Project Title: Active Information Gathering for Non-Cooperative Resident Space Objects Using Reinforcement Learning

Team Members: Rahul Ayanampudi, Sebastian Martinez

Emails: rayanam@stanford.edu, sebasmp@stanford.edu

1 Motivation

Future missions for in-orbit servicing and active debris removal will require a servicer spacecraft to autonomously approach and characterize a non-cooperative Resident Space Object (RSO). Before proximity operations can be attempted safely, the servicer must fully characterize the target. This process involves determining the relative orbit and 3D shape of the target Kruger et al. (2024).

Current observer systems are passive, leading to issues with range ambiguity and slow, uncertain convergence. This project will enable the agent to actively decide where to maneuver to acquire the most informative observations Kruger and D'Amico (2025). However, the spacecraft active sensing problem can be very complex as it involves image processing, navigation, guidance, and control subproblems coupled with the fact that maneuvers that are optimal for orbit determination (e.g., those that induce parallax) are not necessarily optimal for 3D shape reconstruction, which requires a diverse set of viewing angles.

In order to bound the problem, the team will assume the agent has perfect knowledge of the state, dynamics, and control for itself and the target. Hence, the team will focus on determining the optimal sequence of maneuvers to most efficiently reconstruct a target's 3D shape.

2 Methods

2.1 Problem Description

The simplified active sensing problem can be formulated as a Partially Observable Markov Decision Process (POMDP). The environment is a representation of the Low Earth Orbit (LEO) orbital space containing both the servicer (the agent) and the target RSO. The environment is partially observable.

The physical state (s_{phys}) of the servicer relative to the target (position, velocity, attitude) is assumed to be perfectly known to the agent. However, the true 3D shape of the RSO ($\mathcal{S}_{RSO_{shape}}$) is the environment's hidden state. The agent never knows $\mathcal{S}_{RSO_{shape}}$ directly. It can only infer it by taking observations.The agent's internal belief state ($b_{RSO_{shape}}$) is its own model of this hidden state.

The environment's state changes in response to the agent's actions. The physical part of the environment is deterministic. When the agent takes an action a (a Δv maneuver), the resulting transition to physical state s'_{phys} is exactly predictable using the relative orbital elements propagator (no randomness in the physics). The RSO's true shape $\mathcal{S}_{RSO_{shape}}$ is unknown to the agent and does not change over time.

The interaction loop between the agent and the environment is simple. The agent (MCTS planner) selects and executes an action a (a Δv maneuver) and the environment responds by transitioning the agent to a new, deterministic physical state s'_{phys} (via orbital mechanics). It provides the agent's camera with a new observation o (an image/scan) of the hidden $\mathcal{S}_{RSO_{shape}}$ from the new vantage point in the trajectory. The agent calculates its own reward $R = \text{InfoGain} - \text{Cost}$. The $Cost$ comes from the action a, and the InfoGain is calculated by the agent itself based on how the observation o changed its internal belief.

The primary source of uncertainty is the agent's lack of knowledge about the RSO's 3D shape, which is modeled by the agent's internal belief state, b_{shape}. This could be represented as a probabilistic voxel grid, where each 3D cell i has an associated probability $P_i(\text{occupied})$ of containing mass. The mission goal is to drive these probabilities from their initial 0.5 (maximum uncertainty as all possible shapes are equally likely) to 0 or 1.

The second source is the inherent sensor noise in the observation process. Even with a perfect maneuver, the camera used to observe the target is not perfect. An observation o (e.g., a point cloud) is only a noisy, probabilistic (sampled from distribution) reflection of the target's true geometry, not a perfect ground-truth snapshot. The sensor noise is modeled within the observation likelihood

function, $P(o|\mathcal{S}, s_{phys})$. This function defines the probability of getting observation o given a true underlying shape $\mathcal{S}$ and a viewing state s_{phys}.

2.2 Reinforcement Learning Approach

While Monte Carlo Tree Search (MCTS) provides a solid foundation for planning over the agent's belief space, its efficiency and scalability can be further improved through reinforcement learning (RL). The team adopts an AlphaZero-inspired framework Silver et al. (2017) in which a neural network $f_\theta(s)$ jointly predicts a policy $\pi_\theta(a|s)$ and a value $V_\theta(s)$, representing the expected discounted return.

During training, trajectories are generated by running MCTS over the propagator dynamics. Then, at each root state s_t, MCTS uses the current network predictions to guide exploration and evaluates nodes using the predicted value V_θ instead of performing full rollouts.

The resulting search statistics, visit counts and estimated returns, are used to update the network parameters by minimizing:

$$L(\theta) = (R - V_\theta(s))^2 - \pi_{MCTS}(a \mid s)^\top \log \pi_\theta(a|s),$$

where the MCTS-improved policy at the root is defined using visit counts:

$$\pi^{(t)}_{MCTS}(a \mid s) = \frac{N(s_t, a)^{1/\tau}}{\sum_{a' \in \mathcal{A}} N(s_t, a')^{1/\tau}}$$

and the discounted return over the episode is

$$R = \sum_{t=0}^{T} \gamma^t \left[\text{InfoGain}(b_{\text{shape},t}, b_{\text{shape},t+1}) - c\|\Delta v_{a_t}\|\right]$$

The overall planning-and-learning process alternates between two stages:

1. **MCTS planning**: At each step, MCTS is run from the current state to compute an improved policy π_{MCTS}. Then, the tuple (s_t, π_{MCTS}, R) is stored as a training example.
2. **Neural Network Training**: Using the collected MCTS data, the policy–value network is trained via supervised learning. Once the value head is sufficiently accurate, its predictions are used as bootstrap estimates in subsequent MCTS calls, allowing the search to avoid expensive rollouts and evaluate nodes more efficiently.

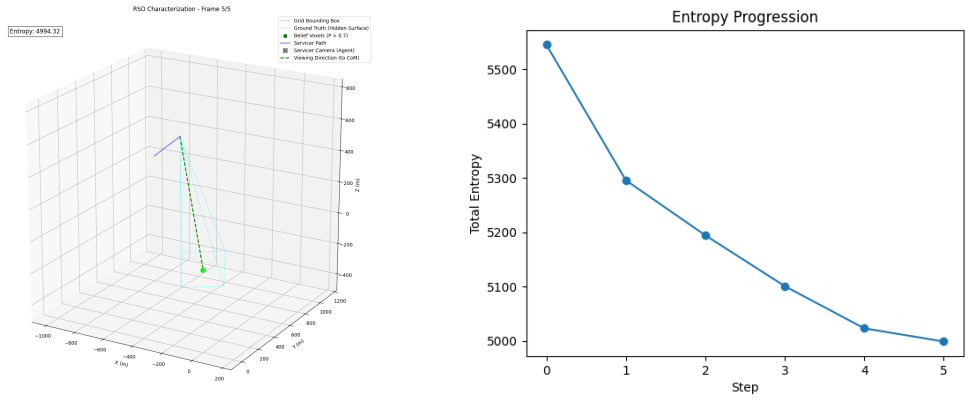
In this setup, MCTS acts to improve the policy, while the network learns to predict both the discounted return and the policy. Over repeated iterations, the system converges toward a maneuver policy that selects Δv actions that are the most informative under orbital constraints, enabling faster and more efficient 3D shape reconstruction with fewer simulated observations.

3 Preliminary Experiments and Results

Simulating active characterization of a non-cooperative Resident Space Object (RSO) requires an orbital dynamics environment. The team has developed a custom simulator shown in Figure 1a that defines the servicer and target spacecraft configurations, establishes their initial states, and propagates their relative orbital elements. The servicer can maneuver (impulsive Δv) and make observations of the target through its camera (modeled with specific resolution, field of view, and noise) at each time step. A ray casting algorithm (3D DDA) determines which cells of the target's probabilistic voxel grid the servicer can observe Amanatides and Woo (1987). The extent of characterization of the target is determined by computing the Shannon entropy of the agent's voxel grid belief.

Monte Carlo Tree Search (MCTS) has been implemented as a baseline for the reinforcement learning. At each time step, a new tree is generated. Each node in the tree represents a state and it has a branch for each of the 13 possible actions. The agent can continue along its current path by making no maneuver or a small/large Δv in the radial, tangential, and normal (RTN) directions. A tree depth of 3 was used for the experiments conducted thus far to bound the computation during testing. The value, or reward, of each node is the difference between the information gained, which is quantified by the entropy reduction, and the cost of the action. Based on the action value functions, $Q(s, a)$

computed with MCTS, of each node of the tree, the optimal actions are determined. The agent's current simulation state, immediate optimal action, associated reward, and corresponding next state are saved to a replay buffer. Then, the optimal action for the current state of the agent is taken in the simulation and the agent's state is propagated to the next time step. This process is repeated at the new state of the agent.



(a) Custom Orbital Simulator

(b) Entropy Progression

Figure 1: Current Implementation

As shown in Figure 1b, the MCTS is applying actions that decrease the entropy, as expected. The next step of the project is to use the replay buffer to train a neural network to approximate the MCTS controller.

4 Next Steps

- Baseline MCTS works in a single episode. To integrate the method into the RL pipeline, the code must be adjusted to run through multiple episodes.
- For every episode, store the tuple (π_{MCTS}, state, rollout return) computed at the root state. These samples will be used for supervised training of the policy and value network.
- Implement the neural policy-value network in Pytorch.
- Build the full training loop. Alternate between running MCTS with frozen network to generate data, and update the network using the stored samples. This is the self-play loop.

5 Team Contributions

- **Rahul Ayanampudi**: Developed probabilistic voxel grid to characterize how much of the RSO has been seen and the relative orbital dynamics simulator. Will help refine the reinforcement learning problem.
- **Sebastian Martinez**: Configured the reinforcement learning problem.

References

John Amanatides and Andrew Woo. 1987. A Fast Voxel Traversal Algorithm for Ray Tracing. *EuroGraphics* (1987).

Justin Kruger and Simone D'Amico. 2025. Autonomous Navigation of a Satellite Swarm using Inter-Satellite Bearing Angles. *IEEE Trans. Aerospace Electron. Systems* (2025).

Justin Kruger, Tommaso Guffanti, Tae Ha Park, Mason Murray-Cooper, Samuel Low, Toby Bell, Simone D'Amico, Christopher Roscoe, and Jason Westphal. 2024. Adaptive End-to-End Architecture for Autonomous Spacecraft Navigation and Control During Rendezvous and Proximity Operations. In *AIAA SCITECH 2024 Forum*. doi:10.2514/6.2024-0001

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815 [cs.AI] https://arxiv.org/abs/1712.01815