501-x

[50] Dietrich, A., and McMahon, J. W., "Orbit Determination Using Flash Lidar Around Small Bodies," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 3, 2017, pp. 650–665. https://doi.org/10.2514/1.G000615

[51] Sanchez, J. C., and Schaub, H., "Small Body Navigation and Gravity Estimation Using Kalman Filter and Least-Squares Fitting," *AAS Spaceflight Mechanics Meeting*, AAS Paper 23-126, 2023.

[52] Schutz, B., Tapley, B., and Born, G. H., *Statistical Orbit Determination*, Elsevier, New York, 2004.

[53] Kochenderfer, M. J., "Sequential Problems," *Decision Making Under Uncertainty: Theory and Application*, Massachusetts Inst. of Technology, Cambridge, MA, 2015, pp. 102–103.

[54] Herrmann, A. P., and Schaub, H., "Monte Carlo Tree Search Methods for the Earth-Observing Satellite Scheduling Problem," *Journal of Aerospace Information Systems*, Vol. 19, No. 1, 2021, pp. 1–13. https://doi.org/10.2514/1.I010992

[55] Herrmann, A., and Schaub, H., "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, Inst. of Electrical and Electronics Engineers, New York, 2023, pp. 1–13. https://doi.org/10.1109/TAES.2023.3251307

[56] Lauretta, D., Bartels, A., Barucci, M., Bierhaus, E., Binzel, R., Bottke, W., Campins, H., Chesley, S., Clark, B., Clark, B., et al., "The OSIRIS-REx Target Asteroid (101955) Bennu: Constraints on Its Physical, Geological, and Dynamical Nature from Astronomical Observations," *Meteoritics & Planetary Science*, Vol. 50, No. 4, 2015, pp. 834–849. https://doi.org/10.1111/maps.12353

[57] Scheeres, D., McMahon, J., French, A., Brack, D., Chesley, S., Farnocchia, D., Takahashi, Y., Leonard, J., Geeraert, J., Page, B., et al., "The Dynamic Geophysical Environment of (101955) Bennu Based on OSIRIS-REx Measurements," *Nature Astronomy*, Vol. 3, No. 4, 2019, pp. 352–361. https://doi.org/10.1038/s41550-019-0721-3

[58] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-Level Control Through Deep Reinforcement Learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533. https://doi.org/10.1038/nature14236

[59] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous Methods for Deep Reinforcement Learning," *International Conference on Machine Learning*, PMLR, New York, 2016, pp. 1928–1937.

[60] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," 2017. https://doi.org/10.48550/arXiv.1707.06347

[61] Shekhtman, L., "OSIRIS-REx Engineers Pull Off a Daring Rescue of a Monumental Asteroid Mission," 2019, https://www.nasa.gov/feature/goddard/2019/osiris-rex-engineers-pull-off-a-daring-rescue-of-a-monumental-asteroid-mission [retrieved 10 Sept. 2019].

T. McLain
Associate Editor