Safe, Autonomous Multi-Agent Inspection of Space Objects Leveraging Relative Orbit Dynamics

Mark Stephenson and Hanspeter Schaub
University of Colorado, Boulder

ABSTRACT

The close-proximity inspection of objects in low Earth orbit is important to operations such as rendezvous, debris removal, servicing, and resident space object (RSO) characterization, all which are of increasing interest to commercial and government organizations. Complex relative motion dynamics in low Earth orbit make the problem of path planning for autonomous multi-agent inspection challenging. Agents must be able to fully inspect an object subject to illumination constraints while avoiding collision with the RSO or—in the multi-agent case—each other. In this paper, autonomous satellite inspection with impulsive maneuvers is considered by learning a policy on a multi-agent semi-Markov decision process formulation of the inspection task while ensuring safety via an optimization-based shield for collision avoidance based on analytical equations of relative motion. This work demonstrates closed-loop, autonomous, safe multi-agent inspection of an RSO with shielded deep reinforcement learning over all low Earth orbit (LEO) orbits.

1. INTRODUCTION

With the proliferation of satellites in low Earth orbit (LEO), rendezvous and proximity operations (RPO) are becoming increasingly important. These include servicing and interacting with active spacecraft, deorbiting defunct satellites, or inspecting assets for damage. Prior to many of these operations, the resident space object (RSO) must be inspected to determine docking points, damage, or other properties. This inspection task is complex, requiring the servicer to maneuver around the RSO, inspecting all illuminated surfaces while avoiding collision. Currently, close-proximity operations are challenging to operate and require significant ground support to upload open-loop command sequences and monitor the resulting performance. Close relative motion maneuvers must be fuel efficient, avoid the potential for collisions, and satisfy any imaging requirements. In some cases, multiple servicers may be in operation around a single RSO at the same time.

A variety of path-planning methods have been proposed for the problem. References [1, 2, 3] demonstrate pipelines for global planning of inspection trajectories for a swarm of satellites, for impulsive and continuous thrust control. These plan a set of stable relative orbits that should provide coverage of the RSO, then assign orbits to individual servicers within the swarm, calculating optimal transitions between orbits. Another paper [4] decomposes a large RSO into primitive shapes and projects inspection paths onto them. In [5, 6], traditional A*-based path-planning methods are combined with methods for robustness to thruster failures.

More recently, reinforcement learning (RL) has been posed as a method for closed-loop autonomy on the inspection task. RL has the benefits of directly finding a policy to maximize a reward function in an arbitrary environment. In [7], a neural network-based policy is used for high-level planning between predetermined inspection waypoints. Another paper [8] approaches the problem with a continuous action space for continuous, low-thrust control, considering lighting conditions. In recent work [9, 10], the waypoint-based approach is used for distributed and centralized control of a swarm of servicers. In [11, 12], the problem is considered with a simultaneous localization and mapping (SLAM) component. However, a drawback of this class of methods is a lack of safety guarantees. While penalties can be included to disincentivize unsafe actions, there is no guarantee that the agent will not take an unsafe action.

To address this challenge, a variety of methods for bounding the performance and guaranteeing safety of RL-based polices have been developed [13]. Several references combine RL with trajectory optimization steps: A few authors [14, 15] leverage the transformer architecture for RPO, while [16] uses RL with traditional path planning methods as a way of directing the search of the traditional methods while maintaining robustness. Shielded RL [17] has been

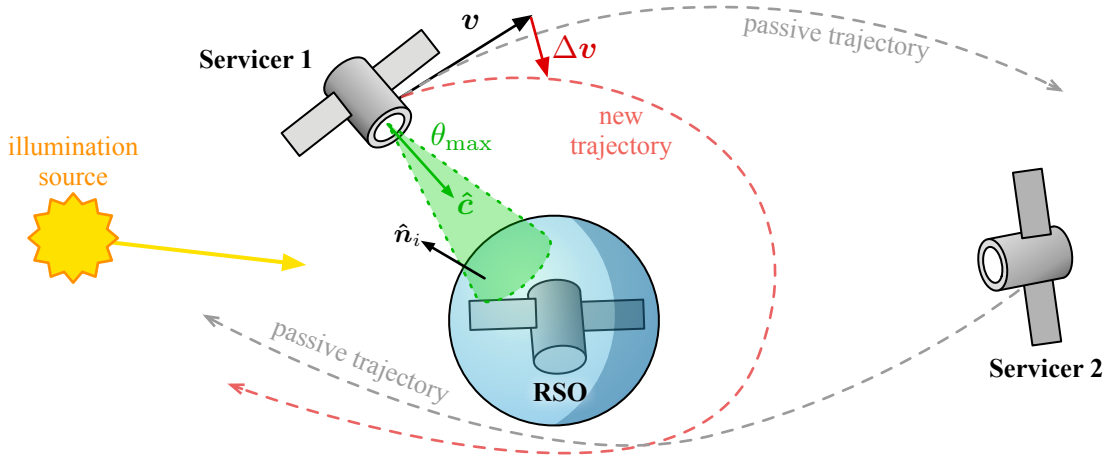


Fig. 1: The multi-agent inspection task with $N = 2$ servicers.

demonstrated as an effective method for various other spacecraft tasking problems, including resource management [18], small-body proximity operations [19], and agile Earth observation under various conditions [20, 21]. Control barrier functions (CBFs) are an alternative to shields that provide guarantees on performance: Most similar to this work, two recent papers [22, 23] ensure safety in RL-controlled inspection task with continuous control inputs using a CBF. Their safety constraints include the keep-out zone present in this work, fixed-horizon passive safety, velocity constraints, a Sun pointing constraint, and a keep-in zone.

To develop safety guarantees for these tasks, existing work on safe coordinated motion planning in LEO can be leveraged. Morgan [24, 25] uses sequential convex programming (SCP) to plan collision-free continuous thrust trajectories. Others [26, 27] use analytical methods to derive swarm reconfiguration laws with impulsive thrusts in terms of relative orbital elements.

In this work, the RSO inspection problem is considered for an RSO in an eccentric orbit with a Hill-frame-fixed attitude and multiple impulsively thrusting servicers, subject to range, pointing, and lighting constraints for inspection. This builds on [28], which considers only a single servicer and Clohessy-Wiltshire-Hill (CWH) circular orbit dynamics, which is a common constraint in existing literature. Considering multiple servicers creates a more complex collision avoidance scenario. In addition to single-agent analysis, this study researches if multi-agent training is able to create novel cooperative inspection behavior not seen when a single-agent policy is used by multiple agents in the same environment. The problem is formulated as a Markov decision process (MDP) and solved using deep reinforcement learning (DRL), with a formulation that is specifically designed to reflect the operations of an impulsively maneuvered spacecraft. Unlike other RL-based approaches to the inspection task, a shield that guarantees safe trajectories relative to the RSO or any other nearby objects in orbit is employed; this shield is based on an optimization-based analysis of relative motion dynamics for eccentric orbits under impulsive control.

2. PROBLEM STATEMENT

In the multi-agent inspection task, N servicer spacecraft $\{S_1, ..., S_N\}$ have the objective of inspecting all facets of an RSO S_0 in LEO, subject to illumination constraints. To achieve this, the servicers must use discrete thrusts to control their motion relative to the RSO while avoiding collision between themselves and any other spacecraft.

2.1 RSO Model

The RSO is a nadir-pointing satellite in LEO. This attitude assumption is reflective of the attitude control for many active LEO satellites. The RSO has a set of body-fixed inspection points $p_i \in \mathcal{P}$, each with an associated normal vector $\hat{\boldsymbol{n}}_i$. In this work, the RSO's geometry is modelled as a 1-meter radius sphere of $|\mathcal{P}| = 100$ uniformly distributed points with surface normals in the radial direction. The RSO's orbit is perturbed by J_2 effects and atmospheric drag. The RSO's orbit is a randomly sampled LEO orbit in each instance of the environment. The altitude of apogee and

perigee are uniformly sampled between 500 and 1100 km, and the angular orbital elements are uniformly sampled in their respective domains.

2.2 Servicer Spacecraft Model

Each servicer spacecraft is equipped with a body-fixed camera $\hat{\boldsymbol{c}}_s$ to inspect the RSO. A flight software algorithm controls the attitude of the spacecraft using reaction wheels to point the camera boresight at the RSO. To inspect a point on the RSO, the angle between the normal $\hat{\boldsymbol{n}}_i$ and the camera boresight $\hat{\boldsymbol{c}}_s$ must satisfy

$$\theta_{i,s} = \angle(\hat{\boldsymbol{n}}_i, -\hat{\boldsymbol{c}}_s) < \theta_{\max} \tag{1}$$

A servicer must also be within $r_{\text{inspect}} = 250$ m of the RSO to inspect a point. When a point is inspected, it is added to the global inspected set $\mathcal{I}$. Since the servicers are close to each other, it is reasonably assumed that they can freely communicate information with one another.

As with the RSO, each servicer's orbit is modelled to high fidelity, including J_2 perturbations and atmospheric drag. As a result, the relative motion between the servicer and the RSO is governed by perturbed equations of relative motion for an eccentric chief [29]. Each servicer can control its relative motion with a cluster of impulsive thrusters. These thrusters can produce impulsive thrusts $\Delta\boldsymbol{v}$ of magnitude up to $\Delta v_{\max} = 1$ m/s in an arbitrary direction. Because the period between thrusts tends to be long, this could be achieved by a variable-thrust thruster on an actuated platform that reorients between maneuvers. At most, each servicer may use 10 m/s of Δv to complete the task.

For each pair of satellites S_d and S_c, a keepout ellipsoid is defined. Taking S_d to be the deputy and S_c to be the chief (which may be the RSO or another servicer), the Hill frame centered on the chief is $\mathcal{H}_c$ and the displacement of the deputy in that Hill frame is $\boldsymbol{r}_{d/c}$. A Hill-frame-fixed keepout ellipsoid $\boldsymbol{K}_{d/c}$ is defined for the satellite pair. If any pair of satellites violates their keepout ellipsoid, the inspection task is considered a failure. Mathematically, collision between S_d and S_c is defined as

$$\boldsymbol{r}_{d/c}^T(t)\boldsymbol{K}_{d/c}\boldsymbol{r}_{c/d}(t) \leq 1 \tag{2}$$

being satisfied at any time t.

The initial position of each servicer is uniformly sampled between 250 and 750 m from the RSO in a random direction. The initial velocity of each servicer relative to the RSO is uniformly sampled between 0 and 1 m/s in a random direction.

2.3 Illumination Constraints

In addition to the view incidence angle, the inspection points on the RSO must be illuminated by the Sun. To be sufficiently illuminated for imaging, the incidence angle between the Sun and the point normal must be less than $\phi_{\max} = 60^\circ$. The RSO also must not be eclipsed by Earth. Because the RSO is modelled as a spherical satellite, no self-shadowing is considered in this work. Likewise, inspector shadowing of the RSO is neglected. The subset of the points $\mathcal{P}$ that are illuminated at some point during the orbit is $\mathcal{L}$.

Fig. 2 shows the percent of the RSO that is illuminated over 1000 random cases as a function of the β angle, the angle between the orbital plane and the Sun vector. Orbits that are closer to a Sun-synchronous orbit (SSO) at the dusk-dawn boundary tend to have the smallest fraction of points ever illuminated, as the orbit-frame-fixed attitude leads to one side of the spacecraft constantly facing away from the Sun.

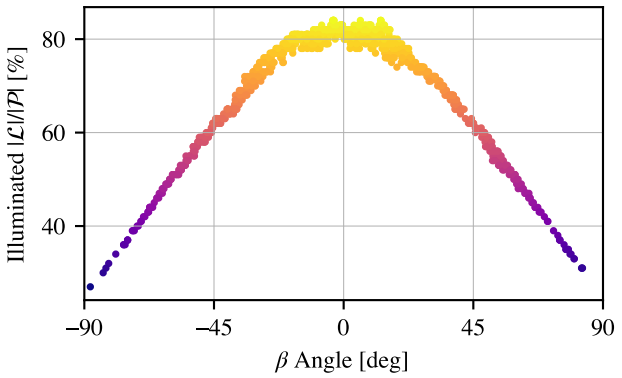


Fig. 2: Percent of RSO meeting illumination conditions for inspection due to orbit geometry.

2.4 Objective Function

Mathematically, the objective of the problem is to maximize the percentage of the RSO that is inspected, subject to a penalty for fuel consumption, non-collision constraints, and the environment dynamics. The full problem can be

expressed as

$$\underset{\{\Delta \boldsymbol{v}_{1,1},\ldots,\Delta \boldsymbol{v}_{N,n_N}\}}{\text{maximize}} \quad \frac{|\mathcal{I}|}{|\mathcal{P}|} - \alpha \sum_{s=1}^{N} \sum_{i=1}^{n_s} |\Delta \boldsymbol{v}_{s,i}| \tag{3}$$

$$\text{such that} \quad \boldsymbol{r}_{i/j}^T(t) \boldsymbol{K}_{i/j} \boldsymbol{r}_{i/j}(t) > 1 \; \forall \, t \in [0, \infty), \; (i,j) \in \{0,\ldots,N\}^2, i \neq j \tag{4}$$

$$\sum_{i=1}^{n_s} |\Delta \boldsymbol{v}_{s,i}| \leq \Delta v_{\text{avail}} \; \forall \, s \tag{5}$$

$$t_{\text{success}} \leq t_{\text{avail}} \tag{6}$$

The maximization is performed over the set of n_s burns and their times for all satellites s. The first term of the objective, $|\mathcal{I}|/|\mathcal{P}|$, is the fraction of points inspected due to the servicer trajectories under that control. In the second term, the sum of fuel used for all burns across all servicers is subtracted from the objective function, weighted by the fuel use penalty α. A keepout constraint (4) is enforced at all times for all pairs of satellites. Finally, a per-satellite maximum fuel (5) and maximum time (6) constraint are also specified.

Of course, directly optimizing this objective function for a complex path planning problem is prohibitively challenging. Instead, RL can be used to find a closed-loop policy that aims to maximize this objective.

3. REINFORCEMENT LEARNING

Reinforcement learning provides a generalized framework for formulating and solving sequential decision-making problems, a broad category that includes path planning, scheduling, and control. As a hybrid of these problem types, the RSO inspection task is a strong candidate for RL-based solution. This section will include a brief overview of RL, specialized modifications to the framework necessary to accurately represent this problem, and the MDP formalization of the RSO inspection problem.

3.1 Learning on Markov Decision Processes

MDPs are a framework for representing sequential optimization problems [30]. An MDP is defined by a state space S, action space A, transition function $T(s,a,s')$, and reward function $R(s,a,s')$. In an MDP, the state s evolves in response to some action a according to the transition function, yielding a new state s' and reward r. The goal of reinforcement learning is to find a policy $\pi(a|s)$ that maximizes the expected γ-discounted sum of rewards

$$V_{\text{MDP}}^{\pi}(s) = r + \gamma \mathbb{E}\left[V_{\text{MDP}}^{\pi}(s') \middle| T(s'|s,a), \pi(a|s)\right] \tag{7}$$

$$= r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots = \sum_{i=0}^{\infty} \gamma^i r_i \tag{8}$$

at all states $s \in S$. However, the transition and reward functions are not directly known to the learning agent. Instead, the agent must iteratively interact with the environment, balancing exploration of new regions of the state space with exploitation of known reward-rich regions to determine the best policy.

While exact methods exist for discrete, low-dimensional state and action spaces, larger and continuous problems are intractable for these approaches. To find solutions to the more challenging (but often more relevant) class of problems, deep RL has been identified as an effective set of methods. In DRL, portions of the RL algorithms are replaced with neural network function approximators, which tend to perform well with high-dimensional and continuous data. DRL algorithms such as proximal policy optimization (PPO), used in this work, have been shown to find strong policies for challenging problems across many domains, including robotics and control [31].

3.2 Semi-Markov Decision Processes

Many sequential problems have a fixed decision interval associated with them, such as a control frequency or a turn in a game. However, the best solutions to planning and scheduling problems often have dynamically determined decision intervals. The RSO inspection problem follows this observation: When performing impulsive maneuvers, a spacecraft generally thrusts then drifts (i.e., does not perform any control inputs) until the next thrust is scheduled. To represent this type of problem, the MDP must be extended to allow for variable-duration timesteps.

A MDP with variable-duration timesteps is called a semi-Markov decision process (sMDP). In a sMDP, each step has some Δt associated with it, which represents the time-opportunity cost of the step, and a reward density $\rho_t(t)$ instead of reward r_t [32]. The γ-discounted reward for a step is then given by the integral

$$r_t^{(\gamma)} = \int_{t_t}^{t_t+\Delta t_t} e^{\gamma(t-t_t)} \rho_t(t) \mathrm{d}t \tag{9}$$

When computing value, Equation 8 also needs to be modified to discount according to the sum of times elapsed over preceding steps

$$V_{\text{sMDP}}(s_0) = \gamma^{\Delta t_0} r_0^{(\gamma)} + \gamma^{\Delta t_0 + \Delta t_1} r_1^{(\gamma)} + \gamma^{\Delta t_0 + \Delta t_1 + \Delta t_2} r_2^{(\gamma)} + \cdots = \sum_{t=0}^{\infty} \gamma^{\sum_{i=0}^{t} \Delta t_i} r_t^{(\gamma)} \tag{10}$$

and any value-like discounted computations in the learning algorithm must be similarly modified.

3.3 Partial Observability

In an MDP with partial observability, the agent learns on observations of the state $o = Z(s)$ instead of the full state. In some cases, estimating the true state is of great importance to solving the problem. The RSO inspection problem has a weaker form of partial observability—the true state of the simulator is very high dimensional, but only a subset of those states are particularly relevant for the control problem. An observation space is hand-crafted to give agents sufficient, but not excessive, information about the state.

3.4 Decentralized Asynchronicity

Two final modifications to the standard MDP framework are necessary to express the multi-agent case: decentralization and asynchronicity. In a decentralized MDP, each agent individually receives and learns on individual observations of part of the environment. As a result, decentralized MDPs are inherently partially observable to each agent, and may still be partially observable in the joint observation space.

The combination of decentralization and sMDPs yields asynchronicity: If agents are acting individually with actions of different durations, their decisions will likely occur at different times [33]. This can be both advantageous and challenging.

3.5 MDP Formulation

The inspection task is formalized as (asynchronous, decentralized, partially observable, semi-) MDP for N agents, which are the N servicer spacecraft. The elements of the MDP tuple for the inspection task are as follows:

- **State Space:** The state of the simulator providing the generative model for the MDP. This includes satellite dynamic states, flight software states, and environment states. Terminal states for a servicer are encountered due to various conditions:
  - $\geq 90\%$ of illuminated points are jointly inspected by all servicers, $|\mathcal{I}|/|\mathcal{L}| \geq 0.9$. This threshold is set since some points on the limb of the spacecraft may only be instantaneously illuminated.
  - A servicer collides with the RSO or another servicer, in violation of Equation 4.
  - A servicer leaves the RSO: $|\boldsymbol{r}_{d/rso}| > 1$ km. This condition is included to encourage "good" behavior when training, but is not a hard constraint enforced by the shield when using the policy.
  - A servicer runs out of available fuel: $\Delta v_{\text{avail}} = 0$.
  - The episode times out: $t \geq 10$ orbits.
- **Observation Space:** The observation for each servicer is composed of a subset of the elements of the state space and transformations thereof. The elements of the observation space are given in Table 1. Two reference frames are used: the RSO Hill frame $\mathcal{H}$, and the Earth-Sun Hill frame $\mathcal{S}$.

  The region inspection status $\%_{\text{inspect}}(\mathcal{P})$ is used as a more compact way of representing what points on the RSO have been jointly inspected, rather than a vector of $|\mathcal{P}|$ Booleans. For this observation, M clusters of inspection points are defined by vectors. Each element of the observation is the fraction of points in the cluster that have been inspected. For this environment, $M = 15$ clusters evenly spaced on the surface of the sphere are used, as it results in regions roughly the same size as a servicer's field of view. This observation does not tell the agent what points will be illuminated at some point; the agent must infer that from the observations of the orbital state.

Table 1: Elements of the observation space for each servicer.

Inclusion	Element	Dim.	Description
Single- and Multi-Agent Training	$\boldsymbol{r}^{\mathcal{H}}_{d/rso}$	3	RSO-relative position
	$\boldsymbol{v}^{\mathcal{H}}_{d/rso}$	3	RSO-relative velocity
	$\hat{\boldsymbol{s}}^{\mathcal{H}}$	3	Sun direction
	$\boldsymbol{e}^{\mathcal{S}}$	3	orbital eccentricity vector
	$\boldsymbol{h}^{\mathcal{S}}$	3	orbital angular momentum vector
	$\boldsymbol{r}^{\mathcal{S}}_{DE}$	3	orbital position
	Δv_{avail}	1	available Δv
	t	1	time since start of episode
	$[\tau^{o}_{\text{ecl}}, \tau^{c}_{\text{ecl}}]$	2	next eclipse start and end times
	$\%_{\text{inspect}}(\mathcal{P})$	M	region inspection status
Multi-Agent Training Only	$\boldsymbol{r}^{\mathcal{H}}_{s/rso}$	$3(N-1)$	RSO-relative position of servicer s
	$\boldsymbol{v}^{\mathcal{H}}_{s/rso}$	$3(N-1)$	RSO-relative velocity of servicer s

Some elements of the observation are always included, while others are only included when training in a multi-agent environment (as opposed to training in a single-agent environment and deploying in a multi-agent environment). The additional elements give each servicer information about the other servicers' positions and velocities. Since these elements are "baked in" to the policy network, the resulting policies do not generalize to different agent counts N. Practically, the number of servicers would be known well before a mission and has a low upper bound.

- **Action Space:** Each servicer's action space is $a \in \Delta v_{\max}\mathcal{B}^3 \times [0, \Delta t_{\max}]$. The first three elements specify the direction and magnitude of an impulsive thrust within a ball, and the last element specifies the duration to drift before executing the next thrust.

- **Reward Function** The reward density $\rho(t)$ for each servicer is the sum of four functions. First, reward is yielded for inspecting points on the RSO that have not yet been inspected by any servicer.

$$\rho_{\text{inspection}}(t) = \frac{1}{|\mathcal{P}|}\frac{\mathrm{d}}{\mathrm{d}t}|\mathcal{I}|. \tag{11}$$

A reward is also given to all agents for jointly reaching the goal state of $|\mathcal{I}|/|\mathcal{L}| \geq 90\%$ inspection of illuminated points, using the Dirac delta function.

$$\rho_{\text{success}}(t) = \beta_{\text{success}}\delta(t - t_{\text{success}}) \tag{12}$$

A penalty for fuel use is given at the time of a burn, weighted by the fuel use penalty α.

$$\rho_{\text{fuel}}(t) = -\alpha|\Delta \boldsymbol{v}|\delta(t - t_{\text{burn}}) \tag{13}$$

Finally, a failure penalty is given to a servicer that reaches any of the negative terminal states (collision, running out of fuel, or leaving the region of space around the RSO)

$$\rho_{\text{failure}}(t) = -\beta_{\text{fail}}\delta(t - t_{\text{failure}}) \tag{14}$$

Objective completion or failure of all agents terminates the episode.

- **Transition Model:** Instead of a probabilistic transition function, the transition model is implemented as a deterministic generative model. When an action is taken, the environment propagates the simulation forward in time until the next action is to be taken. The sMDP Δt is the duration for which the simulator propagated the step, as selected by the action.

The environment is implemented in accordance with the Gymnasium API [34] using BSK-RL[1], a package for creating modular, high-fidelity RL environments for spacecraft tasking [35]. The underlying spacecraft simulation is Basilisk[2], a high-performance spacecraft simulation package [36]. Rigid multi-body dynamics in the perturbed orbital environment and flight-proven flight software algorithms are used to simulate the environment. Basilisk is ideal for offline agent training as its high computation speed enables efficient training with a complex physics-based simulation.

The RLlib implementation of PPO is used to train policies in this environment [31, 37]. The algorithm and training pipeline are modified to use semi-Markov discounting for advantage estimation and to retask agents asynchronously, only when their previous action has completed. There are two options for obtaining policies for the multi-agent case: deploying the single-agent policy in a multi-agent setting with cooperation induced via the sharing of task completion status (which was effective in [21]), and directly training agents in the multi-agent environment with awareness of the other agents' activities through additional observation elements and the sharing of task completion status. In the latter case, all agents act independently but contribute experience to the same policy, which is duplicated across the team.

4. OPTIMIZATION-BASED SHIELD

Collision with other spacecraft is of high concern for each servicer spacecraft. Shielding is a method for guaranteeing operational safety of a RL-based agent by preventing the selection of actions that are known to be unsafe. For this problem, a shield is developed that prevents the servicer from executing a burn that may lead to eventual collision with another spacecraft under natural dynamics.

4.1 Shielding to Guarantee Safety

A shield interacts with a policy in order to prevent unsafe actions (i.e. those actions that may cause a transition into $s' \in S_{\text{unsafe}}$ in the next step, or for more robustness, in any future step) [17]. The shield in this work uses the action projection approach to shielding [13]: if the desired action falls in the space of disallowed actions, the closest safe action is selected instead.

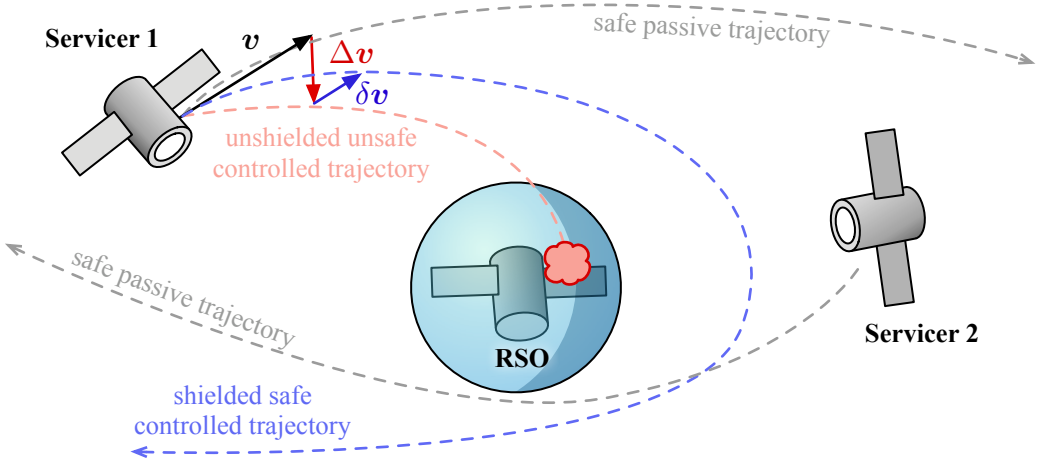


Fig. 3: Shield guaranteeing safety when a servicer performs a maneuver.

The conceptual behavior of the shield is shown in Fig. 3. All servicers are assumed to be on trajectories that are passively safe over an infinite horizon relative to all other spacecraft, due to previous iterations of the shield. Since the agents act asynchronously, at any time only a single servicer will need to select a new action. The policy selects some $\Delta\boldsymbol{v}$; then, the shield determines the smallest deviation $\delta\boldsymbol{v}$ that makes the new trajectory passively safe under relative motion dynamics relative to every other spacecraft. Since the servicer was already known to be on a passively safe trajectory from the previous iterations of shielded tasking, there is always at least the trivial solution $\delta\boldsymbol{v} = -\Delta\boldsymbol{v}$ that satisfies the safety constraints by keeping the servicer in the same passively safe orbit as before.

[1] avslab.github.io/bsk_rl/
[2] avslab.github.io/basilisk/

4.2 Relative Motion Dynamics

Analysis of the system dynamics is necessary to formalize the shield constraints. The motion of the spacecraft is described with a relative motion state transition matrix for the unperturbed motion of a deputy spacecraft relative to a chief spacecraft in an eccentric orbit. In the Hill frame of the chief satellite S_c, the state transition matrix for the state vector $\boldsymbol{x}_{d/c} = \begin{bmatrix} {}^{\mathcal{H}_c}\boldsymbol{r}_{d/c} & {}^{\mathcal{H}_c}\boldsymbol{v}_{d/c} \end{bmatrix}^T$ composed of the Hill-frame position and velocity of the deputy relative to the chief is given by

$$\boldsymbol{\Phi}_c^x(t,t_0) = \boldsymbol{A}_c(t)\boldsymbol{\Phi}_c^{\delta\boldsymbol{\alpha}}(t,t_0)\boldsymbol{A}_c^{-1}(t_0) \tag{15}$$

where $\boldsymbol{A}_c(t)$ is the linearized relative orbital element to Hill frame mapping matrix, and $\boldsymbol{\Phi}_c^{\delta\boldsymbol{\alpha}}(t,t_0)$ is the relative orbital element state transition matrix (see [29], chapter 14.5 for the full 6×6 matrices for $\boldsymbol{A}_c$, $\boldsymbol{A}_c^{-1}$, and $\boldsymbol{\Phi}^{\delta\boldsymbol{\alpha}}$).

It is also useful to bound the long-term trajectories of a deputy relative to a chief, in order to provide infinite horizon safety guarantees in the event that the satellite becomes uncontrolled (simply put, satellites should never be on a collision course). Considering the closed-form solution to relative motion with first order eccentricity (see [29], chapter 14.6.3), two cases lead to passive safety: Either the relative motion is periodic, requiring $\delta a = 0$, or some secular term dominates, and the deputy is moving away from the chief.

Only the y component has such a secular term. The x and z components both oscillate through zero as $t \to \infty$, so they cannot be used to improve the lower bound on the distance between spacecraft. The y component motion is described in terms of orbit element differences

$$y(f,\delta M) \approx a\left(\frac{\delta M}{\eta} + \delta\omega + \cos i\,\delta\Omega\right) - a\delta_y \sin(f - f_y) - \frac{ae}{2}\sin(2f)\delta e \tag{16}$$

where

$$\delta_y(\delta M) = \sqrt{4\delta e^2 + e^2\left(\frac{\delta M}{\eta} - \delta\omega - \cos i\,\delta\Omega\right)^2} \tag{17}$$

By minimizing and maximizing the terms that are periodic in f, the y component motion is bounded by

$$y(\delta M) \geq \lfloor y(\delta M)\rfloor = a\left(\frac{\delta M}{\eta} + \delta\omega + \cos i\,\delta\Omega\right) - a\delta_y(\delta M) - \frac{ae}{2}\delta e \tag{18}$$

$$y(\delta M) \leq \lceil y(\delta M)\rceil = a\left(\frac{\delta M}{\eta} + \delta\omega + \cos i\,\delta\Omega\right) + a\delta_y(\delta M) + \frac{ae}{2}\delta e \tag{19}$$

which are increasing in δM for $\delta a \neq 0$. If $\delta\dot{M} > 0$ (equivalent to $\delta a < 0$), the deputy is guaranteed to be clear of the chief keepout $\boldsymbol{K}$ for all $t > t^*$ if

$$\lfloor y(\delta M(t^*))\rfloor \geq (\hat{\boldsymbol{y}}^T\boldsymbol{K}\hat{\boldsymbol{y}})^{-\frac{1}{2}} \tag{20}$$

Likewise, if $\delta\dot{M} < 0$ (equivalent to $\delta a > 0$), the condition is

$$\lceil y(\delta M(t^*))\rceil \leq -(\hat{\boldsymbol{y}}^T\boldsymbol{K}\hat{\boldsymbol{y}})^{-\frac{1}{2}} \tag{21}$$

As mentioned earlier, the final case $\delta a = 0$ is always acceptable as it yields periodic motion. These three cases can be combined into two concurrently active constraints, leveraging the increasing nature of the functions so that the cases overlap:

$$\begin{aligned} g_{\text{low}}(\boldsymbol{\alpha},\delta\boldsymbol{\alpha}) &= \delta a\left(\lfloor y\rfloor - (\hat{\boldsymbol{y}}^T\boldsymbol{K}\hat{\boldsymbol{y}})^{-\frac{1}{2}}\right) \leq 0 \\ g_{\text{high}}(\boldsymbol{\alpha},\delta\boldsymbol{\alpha}) &= \delta a\left(\lceil y\rceil + (\hat{\boldsymbol{y}}^T\boldsymbol{K}\hat{\boldsymbol{y}})^{-\frac{1}{2}}\right) \leq 0 \end{aligned} \tag{22}$$

4.3 Formalizing the Optimization Problem

With the relative motion dynamics defined, the shield can be formulated as an optimization problem. In short, the problem is to find the thrust for spacecraft d closest to that selected by the policy $\pi(s) = \Delta\boldsymbol{v}$ that does not lead to

collision with the RSO or any other inspector under the relative motion dynamics. Let $\delta \boldsymbol{v}$ be the change in selected thrust required for safety. The optimization problem can be written as

$$\begin{aligned}
\text{minimize} \quad & \|\delta \boldsymbol{v}\| && (23)\\
\text{such that} \quad & \|\Delta \boldsymbol{v} + \delta \boldsymbol{v}\| \leq \Delta v_{\max} && (24)\\
& \text{for each } c \in \{0,...,N\}, c \neq d\\
& \quad \boldsymbol{x}_{d/c}(t_0) = \begin{bmatrix} {}^{\mathcal{H}_c}\boldsymbol{r}_{d/c} \\ \mathcal{H}_c(\boldsymbol{v}_{d/c} + \Delta \boldsymbol{v} + \delta \boldsymbol{v}) \end{bmatrix}\\
& \quad \boldsymbol{x}_{d/c}(t) = \boldsymbol{\Phi}_c^x(t,t_0)\boldsymbol{x}_{d/c}(t_0)\\
& \quad \left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{x}_{d/c}(t)\right)^T \boldsymbol{K}_{d/c} \left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{x}_{d/c}(t)\right) > 1 \quad \forall\, t \in [t_0, t_0 + t_{\max}] && (25)\\
& \quad g_{\text{low,high}}(\boldsymbol{\alpha}(t_{\max}), \delta\boldsymbol{\alpha}(t_{\max}) = \boldsymbol{A}_c^{-1}(t_{\max})\boldsymbol{x}_{d/c}(t_{\max})) \leq 0 && (26)
\end{aligned}$$

The objective (23) minimizes the change in requested thrust. The first constraint (24) ensures that the thrust does not exceed the maximum thrust the servicer can produce. Collision constraints are added for each other spacecraft: Constraint (25) ensures that the shielded action does not lead to collision over some upcoming timespan $[t_0, t_0 + t_{\max}]$. The final constraint (26) uses Equation 22 to maintain long term passive safety on domain $[t_0 + t_{\max}, \infty)$, finding the relative orbital elements using the same mapping matrix $\boldsymbol{A}_c^{-1}$ used in the state transition matrix.

4.4 Iterative Shield Solution

Evaluation times in Equation 25, the keepout constraint, must be discretized to make the number of constraints finite. However, if the discretization is too coarse, this introduces the possibility for interstep collisions. To address this while keeping the number of constraints low (relative to the naïve method of using a small fixed Δt), an iterative scheme to dynamically determine the step size. The position and velocity at each discretized point is used to bound the interstep motion of the deputy.

The minimum distance between the deputy and keepout region $R_{d/Kc}$ is defined as the objective of the optimization problem

$$\begin{aligned}
R_{d/Kc} = \text{minimize} \quad & \left|\boldsymbol{r}^* - \boldsymbol{r}_{d/c}\right| && (27)\\
\text{such that} \quad & \boldsymbol{r}^{*T} \boldsymbol{K}_{d/c} \boldsymbol{r}^* \leq 1 && (28)
\end{aligned}$$

which is a simple convex problem for nonspherical keepouts and trivially

$$R_{d/Kc} = |\boldsymbol{r}_{d/c}| - R_{d/c} \tag{29}$$

for spherical keepouts with radius $R_{d/c}$. It follows that the worst-case maximum timestep between t_i and t_{i+1} that could at most violate the keepout by ε is given by

$$t_{i+1} - t_i \leq \frac{R_{d/Kc}(t_i) + \varepsilon}{\max\limits_{t \in [t_i, t_{i+1}]} |\boldsymbol{v}_{d/c}(t)|} \tag{30}$$

Since this expression still contains a continuous term, it is approximated using the maximum of the endpoints of the segment

$$t_{i+1} - t_i \leq \min\left(\frac{R_{d/Kc}(t_i) + \varepsilon}{\max\left(|\boldsymbol{v}_{d/c}(t_i)|, |\boldsymbol{v}_{d/c}(t_{i+1})|\right)}, \Delta t_{\max}\right) \tag{31}$$

This approximation is exact when $\boldsymbol{v}$ is monotonic on $[t_i, t_{i+1}]$ and is otherwise reasonable due to the slow speed of orbital dynamics, as long as some $\Delta t_{\max}$ is selected that is small relative to the dynamics. Additionally, since the timestep must be smaller in higher-risk situations with a higher velocity or lower distance, the period covered by the step is more likely to be monotonic and thus the approximation exact.

The method solves the optimization problem, updating the time discretization each iteration until Equation 31 is satisfied at all sample points. Separate sets of sample points are maintained for each other satellite, as it is unnecessary to optimize over dense sampling times if a satellite is far away. The complete method is given in Algorithm 1 and Algorithm 2.

Algorithm 1 Iterative Shield Solution

$\boldsymbol{t}^{(c)} \leftarrow [t_0 : \Delta t_{\max} : t_{\max}]\ \forall c \in \{0,...,N\}, c \neq d$ ▷ Seed the sample time vector for all satellites.
while not converged **do**
  solution ← solve Equation 23-26 ▷ Solve the optimization problem with discretization $\boldsymbol{t}$
  if Equation 31 $\forall\ t_i \in \boldsymbol{t}\ \forall c \in \{0,...,N\}, c \neq d$ **then** ▷ Check for possible interstep violations across satellites
    return solution
  else
    for $c \in \{0,...,N\}, c \neq d$ **do**
      $\boldsymbol{t}^{(c)} \leftarrow$ RESAMPLETIMES(solution$^{(c)}$) ▷ Resample the time vector for each satellite with Algorithm 2
    end for
  end if
end while

Algorithm 2 Resample Times

function RESAMPLETIMES(solution, $\kappa = 0.1$)
  $\boldsymbol{t}' \leftarrow [t_0]$
  $\Delta t_{\text{interp}}(t) \leftarrow$ linear interpolation of the first argument of Equation 31 at all points in solution
  if $\boldsymbol{t}'_{-1} \geq t_0 + t_{\max}$ **then** ▷ Check if the desired timespan has been resampled
    $\boldsymbol{t}'_{-1} = t_0 + t_{\max}$
    return $\boldsymbol{t}'$
  else
    $t_i = \boldsymbol{t}'_{-1}$
    $t_{\text{new}} \leftarrow t_{\text{new}} - t_i = (1-\varepsilon) \min_{t \in [t_i, t_{\text{new}}]} \Delta t_{\text{interp}}(t)$ ▷ Find the next time satisfying the required Δt interpolator
    if $\exists\ t_{\text{old}} \in \boldsymbol{t}$ s.t. $t_i < t_{\text{old}} < t_{\text{new}}$ **then** ▷ Check for existing sample from $\boldsymbol{t}$ to add
      append($\boldsymbol{t}', t_{\text{old}}$)
    else ▷ Otherwise, add the new sample
      append($\boldsymbol{t}', t_{\text{new}}$)
    end if
  end if
end function

The purpose of retaining all old sample points when resampling the time discretization in Algorithm 2 is to prevent chatter between two different classes of solutions with different discretizations. By retaining the old points, the constraints that lead to previous iterations' solutions are always maintained.

5. RESULTS: SINGLE-AGENT INSPECTION

The single-servicer ($N = 1$) formulation of the problem is examined first. Policies are trained and benchmarked for a range of fuel use penalties α in order to expose the Pareto front between fuel use and inspection time; this Pareto front is expected from the interaction between the explicit penalty on fuel use and the implicit penalty on time caused by discounting. The performance and effectiveness of a chosen policy is investigated on various cases with and without the shield.

5.1 Training a Single-Agent Policy

PPO is used to train policies in the single-agent environment. A hyperparameter search finds discount rate $\gamma = 0.9999$, learning rate 5×10^{-6}, and batch size 4650 perform well, along with RLlib defaults for other hyperparameters. Seven single-agent policies are trained for fuel use penalty $\alpha \in [0.0 : 0.05 : 0.3]$. Training is performed until the policy converges on 32 cores; this takes 48 to 96 hours for each policy, yielding at least 10M and up to 20M environment interactions.

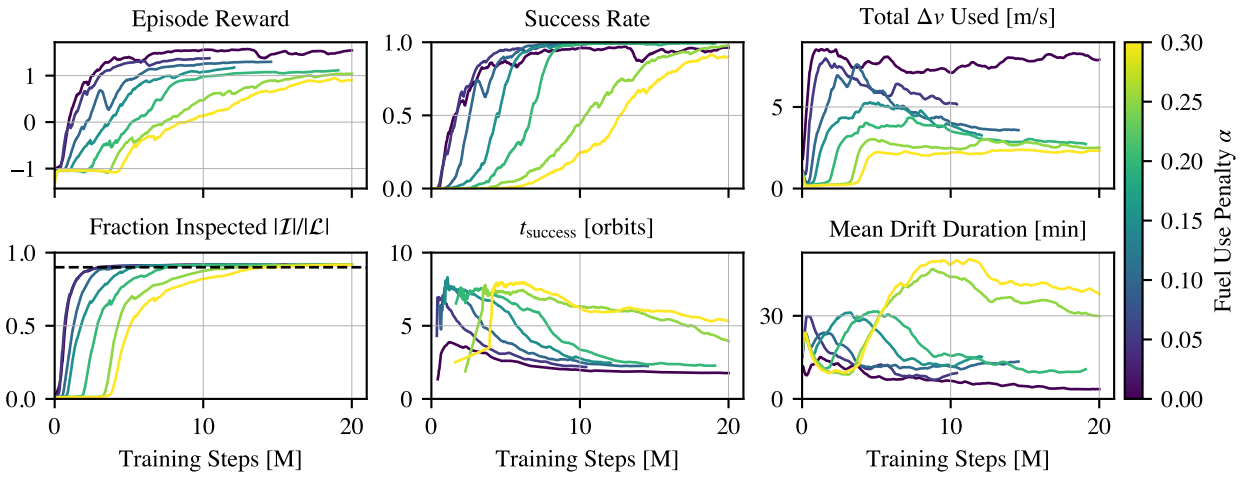


Fig. 4: Training curves for the single-agent environment.

Fig. 4 shows how various metrics evolve as the policies learn to complete the task. As expected, policies for lower α are faster (lower t_{success}) but less fuel-efficient (higher total Δv used) at completing the task. The policies tend to prefer relatively short drifts between a high number of small impulsive thrusts, even when additional policies were trained with a constant per-thrust penalty to discourage this behavior.

5.2 Unshielded Single-Agent Inspection Performance

To evaluate the performance of the policies across a range of cases, each policy is executed in 1000 random instances of the environment. Each instance samples a different orbit for the RSO and the initial relative position for the servicer.

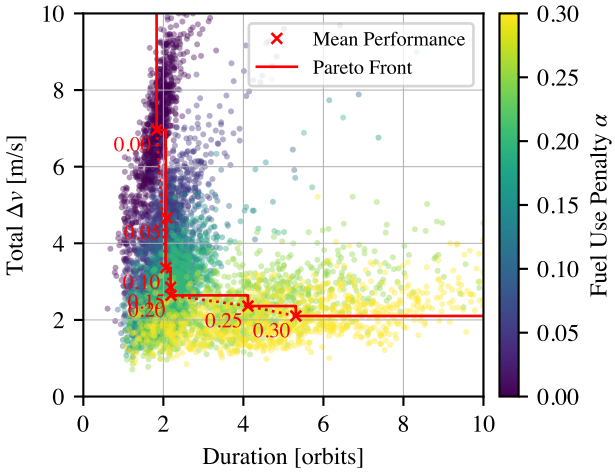


Fig. 5: Unshielded performance of single-agent policies across fuel use penalty α, evaluated at 1000 trials per α.

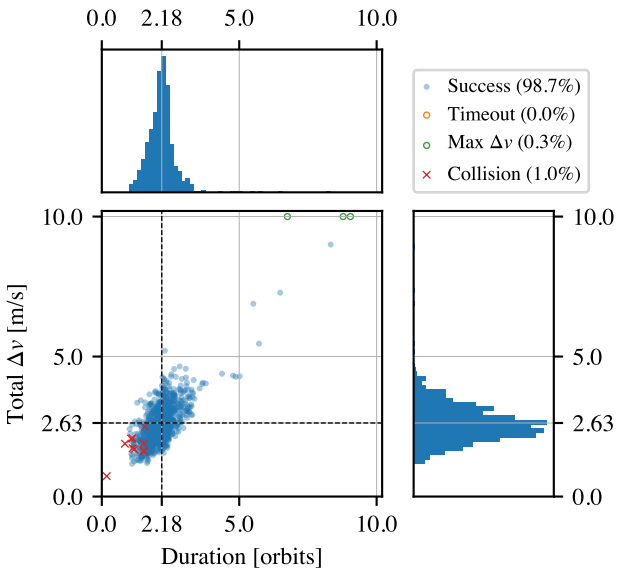
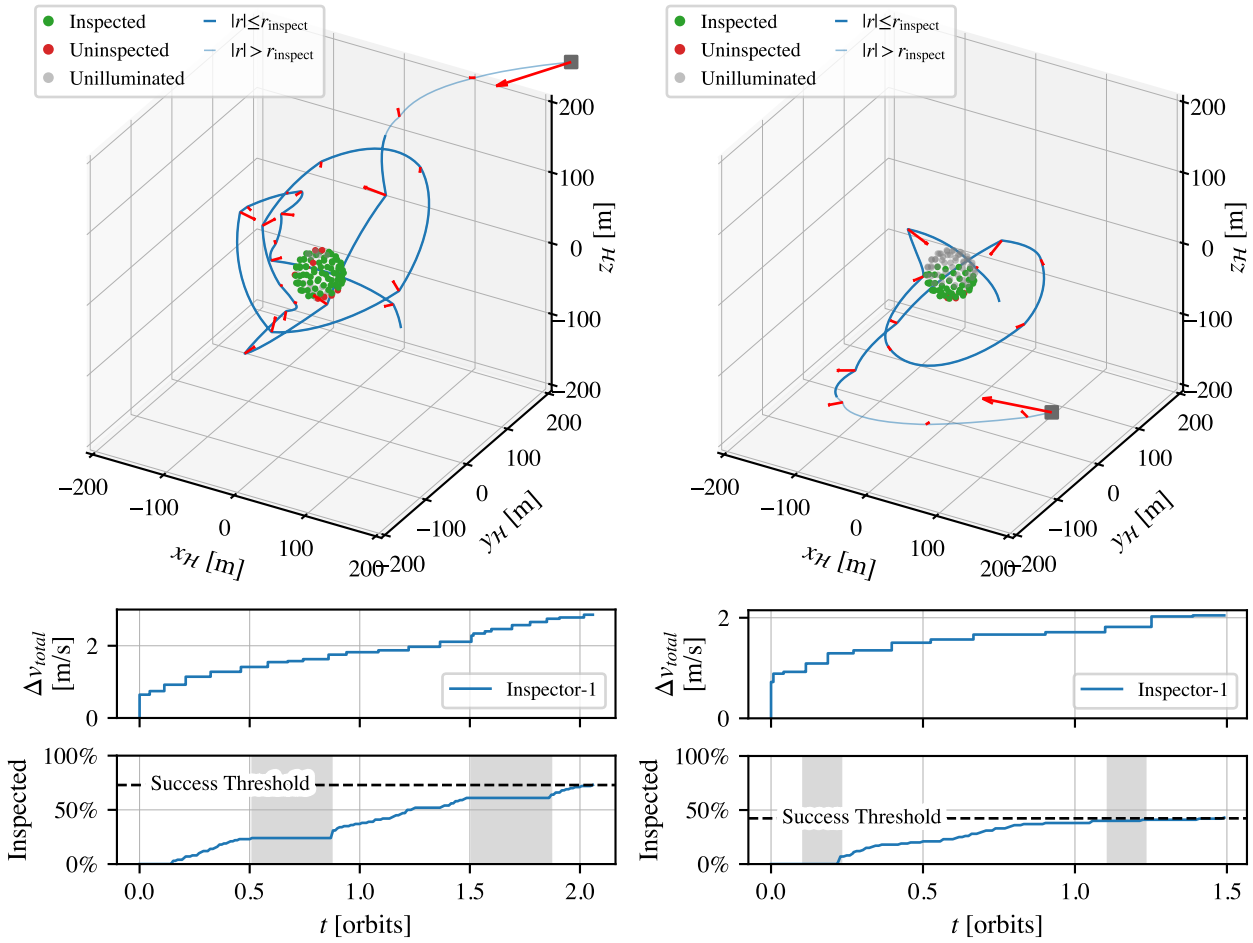


Fig. 6: Unshielded performance of the $\alpha = 0.20$ single-agent policy.

Fig. 5 shows the distribution of unshielded performances for each of the 7 policies. A clear Pareto front between fuel use and inspection time is evident across the policies, with the mean of each policy's performance marked in red. While some improvement in inspection time or fuel use can be found, most of the performance space is dominated by policies near the lower-left corner. This indicates that within the bounds of this formulation, there is a limited trade space between fuel and time optimality.

The $\alpha = 0.20$ policy is selected for further benchmarking, as it balances inspection time and fuel use well. More detail of the benchmark is given in Fig. 6. The task is successfully completed within the time and fuel bounds 99.7% of the time. On average, the servicer uses 2.63 m/s of fuel to complete the inspection in 2.18 orbits. Almost all the cases are successful with respect to the time and fuel limits. However, this benchmark shows the necessity of the shield. Despite learning that collisions between the servicer and RSO are negative, 1% of the cases still ended in collision.



(a) High illumination case, $\beta = 7.6°$.

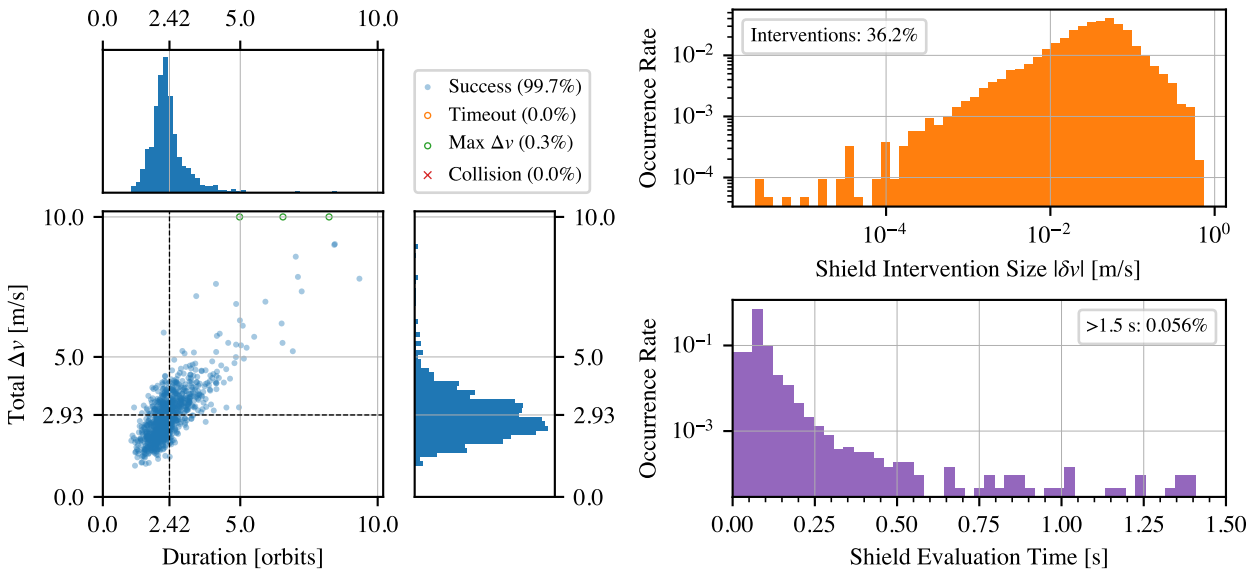
(b) Low illumination case, $\beta = -62.9°$.

Fig. 7: Trajectories generated by the unshielded $\alpha = 0.20$ policy.

Two instances of the $\alpha = 0.20$ policy's behavior—a high illumination and low illumination case—are given in Fig. 7. The servicer starts relatively close to the RSO in both cases but still requires initial large thrusts to move towards the inspection area. In the high illumination case, the servicer makes three passes tracking the illuminated side of the RSO. In the low illumination case, one side is almost always illuminated, meaning that the servicer only needs to maneuver around that side of the RSO to fully inspect it.

5.3 Shielded Single-Agent Inspection Performance

The $\alpha = 0.20$ single-agent policy is benchmarked again, this time with the shield enforcing a 10-meter spherical keepout around the RSO. The initial time discretization is 500 seconds, with a 4-orbit horizon. The results are given in Fig. 8a, with shield evaluation statistics in Fig. 8b. Most importantly, the shield eliminates the three collisions present in the unshielded benchmark (Fig. 6). The shield is shown to be fairly unintrusive. While 36.7% of actions are effected by the shield, the magnitude of the interventions averages 0.052 m/s. The overall impact across scenarios is an increase of 0.24 orbits and 0.30 m/s to complete the inspection task. The worst impact is that a higher fraction of cases (2%)



(a) Performance histograms.

(b) Shield evaluation statistics.

Fig. 8: Shielded performance of the $\alpha = 0.20$ single-agent policy.

do not succeed within the fuel and time bounds. Computationally, the shield takes on average 0.079 seconds to solve, with only a small fraction taking more than half a second. Even if the shield did not converge before a new action was planned, the servicer would already be on a passively safe trajectory and could attempt again for a later thrust time.

A benefit of the shield formulation is that a more conservative keepout region than that used in training can be enforced in deployment. In Fig. 9, a trajectory is generated with the shield enforcing a 100-meter spherical keepout around the RSO; the initial conditions are the same as those in Fig. 7a. While the inspection task is completed less efficiently because the shield is intervening more aggressively, the RSO is still successfully inspected and the 100-meter keepout radius is respected over the entire duration of the task.

6. RESULTS: MULTI-AGENT INSPECTION

Next, a multi-agent formulation of the problem with $N = 2$ servicers is considered. In the first approach, the single-agent policy is deployed in the multi-agent environment, using the agents' joint observation of inspection level to attempt to induce collaboration. In the second approach, policies are trained in the multi-agent environment so that they actively learn to work together. These policies are trained and benchmarked across a range of fuel use penalties α, both with and without the shield.

6.1 Single-Agent Policy in Multi-Agent Deployment

A first attempt at multi-agent inspection is inspired by [21], in which multiple Earth-observing satellites trained in a single-agent environment were successfully deployed in a multi-agent environment. In that work cooperation was induced via a shared task completion list. In this work, two servicers similarly execute a single agent policy while sharing their observation of the joint inspection status of the RSO.

This method is benchmarked unshielded (Fig. 10a) and shielded (Fig. 10b) across 1000 cases. The mean inspection time is significantly lower than in the single agent case using the same policy, but at the cost of twice as much fuel used per time, as the agents could not know to proactively save fuel due to the other agent's presence. The decrease in time is likely reflective of the challenge faced by a single agent, in which a few edge points are barely missed by the servicer (such as in Fig. 7b); with two agents, even on very similar trajectories, the effectively increased field of view can eliminate those cases. Ultimately, the performance only barely dominates the single-agent Pareto front in Fig. 5.

This benchmark is the strongest test of the shield, which successfully eliminates very frequent collisions (15% of cases) between the two servicers in all cases, even though the servicers have no knowledge of the other's existence or

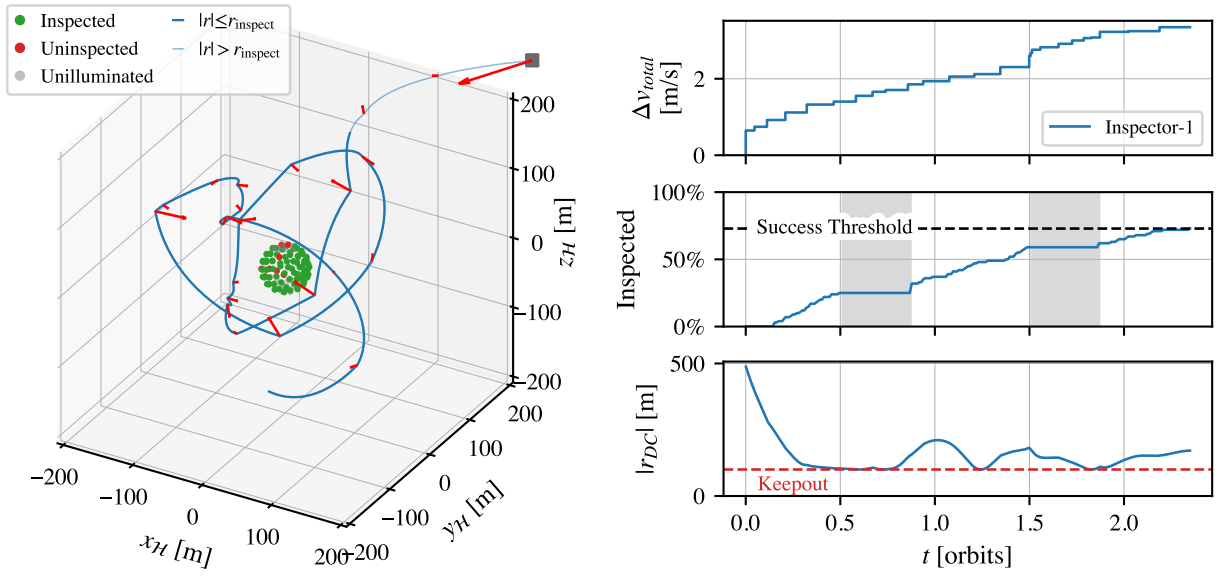
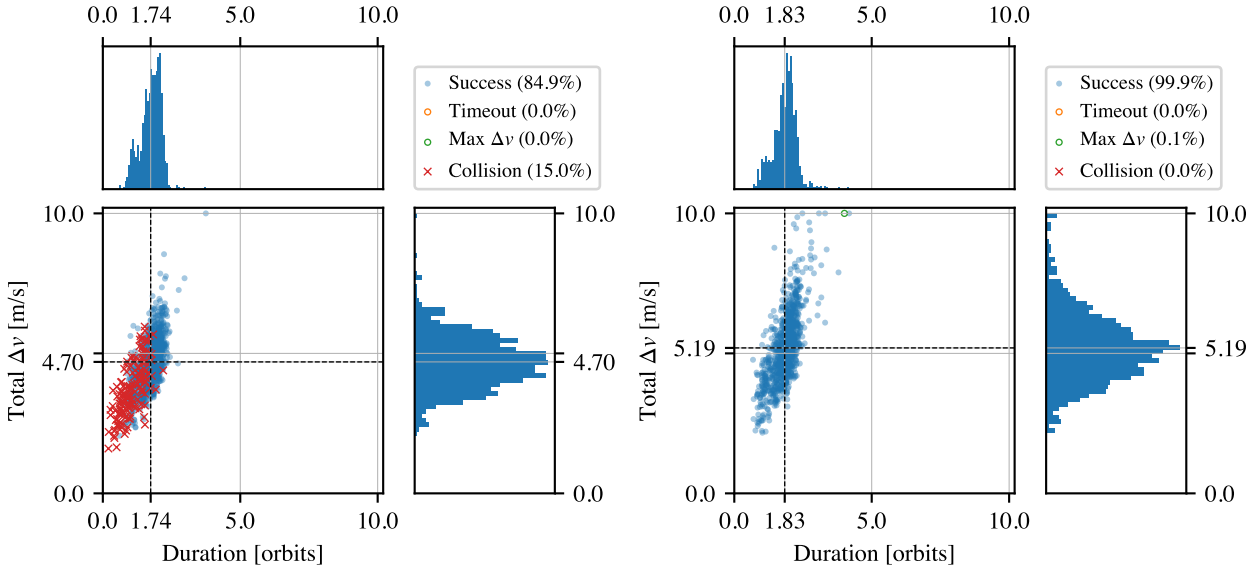


Fig. 9: A trajectory generated by the $\alpha = 0.20$ policy with a 100-meter spherical keepout.



(a) Unshielded benchmark.

(b) Shielded benchmark.

Fig. 10: Performance of the $\alpha = 0.20$ single-agent policy in the multi-agent environment.

intrinsic motivation to avoid collision. As a result, the shield increase fuel use by an average of 0.51 m/s. Clearly, the shield is very necessary for this architecture.

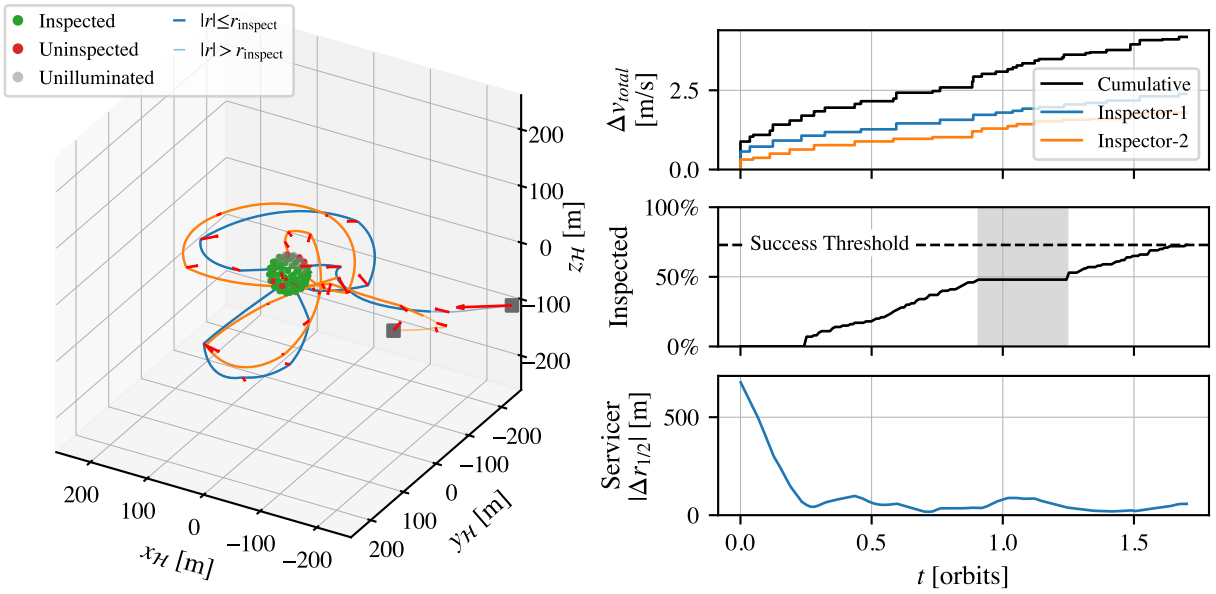


Fig. 11: A trajectory generated by the $\alpha = 0.20$ single-agent policy in a multi-agent setting.

The performance shown in the benchmark is tempered by an undesirable behavior frequently encountered when using this method, such as in the case shown in Fig. 11. Even though a stochastic policy is being used, the distribution of actions it has learned to select is narrow. As a result, the agents spend initial effort reaching an inspection position that has good illumination, then both follow a similar trajectory over the course of the episode (see the low value of the distance between servicers $|\boldsymbol{r}_{1/2}|$ over time in Fig. 11). While the slight difference in trajectory completes the task quickly, it is a poor use of two servicers since the images they collect at any given time are likely to be very similar. This behavior also explains the high number of collisions between servicers in the unshielded benchmark, as the agents want to be in the same location at the same time.

6.2 Multi-Agent Training

The undesirable behavior of a single-agent policy in the multi-agent setting motivates the need for policies trained in the multi-agent environment that can actively diversify the trajectories being executed. A single policy is learned that all agents contribute experience to. As in the single-agent training, PPO is used with the necessary modifications for asynchronicity and sMDPs; the same hyperparameters are also used.

In Fig. 12, the multi-agent training pipeline is shown to be successful. Training converged in a reasonable time for three values of α (0.0, 0.1, 0.2), while $\alpha = 0.3$ was slower to train than in the single-agent case. Overall, the trends are comperable to those seen in single-agent training (Fig. 4).

6.3 Multi-Agent Inspection Performance

As in the single-agent case, a benchmark is performed over 1000 trials for the unshielded and shielded $\alpha = 0.20$ policy; the results are given in Fig. 13a and Fig. 13b, respectively. Unlike with the multi-agent-deployed single agent policies (Fig. 10a and 10b), the fuel use does not increase proportionally to the number of agents, implying that coordination has successfully led to fuel efficiency. However, the inspection time is worse than the single-agent case. This means that the policy does not dominate the single-agent Pareto front in Fig. 5. The multi-agent shield is again shown to effectively prevent collisions between all agents, at the cost of 0.57 m/s and 0.60 orbits, on average.

Fig. 14 shows how the multi-agent policy yields more desirable behavior than the single agent policy in the same multi-agent setting as Fig. 11. The two agents travel on significantly different trajectories, often with one inspecting while the other is out of inspection range on a larger passive maneuver. In the case of a spherical RSO this may be

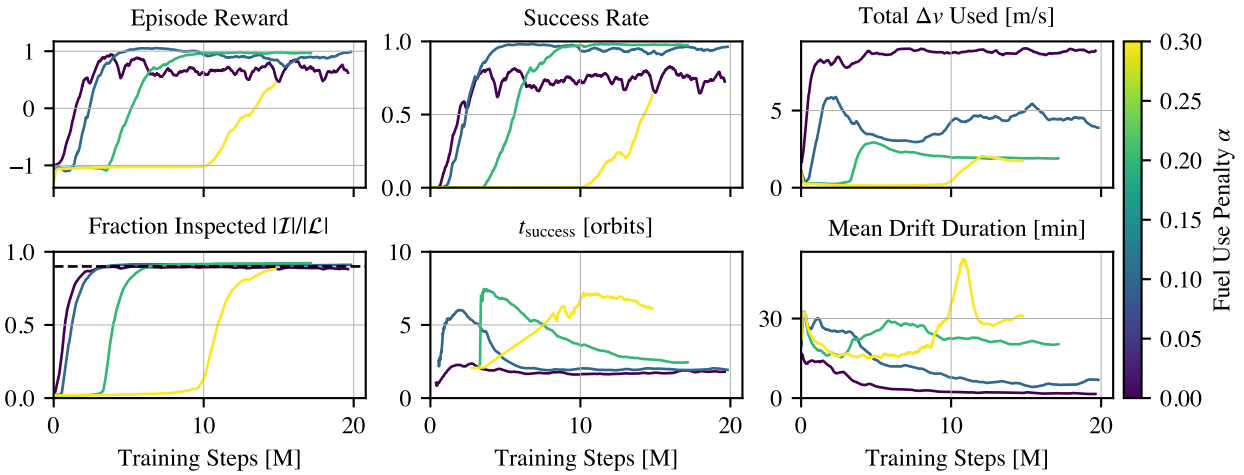
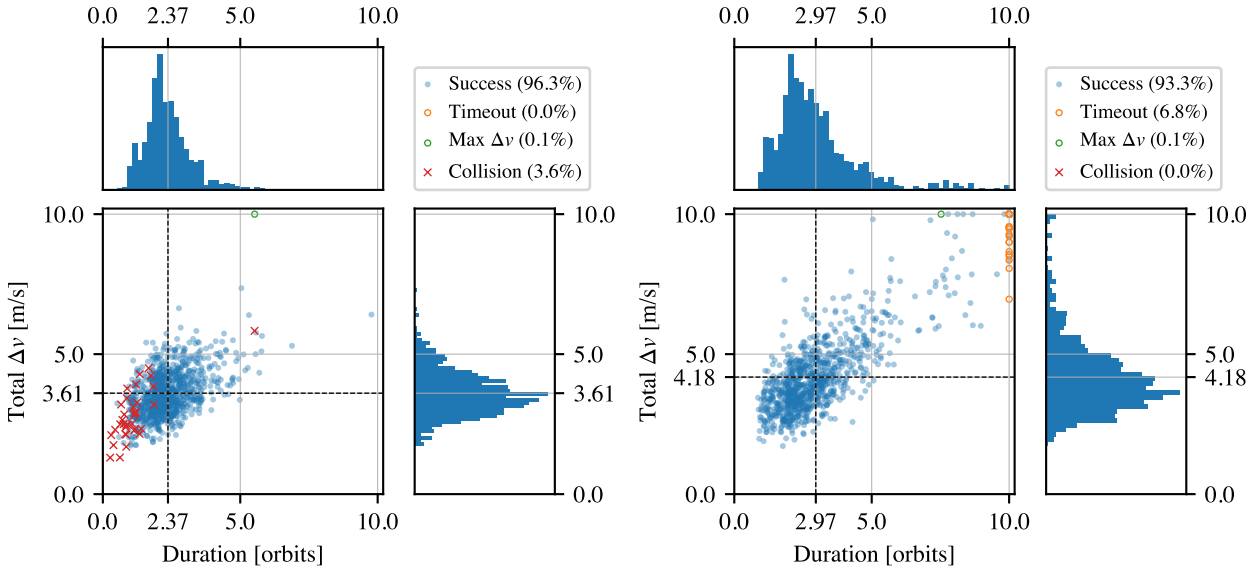


Fig. 12: Training curves for the multi-agent environment.



(a) Unshielded benchmark.

(b) Shielded benchmark.

Fig. 13: Performance of the $\alpha = 0.20$ multi-agent policy.

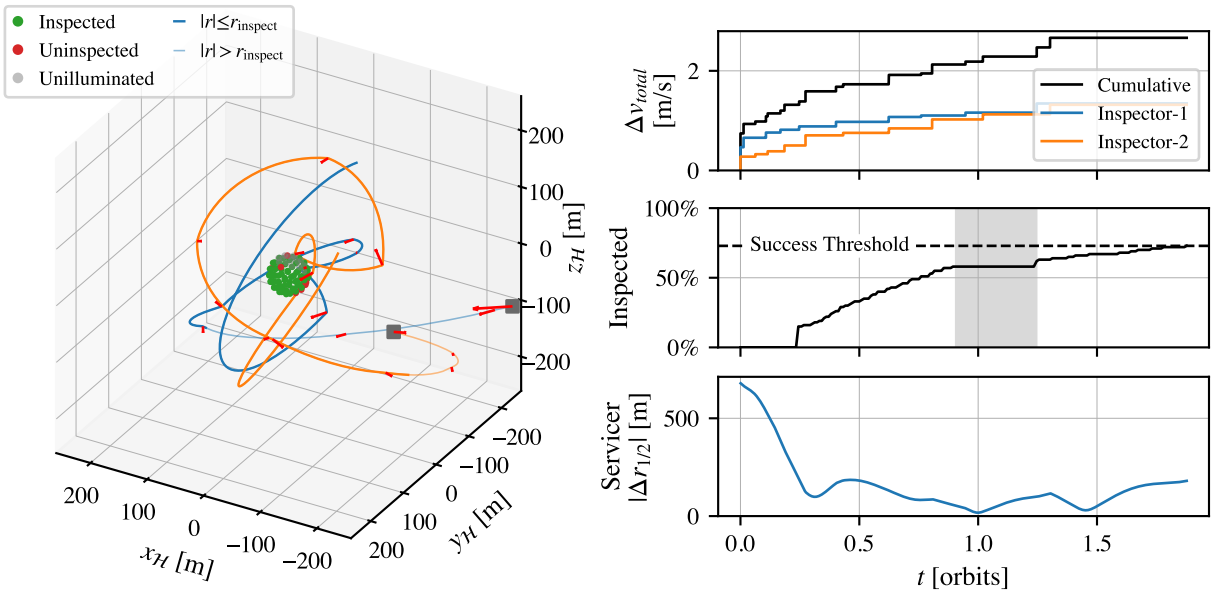


Fig. 14: A trajectory generated by the $\alpha = 0.20$ multi-agent policy, with the same initial conditions as Fig. 11.

unnecessary, but as more complex geometries are considered, agents that can effectively diversify their trajectories will be necessary.

7. CONCLUSIONS

This work demonstrates that RL can effectively learn policies for the RSO inspection task and that safety can be guaranteed even in multi-agent environments with eccentric orbits.

In the single agent environment, policies learn what parts of the RSO will be illuminated and how to track the illuminated region over the course of an orbit. The fuel-time Pareto front shows that there is limited ability to choose between fuel efficiency and time efficiency, with most solutions being dominated by those from a small range of $\alpha \in [0.10, 0.20]$. For this formulation of the problem, this indicates that operators have relatively little choice in how to complete the task. The overall performance is strong, using only 10-20% of the δv consumed in the continuous-thrust formulation of the problem in [22].

For the simple sphere-like satellite considered here, multi-agent inspection is possible but not particularly beneficial. The two methods of multi-agent tasking—single agent policies with a joint observation of inspection, and multi-agent policies—yield different efficiencies and types of solutions, but neither strongly dominates the single-agent Pareto front relative to the cost of having two servicers present. Most likely, the additional fuel cost for collision avoidance from the second servicer greatly outweighs any marginal improvements in speed, and may even impede speedy inspection. However, the method with multi-agent-trained policies is probably necessary for more complex RSO geometries, such as a space station, where diversifying the inspection trajectories would be required to effectively complete the task.

Across all cases, the shield formulation and solution method guarantee the safety of the agents. In cases where the policy learned to avoid collisions, the shield's interventions are minimal and only slightly deteriorate performance. Even in cases where the policy leads to frequent collisions and near-misses or a greater degree of safety is desired than the policy learned in training, the shield is able to maintain safety. This performance is aligned with the analysis of relative orbital dynamics being leveraged.

8. APPENDIX

8.1 Acknowledgements

This work is supported by NASA Space Technology Graduate Research Opportunity grant 80NSSC23K1182.

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).

8.2 Code Availability

Basilisk (`avslab.github.io/basilisk/`) and BSK-RL (`avslab.github.io/bsk_rl/`) are open-source projects. The RSO inspection environment implementation can be found at `avslab.github.io/bsk_rl/examples/rso_inspection.html`. The asynchronous, sMDP-discounted multi-agent training pipeline can be found at `avslab.github.io/bsk_rl/examples/async_multiagent_training.html`.

9. REFERENCES

[1] M. Sabatini, R. Volpe, and G.B. Palmerini. Centralized visual based navigation and control of a swarm of satellites for on-orbit servicing. *Acta Astronautica*, 171:323–334, June 2020.

[2] Benjamin Bernhard, Changrak Choi, Amir Rahmani, Soon-Jo Chung, and Fred Hadaegh. Coordinated Motion Planning for On-Orbit Satellite Inspection using a Swarm of Small-Spacecraft. In *2020 IEEE Aerospace Conference*, pages 1–13, March 2020.

[3] Yashwanth Kumar Nakka, Wolfgang Hönig, Changrak Choi, Alexei Harvard, Amir Rahmani, and Soon-Jo Chung. Information-Based Guidance and Control Architecture for Multi-Spacecraft On-Orbit Inspection. *Journal of Guidance, Control, and Dynamics*, 45(7):1184–1201, July 2022.

[4] Timothy Lauinger and Steve Ulrich. Path Planning for Optimal Coverage of Orbiting Space Structures Using Lissajous Curves. In *AAS/AIAA Space Flight Mechanics Meeting*, January 2025.

[5] Markus Iversflaten, Alex Hansson, David Sternberg, Oliver Jia-Richards, and Keenan Albee. Robust Replanning for Multi-Agent SmallSat Inspection in Failure Scenarios. In *AIAA SCITECH 2025 Forum*, Orlando, FL, January 2025. American Institute of Aeronautics and Astronautics.

[6] Keenan Albee, David C Sternberg, Alexander Hansson, David Schwartz, Ritwik Majumdar, and Oliver Jia-Richards. Architecting Autonomy for Safe Microgravity Free-Flyer Inspection. In *IEEE Aerospace Conference*, March 2025.

[7] Joshua Aurand, Steven Cutlip, Henry Lei, Kendra Lang, and Sean Phillips. Exposure-Based Multi-Agent Inspection of a Tumbling Target Using Deep Reinforcement Learning, May 2023.

[8] David van Wijk, Kyle Dunlap, Manoranjan Majji, and Kerianne L. Hobbs. Deep Reinforcement Learning for Autonomous Spacecraft Inspection using Illumination, August 2023.

[9] Henry H. Lei, Matt Shubert, Nathan Damron, Kendra Lang, and Sean Phillips. Deep Reinforcement Learning For Multi-Agent Autonomous Satellite Inspection. In Matt Sandnas and David B. Spencer, editors, *Proceedings of the 44th Annual American Astronautical Society Guidance, Navigation, and Control Conference, 2022*, pages 1391–1412, Cham, 2024. Springer International Publishing.

[10] Henry Lei, Zachary S Lippay, Anonto Zaman, Joshua Aurand, Amin Maghareh, and Sean Phillips. Stability Analysis of Deep Reinforcement Learning for Multi-Agent Inspection in a Terrestrial Testbed. In *AIAA SCITECH 2025 Forum*, Orlando, FL, January 2025.

[11] Andrea Brandonisio, Michèle Lavagna, and Davide Guzzetti. Reinforcement Learning for Uncooperative Space Objects Smart Imaging Path-Planning. *The Journal of the Astronautical Sciences*, 68(4):1145–1169, December 2021.

[12] Andrea Brandonisio, Lorenzo Capra, and Michèle Lavagna. Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target. *Advances in Space Research*, page S0273117723005276, July 2023.

[13] Hanna Krasowski, Jakob Thumm, Marlon Müller, Lukas Schäfer, Xiao Wang, and Matthias Althoff. Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking, November 2023.

[14] Tommaso Guffanti, Daniele Gammelli, Simone D'Amico, and Marco Pavone. Transformers for Trajectory Optimization with Application to Spacecraft Rendezvous. In *2024 IEEE Aerospace Conference*, pages 1–13, Big Sky, MT, USA, March 2024. IEEE.

[15] Yuji Takubo, Tommaso Guffanti, Daniele Gammelli, Marco Pavone, and Simone D'Amico. Towards Robust Spacecraft Trajectory Optimization via Transformers, October 2024.

[16] Ritwik Majumdar, David C. Sternberg, Keenan Albee, and Oliver Jia-Richards. Demonstration of the Dyna Reinforcement Learning Framework for Reactive Close Proximity Operations. In *AIAA SCITECH 2025 Forum*, Orlando, FL, January 2025. American Institute of Aeronautics and Astronautics.

[17] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe Reinforcement Learning via Shielding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018.

[18] Andrew Harris, Trace Valade, Thibaud Teil, and Hanspeter Schaub. Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning. *Journal of Spacecraft and Rockets*, 59(2):611–626, March 2022.

[19] Adam Herrmann and Hanspeter Schaub. Reinforcement Learning for Small Body Science Operations. In *AAS Astrodynamics Specialist Conference*, Charlotte, North Carolina, August 2022.

[20] Mark Stephenson and Hanspeter Schaub. Reinforcement Learning For Earth-Observing Satellite Autonomy With Event-Based Task Intervals. In *AAS Rocky Mountain GN&C Conference*, Breckenridge, CO, February 2024.

[21] Mark Stephenson, Lorenzzo Mantovani, Anaïs Cheval, and Hanspeter Schaub. Quantifying the Optimality of a Distributed Rl-Based Autonomous Earth-Observing Constellation. In *AAS GN&C Conference*, Breckenridge, CO, February 2025.

[22] David Van Wijk, Kyle Dunlap, Manoranjan Majji, and Kerianne Hobbs. Safe Spacecraft Inspection via Deep Reinforcement Learning and Discrete Control Barrier Functions. *Journal of Aerospace Information Systems*, 21(12):996–1013, December 2024.

[23] Kyle Dunlap, Nathaniel Hamilton, and Kerianne L Hobbs. Deep Reinforcement Learning for Scalable Multiagent Spacecraft Inspection. In *AAS/AIAA Space Flight Mechanics Meeting*, January 2025.

[24] Daniel Morgan, Soon-Jo Chung, and Fred Y. Hadaegh. Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740, November 2014.

[25] Daniel Morgan, Giri P Subramanian, Soon-Jo Chung, and Fred Y Hadaegh. Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *The International Journal of Robotics Research*, 35(10):1261–1285, September 2016.

[26] G. Gaias and S. D'Amico. Impulsive Maneuvers for Formation Reconfiguration Using Relative Orbital Elements. *Journal of Guidance, Control, and Dynamics*, 38(6):1036–1049, June 2015.

[27] Adam W. Koenig and Simone D'Amico. Robust and Safe N-Spacecraft Swarming in Perturbed Near-Circular Orbits. *Journal of Guidance, Control, and Dynamics*, 41(8):1643–1662, August 2018.

[28] Mark Stephenson, Daniel Huterer Prats, and Hanspeter Schaub. Autonomous Satellite Inspection in Low Earth Orbit with Optimization-Based Safety Guarantees. In *International Workshop on Planning & Scheduling for Space*, Toulouse, France, April 2025.

[29] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. AIAA Education Series. American Institute of Aeronautics and Astronautics, Inc, Reston, VA, fourth edition edition, 2018.

[30] Richard S. Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts London, England, 2 edition, 2018.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

[32] Steven Bradtke and Michael Duff. Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994.

[33] Kunal Menda, Yi-Chun Chen, Justin Grana, James W. Bono, Brendan D. Tracey, Mykel J. Kochenderfer, and David Wolpert. Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes. *IEEE Transactions on Intelligent Transportation Systems*, 20(4):1259–1268, April 2019.

[34] Mark Towers, Jordan K Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Jin Shen Tan, and Omar G. Younis. Gymnasium, October 2023.

[35] Mark A Stephenson and Hanspeter Schaub. BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking. In *75th International Astronautical Congress*, Milan, Italy, October 2024.

IAF.
[36] Patrick W. Kenneally, Scott Piggott, and Hanspeter Schaub. Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework. *Journal of Aerospace Information Systems*, 17(9):496–507, September 2020.
[37] Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for Distributed Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3053–3062, June 2018.