(Preprint) AAS 26-XXX

COMPARING PROBABILISTIC ROADMAPS AND REINFORCEMENT LEARNING FOR RELATIVE MOTION INSPECTION OF SPACE OBJECTS

Mark Stephenson * **and Hanspeter Schaub** †

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 considered: a deep reinforcement learning (RL)-based solution developed in previous work is reviewed, and a probabilistic roadmap (PRM)-based solver is developed. The overall performance of solutions produced by the two methods are 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 computationally expensive, open-loop solutions as opposed to the closed-loop, onboard control provided by RL-based policies.

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. As with any motion planning task, offline planners and closed-loop approaches both offer strengths and weaknesses for solving the problem.

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

*Ph.D. Candidate, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80303. AIAA Member.

†Distinguished Professor and Department Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 3775 Discovery Dr., Colorado Center for Astrodynamics Research, Boulder, CO, 80303. AAS Fellow, AIAA Fellow.

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 such as 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 reference.[6,7] 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 for 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. Another paper approaches the problem with a continuous action space for continuous, low-thrust control, considering lighting conditions.[9] A recent study in Reference 10 uses the waypoint-based approach for distributed and centralized control of a swarm of servicers. In References 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 papers leverage the transformer architecture for RPO,[14,15] while Reference 16 uses RL with traditional path planning methods as a way of directing the search of the traditional methods while maintaining robustness. Shielded RL has been demonstrated as an effective method for various other spacecraft tasking problems,[17] 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 ensure safety in an RL-controlled inspection task with continuous control inputs using a CBF.[22,23] 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.

In this work, the RSO inspection problem (Figure 1) is considered for an RSO in an eccentric orbit with a Hill-frame-fixed attitude and an impulsively maneuvering servicer, subject to range, pointing, and lighting constraints for inspection. This builds on References 24 and 25, which finds RL-based solutions for one or many servicers and develop safety guarantees for those agents under various dynamic regimes. This paper provides an alternative offline solution method for the problem, combining probabilistic roadmaps (PRMs) and mixed-integer programs (MIPs) to find a dynamically feasible trajectory that completes the task while satisfying all constraints. This method is compared to the RL solutions, demonstrating stronger performance in fuel and time at the cost of higher computation time and open-loop solutions. While the PRM solutions more optimal than the RL solutions, they come at the cost of expensive, open-loop planning as opposed to autonomous closed-loop execution.

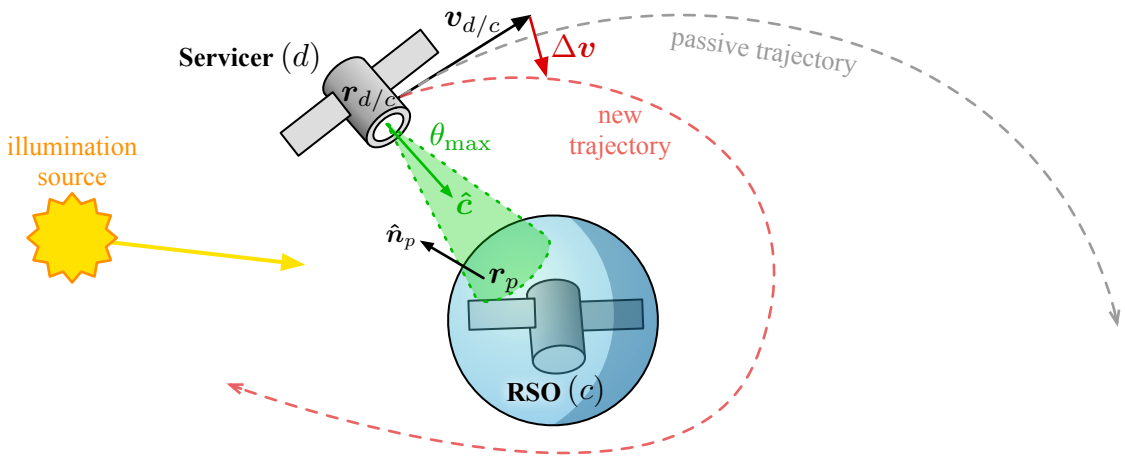


Figure 1: The RSO inspection problem.

PROBLEM FORMULATION

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.

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 $\boldsymbol{r}_{d/c}$ and velocity $\boldsymbol{v}_{d/c}$. The servicer is equipped with a body-fixed camera $\hat{\boldsymbol{c}}$ to inspect the RSO, and the attitude is actively controlled to point the instrument at the RSO (i.e., $\hat{\boldsymbol{c}}$ is kept antiparallel to $\boldsymbol{r}_{d/c}$). The servicer can control its motion with a thruster that can produce impulsive thrusts $\Delta\boldsymbol{v}$ 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 $\boldsymbol{K}$ consisting of the reciprocal squared semi-axes with a constraint in the general form

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

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

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

Analysis of the relative motion dynamics of the system is necessary 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 $\boldsymbol{x}_{d/c} = \begin{bmatrix} {}^{\mathcal{H}}\boldsymbol{r}_{d/c} & {}^{\mathcal{H}}\boldsymbol{v}_{d/c} \end{bmatrix}^T$ composed of the Hill-frame position and velocity of the deputy relative to the chief is given by

$$\boldsymbol{\Phi}_c^x(t, t_0) = \boldsymbol{A}_c(t)\boldsymbol{\Phi}_c^{\delta\boldsymbol{\alpha e}}(t, t_0)\boldsymbol{A}_c^{-1}(t_0) \tag{3}$$

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

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 (relative to the RSO origin) and normal vector $p_p = (\boldsymbol{r}_p, \hat{\boldsymbol{n}}_p)$. 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. Points are illuminated when the angle between the sun vector $\boldsymbol{r}_{\text{sun}}$ 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:

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

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

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

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:

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

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

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

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

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

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

When Equation 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.

*see Reference 26, chapter 14.5 for the full 6×6 matrices for $\boldsymbol{A}_c$, $\boldsymbol{A}_c^{-1}$, and $\boldsymbol{\Phi}^{\delta\boldsymbol{\alpha e}}$

Simulation Environment

A simulation of the problem is implemented using BSK-RL[†], a package for creating modular, high-fidelity RL environments for spacecraft tasking.[27] The underlying spacecraft simulation is Basilisk[‡], a high-performance spacecraft simulation package.[28] Rigid multi-body dynamics in the perturbed orbital environment and flight-proven flight software algorithms are used to simulate the environment.

The environment is used for training and validating the 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[29] for compatibility with other RL tooling. Because both methods are evaluated in the same realistic simulation environment, the comparison is apples-to-apples.

METHODS: 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.[30] Notably, the learning agent does not know about the environment explicity; 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.[25]

MDP Formulation

The single-servicer inspection task is formalized as partially-observable semi-Markov decision process (POsMDP). 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 Equation 1.
  - The servicer leaves the region surrounding RSO: $|\boldsymbol{r}_{d/c}| > 1$ 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: $t \geq 10$ 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. Two reference frames are used: the RSO Hill/body frame $\mathcal{H}$, and the Earth-Sun Hill frame $\mathcal{S}$.

[†] avslab.github.io/bsk_rl/
[‡] avslab.github.io/basilisk/

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

Element	Dim.	Description
${}^{\mathcal{H}}\boldsymbol{r}_{d/c}$	3	RSO-relative position
${}^{\mathcal{H}}\boldsymbol{v}_{d/c}$	3	RSO-relative velocity
${}^{\mathcal{H}}\hat{\boldsymbol{s}}$	3	Sun direction
${}^{\mathcal{S}}\boldsymbol{e}$	3	orbital eccentricity vector
${}^{\mathcal{S}}\boldsymbol{h}$	3	orbital angular momentum vector
${}^{\mathcal{S}}\boldsymbol{r}_{DE}$	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

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, of inspection points are grouped into M clusters. Each element of the observation is the fraction of points in the corresponding cluster that have been inspected. In this environment, $M = 15$ 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.

- **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.

- **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.

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

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

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

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

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

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

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

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)

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

- **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.

Training

Policies are trained in the high-fidelity implementation of this environment for a range of fuel use penaltys $\alpha \in [0.0, 3.0]$. The RLlib implementation of proximal policy optimization (PPO) is used to train policies.[31,32] The algorithm and training pipeline are modified to use semi-Markov discounting for advantage estimation.[33] The policies are represented by a fully connected multilayer perceptron (MLP) neural network of size 2×256, which takes about 100 μs per decision to evaluate in inference. The time use penalty $\alpha_t = 0.05$/orbit, and $\beta_{\text{success}} = \beta_{\text{fail}} = 1.0$. The policies presented in this paper are a refined version of those presented in reference,[25] with a 20M-step learning rate schedule of 5M steps at 10^{-4}, 5M decreasing to 10^{-5}, 5M decreasing to 10^{-6}, and 5M holding at 10^{-6}. Values are selected following an ablation study across algorithm and network parameters.

METHODS: PROBABILISTIC ROADMAP

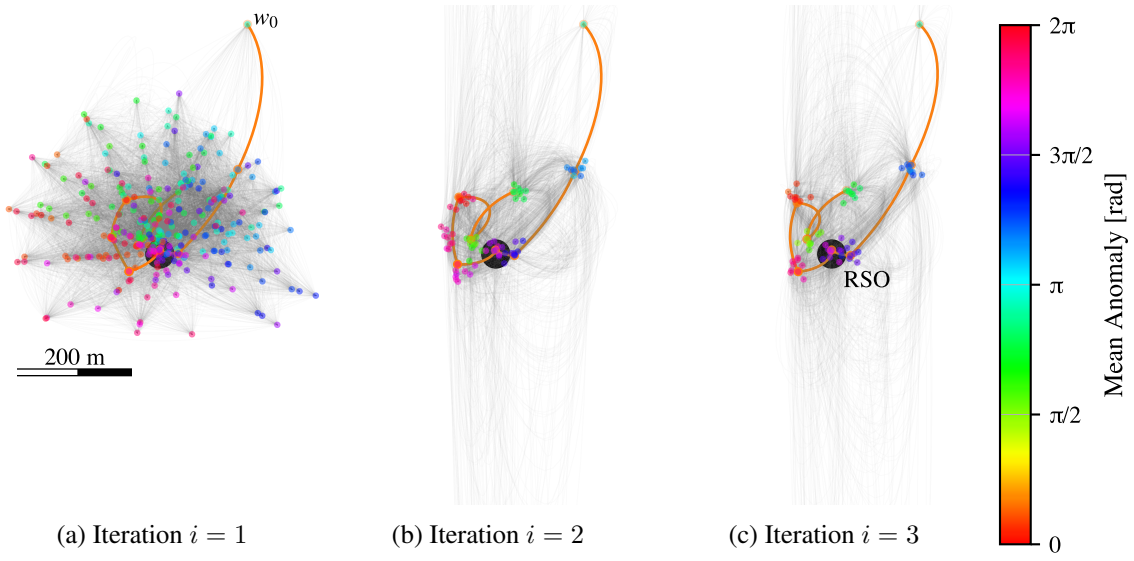
The PRM-based approach for the problem consists of iteratively constructing a roadmap of waypoints, then using a mixed-integer quadratically constrained quadratic program (MIQCQP) to find a trajectory in the roadmap that satisfies the solution constraints.

Algorithm Structure

Algorithm 1, 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, $w_w = (\boldsymbol{r}_w, M_w)$. 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 1 Inspection trajectory planning function

function PLANINSPECTIONTRAJECTORY($\boldsymbol{r}_0, M_0, \mathcal{L}$)
  $w_0 \leftarrow (\boldsymbol{r}_0, M_0)$
  for $i \in 1, 2, 3, \ldots$ **do**
    if $i = 1$ **then**
      $\mathcal{W} \leftarrow \{w_0\} \cup$ SEEDWAYPOINTS($\mathcal{L}, N_{\text{seed}}$)
    else
      $\mathcal{W} \leftarrow \{w_0\} \cup$ RESAMPLEWAYPOINTS($\mathcal{W}_{\text{sol}}[1:], N_{\text{resample}}, \sigma_r, \sigma_M$)
    $\mathcal{E} \leftarrow$ CONNECTGRAPH($\mathcal{W}, N_{\text{conn}}$)
    $\boldsymbol{x} \leftarrow$ construct and solve MIQCQP (Equation 18-27)
    $\mathcal{W}_{\text{sol}} \leftarrow$ PARSESOLUTION($\boldsymbol{x}, \mathcal{W}$)
  return $\mathcal{W}_{\text{sol}}$



(a) Iteration $i = 1$ (b) Iteration $i = 2$ (c) Iteration $i = 3$

Figure 2: PRM solution iterations.

Figure 2 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.

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 2, 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 $\boldsymbol{r}_i$ is sampled uniformly between the conjunction radius R_k and the maximum inspection radius R_{max} in the inspection point normal direction $\hat{\boldsymbol{n}}_p$. The mean anomaly M_i is sampled uniformly over times where the point is illuminated.

Algorithm 2 Generate N_{seed} waypoints with visibility of each illuminated point

function SEEDWAYPOINTS($\mathcal{L}, N_{\text{seed}}$)
  $\mathcal{W} \leftarrow \{\}$
  for $p_p \in \mathcal{L}$ **do**
    for $i \in 1, ..., N_{\text{seed}}$ **do**
      $\boldsymbol{r}_i \leftarrow \boldsymbol{r}_p + \hat{\boldsymbol{n}}_p \cdot \text{Uniform}(R_k - \|\boldsymbol{r}_p\|, R_{\text{max}})$
      $M_i \leftarrow \text{Uniform}(M \in [0, 2\pi] \mid L(p_p, M) = \text{True})$
      $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\boldsymbol{r}_i, M_i)\}$
  return $\mathcal{W}$

In further iterations, Algorithm 3, 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 3 Generate N_{resample} waypoints around each waypoint in the solution

function RESAMPLEWAYPOINTS($\mathcal{W}_{\text{sol}}, N_{\text{resample}}, \sigma_r, \sigma_M$)
  $\mathcal{W} \leftarrow \{\}$
  for $w_w \in \mathcal{W}_{\text{sol}}$ **do**
    $\mathcal{W} \leftarrow \mathcal{W} \cup \{w_w\}$
    for $i \in 1, ..., N_{\text{resample}}$ **do**
      do
        $\boldsymbol{r}_i \leftarrow \text{Normal}(\boldsymbol{r}_w, \sigma_r \boldsymbol{I}_{3\times 3})$
      while $\|\boldsymbol{r}_i\| < R_k$
      $M_i \leftarrow \text{Normal}(M_w, \sigma_M)$
      $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\boldsymbol{r}_i, M_i)\}$
  return $\mathcal{W}$

Algorithm 4, CONNECTGRAPH, builds the PRM by drawing edges between nearby waypoints. The structure of the problem is exploited to use the difference in mean 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 (Equation 17).

Algorithm 4 Add edges between nearby waypoints

function CONNECTGRAPH($\mathcal{W}, N_{\text{conn}}$)
  $\mathcal{E} \leftarrow \{\}$
  for $w_i \in \mathcal{W}$ **do**
    for $w_j \in \mathcal{W}$ **do**
      $d_j \leftarrow (M_j - M_i) \bmod (0, 2\pi]$
    for N_{conn} smallest d_j where COLLISIONFREE(w_i, w_j) **do**
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(w_i \to w_j)\}$
  return $\mathcal{E}$

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 $(i \to j)$ 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 $\boldsymbol{x}$.
- $\Delta v_{0,i}^{\text{lin}}$ 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 $\boldsymbol{z}$.
- $\mathbb{I}$ is the inspectability matrix, which is size $\text{len}(\boldsymbol{z}) \times \text{len}(\boldsymbol{x})$. 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:

$$\mathbb{I}_{p,(i,j)} = \bigwedge_{M=M_i}^{M_j} I(p, \boldsymbol{r}_{i\to j}(M), M) \tag{14}$$

  where $\boldsymbol{r}_{i\to j}(M)$ is the position along the trajectory between two waypoints w_i and w_j as a function of mean anomaly M.

The linear and quadratic Δv costs are given by

$$\Delta v_{0,i}^{\text{lin}} = \|\boldsymbol{v}_0 - \boldsymbol{v}_{\text{in}}(w_j, w_k)\| \tag{15}$$

$$\Delta v_{ijk}^{\text{quad}} = \|\boldsymbol{v}_{\text{out}}(w_i, w_j) - \boldsymbol{v}_{\text{in}}(w_j, w_k)\| \tag{16}$$

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

$$\begin{bmatrix} \boldsymbol{r}_j \\ \boldsymbol{v}_{\text{out}} \end{bmatrix} = \boldsymbol{\Phi}\big(M_i + \big((M_j - M_i) \bmod 2\pi\big), M_i\big) \begin{bmatrix} \boldsymbol{r}_i \\ \boldsymbol{v}_{\text{in}} \end{bmatrix} \tag{17}$$

is solved for $\boldsymbol{v}_{\text{in}}$ and $\boldsymbol{v}_{\text{out}}$.

The complete MIQCQP for path planning over a set of waypoints $\mathcal{W}$ and edges $\mathcal{E}$ is formulated in Equation 18 through 27. 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).

$$\text{minimize } \zeta \left(\sum_i \Delta v_{0,i}^{\text{lin}} x_{0,i} + \sum_i \sum_j \sum_k x_{i,j} \Delta v_{i,j,k}^{\text{quad}} x_{j,k} \right) + \frac{1-\zeta}{n} \sum_i \sum_j \Delta M_{i,j} x_{i,j} \tag{18}$$

$$\begin{aligned}
\text{subject to } & x_{i,j} \in \{0,1\} && \forall\, (w_i \to w_j) \in \mathcal{E} && (19)\\
& \sum_k x_{j,k} \le \sum_i x_{i,j} + b \le 1 && \forall\, j;\ b = 1 \text{ if } j = 0 \text{ else } b = 0 && (20)\\
& n_i \in \{0, ..., n_{\max}\} && \forall\, w_i \in \mathcal{W};\ n_{\max} = 10 && (21)\\
& 0 \le \big(2\pi(n_j - n_i) + M.j - M.i\big) x_{i,j} \le 2\pi && \forall\, (w_i \to w_j) \in \mathcal{E} && (22)\\
& x_{0,i} = 0 && \forall\, i \mid \Delta v_{0,i}^{\text{lin}} > \Delta v_{\max} && (23)\\
& x_{i,j} + x_{j,k} \le 1 && \forall\, i,j,k \mid \Delta v_{i,j,k}^{\text{quad}} > \Delta v_{\max} && (24)\\
& z_p \in \{0,1\} && \forall\, p \in \mathcal{L} && (25)\\
& \mathbb{I}\boldsymbol{x} \ge -\mathbb{M}(1 - \boldsymbol{z}) && \mathbb{M} \gg 1 && (26)\\
& \sum_p z_p \ge 0.9|\mathcal{L}| && && (27)
\end{aligned}$$

The objective function (Equation 18) 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 19 through 22 are used to guarantee a single trajectory originating at w_0. Equation 20 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 disconected cycles in the solution. To prevent these, Equation 22 is introduced which assigns an integer orbit number n_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 23 and 24 are added for every Δv that exceeds $\Delta v_{\max}$. For the linear objective term, the corresponding constraint (23) trivially sets infeasible edges to be unselected; for the quadratic

objective, the corresponding constraint (24) allows for incoming or outcoming edges to be selected but prevents both from being selected for maneuvers that are infeasible.

Finally, constraints 25 through 27 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 $\boldsymbol{x}$, $\mathbb{I}\boldsymbol{x}$ 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.

Parsing the Optimization Solution

Algorithm 5 gives the method for extracting the sequence of waypoints in the solution from the optimization variable $\boldsymbol{x}$.

Algorithm 5 Parse the MIQCQP solution

```
function ParseSolution(x, W)
    W_sol ← [w_0]
    while True do
        w_i ← W_sol[−1]
        for j | x_{i,j} ∈ x do
            if x_{i,j} = 1 then
                Append(W_sol, w_j)
                break
        else                                   ▷ Return if no w_j is connected from w_i
            return W_sol
```

Algorithm Parameters

When evaluating the algorithm, the following parameters are used: $N_{\text{conn}} = 40$, $N_{\text{seed}} = 5$, $N_{\text{resample}} = 8$, $\sigma_\rho = 10$ m, $\sigma_M = 0.2$, and iterations $= 3$. 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 seconds beyond when the first incumbent solution is found; on the following warm-started iterations, the solver is run for 120 seconds per iteration or until the optimal solution is found. Gurobi is used to solve the MIQCQP.[34]

When executing the plan, the 2-point boundary value problem in Equation 17 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.

RESULTS

The PRM-based solver is compared to the RL benchmark over randomly seeded environments. The PRM-based solver is solved for a range of objective weights ζ while the RL policies are trained with a range of fuel penalties α to find the fuel-time Pareto front for each method. The resulting trajectories for some representative cases are also compared.

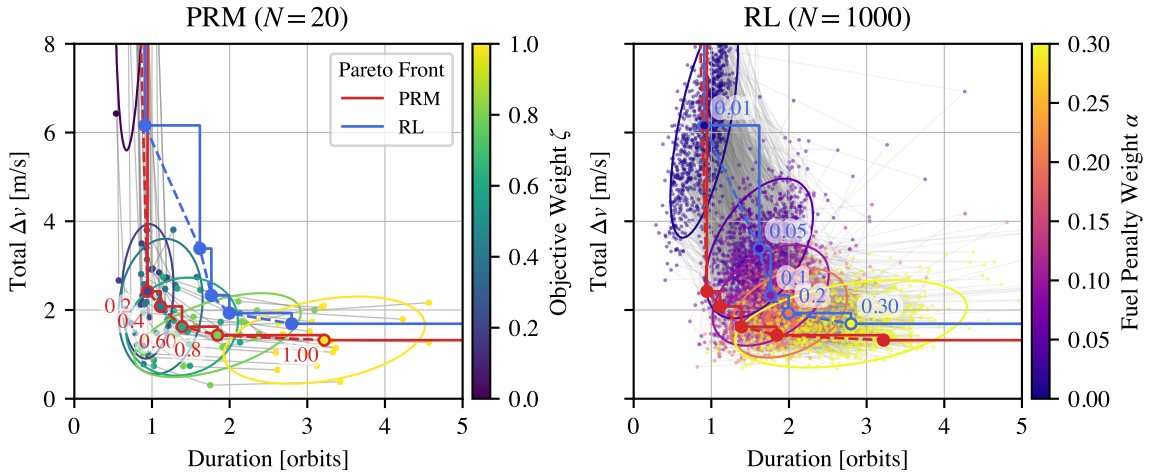


Figure 3: Pareto fronts for PRM and RL solutions for N seeds at each value of ζ or α, respectively; 2σ covariances.

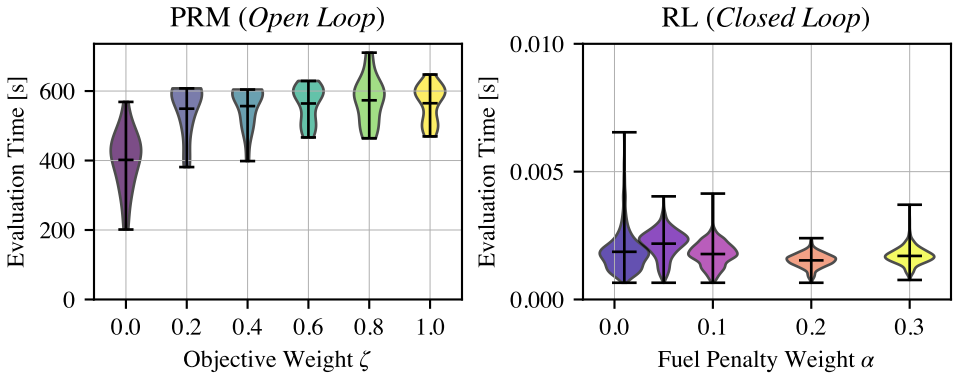
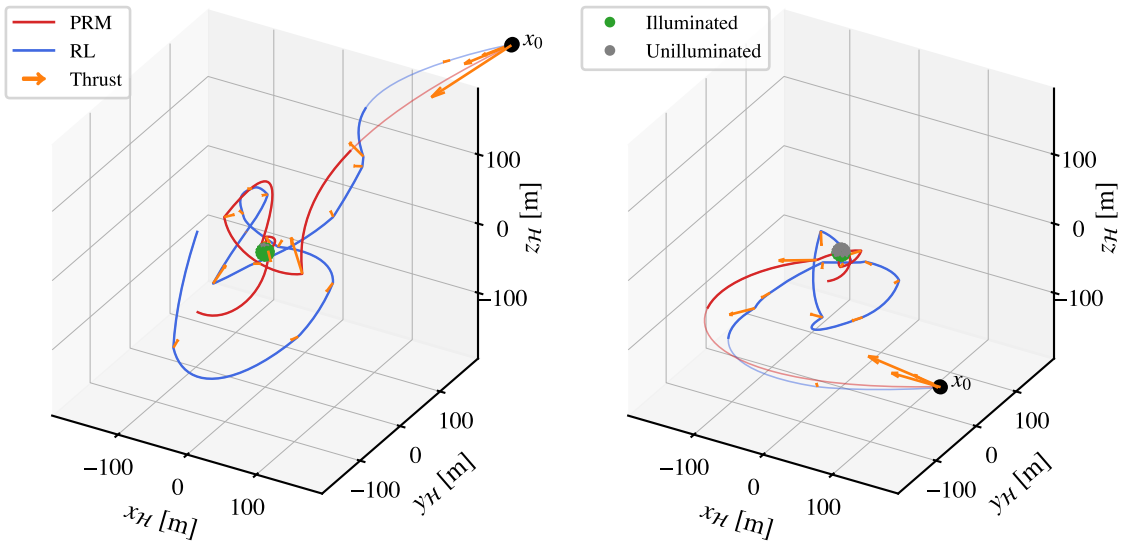


Figure 4: Open loop PRM planning time versus closed loop RL execution time.

Pareto Fronts

Figure 3 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 4 compares the execution times of each method: The PRM method takes about 10 minutes 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



(a) High illumination case.

(b) Low illumination case.

Figure 5: Comparison of PRM and RL solutions.

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

Example Solutions

Figure 5 compares the trajectories of the PRM and RL solutions in two cases, one with high illumination and one with low illumination due to the β angle of the orbit. The PRM uses $\zeta = 0.5$ and the RL solution uses the $\alpha = 0.1$ policy. 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 is has learned.

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. Depending on application, both methods offer novel developments for the RSO inspection task.

ACKNOWLEDGEMENT

This work is supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) 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).

REFERENCES

[1] M. Sabatini, R. Volpe, and G. Palmerini, "Centralized Visual Based Navigation and Control of a Swarm of Satellites for On-Orbit Servicing," *Acta Astronautica*, Vol. 171, June 2020, pp. 323–334, 10.1016/j.actaastro.2020.03.015.

[2] B. Bernhard, C. Choi, A. Rahmani, S.-J. Chung, and F. Hadaegh, "Coordinated Motion Planning for On-Orbit Satellite Inspection Using a Swarm of Small-Spacecraft," *2020 IEEE Aerospace Conference*, Mar. 2020, pp. 1–13, 10.1109/AERO47225.2020.9172747.

[3] Y. K. Nakka, W. Hönig, C. Choi, A. Harvard, A. Rahmani, and S.-J. Chung, "Information-Based Guidance and Control Architecture for Multi-Spacecraft On-Orbit Inspection," *Journal of Guidance, Control, and Dynamics*, Vol. 45, July 2022, pp. 1184–1201, 10.2514/1.G006278.

[4] T. Lauinger and S. Ulrich, "Path Planning for Optimal Coverage of Orbiting Space Structures Using Lissajous Curves," *AAS/AIAA Space Flight Mechanics Meeting*, Jan. 2025.

[5] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically Optimal Sampling-based Kinodynamic Planning," Feb. 2016.

[6] M. Iversflaten, A. Hansson, D. Sternberg, O. Jia-Richards, and K. Albee, "Robust Replanning for Multi-Agent SmallSat Inspection in Failure Scenarios," *AIAA SCITECH 2025 Forum*, Orlando, FL, American Institute of Aeronautics and Astronautics, Jan. 2025, 10.2514/6.2025-0183.

[7] K. Albee, D. C. Sternberg, A. Hansson, D. Schwartz, R. Majumdar, and O. Jia-Richards, "Architecting Autonomy for Safe Microgravity Free-Flyer Inspection," *IEEE Aerospace Conference*, Mar. 2025.

[8] J. Aurand, S. Cutlip, H. Lei, K. Lang, and S. Phillips, "Exposure-Based Multi-Agent Inspection of a Tumbling Target Using Deep Reinforcement Learning," May 2023, 10.48550/arXiv.2302.14188.

[9] D. v. Wijk, K. Dunlap, M. Majji, and K. L. Hobbs, "Deep Reinforcement Learning for Autonomous Spacecraft Inspection Using Illumination," Aug. 2023, 10.48550/arXiv.2308.02743.

[10] H. Lei, Z. S. Lippay, A. Zaman, J. Aurand, A. Maghareh, and S. Phillips, "Stability Analysis of Deep Reinforcement Learning for Multi-Agent Inspection in a Terrestrial Testbed," *AIAA SCITECH 2025 Forum*, Orlando, FL, Jan. 2025.

[11] A. Brandonisio, M. Lavagna, and D. Guzzetti, "Reinforcement Learning for Uncooperative Space Objects Smart Imaging Path-Planning," *The Journal of the Astronautical Sciences*, Vol. 68, Dec. 2021, pp. 1145–1169, 10.1007/s40295-021-00288-7.

[12] A. Brandonisio, L. Capra, and M. Lavagna, "Deep Reinforcement Learning Spacecraft Guidance with State Uncertainty for Autonomous Shape Reconstruction of Uncooperative Target," *Advances in Space Research*, July 2023, p. S0273117723005276, 10.1016/j.asr.2023.07.007.

[13] H. Krasowski, J. Thumm, M. Müller, L. Schäfer, X. Wang, and M. Althoff, "Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking," Nov. 2023, 10.48550/arXiv.2205.06750.

[14] T. Guffanti, D. Gammelli, S. D'Amico, and M. Pavone, "Transformers for Trajectory Optimization with Application to Spacecraft Rendezvous," *2024 IEEE Aerospace Conference*, Big Sky, MT, USA, IEEE, Mar. 2024, pp. 1–13, 10.1109/AERO58975.2024.10521334.

[15] Y. Takubo, T. Guffanti, D. Gammelli, M. Pavone, and S. D'Amico, "Towards Robust Spacecraft Trajectory Optimization via Transformers," Oct. 2024.

[16] R. Majumdar, D. C. Sternberg, K. Albee, and O. Jia-Richards, "Demonstration of the Dyna Reinforcement Learning Framework for Reactive Close Proximity Operations," *AIAA SCITECH 2025 Forum*, Orlando, FL, American Institute of Aeronautics and Astronautics, Jan. 2025, 10.2514/6.2025-1002.

[17] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe Reinforcement Learning via Shielding," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, Apr. 2018, 10.1609/aaai.v32i1.11797.

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

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

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

[21] M. Stephenson, L. Mantovani, A. Cheval, and H. Schaub, "Quantifying the Optimality of a Distributed RL-Based Autonomous Earth-Observing Constellation," *AAS GN&C Conference*, Breckenridge, CO, Feb. 2025.

[22] D. Van Wijk, K. Dunlap, M. Majji, and K. Hobbs, "Safe Spacecraft Inspection via Deep Reinforcement Learning and Discrete Control Barrier Functions," *Journal of Aerospace Information Systems*, Vol. 21, Dec. 2024, pp. 996–1013, 10.2514/1.I011391.

[23] K. Dunlap, N. Hamilton, and K. L. Hobbs, "Deep Reinforcement Learning for Scalable Multiagent Spacecraft Inspection," *AAS/AIAA Space Flight Mechanics Meeting*, Jan. 2025.

[24] M. Stephenson, D. H. Prats, and H. Schaub, "Autonomous Satellite Inspection in Low Earth Orbit with Optimization-Based Safety Guarantees," *International Workshop on Planning & Scheduling for Space*, Toulouse, France, Apr. 2025.

[25] M. Stephenson and H. Schaub, "Safe, Autonomous Multi-Agent Inspection of Space Objects Leveraging Relative Orbit Dynamics," *Advanced Maui Optical and Space Surveillance Technologies Conference*, Maui, Hawaii, Sept. 2025.

[26] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. AIAA Education Series, Reston, VA: American Institute of Aeronautics and Astronautics, Inc, fourth edition ed., 2018.

[27] M. A. Stephenson and H. Schaub, "BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking," *75th International Astronautical Congress*, Milan, Italy, IAF, Oct. 2024.

[28] P. W. Kenneally, S. Piggott, and H. Schaub, "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *Journal of Aerospace Information Systems*, Vol. 17, Sept. 2020, pp. 496–507, 10.2514/1.I010762.

[29] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. de Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. J. S. Tan, and O. G. Younis, "Gymnasium," Oct. 2023.

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

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, 10.48550/arXiv.1707.06347.

[32] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80, June 2018, pp. 3053–3062, 10.48550/arXiv.1712.09381.

[33] S. Bradtke and M. Duff, "Reinforcement Learning Methods for Continuous-Time Markov Decision Problems," *Advances in Neural Information Processing Systems*, Vol. 7, MIT Press, 1994, 10.5555/2998687.2998736.

[34] "Gurobi Optimizer Reference Manual," Gurobi Optimization LLC, 2023.