



Two Approaches to Autonomous Spacecraft Inspection: Shielded Reinforcement Learning versus Probabilistic Roadmaps

Mark Stephenson¹ · Hanspeter Schaub¹

Received: 28 January 2026 / Accepted: 24 August 2026

© The Author(s) 2026

Abstract

The close-proximity inspection of objects in low Earth orbit (LEO) is important to operations such as rendezvous, debris removal, servicing, and resident space object (RSO) characterization, all of which are of increasing interest to commercial and government organizations. Complex relative motion dynamics in eccentric LEO make the problem of path planning for autonomous inspection challenging. Agents must be able to fully inspect an object subject to illumination constraints while avoiding collision with the RSO. In this work, two approaches to the problem are developed and compared: First, a deep reinforcement learning (RL)-based solution that allows for closed-loop autonomous inspection is formulated, using analysis of equations of motion to derive a shield that guarantees safe actions. Second, a solver using probabilistic roadmaps (PRMs) and mixedinteger quadratically-constrained quadratic programs (MIQCQPs) is designed to provide more performant but computationally expensive open-loop inspection trajectories. The overall performance of solutions produced by the two methods is compared across the fuel-time trade space in a high-fidelity simulation environment: These experiments demonstrate that while PRMs result in more fuel- and time-efficient solutions than RL, this comes at the cost of offline, open-loop solutions as opposed to the closed-loop, onboard control provided by RL-based policies.

Keywords Spacecraft inspection · Reinforcement learning · Path planning · Relative motion · Autonomy

✉ Mark Stephenson
Mark.A.Stephenson@colorado.edu
Hanspeter Schaub
Hanspeter.Schaub@colorado.edu

¹ Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 3775 Discovery Dr., Boulder, CO 80304, USA

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.

A variety of path planning methods have been proposed for the inspection task. References [1–3] 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. Reference [4] decomposes a large RSO into primitive shapes and projects inspection paths onto them. Such two-phase methods introduce significant suboptimality by constraining the solution space to certain classes of trajectories.

Sample-based motion planners offer a more general framework for trajectory planning and optimization. Algorithms building on rapidly-exploring random tree RRT (RRT), such as stable sparse RRT (SST) [5], among many others, can solve kinodynamically-constrained path planning problems across domains. Considering RPO tasks specifically, traditional A*-based path planning methods are combined with methods for robustness to thruster failures in References [6] and [7]. However, for the inspection task, the complex goal state of complete RSO observation makes the problem requirements considerably more complex than point-to-point motion planning is able to efficiently consider because the goal of complete coverage can only be represented with a very high-dimensional state.

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 Reference [8], a neural network-based policy is used for high-level planning between predetermined inspection waypoints. Van Wijk [9] approaches the problem with a continuous action space for continuous, low-thrust control, considering lighting conditions. In Lei's work [10, 11], the waypoint-based approach is used for distributed and centralized control of a swarm of servicers. Brandonisio [12, 13] and Quinn [14] consider the problem 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 policies have been developed [15]. Several references combine RL with trajectory optimization steps: References [16] and [17] leverage

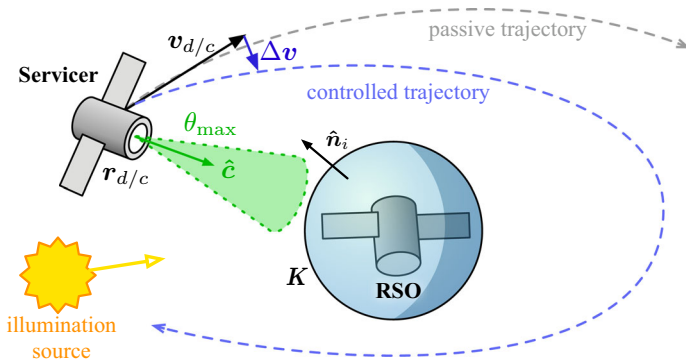


Fig. 1 The RSO inspection task

the transformer architecture for RPO, while Majumdar [18] uses RL with traditional path planning methods as a way of directing the search of the traditional methods while maintaining robustness. Shielded RL [19] has been demonstrated as an effective method for various other spacecraft tasking problems, including resource management [20], small-body proximity operations [21], and agile Earth observation under various conditions [22, 23]. Control barrier functions (CBFs) are an alternative to shields that provide guarantees on performance: Most similar to this work, papers by Van Wijk and Dunlap [24, 25] ensure safety in RL-controlled inspection task with continuous control inputs using a CBF. While Van Wijk's paper is most similar to this work, a few key differences are present: We consider discrete burn events rather than continuous control, and our environment and safety constraints are formulated for an ellipsoidal RSO in an eccentric orbit with infinite-horizon safety guarantees, rather than a spherical RSO in a circular orbit with finite-horizon safety.

To develop safety guarantees for these tasks, existing work on safe coordinated motion planning in LEO can be leveraged. Morgan [26, 27] uses sequential convex programming (SCP) to plan collision-free continuous thrust trajectories. References [28, 29] use analytical methods to derive swarm reconfiguration laws with impulsive thrusts in terms of relative orbital elements.

Across the literature, fast (< 1 orbit) and high- Δv cost (dozens of m/s) solutions are generated. This paper aims to find a slower and more fuel-efficient class of solutions that leverage natural relative motion behaviors in LEO. The RSO inspection problem is considered for an RSO in an eccentric orbit with a Hill-frame-fixed attitude and an impulsive thrusting servicer, subject to range, pointing, and lighting constraints for inspection (Fig. 1). This paper develops and compares solutions using deep reinforcement learning (DRL) and probabilistic roadmaps (PRMs). The DRL approach generates closed-loop policies that have safety guarantees via an optimization-based shield that is derived from the analytical equations of relative motion. The second approach produces open-loop solutions using PRMs and mixed-integer quadratically-constrained quadratic programs (MIQCQPs), which yields trajectories that are more performant but computationally more expensive to produce. Earlier versions of the shielded RL and PRM methods are developed in earlier conference papers (Reference

[30] and [31], respectively); this work presents a deeper analysis comparing the two approaches. To evaluate the utility of these methods, they are tested in a high-fidelity simulation environment, analyzing their performance in the fuel-time trade space. While PRMs yields marginally more optimal open-loop solutions, the closed-loop execution of the RL policies may prove more useful in uncertain environments.

2 Problem Statement

In the inspection task, a servicer spacecraft has the objective of inspecting all facets of an RSO in LEO, subject to illumination constraints. To achieve this, the servicer must use discrete thrusts to control its motion relative to the RSO while avoiding collision between the two spacecraft.

2.1 Satellite Dynamics

The RSO (i.e., the chief) is a non-maneuvering satellite in an eccentric LEO orbit with a Hill-frame-fixed, nadir-pointing attitude; this attitude assumption is reflective of the attitude control for many active LEO satellites. The origin of this Hill/body frame $\mathcal{H}$ is used for all displacement vectors unless otherwise noted. 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.

The servicer (i.e., the deputy) is an actively maneuvering satellite operating near the RSO. The Hill-frame state of deputy d relative to the chief c is written as the position $\mathbf{r}_{d/c}$ and velocity $\mathbf{v}_{d/c}$. The servicer is equipped with a body-fixed camera $\hat{\mathbf{c}}$ to inspect the RSO, and the attitude is actively controlled to point the instrument at the RSO (i.e., $\hat{\mathbf{c}}$ is kept antiparallel to $\mathbf{r}_{d/c}$). The servicer can control its motion with a thruster that can produce impulsive thrusts $\Delta \mathbf{v}$ of magnitude up to $\Delta v_{\max} = 1$ m/s in an arbitrary direction, followed by a passive drift period. 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.

To avoid collision, the deputy must stay out of the chief's keep-out ellipsoid represented by a 3×3 positive definite matrix $\mathbf{K}$ consisting of the reciprocal squared semi-axes with a constraint in the general form

$$\mathbf{r}_{d/c}^T(t) \mathbf{K} \mathbf{r}_{d/c}(t) \geq 1 \quad (1)$$

For a spherical keep-out region of radius R_k (i.e., $\mathbf{K} = \mathbf{I}/R_k^2$), as is primarily considered in this work, this constraint reduces to

$$\|\mathbf{r}_{d/c}(t)\| \geq R_k \quad (2)$$

Analysis of the relative motion dynamics of the system is necessary for the shield and for the PRM-based solution. The motion of the servicer can be 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 $\mathcal{H}$ of the chief satellite C (which is the same as its body frame due to the Hill-fixed attitude assumption), the state transition matrix for the state vector $\mathbf{x}_{d/c} = [\mathcal{H}\mathbf{r}_{d/c}^T \ \mathcal{H}\mathbf{v}_{d/c}^T]^T$ composed of the Hill-frame position and velocity of the deputy relative to the chief is given by

$$\Phi_c^x(t, t_0) = \mathbf{A}_c(t) \Phi_c^{\delta\alpha}(t, t_0) \mathbf{A}_c^{-1}(t_0) \quad (3)$$

where $\mathbf{A}_c(t)$ is the linearized relative orbital element to Hill frame mapping matrix, and $\Phi_c^{\delta\alpha}(t, t_0)$ is the relative orbital element state transition matrix.¹ Left superscripts are used to denote the frame the vector is expressed in.

2.2 Inspection Task

For the inspection task, a set of required inspection points $\mathcal{P}$, the criteria for inspecting them, and the criteria for task completion are established.

The RSO has a set of body-fixed inspection point structures $p \in \mathcal{P}$, each with an associated body-fixed position $\mathbf{r}_p$ (relative to the RSO origin) and normal vector $\hat{\mathbf{n}}_p$, such that $p_p = (\mathbf{r}_p, \hat{\mathbf{n}}_p)$. In this work, the RSO's geometry is modeled as a 1-meter radius sphere of $|\mathcal{P}| = 100$ uniformly distributed points with surface normals in the radial direction. Points are illuminated when the angle between the sun vector $\mathbf{r}_{\text{sun}}$ and the point normal is less than $\phi_{\text{max}} = 60^\circ$ and the RSO is not being eclipsed by the Earth. Illumination can be reduced to a function of the RSO's mean anomaly M :

$$L(p_p, M) = \neg \text{Eclipse}(p_p, M) \wedge \angle(\hat{\mathbf{n}}_p, \mathbf{r}_{\text{sun}}(M) - \mathbf{r}_p) < \phi_{\text{max}} \quad (4)$$

The set $\mathcal{L}$ of points that are illuminated for at least some time during the orbit is thus defined as

$$\mathcal{L} = \{p \in \mathcal{P} \mid \exists M \in [0, 2\pi) \text{ such that } L(p, M) = \text{True}\} \quad (5)$$

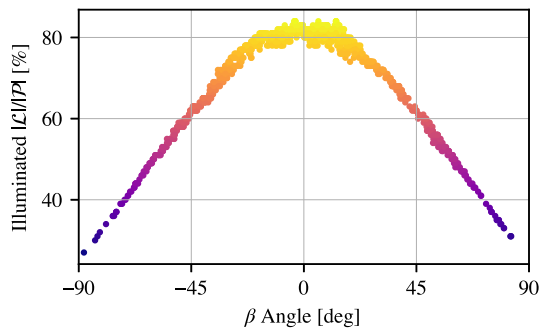
Fig. 2 shows the percent of the RSO that is illuminated over 1000 random cases as a function of the β angle, the angle between the orbital plane and the Sun vector. Orbits that are closer to a Sun-synchronous orbit (SSO) at the dusk-dawn boundary tend to have the smallest fraction of points ever illuminated,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

To inspect an inspection point, the servicer must be in range of the point, the instrument must have an acceptable relative angle, and the point must be illuminated. The range constraint requires that the servicer is within $R_{\text{max}} = 250$ m of the point p_p being inspected:

$$\|\mathbf{r}_{d/c}(M) - \mathbf{r}_p\| < R_{\text{max}} \quad (6)$$

¹ see Reference [32], chapter 14.5 for the full 6×6 matrices for $\mathbf{A}_c$, $\mathbf{A}_c^{-1}$, and $\Phi_c^{\delta\alpha}$

Fig. 2 Fraction of RSO illuminated as a function of orbital β angle



The squint angle constraint requires that the servicer's instrument views the point p_p from an angle no more than $\theta_{\max} = 30^\circ$:

$$\angle(\hat{n}_p, \mathbf{r}_{d/c}(M) - \mathbf{r}_p) < \theta_{\max} \quad (7)$$

And the illumination constraint requires that $L(p, M)$ is True. Combined, these constraints define an inspectability indicator function

$$I(p, \mathbf{r}_{d/c}, M) = \text{Equation 6} \wedge \text{Equation 7} \wedge L(p, M) \quad (8)$$

that is a function only of mean anomaly and servicer position.

When Eq. 8 is satisfied for some point p , the point is added to the inspected set $\mathcal{I}$. The RSO is considered fully inspected and the task is complete when at least 90% of the points in $\mathcal{L}$ are in $\mathcal{I}$. This threshold is set since some points on the limb of the spacecraft may only be instantaneously illuminated. The complete problem is then a constrained multiobjective optimization problem: minimize the fuel consumed and time taken while completing the inspection task.

2.3 Simulation Environment

A simulation of the problem is implemented using BSK-RL,² a package for creating modular, high-fidelity RL environments for spacecraft tasking [33]. The underlying spacecraft simulation is Basilisk,³ a high-performance spacecraft simulation package [34, 35]. Rigid multi-body dynamics in the perturbed orbital environment and flight-proven flight software algorithms are used to simulate the environment. Perturbations include gravitational spherical harmonics up to order and degree 10, solar radiation pressure, and atmospheric drag.

The environment is used for training and validating the shielded RL solutions and validating the PRM solutions. Basilisk is ideal for offline agent training as its high computation speed enables efficient training with a complex physics-based simulation, and BSK-RL implements the Gymnasium API [36] for compatibility with other

² avslab.github.io/bsk_rl/

³ avslab.github.io/basilisk/

RL tooling. Because both methods are evaluated in the same realistic simulation environment, the comparison is apples-to-apples.

3 Reinforcement Learning

The first of the two approaches considered in this paper is RL. RL is a method of solving sequential decision-making problems by first formulating the problem as a Markov decision process (MDP), then using an algorithm to find a policy, or mapping from states to actions, that maximizes the expected sum of future rewards [37]. This problem considers continuous state and action spaces. Notably, the learning agent does not know about the environment explicitly; instead, it must explore for and exploit high-reward regions of the environment through trial and error. This section briefly reviews the formulation and methods developed more completely in Reference [30].

3.1 MDP Formulation

The single-servicer inspection task is formalized as partially-observable semi-Markov decision process (POsMDP) [38–40]. 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 the servicer are encountered at the following conditions:
 - $\geq 90\%$ of illuminated points are inspected, $|\mathcal{I}|/|\mathcal{L}| \geq 0.9$.
 - The servicer collides with the RSO, in violation of Eq. 1.
 - The servicer leaves the region surrounding RSO: $|\mathbf{r}_{d/c}| > 1$ 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.
 - The servicer runs out of available fuel: $\Delta v_{\text{avail}} = 0$.
 - The episode times out: $t \geq 10$ RSO orbits.
- **Observation Space:** The observation for the 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, which are then normalized to be on a similar order of magnitude. Two reference frames are used: the RSO Hill/body frame $\mathcal{H}$, and the Earth-Sun Hill frame $\mathcal{S}$.

The region inspection status $\%_{\text{inspect}}(\mathcal{P})$ is a compact way of representing what points on the RSO have been jointly inspected, rather than a vector of $|\mathcal{P}|$ Booleans. For this observation, inspection points are grouped into N clusters. Each element of the observation is the fraction of points in the corresponding cluster that have been inspected. In this environment, $N = 15$ clusters are evenly spaced on the surface of the sphere, resulting in regions roughly the same size as the servicer’s field of view. This observation does not explicitly 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

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

- **Action Space:** The 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. This continuous action space is in contrast to prior work that uses a flight mode-based approach [40] and ongoing work that consider hybrid action spaces [41].
- **Reward Function** The reward $R(s, s')$ for the servicer is the sum of five functions. First, reward is yielded for inspecting points on the RSO that have not yet been inspected by any servicer.

$$R_{\text{inspection}}(s, s') = \frac{|\mathcal{I}'| - |\mathcal{I}|}{|\mathcal{P}|} \quad (9)$$

A reward is also given to all agents for jointly reaching the goal state of 90% inspection of illuminated points.

$$R_{\text{success}}(s') = \beta_{\text{success}} \quad \text{if } \frac{|\mathcal{I}'|}{|\mathcal{L}|} \geq 90\% \quad (10)$$

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

$$R_{\text{fuel}}(\Delta \mathbf{v}) = -\alpha \|\Delta \mathbf{v}\| \quad (11)$$

A penalty for time use is given per-step, weighted by the time-use penalty α_t .

$$R_{\text{time}}(\Delta t) = -\alpha_t \Delta t \quad (12)$$

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)

$$R_{\text{failure}}(s') = -\beta_{\text{fail}} \quad \text{if } s' \in \mathcal{S}_{\text{fail}} \quad (13)$$

- **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 semi-Markov decision process (sMDP) Δt is the duration for which the simulator propagated the step, as selected by the action.

3.2 Learning on MDPs

While a complete review of RL is outside the scope of this paper, a brief review specific to the work in this paper is given here; Reference [37] gives the complete development. To solve an MDP, one must find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the value—or the expectation of the discounted sum of future rewards—at all states:

$$V_{\text{MDP}}^{\pi}(s) = r + \gamma \mathbb{E} [V_{\text{MDP}}^{\pi}(s') | T(s'|s, a), \pi(a|s)] \quad (14)$$

$$= r_0 + \gamma r_1 + \gamma^2 r_2 + \dots = \sum_{i=0}^{\infty} \gamma^i r_i \quad (15)$$

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, DRL 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 [42].

3.2.1 Semi-MDPs

A MDP with variable-duration timesteps is called a sMDP [38, 39]. In a sMDP, each step has some Δt associated with it, which represents the time-opportunity cost of the step, and a reward density $q_t(t)$ instead of reward r_t [38]. The γ -discounted reward for a step is then given by the integral

$$r_t^{(\gamma)} = \int_{t_t}^{t_t + \Delta t_t} e^{\gamma(t - t_t)} q_t(t) dt \quad (16)$$

However, in this work it is assumed that a single scalar reward is given at the end of the step. When computing value, Eq. 15 also needs to be modified to discount according

to the sum of times elapsed over preceding steps

$$V_{\text{sMDP}}(s_0) = \gamma^{\Delta t_0} r_0^{(\gamma)} + \gamma^{\Delta t_0 + \Delta t_1} r_1^{(\gamma)} + \gamma^{\Delta t_0 + \Delta t_1 + \Delta t_2} r_2^{(\gamma)} + \dots = \sum_{t=0}^{\infty} \gamma^{\sum_{i=0}^t \Delta t_i} r_t^{(\gamma)} \quad (17)$$

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

3.2.2 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.

4 Shield Formulation

Collision with other spacecraft is of high concern for the servicer spacecraft. Shielding is a method for guaranteeing operational safety of a RL-based agent by preventing the execution 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) [19]. The shield in this work uses the action projection approach to shielding [15]: if the desired action falls in the space of disallowed actions, the closest safe action is selected instead.

The conceptual behavior of the shield is shown in Fig. 3. When a policy selects some $\Delta \mathbf{v}$, the shield determines the smallest deviation $\delta \mathbf{v}$ that makes the new trajectory passively safe over an infinite horizon under relative motion dynamics. Since the servicer is already known to be on a passively safe trajectory from the previous iterations of shielded tasking, there is always at least the trivial solution $\delta \mathbf{v} = -\Delta \mathbf{v}$ that satisfies the safety constraints by keeping the servicer in the same passively safe orbit as before. This infinite-horizon safety also means that a missed firing for any reason does not jeopardize safety. As a result of the shield, the servicer is never on a direct collision course that depends on an additional future burn to maintain safety.

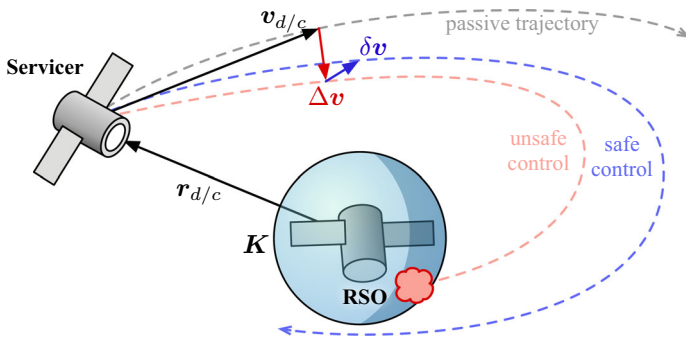


Fig. 3 The shield guarantees safety when the servicer maneuvers

4.2 The Shield as an Optimization Problem

Under the dynamics and constraints defined in the problem statement, 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 its policy $\pi(s) = \Delta \mathbf{v}$ that does not lead to collision with the RSO under the first-order eccentric relative motion dynamics. Let $\delta \mathbf{v}$ be the change in selected thrust required for safety. The optimization problem can be written as

$$\text{minimize } \|\delta \mathbf{v}\| \quad (18)$$

$$\text{such that } \|\Delta \mathbf{v} + \delta \mathbf{v}\| \leq \Delta v_{\max} \quad (19)$$

$$\mathbf{x}_{d/c}(t_0) = \begin{bmatrix} \mathcal{H} \mathbf{r}_{d/c} \\ \mathcal{H}(\mathbf{v}_{d/c} + \Delta \mathbf{v} + \delta \mathbf{v}) \end{bmatrix}$$

$$\mathbf{x}_{d/c}(t) = \Phi_c^x(t, t_0) \mathbf{x}_{d/c}(t_0)$$

$$([\mathbf{I} \ \mathbf{0}] \mathbf{x}_{d/c}(t))^T \mathbf{K} ([\mathbf{I} \ \mathbf{0}] \mathbf{x}_{d/c}(t)) \geq 1 \quad \forall t \in [t_0, t_0 + t_{\max}] \quad (20)$$

$$g_{\text{low,high}}(\alpha(t_{\max}), \delta \alpha(t_{\max}) = \mathbf{A}_c^{-1}(t_{\max}) \mathbf{x}_{d/c}(t_{\max})) \leq 0 \quad (21)$$

The objective (18) minimizes the change in requested thrust. The first constraint (19) ensures that the thrust does not exceed the maximum thrust the servicer can produce. Constraint (20) ensures that the shielded action does not lead to collision over some upcoming timespan $[t_0, t_0 + t_{\max}]$, as computed in the preceding lines. The final constraint (21) uses Eq. A7 as derived in Appendix A to maintain long term passive safety on domain $[t_0 + t_{\max}, \infty)$ by ensuring that the state at $t_0 + t_{\max}$ will evolve in a way such that either the secular term dominates and motion is away from the RSO or motion is periodic relative to the RSO. The relative orbital elements are found using the same mapping matrix $\mathbf{A}_c^{-1}$ used in the state transition matrix. Together, Eqs. 20 and 21 combine to guarantee safety over the infinite horizon $[t_0, \infty)$.

4.3 Iterative Shield Solution

Evaluation times in Eq. 20, 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 $R_{d/K}$ is defined as the objective of the optimization problem

$$R_{d/K} = \text{minimize } \|\mathbf{r}^* - \mathbf{r}_{d/c}\| \quad (22)$$

$$\text{such that } \mathbf{r}^{*T} \mathbf{K} \mathbf{r}^* \leq 1 \quad (23)$$

which is a simple convex problem for nonspherical keepouts and trivially

$$R_{d/K} = \|\mathbf{r}_{d/c}\| - R_{d/c} \quad (24)$$

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

$$t_{i+1} - t_i \leq \frac{R_{d/K}(t_i) + \epsilon}{\max_{t \in [t_i, t_{i+1}]} \|\mathbf{v}_{d/c}(t)\|} \quad (25)$$

Since this condition contains a continuous term, the condition is approximated using the maximum of the endpoints of the segment

$$t_{i+1} - t_i \leq \min \left(\frac{R_{d/K}(t_i) + \epsilon}{\max(\|\mathbf{v}_{d/c}(t_i)\|, \|\mathbf{v}_{d/c}(t_{i+1})\|)}, \Delta t_{\max} \right) \quad (26)$$

This approximation is exact when $\mathbf{v}$ 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 Eq. 26 is satisfied at all sample points. The complete method is given in Algorithm 1 and Algorithm 2. First, the sample time vector is seeded uniformly. Then, the optimization problem is solved to find the shielded action. If the algorithm has converged (i.e. Eq. 26 is satisfied at all times), the algorithm has converged and the solution is returned. If not, the time discretization is resampled based on the solution trajectory.

Algorithm 1 Iterative Shield Solution

```

1:  $t \leftarrow [t_0 : \Delta t_{\max} : t_{\max}]$ 
2: while not converged do
3:   solution  $\leftarrow$  solve Equation 18–21
4:   if Equation 26  $\forall t_i \in t$  then
5:     return solution
6:   else
7:      $t \leftarrow \text{RESAMPLETIMES}(\text{solution})$ 

```

To resample the time discretization t' from the previous discretization t , the necessary discretization at the current steps is approximated as a continuous function with a linear interpolation of Eq. 26 at the sample points. Starting at t_0 , points are added to the sample vector such that the Δt between the new point and the previous point is less than $(1 - \kappa)$ times the lowest interpolated maximum allowable value of Δt in that period, where $0 < \kappa < 1$.

Algorithm 2 Resample Times

```

1: function RESAMPLETIMES(solution,  $\kappa = 0.1$ )
2:    $t' \leftarrow [t_0]$ 
3:    $\Delta t_{\text{interp}}(t) \leftarrow$  linear interpolation of R.H.S. of Equation 26
4:   if  $t'_{-1} \geq t_0 + t_{\max}$  then
5:      $t'_{-1} = t_0 + t_{\max}$ 
6:     return  $t'$ 
7:   else
8:      $t_i = t'_{-1}$ 
9:      $t_{\text{new}} \leftarrow t_{\text{new}} - t_i = (1 - \kappa) \min_{t \in [t_i, t_{\text{new}}]} \Delta t_{\text{interp}}(t)$ 
10:    if  $\exists t_{\text{old}} \in t$  s.t.  $t_i < t_{\text{old}} < t_{\text{new}}$  then
11:      append( $t'$ ,  $t_{\text{old}}$ )
12:    else
13:      append( $t'$ ,  $t_{\text{new}}$ )

```

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. At each iteration, the optimization problem is solved by Gurobi 10.0.1 [43], running in the fully non-convex mode with NumericFocus set to 3 for higher numerical accuracy. The shield solver is seeded with the trivial solution $\delta v = -\Delta v$ on the first iteration, and the solution for δv from the previous iteration on subsequent iterations.

5 Probabilistic Roadmaps

The PRM-based approach for the problem consists of iteratively constructing a roadmap of waypoints, then using a MIQCQP to find a trajectory in the roadmap that satisfies the solution constraints.

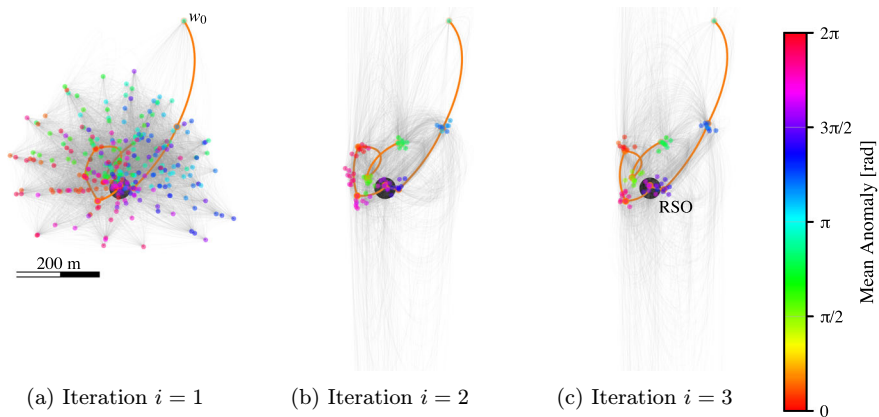


Fig. 4 PRM solution iterations

5.1 Algorithm Structure

Algorithm 3, PLANINSPECTIONTRAJECTORY, gives the general scheme of the trajectory planning algorithm. The trajectory space is discretized into waypoints $w \in \mathcal{W}$ that consist of the RSO Hill- and body-frame position of the servicer and the mean anomaly of the inspection point, $w_w = (\mathbf{r}_w, M_w)$. In the first iteration, these waypoints are sampled (SEEDWAYPOINTS) in regions of the state space that have inspectable RSO points in $\mathcal{L}$. In following iterations, the waypoints are resampled (RESAMPLEWAYPOINTS) around the incumbent solution from the previous iteration, in order to improve beyond the initial discretization. With the waypoints sampled, a roadmap of dynamically feasible transitions is built between waypoints (CONNECTGRAPH). The roadmap is then encoded into a MIQCQP that optimizes for fuel and time (weighted by ζ , which is akin to α in the MDP formulation) with constraints for feasibility and complete inspection. The solution to the MIQCQP provides a trajectory that satisfies the inspection task requirements.

Algorithm 3 Inspection trajectory planning function

```

1: function PLANINSPECTIONTRAJECTORY( $\mathbf{r}_0, M_0, \mathcal{L}$ )
2:    $w_0 \leftarrow (\mathbf{r}_0, M_0)$ 
3:   for  $i \in 1, 2, 3, \dots$  do
4:     if  $i = 1$  then
5:        $\mathcal{W} \leftarrow \{w_0\} \cup \text{SEEDWAYPOINTS}(\mathcal{L}, N_{\text{seed}})$ 
6:     else
7:        $\mathcal{W} \leftarrow \{w_0\} \cup \text{RESAMPLEWAYPOINTS}(\mathcal{W}_{\text{sol}}[1:], N_{\text{resample}}, \sigma_r, \sigma_M)$ 
8:      $\mathcal{E} \leftarrow \text{CONNECTGRAPH}(\mathcal{W}, N_{\text{conn}})$ 
9:      $\mathbf{x} \leftarrow \text{construct and solve MIQCQP (Equation 31–40)}$ 
10:     $\mathcal{W}_{\text{sol}} \leftarrow \text{PARSESOLUTION}(\mathbf{x}, \mathcal{W})$ 
11:   return  $\mathcal{W}_{\text{sol}}$ 

```

Fig. 4 shows three iterations of the method on an example case. The differences between first and later iteration waypoint sampling methods are clearly visible, and the incumbent solution quality improves with each iteration.

Algorithm 4 Generate N_{seed} waypoints with visibility of each illuminated point

```

1: function SEEDWAYPOINTS( $\mathcal{L}$ ,  $N_{\text{seed}}$ )
2:    $\mathcal{W} \leftarrow \{\}$ 
3:   for  $p_p \in \mathcal{L}$  do
4:     for  $i \in 1, \dots, N_{\text{seed}}$  do
5:        $\mathbf{r}_i \leftarrow \mathbf{r}_p + \hat{\mathbf{n}}_p \cdot \text{Uniform}(R_k - \|\mathbf{r}_p\|, R_{\text{max}})$ 
6:        $M_i \leftarrow \text{Uniform}(M \in [0, 2\pi] \mid L(p_p, M) = \text{True})$ 
7:        $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\mathbf{r}_i, M_i)\}$ 
8:   return  $\mathcal{W}$ 

```

5.2 Graph Sampling and Construction

Three algorithms are used for constructing a discrete graph in the waypoints space. SEEDWAYPOINTS and RESAMPLEWAYPOINTS generate sets of waypoints for the PRM, and CONNECTGRAPH draws dynamically feasible edges between the sampled waypoints to produce a graph of possible trajectories.

Algorithm 4, SEEDWAYPOINTS, generates waypoints for the initial iteration of PLANINSPECTIONTRAJECTORY. While PRM-based methods generally sample the waypoint space uniformly, a large portion of the domain does not result in inspected points and is thus irrelevant to the task; including points from this region results in an intractably large optimization problem. Instead, N_{seed} waypoints are sampled for each RSO point in $\mathcal{L}$. The waypoint position $\mathbf{r}_i$ is sampled uniformly between the conjunction radius R_k and the maximum inspection radius R_{max} in the inspection point normal direction $\hat{\mathbf{n}}_p$. The mean anomaly M_i is sampled uniformly over times where the point is illuminated.

In further iterations, Algorithm 5, RESAMPLEWAYPOINTS, is used to generate new waypoints for the PRM in order to improve beyond the previous iteration's discretization. For each waypoint in the solution trajectory $\mathcal{W}_{\text{sol}}$, N_{resample} new waypoints are generated in addition to the waypoint from the incumbent solution. These are sampled in a normal distribution around the solution point with a standard deviation of σ_r in position and σ_M in true anomaly. Because the points in $\mathcal{W}_{\text{sol}}$ are included, the new roadmap will be at worst as good as the previous iteration.

Algorithm 6, CONNECTGRAPH, builds the PRM by drawing edges between nearby waypoints. The structure of the problem is exploited to use the difference in mean

Algorithm 5 Generate N_{resample} waypoints around each waypoint in the solution

```

1: function RESAMPLEWAYPOINTS( $\mathcal{W}_{\text{sol}}$ ,  $N_{\text{resample}}$ ,  $\sigma_r$ ,  $\sigma_M$ )
2:    $\mathcal{W} \leftarrow \{\}$ 
3:   for  $w_w \in \mathcal{W}_{\text{sol}}$  do
4:      $\mathcal{W} \leftarrow \mathcal{W} \cup \{w_w\}$ 
5:     for  $i \in 1, \dots, N_{\text{resample}}$  do
6:       do
7:          $\mathbf{r}_i \leftarrow \text{Normal}(\mathbf{r}_w, \sigma_r \mathbf{I}_{3 \times 3})$ 
8:         while  $\|\mathbf{r}_i\| < R_k$ 
9:          $M_i \leftarrow \text{Normal}(M_w, \sigma_M)$ 
10:         $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\mathbf{r}_i, M_i)\}$ 
11:   return  $\mathcal{W}$ 

```

anomaly as the distance metric. The modulus operator is applied to the mean anomaly difference to transform it into the range $(0, 2\pi]$ to reflect that any transition happens forward in time in at most one orbit. The N_{conn} closest COLLISIONFREE child waypoints are connected from each waypoint, where collisions are checked at discrete points between each pair using the STM to solve the two-point boundary value problem (Eq. 30).

Algorithm 6 Add edges between nearby waypoints

```

1: function CONNECTGRAPH( $\mathcal{W}$ ,  $N_{\text{conn}}$ )
2:    $\mathcal{E} \leftarrow \{\}$ 
3:   for  $w_i \in \mathcal{W}$  do
4:     for  $w_j \in \mathcal{W}$  do
5:        $d_j \leftarrow (M_j - M_i) \bmod (0, 2\pi]$ 
6:       for  $N_{\text{conn}}$  smallest  $d_j$  where COLLISIONFREE( $w_i, w_j$ ) do
7:          $\mathcal{E} \leftarrow \mathcal{E} \cup \{(w_i \rightarrow w_j)\}$ 
8:   return  $\mathcal{E}$ 
  
```

5.3 Mixed-Integer Formulation

With the waypoints $\mathcal{W}$ and edges $\mathcal{E}$ computed, a MIQCQP can be formulated to find the optimal trajectory that satisfies the inspection percentage requirement. New notation is introduced for the optimization variables and weights:

- $\zeta \in [0, 1]$ is the fuel-time cost weight, with $\zeta = 1$ causing the optimizer to only minimize fuel use and $\zeta = 0$ causing the optimizer to only minimize inspection time.
- $x_{i,j}$ is a binary optimization variable that indicates if the edge $(i \rightarrow j)$ is traversed in the solution. One $x_{i,j}$ is created for every edge in $\mathcal{E}$. The vector of all $x_{i,j}$ is $\mathbf{x}$.
- $\Delta v_{0,i}^{\text{lin}}$ is the Δv cost of the initial maneuver to travel from w_0 to w_i .
- $\Delta v_{i,j,k}^{\text{quad}}$ is the Δv cost of maneuver that must take place at w_j when coming from w_i heading to w_k via w_j .
- z_p is a binary optimization variable indicating if inspection point p on the RSO is inspected in the solution. One z_p is created for every point in the illuminated set $\mathcal{L}$. The vector of all z_p is $\mathbf{z}$.
- $\mathbb{I}$ is the inspectability matrix, which is size $\text{len}(\mathbf{z}) \times \text{len}(\mathbf{x})$. It is structured such that entries at the row corresponding to z_p and column corresponding to $x_{i,j}$ are the points of the RSO inspectable on the edge between w_i and w_j :

$$\mathbb{I}_{p,(i,j)} = \bigvee_{M=M_i}^{M_j} I(p, \mathbf{r}_{i \rightarrow j}(M), M) \quad (27)$$

where $\mathbf{r}_{i \rightarrow j}(M)$ is the position along the trajectory between two waypoints w_i and w_j as a function of mean anomaly M . In this implementation, this is evaluated discretely 20 points along each edge.

The linear and quadratic Δv costs are given by

$$\Delta v_{0,i}^{\text{lin}} = \|\mathbf{v}_0 - \mathbf{v}_{\text{in}}(w_0, w_i)\| \quad (28)$$

$$\Delta v_{i,j,k}^{\text{quad}} = \|\mathbf{v}_{\text{out}}(w_i, w_j) - \mathbf{v}_{\text{in}}(w_j, w_k)\| \quad (29)$$

To compute the incoming $\mathbf{v}_{\text{in}}$ and outgoing $\mathbf{v}_{\text{out}}$ velocities for the trajectory edge between w_i and w_j , the equation

$$\begin{bmatrix} \mathbf{r}_j \\ \mathbf{v}_{\text{out}} \end{bmatrix} = \Phi(M_i + ((M_j - M_i) \bmod 2\pi), M_i) \begin{bmatrix} \mathbf{r}_i \\ \mathbf{v}_{\text{in}} \end{bmatrix} \quad (30)$$

is solved for $\mathbf{v}_{\text{in}}$ and $\mathbf{v}_{\text{out}}$, given that $\mathbf{r}_i$ and $\mathbf{r}_j$ are known.

The complete MIQCQP for path planning over a set of waypoints $\mathcal{W}$ and edges $\mathcal{E}$ is formulated in Eq. 31 through 40. It is assumed that all summations are over the indices for which the respective variables are explicitly defined (i.e. $\sum_i \sum_j x_{i,j}$ only sums over existing edges).

$$\text{minimize } \zeta \left(\sum_i \Delta v_{0,i}^{\text{lin}} x_{0,i} + \sum_i \sum_j \sum_k x_{i,j} \Delta v_{i,j,k}^{\text{quad}} x_{j,k} \right) + \frac{1-\zeta}{n} \sum_i \sum_j \Delta M_{i,j} x_{i,j} \quad (31)$$

$$\text{subject to } x_{i,j} \in \{0, 1\} \quad \forall (w_i \rightarrow w_j) \in \mathcal{E} \quad (32)$$

$$\sum_k x_{j,k} \leq \sum_i x_{i,j} + b \leq 1 \quad \forall j; b = 1 \text{ if } j = 0 \text{ else } b = 0 \quad (33)$$

$$m_i \in \{0, \dots, m_{\max}\} \quad \forall w_i \in \mathcal{W}; m_{\max} = 10 \quad (34)$$

$$0 \leq (2\pi(m_j - m_i) + M_j - M_i)x_{i,j} \leq 2\pi \quad \forall (w_i \rightarrow w_j) \in \mathcal{E} \quad (35)$$

$$x_{0,i} = 0 \quad \forall i \mid \Delta v_{0,i}^{\text{lin}} > \Delta v_{\max} \quad (36)$$

$$x_{i,j} + x_{j,k} \leq 1 \quad \forall i, j, k \mid \Delta v_{i,j,k}^{\text{quad}} > \Delta v_{\max} \quad (37)$$

$$z_p \in \{0, 1\} \quad \forall p \in \mathcal{L} \quad (38)$$

$$\mathbb{I}\mathbf{x} - \mathbf{1} \geq -\mathbb{M}(1 - \mathbf{z}) \quad \mathbb{M} \gg 1 \quad (39)$$

$$\sum_p z_p \geq 0.9|\mathcal{L}| \quad (40)$$

The objective function (Eq. 31) consists of two terms: the fuel cost term, weighted by ζ , and the time cost, weighted by $1 - \zeta$ and converted to time by division by the mean motion n . The fuel cost consists of a linear term for the cost of the initial maneuver based on which outgoing edge from w_0 is selected and a quadratic term based on the necessary incoming and outgoing velocities at each visited waypoint. The time cost sums the ΔM between waypoints on traveled edges.

Constraints 32 through 35 are used to guarantee a single trajectory originating at w_0 . Equation 33 is the traditional traveling salesman problem in-out constraint: edges

can only be traveled if they come from the starting waypoint ($b = 1$) or another traveled edge ($b = 0$), and the path cannot branch or merge. However, that alone allows for disconnected cycles in the solution. To prevent these, Eq. 35 is introduced which assigns an integer orbit number m_i to each waypoint w_i and requires that for sequential waypoints in a solution trajectory, the absolute mean anomaly is always increasing with a difference in $(0, 2\pi]$. As a result, no cycles can be formed and the less-than-one orbit connection is used.

Constraints 36 and 37 are added for every Δv that exceeds $\Delta v_{\max}$. For the linear objective term, the corresponding constraint (36) trivially sets infeasible edges to be unselected; for the quadratic objective, the corresponding constraint (37) allows for incoming or outgoing edges to be selected but prevents both from being selected for maneuvers that are infeasible.

Finally, constraints 38 through 40 use a big- $\mathbb{M}$ formulation to ensure that at least 90% of illuminated points are inspected by the trajectory. For a trajectory with traveled edges encoded by $\mathbf{x}$, $\mathbb{I}\mathbf{x}$ results in a vector that is 0 for uninspected points and ≥ 1 for inspected points. The big- $\mathbb{M}$ formulation, with $\mathbb{M}$ arbitrarily set to 10000, allows for a minimum threshold on the number of inspected points to be a constraint.

5.4 Parsing the Optimization Solution

Algorithm 7 gives the method for extracting the sequence of waypoints in the solution from the optimization variable $\mathbf{x}$.

Algorithm 7 Parse the MIQCQP solution

```

1: function PARSE_SOLUTION( $\mathbf{x}, \mathcal{W}$ )
2:    $\mathcal{W}_{\text{sol}} \leftarrow [w_0]$ 
3:   while True do
4:      $w_i \leftarrow \mathcal{W}_{\text{sol}}[-1]$ 
5:     for  $j \mid x_{i,j} \in \mathbf{x}$  do
6:       if  $x_{i,j} = 1$  then
7:         APPEND( $\mathcal{W}_{\text{sol}}, w_j$ )
8:       break
9:   else ▷ Return if no  $w_j$  is connected from  $w_i$ 
10:  return  $\mathcal{W}_{\text{sol}}$ 

```

5.5 Algorithm Parameters

When evaluating the algorithm, the following parameters are used: $N_{\text{conn}} = 40$, $N_{\text{seed}} = 5$, $N_{\text{resample}} = 8$, $\sigma_r = 10$ m, $\sigma_M = 0.2$, and iterations = 3. These parameters were determined by comparing results against parameters that yield larger instances of the problem that search more of the optimization space; the selected parameters produce similar quality solutions in faster time. For iterations $i \geq 2$, the solver can be warm-started with the previous iteration's solution. The problem is likely too large to solve exactly in reasonable time, so a non-optimal but feasible incumbent may be accepted instead. In the first iteration, the solver is run for 120 s beyond when the

first incumbent solution is found; on the following warm-started iterations, the solver is run for 120 s per iteration or until the optimal solution is found. Gurobi 10.0.1 is used to solve the MIQCQP [43]; Gurobi automatically presolves the quadratic term in the objective to ensure convexity of the MIQP, as $\Delta v_{i,j,k}^{\text{quad}}$ is not inherently positive semidefinite.

When executing the plan in the high-fidelity simulator, the 2-point boundary value problem in Eq. 30 is solved for each maneuver according to the current true state to reach the next waypoint. This prevents perturbations from compounding over the course of execution.

6 Results

First, policies are trained and benchmarked for the RL formulation; the performance and impact of the shield is also evaluated. Next, the RL and PRM policies are compared by considering the resulting fuel-time Pareto front from each method. Example trajectories are given for various cases.

6.1 Training

Policies are trained in the high-fidelity implementation of this environment, without a shield, for a range of fuel use penalties $\alpha \in [0.01, 0.05, 0.1, 0.2, 0.3]$.⁴ The RLlib implementation of PPO is used to train policies [42, 44]. The algorithm and training pipeline are modified to use semi-Markov discounting for advantage estimation [38]. The policies are represented by a fully connected multilayer perceptron (MLP) neural network of size 2×256 , which takes about $100 \mu\text{s}$ per decision to evaluate in inference. The time-use penalty $\alpha_t = 0.05/\text{orbit}$, which yields policies that are about 5% more efficient in time than with no penalty, owing to the fact that discounting alone can be weak for encouraging time-efficient policies. The bonus and penalties $\beta_{\text{success}} = \beta_{\text{fail}} = 1.0$; since success without violation of constraints is always possible, the specific values do not strongly impact the policies' behavior and the best practice of scaling them to the same order as other rewards is followed. The policies presented in this paper are a refined version of those presented in Reference [30], with a 40M-step learning rate schedule of 10M steps at 10^{-4} , 10M decreasing to 10^{-5} , 10M decreasing to 10^{-6} , and 10M holding at 10^{-6} . Hyperparameter values are selected following an ablation study across algorithm and network parameters. These include discount rate $\gamma = 0.9999$, batch size 4650 (collected across 30 learning environments), and RLlib's defaults for PPO for all others.

Figure 5 shows how various metrics evolve as the policies learn to complete the task. All the policies reach a success rate of nearly 100% as they learn to inspect at least 90% of illuminated points. The fuel and time used are both driven down as training refines the policies. As expected, policies for lower α are faster (lower t_{success})

⁴ The process training the $\alpha = 0.1$ policy was interrupted at 25.4M steps, but the policy is still considered since training appears to have converged.

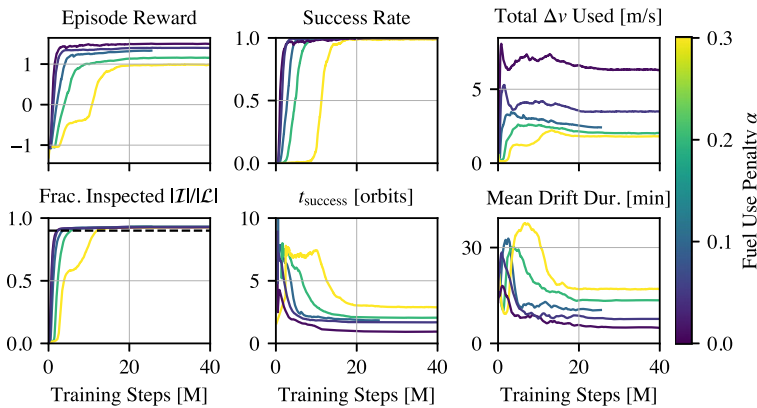


Fig. 5 Training curves for policies with varying fuel use penalty α

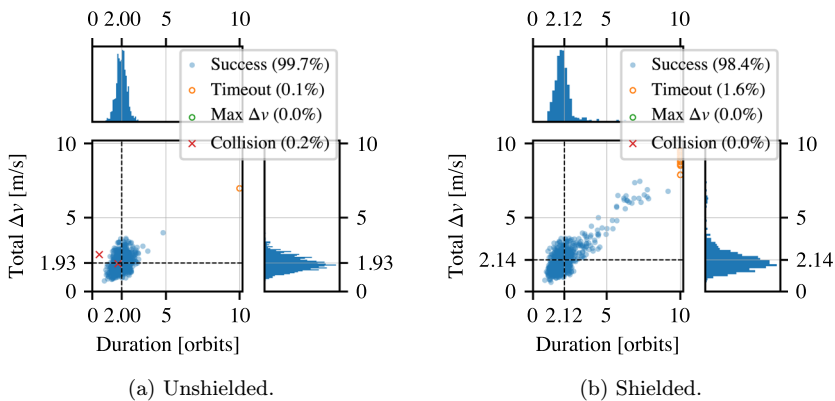


Fig. 6 Performance of the $\alpha = 0.2$ policy

but less fuel-efficient (higher total Δv used) when completing the task. The policies tend to prefer relatively short drifts between a high number of small impulsive thrusts.

6.2 Unshielded and Shielded RL

The $\alpha = 0.2$ policy is evaluated with and without the shield for 1000 environment seeds to investigate the impact of the shield on policy performance. The shield uses a 5 m keepout, $t_{\max} = 4$ orbits, and $\Delta t_{\max} = 500$ seconds.

6.2.1 Benchmarks

The unshielded (Fig. 6a) and shielded (Fig. 6b) performance of the policies is compared. On average, the shield results in trajectories that take 0.12 more orbits and 0.21 m/s more fuel to complete the inspection task. As expected, the shield eliminates the (relatively rare) collisions produced by the unshielded policy. However, this

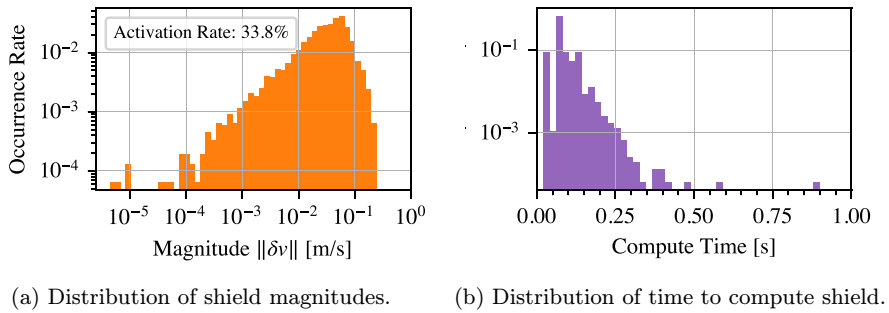


Fig. 7 Shield statistics

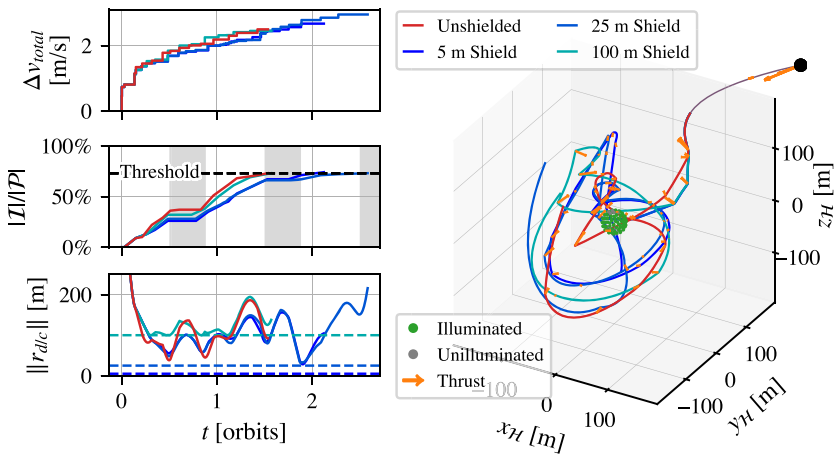


Fig. 8 Unshielded and shielded RL trajectories, $\beta = 7.6^\circ$

comes at the cost of increased time—and thus fuel—to complete the task. Practically, the cases where the shielded policy does not complete task are not critically bad: a $> 80\%$ inspection threshold, rather than the 90% threshold, is met in a reasonable time. Another reason for the shield's strong impact on performance is that it is not *only* preventing collisions—it also introduces infinite-horizon passive safety guarantees that are stricter than the base problem's non-collision constraint.

Figure 7a gives histograms of the size of shield interventions and Fig. 7b shows how long the shield took to compute on an Apple M2 Pro CPU. In the 33.8% of actions where the shield activates, the shield yields an average $\|\delta v\|$ of only 0.04 m/s, showing that it is generally unobtrusive. The evaluation time is minimal, making the shield compatible with the closed-loop paradigm of the RL-based policy.

6.2.2 Example Trajectories

Trajectories are generated for a high illumination case (Fig. 8) and a low illumination case (Fig. 9) for the $\alpha = 0.1$ policy, both unshielded and with a range of shield radii; this highlights the ability of the shield to enforce a different keepout radius than the

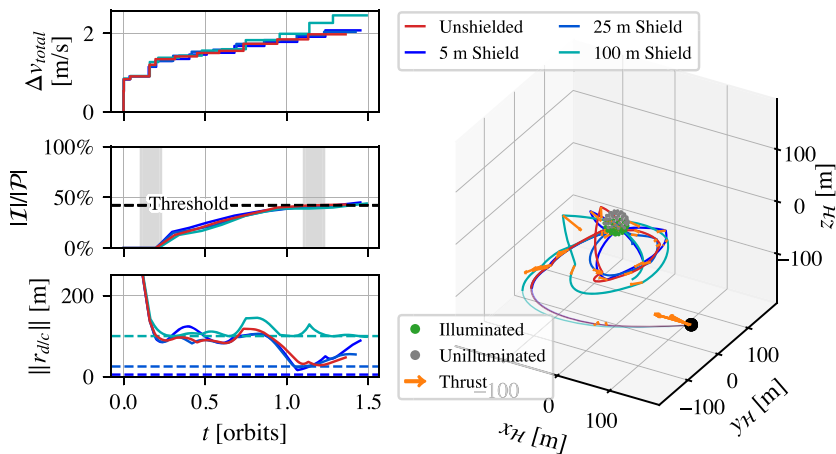


Fig. 9 Unshielded and shielded RL trajectories, $\beta = -62.9^\circ$

policy was trained for. Both the Hill-frame trajectory and various statistics (Δv used, fraction of points inspected, and RSO-servicer distance) are recorded.

In both cases, there is an initial high-cost burn to near the RSO. Once within the inspection radius, points are inspected at a relatively constant rate, other than when the spacecraft is in eclipse (shaded gray). The geometry of trajectories are similar with or without shields: In the high illumination case, the servicer tracks the illuminated side of the RSO as it moves, while in the low illumination case, only one side of the RSO is ever illuminated, so the servicer stays on that side.

The shield is seen to successfully enforce the various keepout radii, as denoted by dashed lines in the plot of $\|r_{d/c}\|$. While the difference from the unshielded trajectory is generally minimal, the 100 m keepout leads to notable differences in the trajectory, with the shield keeping the servicer on the edge of the constraint much of the time.

6.3 Comparing PRM and RL

The benchmarks of all five unshielded RL policies for differing α are compared to a benchmark of the PRM solver over a range of object weights ζ . The Pareto fronts yielded by the two methods are compared, and example trajectories are given.

6.3.1 Pareto Fronts

Figure 10 shows the mean Pareto front for the PRM solver across ζ for $N = 20$ seeds and the RL solutions across α for $N = 1000$ seeds. Both methods produce reasonable results, clearly showing the time-fuel tradeoff as the respective weighting parameter is changed. The PRM solver shows stronger performance than the RL solution across the front.

While the RL Pareto front is expectedly dominated by the PRM solver, this comes at significant computational—and thus operational—cost. Figure 11 compares the

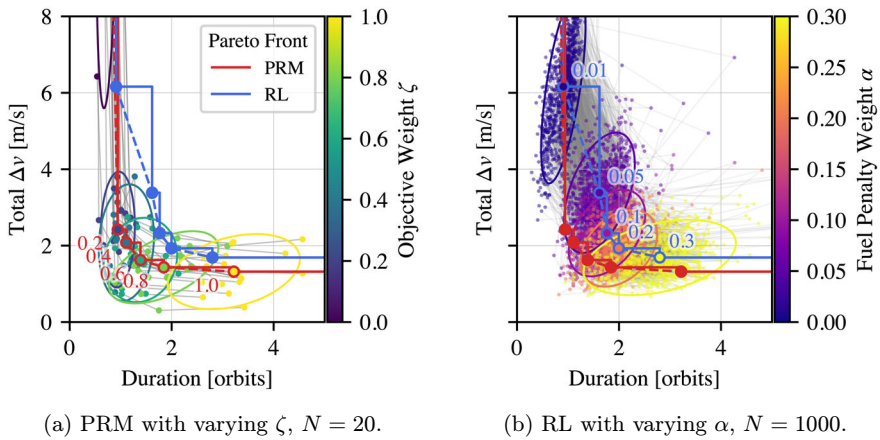


Fig. 10 Mean Pareto fronts across N trials; 2σ covariances

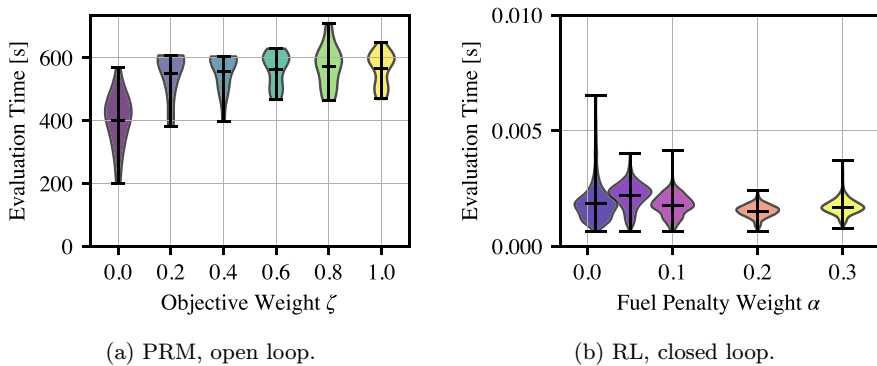


Fig. 11 Computation times; colors as in Fig. 10

execution times of each method: The PRM method takes about 10 min on an Apple M2 Pro with 12 cores to converge, using the Gurobi solver that cannot be (trivially) used on a spacecraft computer. Computing these trajectories onboard is therefore infeasible with typical spacecraft resources, requiring a paradigm of offline planning and open-loop execution. Comparatively, the RL solution is very cheap to evaluate due to its representation as a small (2×256 node) MLP, which requires a trivial amount of storage and compute; even larger and more network architectures that would reasonably be considered for this problem have low enough inference costs to be useable onboard. As a result, it operates in a closed-loop, online manner. Not only does this allow for execution at arbitrary times, if some part of the RSO needed to be dynamically reinspected, the RL solution could trivially handle that case while the PRM would need an additional planning cycle.

As a result, the tradeoff between the two methods is autonomy and responsiveness at the cost of some additional fuel or time. For a science or security mission with opportunistic events, those benefits of RL may outweigh the costs. In scenarios such as

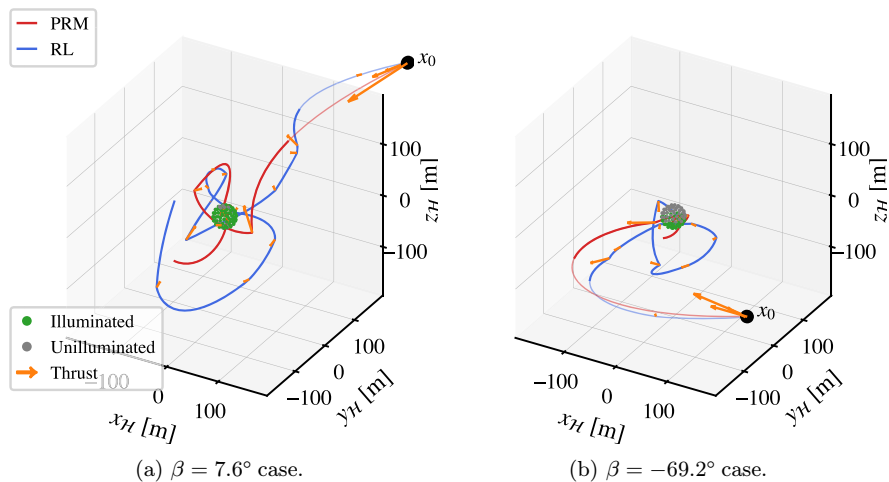


Fig. 12 Comparison of PRM and RL trajectories on two sets of initial conditions

deep space missions where resources are limited and ample planning time is available, the PRM method is superior.

6.3.2 Example Trajectories

Figure 12 compares the PRM solution with $\zeta = 0.5$ to the unshielded RL policy with $\alpha = 0.1$ for the same cases tested in Fig. 8 and reffig:trajectoriespslowpsillumination. The solutions are qualitatively similar. The most notable difference is that the PRM is more “comfortable” with trajectories that get very close to the RSO, while the RL solution is more conservative due to the collision penalty that it encounters during learning.

In Fig. 13, the PRM and RL solutions are compared across all tested values of α and ζ at 20 different environment seeds. Cases where the PRM dominates RL (i.e. the time and fuel costs are both better) are in red, and the opposite are in blue; where no method dominates, the plot is gray. The intensity of the color corresponds to the product of the difference in fuel cost times the difference in time cost, effectively a rectangle with area in units of m/s-orbits. From this, it is clear that the PRM solutions do not only dominate on average (Fig. 10) but in almost all cases (Fig. 13); granted, this dominance is generally small. As seen on the plot of Pareto fronts, the largest gap is present for fast, high-fuel-cost solutions.

6.4 Limitations and Future Directions

This work considers a common case of the RSO inspection problem under the assumptions of a nadir-pointing RSO with impulsive maneuvers. Mathematically, the methods presented here generalize beyond LEO; in practice, differences in dynamical regime being used may qualitatively or quantitative affect the performance of either method. Many of the more realistic factors that one could introduce to the problem statement—

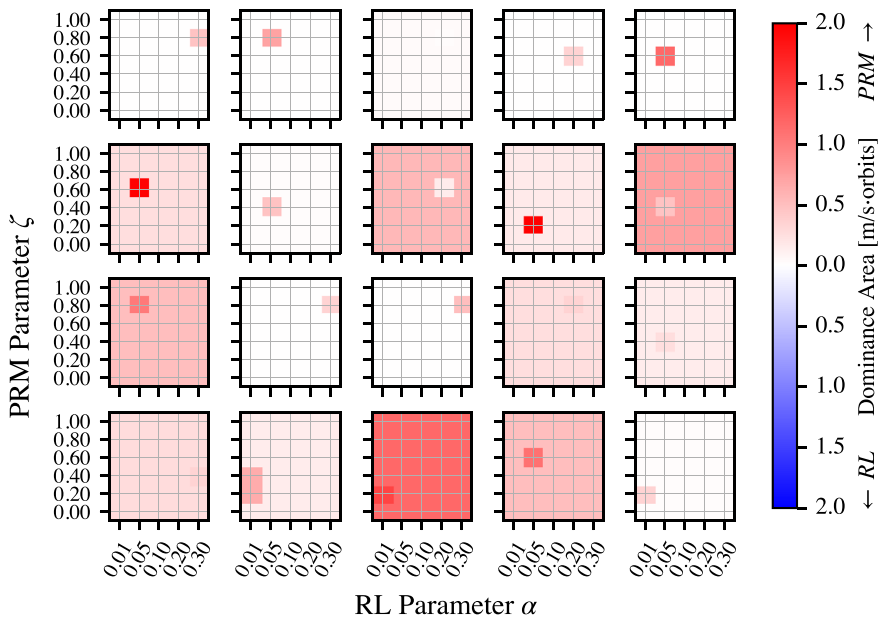


Fig. 13 Comparison of PRM across α and RL across ζ for 20 seeds (one subplot per seed). Color indicates area of dominance, gray indicates no dominance

such as non-impulsive or uncertain burns—are more easily addressed via the reactive policies produced by RL; increasing the complexity or uncertainty of the dynamics poses a large challenge for PRMs and adds some additional complexity to the shield formulation. Likewise, in order of complexity, a tumbling, actively slewing, or reactively slewing RSO leads to considerably more challenging problem variants; again, RL or some other reactive approach is necessary to solve these forms of the problem, as the PRM or any preplanning method requires *a priori* knowledge of the RSO's attitude over time.

7 Conclusions

A PRM-MIQCQP approach to planning for the RSO inspection problem is developed and compared to an RL-based approach presented in previous work. Both methods result in a fuel-time Pareto front, allowing selection between faster and more fuel-efficient trajectories. The PRM front dominates the RL front, but at the cost of a more expensive planner that must be evaluated offline; the RL solution allows for closed-loop and responsive inspection due to its low computational requirements. Both methods offer novel developments for the RSO inspection task; the best method to use is application dependent.

Acknowledgements This work is supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) grant, 80NSSC23K1182.

Author Contributions M.S. created the reinforcement learning pipeline, derived the probabilistic roadmap method, and analyzed the performance of the methods. H.S. developed the simulator and contributed to the relative motion analysis. All authors reviewed the manuscript.

Funding 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, and Colorado State University and with support from NSF grants OAC-2201538 and OAC-2322260.

Data Availability No datasets were generated or analysed during the current study.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A Conditions for Infinite-Horizon Non-Collision

It is necessary to analyze the long-term trajectories of a deputy relative to a chief in order to provide infinite horizon safety guarantees in the event that the deputy 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 Reference [32], 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 $t \rightarrow \infty$, 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

$$y(f, \delta M) \approx a \left(\frac{\delta M}{\eta} + \delta\omega + \cos i \delta\Omega \right) - a\delta_y \sin(f - f_y) - \frac{ae}{2} \sin(2f)\delta e \quad (\text{A1})$$

where

$$\delta_y(\delta M) = \sqrt{4\delta e^2 + e^2 \left(\frac{\delta M}{\eta} - \delta\omega - \cos i \delta\Omega \right)^2} \quad (\text{A2})$$

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

$$y(\delta M) \geq \lfloor y(\delta M) \rfloor = a \left(\frac{\delta M}{\eta} + \delta\omega + \cos i \delta\Omega \right) - a\delta_y(\delta M) - \frac{ae}{2} \delta e \quad (\text{A3})$$

$$y(\delta M) \leq \lceil y(\delta M) \rceil = a \left(\frac{\delta M}{\eta} + \delta\omega + \cos i \delta\Omega \right) + a\delta_y(\delta M) + \frac{ae}{2} \delta e \quad (\text{A4})$$

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 K for all $t > t^*$ if

$$\lfloor y(\delta M(t^*)) \rfloor \geq (\hat{y}^T K \hat{y})^{-\frac{1}{2}} \quad (A5)$$

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

$$\lceil y(\delta M(t^*)) \rceil \leq -(\hat{y}^T K \hat{y})^{-\frac{1}{2}} \quad (A6)$$

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:

$$\begin{aligned} g_{\text{low}}(\alpha, \delta\alpha) &= \delta a \left(\lfloor y \rfloor - (\hat{y}^T K \hat{y})^{-\frac{1}{2}} \right) \leq 0 \\ g_{\text{high}}(\alpha, \delta\alpha) &= \delta a \left(\lceil y \rceil + (\hat{y}^T K \hat{y})^{-\frac{1}{2}} \right) \leq 0 \end{aligned} \quad (A7)$$

References

1. Sabatini, M., Volpe, R., Palmerini, G.B.: Centralized visual based navigation and control of a swarm of satellites for on-orbit servicing. *Acta Astronaut.* **171**, 323–334 (2020). <https://doi.org/10.1016/j.actaastro.2020.03.015>
2. Bernhard, B., Choi, C., Rahmani, A., Chung, S.-J., Hadaegh, F.: Coordinated Motion Planning for On-Orbit Satellite Inspection using a Swarm of Small-Spacecraft. In: 2020 IEEE Aerospace Conference, pp. 1–13 (2020). <https://doi.org/10.1109/AERO47225.2020.9172747>
3. Nakka, Y.K., Hönnig, W., Choi, C., Harvard, A., Rahmani, A., Chung, S.-J.: Information-based guidance and control architecture for multi-spacecraft on-orbit inspection. *J. Guid. Control. Dyn.* **45**(7), 1184–1201 (2022). <https://doi.org/10.2514/1.G006278>
4. Lauinger, T., Ulrich, S.: Path Planning for Optimal Coverage of Orbiting Space Structures Using Lissajous Curves. In: AAS/AIAA Space Flight Mechanics Meeting (2025)
5. Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. *Int. J. Robot. Res.* **35**(5), 528–564 (2016). <https://doi.org/10.1177/0278364915614386>. [arXiv:1407.2896](https://arxiv.org/abs/1407.2896)
6. Iversflaten, M., Hansson, A., Sternberg, D., Jia-Richards, O., Albee, K.: Robust Replanning for Multi-Agent SmallSat Inspection in Failure Scenarios. In: AIAA SCITECH 2025 Forum. American Institute of Aeronautics and Astronautics, Orlando, FL (2025). <https://doi.org/10.2514/6.2025-0183>
7. Albee, K., Sternberg, D.C., Hansson, A., Schwartz, D., Majumdar, R., Jia-Richards, O.: Architecting Autonomy for Safe Microgravity Free-Flyer Inspection. In: IEEE Aerospace Conference (2025)
8. Aurand, J., Cutlip, S., Lei, H., Lang, K., Phillips, S.: Exposure-Based Multi-Agent Inspection of a Tumbling Target Using Deep Reinforcement Learning. *arXiv* (2023). <https://doi.org/10.48550/arXiv.2302.14188>
9. Wijk, D., Dunlap, K., Majji, M., Hobbs, K.L.: Deep Reinforcement Learning for Autonomous Spacecraft Inspection Using Illumination. *arXiv* (2023). <https://doi.org/10.48550/arXiv.2308.02743>
10. Lei, H.H., Shubert, M., Damron, N., Lang, K., Phillips, S.: Deep Reinforcement Learning For Multi-Agent Autonomous Satellite Inspection. In: Sandnas, M., Spencer, D.B. (eds.) Proceedings of the 44th Annual American Astronautical Society Guidance, Navigation, and Control Conference, 2022, pp. 1391–1412. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-51928-4_76
11. Lei, H., Lippay, Z.S., Zaman, A., Aurand, J., Maghareh, A., Phillips, S.: Stability Analysis of Deep Reinforcement Learning for Multi-Agent Inspection in a Terrestrial Testbed. In: AIAA SCITECH 2025 Forum, Orlando, FL (2025)

12. Brandonisio, A., Lavagna, M., Guzzetti, D.: Reinforcement learning for uncooperative space objects smart imaging path-planning. *J. Astronaut. Sci.* **68**(4), 1145–1169 (2021). <https://doi.org/10.1007/s40295-021-00288-7>
13. Brandonisio, A., Capra, L., Lavagna, M.: Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target. *Adv. Space Res.* **73**(11), 5741–5755 (2023). <https://doi.org/10.1016/j.asr.2023.07.007>
14. Quinn, P., Gopu, B.P.R., Nehma, G.M., Tiwari, D.M.: Simulation Based Reward Function Validation for Multi-Agent On Orbit Inspection. In: AIAA SciTech Forum, Orlando, FL (2026)
15. Krasowski, H., Thumm, J., Müller, M., Schäfer, L., Wang, X., Althoff, M.: Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking. arXiv e-prints [arXiv:2205.06750](https://arxiv.org/abs/2205.06750) [cs.LG] (2023). <https://doi.org/10.48550/arXiv.2205.06750>
16. Guffanti, T., Gammelli, D., D’Amico, S., Pavone, M.: Transformers for Trajectory Optimization with Application to Spacecraft Rendezvous. In: 2024 IEEE Aerospace Conference, pp. 1–13. IEEE, Big Sky, MT, USA (2024). <https://doi.org/10.1109/AERO58975.2024.10521334>
17. Takubo, Y., Guffanti, T., Gammelli, D., Pavone, M., D’Amico, S.: Towards Robust Spacecraft Trajectory Optimization via Transformers. arXiv e-prints [arXiv:2410.05585](https://arxiv.org/abs/2410.05585) [math.OC] (2024)
18. Majumdar, R., Sternberg, D.C., Albee, K., Jia-Richards, O.: Demonstration of the Dyna Reinforcement Learning Framework for Reactive Close Proximity Operations. In: AIAA SCITECH 2025 Forum. American Institute of Aeronautics and Astronautics, Orlando, FL (2025). <https://doi.org/10.2514/6.2025-1002>
19. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. *Proc. AAAI Conf. Artif. Intell.* (2018). <https://doi.org/10.1609/aaai.v32i1.11797>
20. Harris, A., Valade, T., Teil, T., Schaub, H.: Generation of spacecraft operations procedures using deep reinforcement learning. *J. Spacecr. Rocket.* **59**(2), 611–626 (2022). <https://doi.org/10.2514/1.A35169>
21. Herrmann, A., Schaub, H.: Reinforcement Learning for Small Body Science Operations. In: AAS Astrodynamics Specialist Conference, Charlotte, North Carolina (2022)
22. Stephenson, M., Schaub, H.: Reinforcement Learning For Earth-Observing Satellite Autonomy With Event-Based Task Intervals. In: AAS Rocky Mountain GN&C Conference, Breckenridge, CO (2024)
23. Stephenson, M., Mantovani, L., Cheval, A., Schaub, H.: Quantifying the Optimality of a Distributed RL-Based Autonomous Earth-Observing Constellation. In: AAS GN&C Conference, Breckenridge, CO (2025)
24. Van Wijk, D., Dunlap, K., Majji, M., Hobbs, K.: Safe spacecraft inspection via deep reinforcement learning and discrete control barrier functions. *J. Aerosp. Inf. Syst.* **21**(12), 996–1013 (2024). <https://doi.org/10.2514/1.I011391>
25. Dunlap, K., Hamilton, N., Hobbs, K.L.: Deep Reinforcement Learning for Scalable Multiagent Spacecraft Inspection. In: AAS/AIAA Space Flight Mechanics Meeting (2025)
26. Morgan, D., Chung, S.-J., Hadaegh, F.Y.: Model predictive control of swarms of spacecraft using sequential convex programming. *J. Guid. Control. Dyn.* **37**(6), 1725–1740 (2014). <https://doi.org/10.2514/1.G000218>
27. Morgan, D., Subramanian, G.P., Chung, S.-J., Hadaegh, F.Y.: Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming. *Int. J. Robot. Res.* **35**(10), 1261–1285 (2016). <https://doi.org/10.1177/0278364916632065>
28. Gaïas, G., D’Amico, S.: Impulsive maneuvers for formation reconfiguration using relative orbital elements. *J. Guid. Control. Dyn.* **38**(6), 1036–1049 (2015). <https://doi.org/10.2514/1.G000189>
29. Koenig, A.W., D’Amico, S.: Robust and safe N-spacecraft swarming in perturbed near-circular orbits. *J. Guid. Control. Dyn.* **41**(8), 1643–1662 (2018). <https://doi.org/10.2514/1.G003249>
30. Stephenson, M., Schaub, H.: Safe, Autonomous Multi-Agent Inspection of Space Objects Leveraging Relative Orbit Dynamics. In: Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, Hawaii (2025)
31. Stephenson, M., Schaub, H.: Comparing Probabilistic Roadmaps and Reinforcement Learning for Relative Motion Inspection of Space Objects. In: AAS Rocky Mountain GN&C Conference, Breckenridge, Colorado (2026)
32. Schaub, H., Junkins, J.L.: Analytical Mechanics of Space Systems. AIAA Education Series, 4th edn. American Institute of Aeronautics and Astronautics, Inc, Reston, VA (2018)
33. Stephenson, M.A., Schaub, H.: BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking. In: 75th International Astronautical Congress. IAF, Milan, Italy (2024)

34. Carnahan, S.J.K., Piggott, S., Schaub, H.: A New Messaging System for Basilisk. In: AAS Guidance and Control Conference, Breckenridge, CO (2020)
35. Kenneally, P.W., Piggott, S., Schaub, H.: Basilisk: a flexible, scalable and modular astrodynamics simulation framework. *J. Aerosp. Inf. Syst.* **17**(9), 496–507 (2020). <https://doi.org/10.2514/1.1010762>
36. Towers, M., Terry, J.K., Kwiatkowski, A., Balis, J.U., de Cola, G., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Tan, A.J.S., Younis, O.G.: *Gymnasium* arXiv e-prints [arXiv:2407.17032](https://arxiv.org/abs/2407.17032) [cs.LG] (2024)
37. Sutton, R.S., Barto, A.: *Reinforcement Learning: An Introduction*, 2nd edn. The MIT Press, Cambridge, Massachusetts London, England (2018)
38. Bradtke, S., Duff, M.: Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. In: *Advances in Neural Information Processing Systems*, vol. 7. MIT Press (1994). <https://doi.org/10.5555/2998687.2998736>
39. Sutton, R.S., Precup, D., Singh, S.: Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **112**(1–2), 181–211 (1999). [https://doi.org/10.1016/S0004-3702\(99\)00052-1](https://doi.org/10.1016/S0004-3702(99)00052-1)
40. Stephenson, M.A., Mantovani, L.Q., Schaub, H.: Learning policies for autonomous earth-observing satellite scheduling over semi-MDPs. *J. Aerosp. Inf. Syst.* **22**(9), 789–799 (2025)
41. Cheval, A., Schaub, H.: Reinforcement Learning with Hybrid Action Representation for Autonomous Strip Imaging Task Scheduling in High-Resolution Super-Agile Satellites, AIAA (2026)
42. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. arXiv e-prints [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) [cs.LG] (2017). <https://doi.org/10.48550/arXiv.1707.06347>
43. Gurobi Optimizer Reference Manual. Gurobi Optimization LLC (2023)
44. Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J.E., Jordan, M.I., Stoica, I.: RLlib: Abstractions for Distributed Reinforcement Learning. In: *Proceedings of 35th International Conference on Machine Learning*, vol. 80, pp. 3053–3062 (2018). <https://doi.org/10.5555/2998687.2998736>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.