(Preprint) AAS 25-285

REINFORCEMENT LEARNING FOR SPACECRAFT NAVIGATION & ENVIRONMENT CHARACTERIZATION IN THE PLANAR-RESTRICTED TWO-BODY PROBLEM

Kenneth M. Getzandanner*† and John R. Martin‡

As science, exploration, and commercial space missions become increasingly complex, so does the need for efficient, autonomous, and integrated spacecraft navigation and operations techniques. Key operational functions, including data collection and transmission, environment characterization, systems constraints, human factors, and navigation, often are intertwined and conflicted. Deep Reinforcement Learning (DRL) offers a framework for addressing integrated spacecraft navigation and planning in an uncertain dynamical environment. The goal of this study is to evaluate the utility of DRL for integrated spacecraft navigation and planning. This is achieved by developing a simple environmental characterization training environment in the Planar-Restricted 2-Body Problem (PR2BP), establishing benchmarks and heuristic baselines, and designing a previously unstudied Markov Decision Process (MDP) formulation. This MDP formulation enables the spacecraft DRL agents to appropriately balance navigation and actuation capabilities. The resulting DRL-derived policy exceeds a random or untrained policy and meets or exceeds the level of performance of a heuristic without actuation. In the process, valuable intuition is gained about the problem with insight into how DRL methods could scale to increasingly more realistic scenarios, including network design and training architectures, efficient state space representations, and methods for encouraging exploration in a parametric action space, among others.

INTRODUCTION

During mission planning and execution, spacecraft operators must balance data collection and downlink, systems constraints, human factors, and navigation. As missions become increasingly complex and ambitious, these factors become more intricately entwined and conflicted. For example, a spacecraft's position must be known accurately in order to point to and image a target. Large position errors may cause missed observations or require additional scanning that increases operations complexity and data volume. Some observations require imaging from specific relative geometries, which add orbit control and timing considerations. Adjusting the orbit may allow for optimal observability of environmental parameters and/or enable more efficient sensor coverage, but maneuver execution error adds uncertainty to the current state, impacting both characterization and coverage objectives.

For Earth-orbiting spacecraft, onboard, autonomous navigation systems that use the Global Positioning System (GPS) and other measurements simplify operations planning. However, resource constraints may limit the ability to perform navigation and other operations tasks simultaneously.

*Flight Dynamics Lead, NASA/GSFC Code 595, 8800 Greenbelt Rd, Greenbelt, MD 20771.
†PhD Student, Department of Aerospace Engineering, University of Maryland, College Park, MD 20742.
‡Assistant Professor, Department of Aerospace Engineering, University of Maryland, College Park, MD 20742.

Spacecraft that rely on ground-based radiometrics must consider tracking and ephemeris update schedules. Furthermore, interplanetary and small-body navigation relies on Earth-based radiometric tracking and, in some cases, *in situ* observations. Generally, this type of tracking conflicts with science observations and data storage, e.g., the spacecraft must slew an antenna toward Earth, capture Optical Navigation (OpNav) images, or collect altimetry measurements. Moreover, environment characterization is tightly coupled with navigation and operations. Uncertainty in geophysical parameters, such as gravity field, spin state, atmosphere, etc., of uncharacterized bodies affects navigation performance and is often scientifically-valuable independent of navigation. The small-body environment presents a particularly complex and stressing case, as experienced by the Origins, Spectral Interpretation, Resource Identification, Security–Regolith Explorer (OSIRIS-REx) spacecraft at the near-Earth asteroid 101955 Bennu[1,2] and by other missions.

Deep Reinforcement Learning (DRL) is a state-of-the-art (SOA) technique for autonomous spacecraft operations planning and autonomy. Examples in the literature include applications of DRL for the Agile Earth Observing Satellite Scheduling Problem (AEOSSP),[3,4] Mars station-keeping and Low-Earth Orbit (LEO) observations,[5] several aspects of spacecraft control,[6] Autonomous Rendezvous & Docking (AR&D),[7] and operations about a small celestial body,[8–11] among others. To date, research has largely simplified or ignored realistic navigation considerations and environment characterization for the spacecraft scheduling and autonomy problem. The goal of the current study is to address this gap in the literature by developing a simple environmental characterization training environment in the Planar-Restricted 2-Body Problem (PR2BP), establishing benchmarks and heuristic baselines, and designing a previously unstudied Markov Decision Process (MDP) formulation that will enable the spacecraft DRL agents to balance navigation and actuation capabilities appropriately. Starting with a lower-fidelity model provides valuable insights into the structure of the problem and potential solutions that can be applied to future iterations with increasing levels of fidelity.

PROBLEM DEFINITION

Consider a spacecraft in a planar, 2D orbit about an arbitrary, planar central body. The central body rotates at a fixed rate about an axis perpendicular to the spacecraft's orbital plane. The only force acting on the spacecraft is the two-body gravitational attraction of the central body, such that the PR2BP governs dynamics. The spacecraft's initial position and velocity vectors, $\mathbf{r}(t_0), \mathbf{v}(t_0)$, the central body's gravitational parameter, μ, initial rotation offset, θ_0, and constant rotation rate, ω, are random variables with uncorrelated Gaussian priors. The system state vector, $\mathbf{X}$, and covariance matrix, P, are defined as:

$$\mathbf{X} = \left[\mathbf{r}^T, \mathbf{v}^T, \mu, \theta, \omega, \right]^T \tag{1}$$

$$P = \mathbb{E}\left[(\hat{\mathbf{X}} - \mathbf{X})(\hat{\mathbf{X}} - \mathbf{X})^T\right] \tag{2}$$

where $\hat{\mathbf{X}} - \mathbf{X}$ is the difference between the estimated and true-state vectors. As discussed in Reference 12, the covariance matrix can be factorized into the following form for improved numerical stability:

$$P = UDU^T \tag{3}$$

where U is an upper triangular matrix with 1s along the diagonal and 0s below it, and D is a diagonal matrix.

The state is propagated at fixed time steps using numerical integration subject to PR2BP dynamics for the spacecraft and constant-rate central body rotation. The State Transition Matrix (STM) is also numerically integrated along with the state and used for linear covariance propagation. In addition, fixed process noise, S, is applied at each time update to capture the effects of un-modeled accelerations and to aid in numerical stability.* The full time update is shown in Equations (4) and (5), where F is the non-linear time update function using numerical integration.

$$\mathbf{X}(t_{k+1}), \Phi(t_k, t_{k+1}) = F(t_k, \mathbf{X}(t_k)) \tag{4}$$

$$P(t_{k+1}) = \Phi(t_k, t_{k+1}) P(t_k) \Phi(t_k, t_{k+1})^T + S(t_k) \tag{5}$$

For the UD-factorized form, the covariance time update, Equation (5), can be restructured to solve for $U(t_{k+1})$ and $D(t_{k+1})$ by using the Modified Gram-Schmidt algorithm (see Reference 12). The analytic approximation for the process noise covariance, $S(t_k)$, based on a first order Taylor Series truncation of $\Phi(t_k, t_{k+1})$, presented in Reference 12, is used here. Process noise is applied to the translational and rotational portions of the state with separate Power Spectral Densities (PSDs), q_t and q_r. $S(t_k)$ is decomposed into the factors Q_k and G_k according to Equations (6), (7), and (8) as inputs into the Modified Gram-Schmidt algorithm.

$$S(t_k) = G_k Q_k G_k^T = \begin{bmatrix} \frac{\Delta t^3}{3} q & \frac{\Delta t^2}{2} q \\ \frac{\Delta t^2}{2} q & \Delta t q \end{bmatrix} \tag{6}$$

$$Q_k = \begin{bmatrix} \frac{\Delta t^3}{12} q & 0 \\ 0 & \Delta t q \end{bmatrix} \tag{7}$$

$$G_k = \begin{bmatrix} 1 & \frac{\Delta t}{2} \\ 0 & 1 \end{bmatrix} \tag{8}$$

A configurable number of ground stations reside on the central body at specified body-fixed coordinates. At any time step, the spacecraft may receive a single instantaneous range measurement from one ground station subject to visibility constraints, i.e., the spacecraft must be above the station's local horizon. A station is considered visible if the spacecraft is above the local horizon, i.e.,

$$\frac{\mathbf{r} - \mathbf{r}_n}{||\mathbf{r} - \mathbf{r}_n||} \cdot \frac{\mathbf{r}_n}{||\mathbf{r}_n||} \geq 0 \tag{9}$$

where $\mathbf{r}$ and $\mathbf{r}_n$ are the inertial positions of the spacecraft and station, respectively.

At each time step, the spacecraft may receive a range measurement from one of the specified number of ground stations, apply an impulsive ΔV to adjust the orbit configuration, or do nothing.

*Unfortunately, the process noise matrix symbol, S, is similar to the MDP state space symbol, $\mathcal{S}$, used later in the paper. Note the use of script font for the latter to help differentiate.

If any of the measurement actions are taken, the covariance is updated by the standard Kalman update:

$$K = P(t_{k+1})H^T(HP(t_{k+1})H^T + R)^{-1} \tag{10}$$

$$P^+(t_{k+1}) = (I - KH)P(t_{k+1}) \tag{11}$$

where H is the measurement partials matrix, and R is the user-defined measurement noise. Reference 12 provides a measurement update for the UD-factorized form using a Carlson rank-one update. If the maneuver action is selected, an impulsive $\Delta\mathbf{V}$ is applied to the spacecraft's velocity component. Maneuver execution errors, defined via the Gates model,[13] are also applied to the covariance factors using a Carlson rank-one update. Figure 1 depicts a representation of the example problem formulated above.

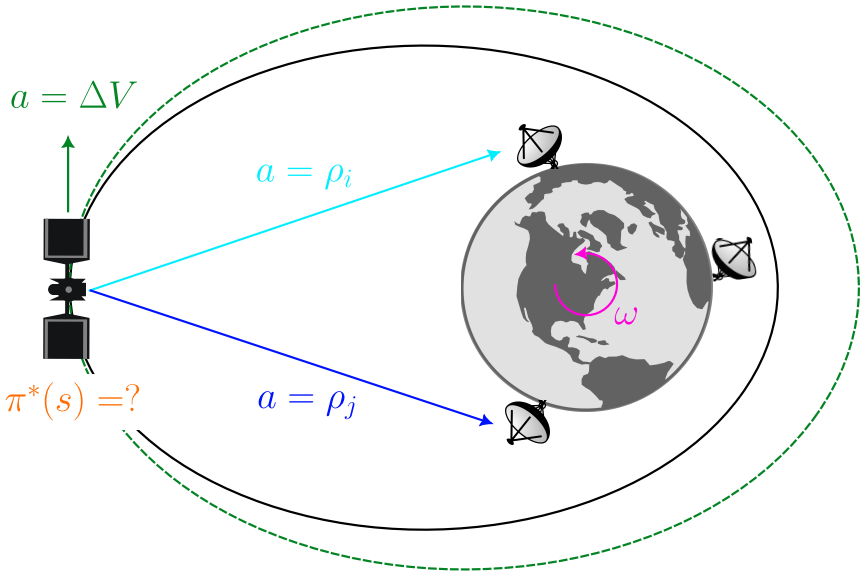


Figure 1: Representation of the planar, 2D spacecraft navigation and characterization example problem with ground station tracking and maneuver actions.

This problem can be formulated as an MDP:

$$\mathcal{P} : \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\} \tag{12}$$

At minimum, the MDP state space, $\mathcal{S}$, consists of the state vector, $\mathbf{X}$:

$$\mathcal{S} = \{\mathbf{X}\} \tag{13}$$

The current study added three other terms to the state space to aid in training:

$$\mathcal{S} = \{\mathbf{X}, \frac{1}{1+\sigma_\mu}, \alpha, t_k\} \tag{14}$$

where σ_μ is the gravitational parameter formal uncertainty, α^* is the orbit semi-major axis (SMA), and t_k is the environment time at the current step, k. For network training and stability considerations, state parameters are scaled by a factor, s. This study calculated scale factors for position and velocity by using approximate maximum orbit radius and velocity magnitude and applied them to the corresponding elements of the state vector.

The action space, $\mathcal{A}$, is defined as:

$$\mathcal{A} = \{\text{Null}, \rho_1, ..., \rho_n, \Delta V\} \tag{15}$$

where "Null" means take no action, ρ_i means process a single range measurement from station i, and ΔV means apply a control action via a specified impulsive $\Delta\mathbf{V}$. All actions are mutually exclusive, i.e., only one action can be performed per time step. For this scenario, there is no incentive to select the "Null" action and it is included largely for evaluation purposes, e.g., the policy should not select a Null action if the policy is working properly. However, there are practical circumstances where a Null action could be valid for alternate scenarios, such as if resource and/or spacecraft constraints limited the total amount of tracking.

The inclusion of a control action, ΔV, introduces both discrete and continuous actions to the action space. If the control action is selected, a continuous vector, specified as a 2D Cartesian vector, must also be supplied. Continuous action parameters that depend on the selection of a discrete action choice forms a parametric action space.[14]

The transition function, $\mathcal{T}$, is deterministic and equal to the generative function, G, where G performs the time update described in Equations (4) and (5) and, depending on the action, a, the measurement update described in Equations (10) and (11).

$$\mathcal{T}(s'|s, a) = G(s, a) \tag{16}$$

Given that the ultimate goal of this work is to incorporate realistic navigation and environment characterization in a DRL spacecraft planning framework, the reward, $\mathcal{R}$, is constructed as a function of the change in the formal gravitational parameter uncertainty, σ_μ:

$$\mathcal{R}(s(t_k), a(t_k)) = \frac{1}{1 + \sigma_\mu(t_k)} - \frac{1}{1 + \sigma_\mu(t_{k-1})} \tag{17}$$

The last component of the MDP, the discount factor, $\gamma \in [0, 1]$, is left as a user-defined value.

METHODOLOGY

The goal of this study was to generate an optimal policy, $\pi(s)$, for the MDP described in the previous section. In other words, we seek a policy that selects the spacecraft navigation or control action that maximizes the future discounted reward, thus minimizing the formal uncertainty in the gravitational parameter, σ_μ, as a proxy for overall environment characterization. The expected value, $\mathcal{V}$, at the current state, s given policy, π, is defined as:

$$\mathcal{V}^\pi(s) = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s'} \mathcal{T}(s'|s, \pi(s))\mathcal{V}^\pi(s') \tag{18}$$

*Orbit semi-major axis (SMA) is denoted as α to differentiate it from an action, a.

The policy that maximizes the value for any state is the optimal policy, $\pi^*(s)$:

$$\pi^*(s) = \arg\max_{\pi} \mathcal{V}^{\pi}(s) \; \forall \; s \in \mathcal{S} \tag{19}$$

We employ Proximal Policy Optimization (PPO)[15] as a solution method given that it is an SOA approach for solving MDPs. Specifically, we leverage the PPO implementation and associated utilities in PyTorch's TorchRL[16] library. PPO is an online, on-policy, gradient method for policy optimization that uses a surrogate objective function that "clips" the ratio between the new and old policy, i.e., large changes to the policy are discouraged between iterations:

$$\mathbf{r}(\Theta) = \frac{\pi_{\Theta}(a|s)}{\pi_{\Theta old}(a|s)} \tag{20}$$

$$L_{obj}^{CLIP}(\Theta) = \mathbb{E}\left[\min\left(\mathbf{r}(\Theta)A_{\Theta}, \text{clip}(\mathbf{r}(\Theta), 1-\epsilon, 1+\epsilon)A_{\Theta}\right)\right] \tag{21}$$

where Θ is the policy parameterization and $A_{\Theta}(s,a) = Q(s,a) - V(s)$ is the advantage function.

In all cases, policies are trained using episodes with spacecraft in a range of randomly selected orbits. Scenario termination occurs after a fixed number of time steps. Both the maximum duration, t_{max}, and the step size, Δt, are configurable parameters. Dynamics and measurement modeling leverage PyTorch `Tensor` objects contained in TorchRL `TensorDicts`. State and covariance propagation is performed with the torchode numerical integration package,[17] which accepts `Tensors` containing the initial states as inputs, as well as a pointer to a function that returns PR2BP derivatives.

Parametric Action Space

To address the challenge of balancing navigation and actuation actions in this problem, we employed a parametric action space formulation similar to Reference 14. Generally, a parametric action space consists of layered actions where the option to perform lower-level, discrete or continuous actions depends on selection of higher-level discrete actions. In this scenario, the decision to perform a maneuver is prescribed by the higher-level, discrete action space described by a Categorical distribution while the maneuver components are calculated by using lower-level, continuous Normal distributions. All other actions are purely discrete with no lower-level continuous components. Figure 2 depicts this scenario.

For training, PPO requires the natural log of the probability for each action, as well as the total entropy of the compound distribution. To calculate these quantities, first consider the general case for a discrete random variable X in the set $\mathcal{X} \in \{1, 2, ..., N\}$ and a continuous random variable Y in the set $\mathcal{Y} \in \mathbb{R}$. The probability density for Y, $q_i(y)$, is dependent on the realization of X, x_i. The joint probability density for X and Y, $p(x, y)$, is defined as:

$$p(x_i, y_j) = p(x_i)q_i(y) \tag{22}$$

The entropy of the joint distribution, $H(X, Y)$, is found by combining the Shannon entropy of X and the differential entropy of Y:

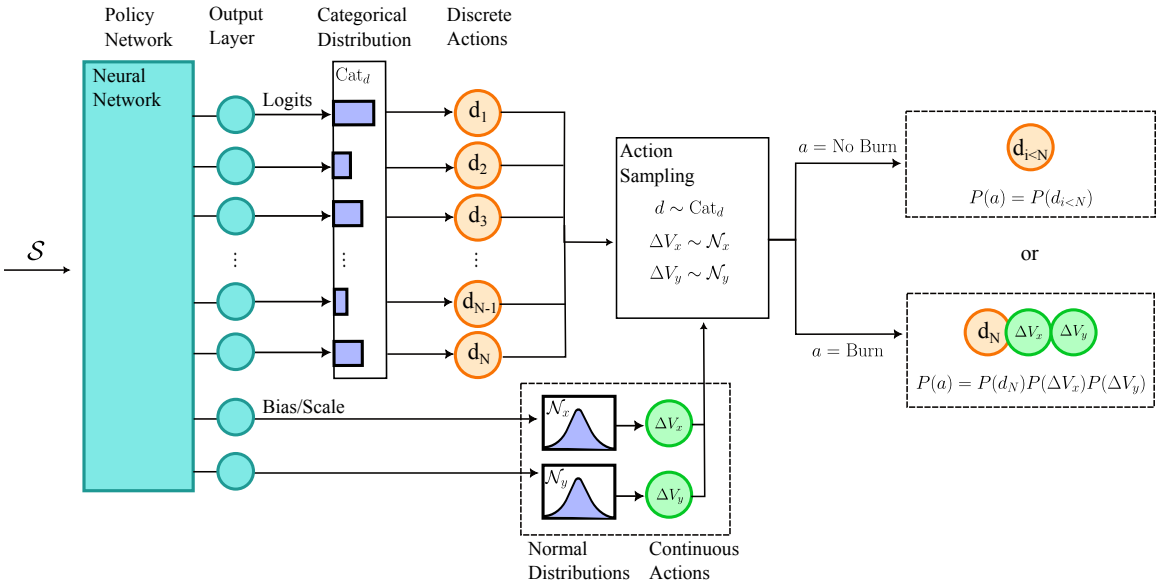


Figure 2: Representation of the parametric action space implemented in this study. Orange circles represent discrete actions, d, with probabilities described by a Categorical distribution, Cat_d. Green circles represent continuous control actions, ΔV_x and ΔV_y, with corresponding Normal distributions, $\mathcal{N}_x$ and $\mathcal{N}_y$. Control actions are applied only when the corresponding discrete action is selected during that environment step.

$$H(X,Y) = -\sum_{x\in\mathcal{X}} \int_{y\in\mathcal{Y}} p(x_i)q_i(y)\log\left(p(x_i)q_i(y)\right)dy \tag{23}$$

$$= H(X) + \sum_{i=0}^{N} p(x_i)H_i(Y) \tag{24}$$

where $H_i(Y)$ is the differential entropy of distribution $q_i(y)$:

$$H_i(Y) = -\int_{y\in\mathcal{Y}} q_i(y)\log q_i(y)dy \tag{25}$$

In our problem, Y is applicable only when X corresponds to the control action, i.e., $x = N$, where N is the total number of actions. Therefore:

$$q_i(y) = \begin{cases} q_N(y), & i = N \\ 1, & otherwise \end{cases} \tag{26}$$

and

$$H_i(Y) = \begin{cases} H_N(Y), & i = N \\ 0, & otherwise \end{cases} \tag{27}$$

Note that, with the two components of the 2D maneuver drawn from independent Normal distributions, the total continuous probability density, $q_N(y)$, is the product of the probability densities from the two distributions. Substituting Equation 27 into Equation 24 yields:

$$H(X,Y) = H(X) + p(x_N)H_N(Y) \tag{28}$$

where $H_N(Y) = \log(\sigma_x\sqrt{2\pi e}) + \log(\sigma_y\sqrt{2\pi e})$ for the two independent Normal distributions with standard deviations of σ_x and σ_y. However, as discussed in Reference 18, differential entropy does not obey all the properties of discrete Shannon entropy and can be negative. Therefore, we add a user-defined bias term, β, to the discrete entropy term in Equation 28:

$$H(X,Y) = H(X) + p(x_N)[H_N(Y) + \beta] \tag{29}$$

In this study, we chose $\beta = 5$ such that differential entropy remains positive for standard deviations $\sigma_{x,y} \gtrapprox 5$ m/s.

Scenario Parameters

The planar spacecraft orbits are initialized randomly according to the orbital element distributions listed in Table 1. Orbit propagation occurs in intervals of $\Delta t = 120.0$ s for a maximum of 24,000 s, or 200 steps. Measurements, if selected, are processed at the beginning of the interval before state and covariance propagation. A total of 100 separate spacecraft scenarios are simulated in parallel. Figure 3 presents an example of 100 randomly initialized trajectories integrated over a 24,000 s period. For this simulation, Earth is used as the central body with the corresponding values of μ_E and ω_E. Table 2 presents the parameters specifying the simulation environment. Table 3 presents the initial state uncertainties for filter parameters.

Noise for the instantaneous range measurements are set at 100 m, 1-σ. The translational and rotational PSDs, q_t and q_r, are set to 1e-20 km^2/s^3 and 1e-24 rad^2/s^3, respectively. Figure 4 presents the planet-fixed station locations.

Table 1: Random initial conditions for spacecraft orbits and central body rotation offset used for training and evaluation.

Element	Dist. (w/o Actuation)	Dist. (w/ Actuation)
Periapsis radius, r_p	$\mathcal{U}(6{,}528.1, 20{,}000)$ km	$\mathcal{U}(6{,}528.1, 10{,}000)$ km
Apoapsis radius, r_a	$\mathcal{U}(r_p, 20{,}000)$ km	$\mathcal{U}(r_p, 10{,}000)$ km
Argument of periapsis, ω_p	$\mathcal{U}(-\pi, \pi)$ rad	$\mathcal{U}(-\pi, \pi)$ rad
Mean anomaly, M	$\mathcal{U}(0, 2\pi)$ rad	$\mathcal{U}(0, 2\pi)$ rad
Central body rotation offset, θ_0	$\mathcal{U}(0, 2\pi)$ rad	$\mathcal{U}(0, 2\pi)$ rad

Network Training and Hyperparameters

Two neural networks were trained concurrently: a "policy" network that takes in a state and outputs a corresponding action, and a "value" network that takes in a state and outputs a corresponding

Table 2: Parameters specifying the simulation environment.

Parameter	Value
Time step, Δt	120.0 s
Maximum time, t_{max}	24,000.0 s
Maximum steps	200
Central body GM, μ_E	3.986e5 km^3/s^2
Central body rotation rate, ω_E	3.636e-05 rad/s
Range measurement noise, σ_ρ	100 m, 1-σ
Translational PSD, q_t	1e-20 km^2/s^3
Rotational PSD, q_r	1e-24 rad^2/s^3
Maneuver execution error	
Magnitude, proportional	0.1%
Magnitude, fixed	1 mm/s
Direction, proportional	0.01%
Direction, fixed	1 mm/s
Termination conditions	
Minimum radius, r_{min}	6,371 km
Maximum radius, r_{max}	10,0000 km

estimate of the value. Both networks consist of 4 hidden linear layers with either 16 or 64 nodes per layer, and use `LeakyReLu` activation functions between each layer.

Each training iteration begins by simultaneously simulating 100, randomly initialized trajectories for a maximum of 200 steps using the current policy network to select measurement actions. The policy network outputs `logits` corresponding to probabilities for each discrete action. The network also produces two `bias` and `scale` parameters that describe independent Normal distributions for the 2D continuous maneuver action. During training, `logits` are normalized and converted to a Categorical distribution from which discrete actions are randomly sampled. The `bias` and `scale` parameters are converted to mean and standard deviations by a biased `softplus` function and used to construct Normal distributions. If the discrete maneuver action is selected, each component of the maneuver is sampled from the corresponding Normal distribution. This simulation results in 200,000 training data points per iteration that are stored in a buffer.

If the spacecraft's radius drops below the defined planet radius or exceeds a configurable maximum radius (100,000 km in this study) as a result of one or more maneuver actions, the corresponding trajectory realization is terminated. A new trajectory realization is then initialized and advanced for the remaining steps of the original trajectory, ensuring that the total number of training data points for the iteration stays consistent. Initial experiments with an explicit penalty subtracted from the reward function for trajectory termination did not yield improved performance; therefore, this penalty was not included in the study. However, trajectory termination implicitly limits the maximum reward available for that realization.

TorchRL's built-in Generalized Advantage Estimator (GAE) module is used to calculate the ad-

Table 3: Initial uncertainties for filter parameters.

State	Uncertainty (1-σ)
Position	10 km
Velocity	1 m/s
Central body GM	0.1%
Central body initial rotation offset	0.1%
Central body rotation rate	0.1%

Table 4: Planet-fixed ground station locations.

Station #	X, km	Y, km
1	6,371.0	0.0
2	-3,185.5	5,517.4
3	-3,185.5	-5,517.4

vantage and produce target values used to train the value network.[19] The loss function is defined by the `ClipPPOLoss` module in TorchRL and includes objective, critic, and entropy terms. The buffer is randomly sampled, without replacement, in mini-batches of 4,096 data points. Losses are calculated for each mini-batch and used to update the policy and value networks with an Adam optimizer[20] and a cosine annealing scheduler.[21] Sampling and training continues until the buffer is depleted. This process is repeated for 10 training epochs per iteration. The objective, critic, and entropy loss values are saved at the end of each training epoch for evaluation. The buffer is cleared before the start of the next iteration.

This study consisted of 2,000 training iterations. Table 5 presents a summary of hyperparameters. The mean cumulative reward was saved at the end of each iteration as a means to assess training performance. Trajectory rewards and visibility percentage plots were generated and saved following each iteration as well. Following the final training iteration, the environment was reset to the initial spacecraft state distribution and evaluated using the final policy network for comparison to the initial/random and heuristic policies. For the case without actuation, actions were selected deterministically by using the maximum output channel rather than a random distribution. For the case with actuation, actions were sampled stochastically from the resulting trained policy distributions.

Training and evaluation were performed on a MacBook Pro with an Apple M3 processor and 8GB RAM. One full training session takes approximately one hour, or an average of 1.8 seconds per iteration, depending on the given scenario.

Input Scaling

To keep network inputs roughly within the range [-1,1], spacecraft states were scaled by the position and velocity scale factors provided in Table 6. The current time, t_k was divided by t_{max} (24,000 sec) such that the scaled value was in the range [0,1]. The central body rotation state, θ,

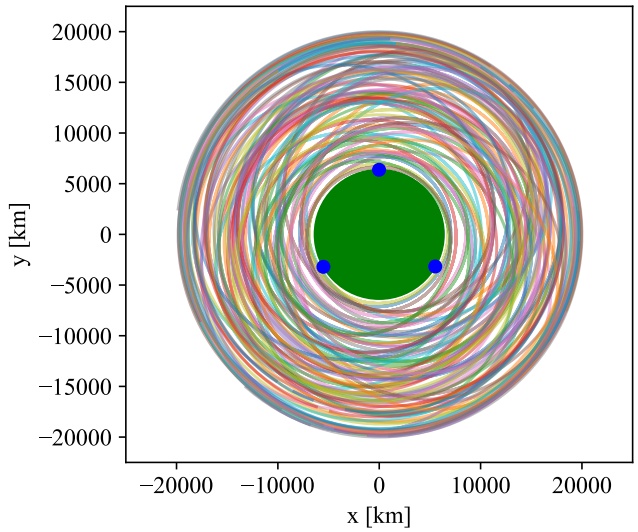


Figure 3: 100 randomly-initialized spacecraft orbits propagated for 24,000 sec. Each color corresponds to a unique spacecraft trajectory realization. The green circle represents the central body radius, and the solid blue dots are notional locations of the ground stations which rotate with the central body.

was replaced by $\sin(\theta)$ and $\cos(\theta)$ in the input state vector to improve overall performance and convergence. Both dimension scaling and *sin*/*cos* transformations used custom TorchRL `Transform` classes.

Policy Evaluation Metrics

The study used three main criteria to assess policy performance: the mean, non-discounted, cumulative reward for all trajectories; the reward as a function of time step for each trajectory; and the station visibility percentage at each time step. The mean, non-discounted, cumulative reward, $\bar{\mathcal{R}}$, is defined as:

$$\bar{\mathcal{R}} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{M}\mathcal{R}(s^n(t_m), a^n(t_m)) \tag{30}$$

where N=100 is the total number of trajectories, and M=200 is the total number of time steps per trajectory. The reward at each time step for each individual trajectory, $\mathcal{R}^n(t_m)$, is also plotted, along with the mean reward at each time step, $\bar{\mathcal{R}}(t_m)$, calculated as:

$$\bar{\mathcal{R}}(t_m) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{R}(s^n(t_m), a^n(t_m)) \tag{31}$$

Intuitively, for this scenario, policies that consistently select measurement actions corresponding to stations that are visible to the spacecraft result in higher cumulative rewards. Therefore, the percentage of selected actions corresponding to visible stations—and thus valid measurements—is

Table 5: Summary of PPO hyperparameters used for policy and value network training.

Parameter	Value
Hidden layers (Policy and Value)	4
Nodes per layer (Policy and Value)	16 or 64
Activation function (Policy & Value)	`LeakyReLu`
Data points per iteration	20,000
Mini-batch size	4,096
Training epochs per iteration	10
Total iterations	2,000
Initial learning rate	8e-5
Maximum gradient norm.	1.0
Clip ϵ	1.0
Discount factor, γ	0.99
Trajectory discount, λ	0.95
Entropy coefficient	1e-4
Discrete entropy bias, β	5.0

Table 6: Factors used to scale policy and value network inputs to ensure that they fall roughly within the range [-1,1] for better performance and convergence.

Dimension	Scale
Position	5e-5 km^{-1}
Velocity	0.13 $(km/s)^{-1}$
Angle	0.159 rad^{-1}
Time (t_k-only)	4.2e-4 sec^{-1}

calculated at each time step. Station visibility is determined via Equation 9. Null and maneuver actions are considered not visible, by default.

RESULTS

Ground Station Tracking (No Actuation)

Figure 4 displays the station visibility percentage for the trained, random, and heuristic policies corresponding to the no-actuation case with a 20,000 km maximum initial orbit radius. The visibility percentage for the trained policy is nearly equivalent to the heuristic policy; however, the heuristic slightly outperforms the trained policy. Investigating the exact cause of the minor performance difference warrants future work. In any event, the performance of the trained policy is clearly much higher than the random, and equivalently untrained, policy.

Figure 5 depicts trajectory rewards for the three policies. The transparent blue lines are the rewards corresponding to the 100 trajectories for each time step using the trained policy. The mean

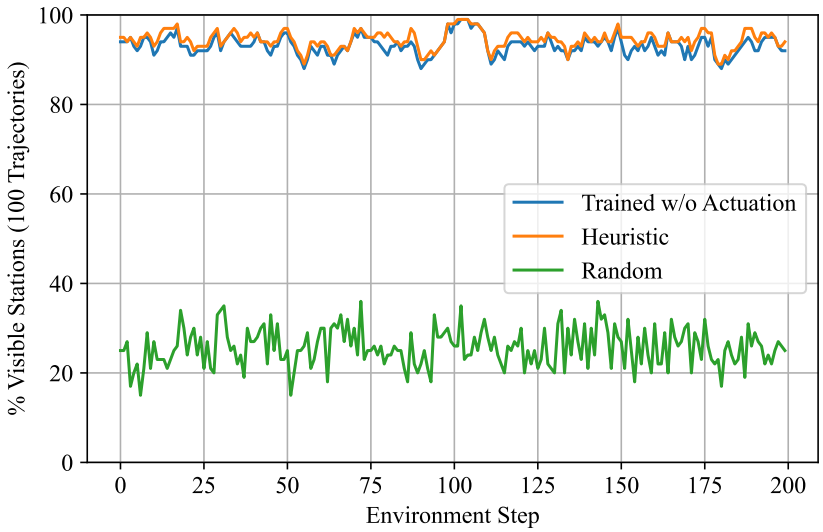


Figure 4: Ground station visibility percentage as a function of environment step for the no-actuation case (20,000 km maximum initial orbit radius).

of those rewards is represented with the dashed red line. The teal and orange dashed lines represent the heuristic and random policy mean rewards, respectively. Similar to the visibility percentage metric, the performance of the trained policy is nearly equivalent to, but still slightly less than, the heuristic policy. Rewards for the trained policy far exceed the performance of the random policy, indicating a significant improvement after training.

Ground Station Tracking with Actuation

Further examining Figure 5 reveals a substantial variance in the cumulative rewards for each trajectory realization (blue curves) due to variation in initial orbit conditions. Plotting the cumulative, non-discounted reward versus the initial orbit SMA and periapsis radius (Figure 6) indicates a strong correlation between the parameters. At orbit radii below approximately 13,000 km, the spacecraft enters and exits spatial regions where no ground station is visible, thus reducing the amount of tracking. Continuous tracking is possible at higher altitudes, but sensitivity to μ decreases proportionally to $1/r^2$. Therefore, in theory, the agent is incentivized to perform actuation in order to reconfigure the orbit to balance station visibility and sensitivity to μ. However, initial experiments revealed the agent's tendency to fall into an apparent local minimum, forgoing actuation in favor of choosing tracking actions. The cause and potential remedy for this behavior points to the need for future work.

In the meantime, we modified the experiment to reduce the maximum initial orbit radius to 10,000 km in order to emphasize the utility of performing maneuvers and adjusting the orbit configuration. The training configuration and process was nearly identical as before, except that the number of nodes in each hidden layer of the policy and value networks increased from 16 to 64. Random action sampling, as opposed to deterministic in the tracking-only case, was used at evaluation because of the stochastic nature of the resulting policy. Figures 7a and 7b show the spacecraft trajectories with and without maneuvers from the trained policy, respectively. None of the resulting trajectories

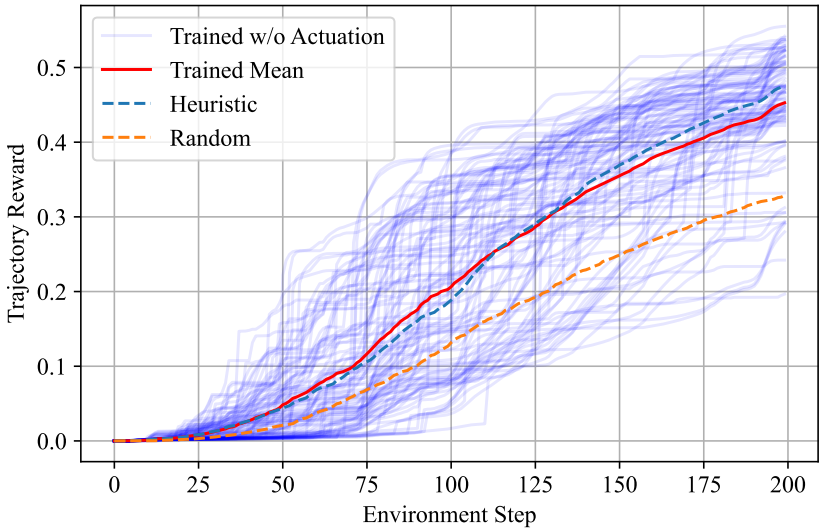


Figure 5: Reward trajectories for each realization (transparent blue) and the mean (solid red) for the final, PPO-trained policy compared to the mean reward for the heuristic (dashed teal) and random (dashed orange) policies. No actuation, 20,000 km maximum initial orbit radius.

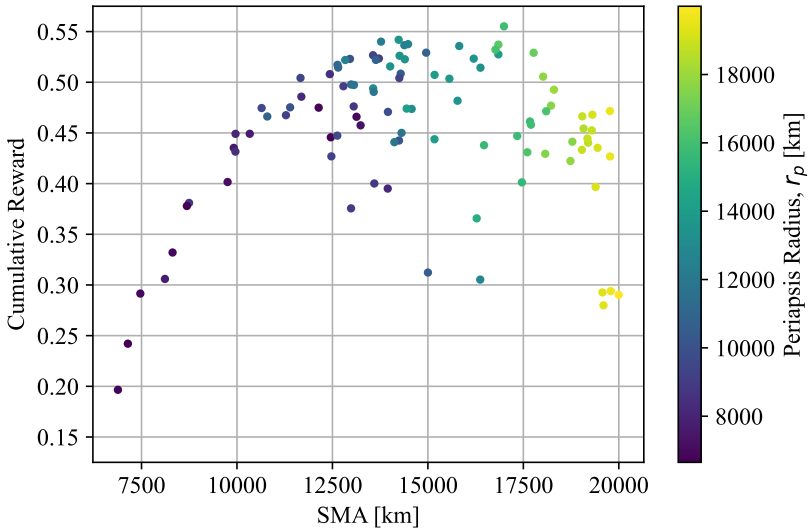
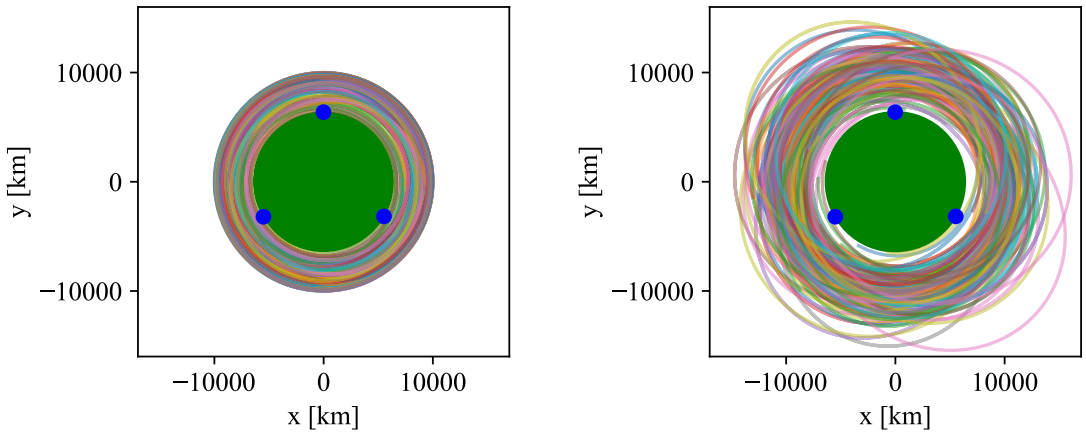


Figure 6: Correlation between initial orbit SMA (and periapsis radius) versus cumulative reward for the ground station tracking scenario without actuation (20,000 km maximum initial orbit radius).



(a) Initial orbit configuration with $r_{max} = 10,000$ km.

(b) Final orbit configuration including maneuvers dictated by the trained policy.

Figure 7: Comparison of the initial (a) and final (b) orbit configurations corresponding to the actuation case (10,000 km maximum initial orbit radius).

violated the termination conditions for the minimum or maximum radius. Figure 8 presents corresponding maneuver magnitudes and flight path angles (FPAs) from the trained policy, where each color represents maneuvers from a different trajectory realization. A total of 281 maneuvers were performed with magnitudes generally between 0.1 and 0.45 km/s and FPAs between $\pm 45°$. The latter indicates an increase in orbital energy, as expected. A notable observation is the tendency for the policy to favor several, smaller-magnitude maneuvers in close proximity rather than one larger maneuver. This is likely attributable to the probabilistic nature of the policy sampling. Implications and possible remedies for employing a stochastic policy demands further study.

Figure 9 shows the station visibility percentage for this case. The heuristic and random policies, neither with actuation, exhibit a lower average visibility percentage because of the lower maximum orbit radius and increased time spent in the non-visible regions. The trained policy starts with a lower average visibility percentage as a consequence of the selection of maneuver actions, followed by an increase in visibility resulting from the improved geometry in the new orbits. There is a noticeable decrease in visibility around approximately 75 to 100 steps (9,000 to 12,000 sec). This apparent cyclical pattern is consistent with orbit periods corresponding to SMAs near 10,000 km (9,952 sec, 83 steps).

Figure 10 depicts trajectory rewards for the case with actuation. The mean reward for the trained policy exceeds the heuristic and random policies (without actuation). There is also less variation among rewards for each trajectory realization (transparent blue) compared to the case without actuation (Figure 5). Figures 11a and 11b present a comparison of the correlation between initial orbit SMA (and periapsis radius) and cumulative reward for the heuristic (no actuation) and trained policy (with actuation), respectively. There is significantly less correlation between the parameters for the trained policy (Figure 11b). All of these observations indicate that the trained policy successfully uses maneuvers to improve visibility and thus increase total reward.

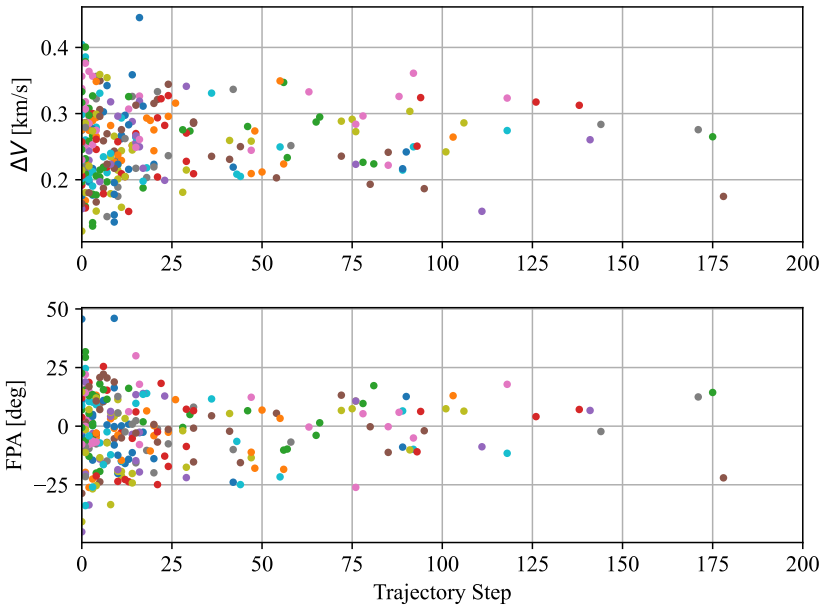


Figure 8: Magnitudes and FPAs for the 281 maneuvers produced by the trained policy for the case with actuation and 10,000 km maximum initial orbit radius. Each color corresponds to a different trajectory realization.

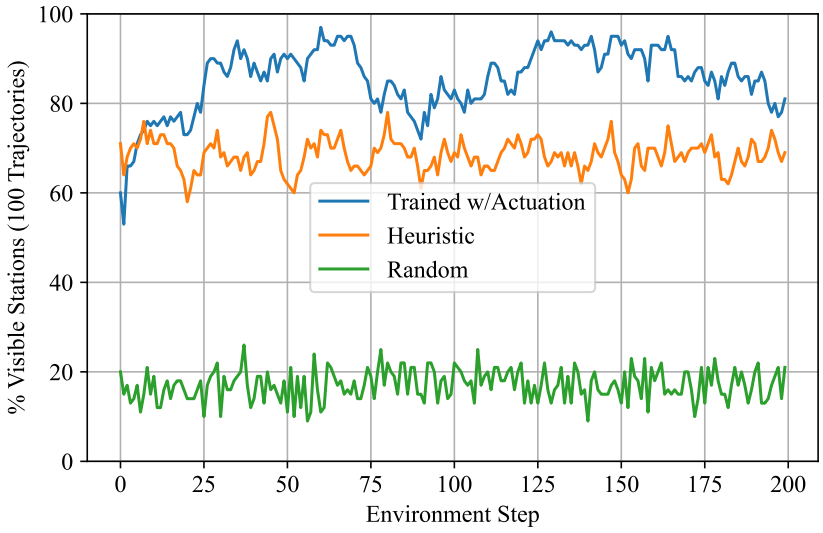


Figure 9: Ground station visibility percentage as a function of environment step for the case with actuation and a 10,000 km maximum initial orbit radius.

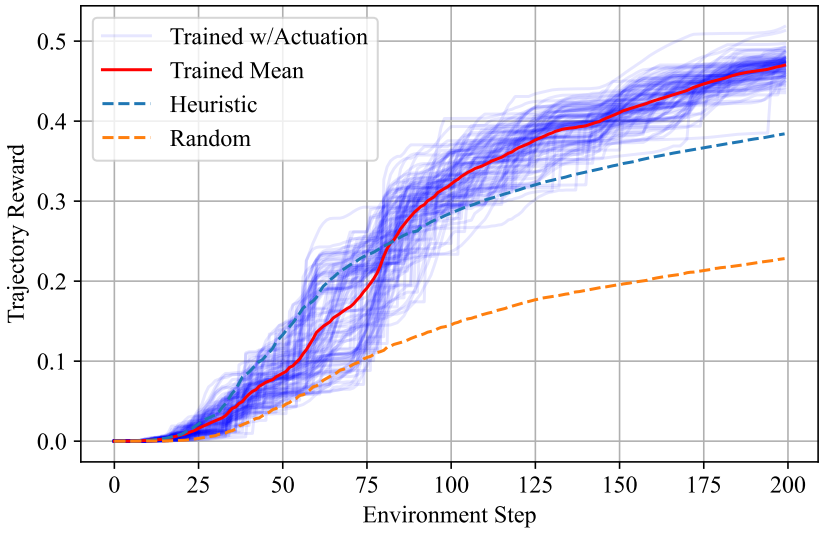
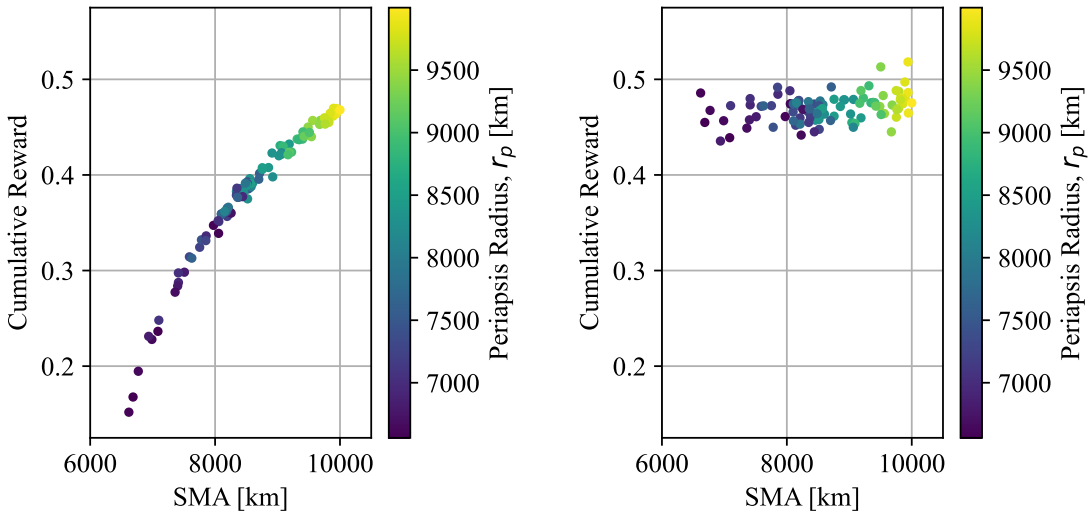


Figure 10: Reward trajectories for each realization (transparent blue) and the mean (solid red) for the final, PPO-trained policy compared to the mean reward for the heuristic (dashed teal) and random (dashed orange) policies. With actuation, 10,000 km maximum initial orbit radius.



(a) Heuristic Policy (No Actuation)

(b) Trained Policy with Actuation

Figure 11: Correlation between initial orbit SMA (and periapsis radius) versus cumulative reward for the heuristic policy without actuation (a) and the trained policy with actuation (b). 10,000 km maximum initial orbit radius.

CONCLUSIONS

Overall, the results indicate that performance of the DRL-derived policy with and without actuation approaches and, in some cases, exceeds that of simple heuristic policies without actuation. Thus, DRL is a valid approach to the problem and provides promise for addressing more complex and realistic scenarios. Despite the relative performance increase over a random or untrained policy, there is still room for improvement, particularly with regard to actuation. Understanding the performance differences and investigating approaches for minimizing the gap will guide network design and training for other scenarios.

Early experimentation and training provided valuable insight into the problem. Network size proved to be an important training parameter. Smaller network sizes on the order of 16 to 64 nodes per layer allowed for significantly faster run times, more training iterations, and resulted in better policies. Additionally, slight improvements in training were observed using a `LeakyReLu` activation function for the policy and value networks compared to `Tanh`. Another consequential design decision that directly affected policy training was input scaling. Most notably, transforming the central body rotation angle, θ, to a $sin(\theta)$ and $cos(\theta)$ resulted in substantial improvement in terms of overall policy performance and convergence time. There was also a noticeable improvement by including the three additional terms in the MDP state space (Equation 14): $\frac{1}{1+\sigma_\mu}$, α, and t_k. In terms of filter design, implementing a UD-factorized form with process noise alleviated numerical issues observed with a non-factorized form. Derivation of entropy for a parametric distribution and inclusion of a bias term (Equation 29) was necessary to achieve proper exploration during training.

Before transitioning to more complex and realistic scenarios, additional investigations are warranted to understand and improve the performance, such as modifications to network training and additional hyperparameter tuning. Additional investigations considering the sensitivity of scenario parameters, including initial uncertainties, measurement noise values, step sizes, and maximum steps, are also indicated. Further understanding of the parametric action space formulation and implementation, particularly with respect to entropy and the bias term sensitivity, is necessary. For more complex scenarios, there may be utility in providing formal uncertainty information to the policy network through the MDP state vector. Analyzing how policies trained on shorter arcs generalize and extrapolate to longer scenarios is another possible area of study. While PPO has proved promising for these experiments, it is worth comparing other techniques such as off-policy methods that may be more sample-efficient and better enable policy exploration. Detailed comparisons of DRL to alternative optimization techniques, such as mixed-integer linear programming or evolutionary algorithms, are also potential future work. Better understanding performance impacts on the simpler case will translate to more complex simulations.

While the preliminary results of this study are promising, significant development and analysis is required in order to employ DRL in a complex, flight-like scenario or operational setting. Subsequent efforts will focus on incrementally increasing fidelity to the simplified scenario described in this study, including transitioning to a 3D spacecraft state, adding higher-fidelity force and measurement models, and considering spacecraft operational constraints. Additional navigation and environment characterization objectives will be incorporated into the reward function, e.g., higher-order gravity coefficients and sensor coverage. Computational requirements and capabilities of current spaceflight processor hardware must also be addressed. The overall goal is to develop tools and techniques that are applicable to real-world spacecraft navigation and planning applications, either on the ground or onboard the spacecraft. The latter would represent a significant step toward the broader vision of fully autonomous spacecraft and spacecraft constellations deployed for a wide

range of ambitious mission concepts for scientific discovery, exploration, and commercial ventures.

REFERENCES

[1] K. M. Getzandanner, K. E. Berry, P. G. Antreasian, J. M. Leonard, C. D. Adam, D. R. Wibben, M. C. Moreau, D. E. Highsmith, and D. S. Lauretta, "Small-Body Proximity Operations & TAG: Navigation Experiences & Lessons Learned from the OSIRIS-REx Mission," *The 32nd AIAA/AAS Space Flight Mechanics Meeting, San Diego, CA*, Jan. 2022.

[2] P. G. Antreasian, C. D. Adam, B. W. Ashman, K. Berry, , J. L. Geeraert, K. M. Getzandanner, D. E. Highsmith, J. M. Leonard, E. Lessac-Chenen, A. Levine, J. V. McAdams, L. K. McCarthy, M. Moreau, D. Nelson, B. Page, J. Y. Pelgrift, S. Rieger, E. Sahr, D. Wibben, B. Williams, K. Williams, and D. Lauretta, "OSIRIS-REx Proximity Operations and Navigation Performance at (101955) Bennu," *The 32nd AIAA/AAS Space Flight Mechanics Meeting, San Diego, CA*, Jan. 2022.

[3] M. Stephenson and H. Shaub, "Reinforcement Learning for Earth-Observing Satellite Autonomy with Event-Based Task Intervals," *46th AAS Guidance and Control Conference*, Breckenridge, CO, Feb. 2024.

[4] A. Herrmann and H. Schaub, "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, Oct. 2023, 10.1109/TAES.2023.3251307.

[5] A. Harris, T. Valade, T. Teil, and H. Schaub, "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 59, Feb. 2022, pp. 611–626, 10.2514/1.A35169.

[6] M. Tipaldi, R. Iervolino, and P. R. Massenio, "Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges," *Annual Reviews in Control*, Vol. 54, Jan. 2022, pp. 1–23, 10.1016/j.arcontrol.2022.07.004.

[7] A. Brandonsio, L. Capra, and M. Lavagna, "Deep Reinforcement Learning Spacecraft Guidance with State Uncertainty for Autonomous Shape Reconstruction of Uncooperative Target," *Advances in Space Research*, 2023, 10.1016/j.asr.2023.07.007.

[8] D. Chan and A. Agha-Mohammadi, "Autonomous Imaging and Mapping of Small Bodies Using Deep Reinforcement Learning," *IEEE Aerospace Conference*, Big Sky, MT, Mar. 2019.

[9] A. Herrmann and H. Schaub, "Reinforcement Learning for Small Body Science Operations," *AAS Astrodynamics Specialist Conference*, Charlotte, NC, Aug. 2022.

[10] S. Takahashi and D. J. Scheeres, "Autonomous Reconnaissance Trajectory Guidance at Small Near-Earth Asteroids via Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, Vol. 46, July 2023, pp. 1280–1297, 10.2514/1.G007043.

[11] Z. Chen, H. Cui, and Y. Tian, "Autonomous Maneuver Planning for Small-Body Reconnaissance via Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, May 2024, pp. 1–13, 10.2514/1.G008011.

[12] J. R. Carpenter and C. N. D'Souza, "Navigation Filter Best Practices," Tech. Rep. NASA/TP–2018–219822, 2018.

[13] C. R. Gates, "A Simplified Model of Midcourse Maneuver Execution Errors," Tech. Rep. 32-504, JPL, Pasadena, CA, 1963.

[14] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement Learning with Parameterized Actions," Oct. 2015.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017.

[16] A. Bou, M. Bettini, S. Dittert, V. Kumar, S. Sodhani, X. Yang, G. D. Fabritiis, and V. Moens, "TorchRL: A data-driven decision-making library for PyTorch," June 2023.

[17] M. Lienen and S. Günnemann, "torchode: A Parallel ODE Solver for PyTorch," *The Symbiosis of Deep Learning and Differential Equations II, NeurIPS*, 2022.

[18] E. T. Jaynes, "Information Theory and Statistical Mechanics," *Brandeis University Summer Institute Lectures in Theoretical Physics*, Vol. 3, 1962, pp. 182–218.

[19] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," 2018.

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017.

[21] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," 2017.