Active Characterization of Non-Cooperative Resident Space Objects Using Reinforcement Learning

Stanford AA228 Project

Rahul Ayanampudi
Department of Aeronautics and Astronautics
Stanford University
rayanam@stanford.edu

Sebastian Martinez
Department of Aeronautics and Astronautics
Stanford University
sebasmp@stanford.edu

Abstract

Future missions for in-orbit servicing and active debris removal require autonomous spacecraft to characterize non-cooperative Resident Space Objects (RSOs) prior to proximity operations. This paper presents a reinforcement learning framework for active characterization, enabling a servicer spacecraft to autonomously plan maneuvers that maximize information gain regarding a target's 3D shape. We formulate the problem as a Partially Observable Markov Decision Process (POMDP) where the agent maintains a probabilistic voxel grid belief of the target. We implement a custom orbital dynamics simulator with Monte Carlo Tree Search (MCTS) as a classical planning baseline and develop an AlphaZero-inspired neural network that learns to approximate MCTS planning through self-play. Our results demonstrate that AlphaZero MCTS achieves superior performance over pure MCTS, reducing belief entropy by 97.1% with only 0.11 m/s fuel expenditure compared to pure MCTS which achieves 96.77% reduction with 0.31 m/s. The AlphaZero agent executes 65% less fuel and 57% fewer maneuvers while achieving better reconstruction quality, validating the approach for fuel-efficient autonomous shape reconstruction in Low Earth Orbit (LEO) environments.

1 Introduction

The proliferation of space debris and the growing need for on-orbit servicing have made the autonomous characterization of Resident Space Objects (RSOs) a critical capability for future space missions. Before a servicer spacecraft can safely attempt docking or debris removal, it must fully characterize the target, a process that involves determining the relative orbit and reconstructing the target's 3D shape [Kruger et al., 2024]. Current observer systems largely rely on passive sensing, which suffers from range ambiguities and slow convergence rates due to a lack of geometric diversity in viewing angles.

To overcome these limitations, we propose an active sensing framework where the agent explicitly decides on maneuvers (Δv) to acquire the most informative observations. This problem presents significant challenges in several domains: orbital mechanics, guidance, navigation, and control (GNC), and computer vision. Furthermore, maneuvers that are optimal for orbit determination, such as those inducing parallax, are not necessarily optimal for shape reconstruction, which requires a diverse set of viewing angles to resolve occlusions.

We model this problem as a POMDP. The input to our algorithm includes the fully observable physical state of the servicer relative to the target and the agent's internal belief state regarding the target's shape. The output is a discrete action representing an impulsive maneuver in the Radial-Tangential-Normal (RTN) frame. Training a policy that balances the cost of fuel expenditure against the information gain derived from reducing uncertainty in the target's volumetric model is done by leveraging an AlphaZero-style reinforcement learning architecture.

The discrete action space problem formulation offers some advantages. First, impulsive spacecraft maneuvers naturally correspond to discrete actions, as thrusters execute finite-duration burns modeled as instantaneous velocity changes. Second, tree search can systematically evaluate possible maneuvers at each decision point [Kocsis and Szepesvári, 2006], enabling principled long-horizon planning. This is critical for active sensing, where the information gain of future observations depends on selecting the right sequence of maneuvers.

The remainder of this paper is organized as follows: Section 2 reviews related work in autonomous spacecraft navigation and reinforcement learning. Section 3 describes the simulation environment and POMDP formulation. Section 4 presents the MCTS baseline and AlphaZero training framework. Section 5 reports experimental results comparing pure MCTS and AlphaZero performance. Section 6 concludes with future work directions.

2 Related Work

The challenge of autonomous spacecraft navigation has been extensively studied in the context of cooperative rendezvous. However, non-cooperative proximity operations introduce high uncertainty. Kruger et al. [2024] proposed an adaptive end-to-end architecture for autonomous navigation, highlighting the necessity of robust estimation filters for proximity operations. Building on this, Kruger and D'Amico [2025] explored autonomous navigation using inter-satellite bearing angles, emphasizing the value of active maneuvering to improve observability in orbital regimes.

Our work draws inspiration from general reinforcement learning advancements in complex decision-making domains. Specifically, Silver et al. [2017] introduced the AlphaZero algorithm, which uses Monte Carlo Tree Search (MCTS) with a deep neural network to master games like Chess and Go without human domain knowledge. The goal is to adapt this "planning-and-learning" paradigm to the aerospace domain.

The simulation utilizes Relative Orbital Elements (ROE), a geometric state representation developed by Willis [2023] that describes the relative motion trajectory. For the sensor model and belief update, we utilize the fast voxel traversal algorithm by Amanatides and Woo [1987] to efficiently simulate ray casting and occlusion in 3D space.

3 Dataset and Simulation Environment

We formulate active RSO characterization as a POMDP. The custom LEO simulator contains the servicer and the target RSO. The physical state (s_{phys}) of the servicer relative to the target (position and velocity) is assumed to be perfectly known to the agent. However, the true 3D shape of the target ($\mathcal{S}_{RSO_{shape}}$) is unknown and does not change with time. The agent's belief state $b_{RSO_{shape}}$ of the target is a probabilistic voxel grid where each cell i contains a probability P_i of being occupied. Initially, all cells are set to $P_i = 0.5$, representing maximum uncertainty. At the beginning of each episode, the relative orbital elements of the agent is initialized according to a random uniform Monte Carlo dispersion around a nominal scenario. The agent selects discrete impulsive maneuvers (13 actions: no-burn and $\pm\delta v_{\text{small}}, \pm\delta v_{\text{large}}$ along RTN axes) to minimize the entropy of its belief map while minimizing fuel expenditure.

The simulator propagates the ROE deterministically given an impulsive maneuver Δv. At each time step, the servicer captures an observation o of the target with its 10° FOV, 64x64 resolution camera. The camera is assumed to always point at the center of the target and with optimal lighting conditions. We employ a ray-casting algorithm (3D DDA) to determine which voxels are visible from the current state [Amanatides and Woo, 1987]. The observation process includes sensor noise modeled within the likelihood function $P(o|\mathcal{S}, s_{phys})$. 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 projection of the camera observation of the orbiting servicer onto the probabilistic voxel grid enveloping the target is depicted in Figure 1.

The dataset is generated dynamically in a self-play loop, as explained in Section 4.2.

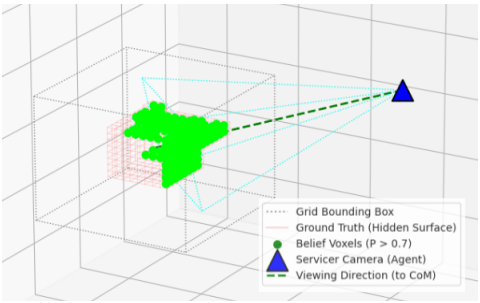


Figure 1: Probabilistic Voxel Belief Grid

4 Methods

The reward R for a trajectory of length T is defined as:

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

where InfoGain corresponds to the decrease in normalized total entropy of the probabilistic grid, and λ is a weighting factor penalizing fuel consumption (Δv).

4.1 Monte Carlo Tree Search (MCTS)

MCTS [Czech et al., 2020] builds a search tree iteratively where nodes represent states and edges represent actions (maneuvers). Each iteration consists of four phases: (1) **Selection** traverses the tree from root to a leaf by selecting actions that maximize the Upper Confidence Bound formula $\text{UCB1}_i = Q(s, a_i) + c\sqrt{\ln N(s)/N(s, a_i)}$, where $Q(s, a_i)$ is the mean reward for action a_i, $N(s)$ is the total visits to state s, $N(s, a_i)$ is visits to action a_i, and c controls the exploration-exploitation tradeoff. (2) **Expansion** adds a new child node to the tree when a leaf is reached. (3) **Simulation** performs a rollout from the new node using random action selection until a terminal condition is met, providing a reward estimate. (4) **Backpropagation** updates the Q-values and visit counts along the path from the expanded node back to the root. After a fixed iteration budget, the best root action is selected via $a^* = \arg\max_i Q(s, a_i)$.

4.2 Reinforcement Learning Framework

An AlphaZero-inspired framework [Silver et al., 2018] is adopted, where a neural network learns to approximate MCTS planning through iterative self-play. The training loop alternates between two phases: (1) **Self-Play Generation** uses MCTS guided by the current network to generate episodes, storing tuples $(s_t, \pi_{MCTS}(s_t), z_t)$ in a replay buffer, where $\pi_{MCTS} = N(s, a)/\sum_{a'} N(s, a')$ is the visit-count policy distribution and z_t is the observed discounted return, and (2) **Network Training** samples mini-batches from the replay buffer to update the network parameters.

The observable state $s = (s_{phys}, b_{shape})$ consists of the ROE and the probabilistic voxel grid. The network processes these inputs through separate branches (Figure 2): 3D convolutional layers extract spatial features from the voxel grid, while fully-connected layers encode the ROE vector. After concatenation, a shared trunk outputs $f_\theta(s) = (\mathbf{p}, V_\theta(s))$. The **policy head** produces action probabilities $\pi_\theta(a|s) = \text{softmax}(\mathbf{p})$ that guide MCTS tree search via the PUCT exploration bonus [Silver et al., 2017], replacing the uniform exploration of standard UCB1. The **value head** outputs $V_\theta(s)$, predicting expected discounted return to replace expensive random rollouts at leaf nodes when network confidence is sufficient.

The network is trained by minimizing the loss function $\mathcal{L}(\theta) = (z - V_\theta(s))^2 - \pi_{MCTS}(a|s)^\top \log \pi_\theta(a|s) + c||\theta||^2$, which combines three components: (1) **value loss** $(z - V_\theta(s))^2$ trains the value head to predict episode returns, (2) **policy loss** $-\pi_{MCTS}^\top \log \pi_\theta$ trains the policy head to match the improved MCTS policy via cross-entropy, and (3) **L2 regularization** $c||\theta||^2$ prevents overfitting. Mini-batches of size 64 are sampled from a replay buffer of 20,000 transitions, trained

using the Adam optimizer with learning rate 5×10^{-4}. As training progresses, the network's policy and value estimates improve, enabling faster and higher-quality MCTS planning in subsequent self-play iterations.

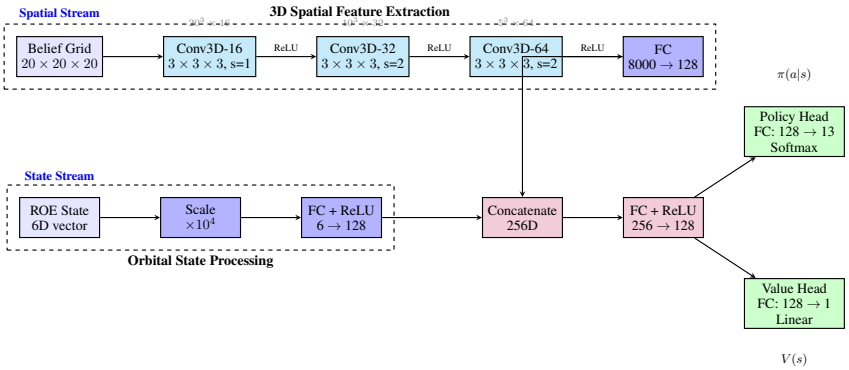


Figure 2: AlphaZero Policy-Value Network Architecture

5 Experiments and Results

5.0.1 MCTS Hyperparameter Tuning

Hyperparameter sweeps validate MCTS performance using entropy reduction $(H_{\text{initial}} - H_{\text{final}})/H_{\text{initial}} \times 100$ and total Δv (m/s) as metrics. Experiments swept horizon $h \in \{10, 15, 20\}$, exploration $c \in \{1.4, 2.0, 3.0\}$, discount $\gamma \in \{0.9, 0.95, 0.99\}$, and fuel penalty $\lambda_{\Delta v} \in \{0.01, 0.1, 0.5, 1.0, 5.0\}$ across 500 and 1000 MCTS iterations. Table 1 depicts the simulation parameters used for the top performers out of a total of 22 sweeps.

Table 1: MCTS hyperparameter sweep

c	h	γ	$\lambda_{\Delta v}$	Iters	Ent. Red. (%)	Δv (m/s)
1.4	15	0.99	1.0	1000	**96.77**	**0.31**
1.4	20	0.99	0.01	500	96.60	0.75
1.4	20	0.99	0.1	500	96.58	0.79
3.0	10	0.99	0.5	1000	96.37	1.10
3.0	10	0.99	1.0	1000	94.96	1.17

Experiments reveal a fundamental tradeoff between planning depth and exploration breadth. Deep planning ($h = 20$) achieves $\approx 96.6\%$ reconstruction using only 0.75 m/s, while shallow planning ($h = 10$) requires aggressive exploration and 1000 iterations to achieve similar performance (96.37%) but consumes 47% more fuel (1.10 m/s), demonstrating that lookahead depth is more fuel-efficient than exploration breadth. The best configuration ($c = 1.4$, $h = 15$, $\gamma = 0.99$, $\lambda = 1.0$, 1000 iterations) achieves 96.77% entropy reduction with 0.31 m/s, validating MCTS as a strong baseline for fuel-constrained active sensing.

5.1 AlphaZero MCTS Planner

5.1.1 Simulation Parameters and Neural Network Hyperparameter Tuning

The AlphaZero MCTS planner hyperparameters were tuned with sweeps similar to the top performing configuration for pure MCTS. Tables 3 and 2 exhibit a parameter configuration that balances four key considerations: physical realism, computational tractability, learning complexity, and adherence to standard hyperparameter conventions.

After each batch of self-play episodes, the network undergoes 5 training epochs to prevent underfitting while avoiding excessive optimization on a single data batch. The replay buffer maintains 20,000

transitions (approximately 6 full episodes), providing sample diversity for decorrelating batches while keeping the training distribution relatively on-policy.

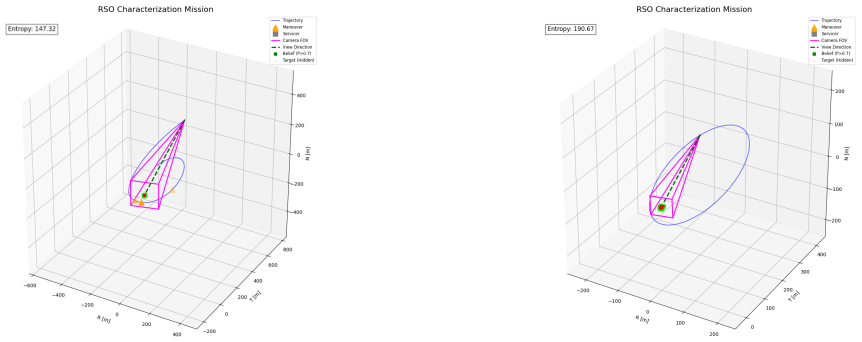
Table 2: Environment Configuration

Category	Parameter	Value
Simulation	Max Horizon	5
	Num Steps	50
	Time Step (s)	120.0
	Num Episodes	130
Orbit	μ_{Earth} (km^3/s^2)	398600.4
	Semi-major a_{chief}	7000.0
	Eccentricity e_{chief}	0.001
	Inclination i_{chief}	98.0°
	RAAN Ω_{chief}	30.0°
Initial ROE (Nominal $\pm$ Bounds) $[m]$	δa	0 ± 0
	$\delta\lambda$	200 ± 20
	δe_x	100 ± 10
	δe_y	0 ± 10
	δi_x	50 ± 10
	δi_y	0 ± 10

Table 3: Hyperparameters

Category	Parameter	Value
AlphaZero Training	Batch Size	64
	Learning Rate	5e-4
	MCTS Iterations	100
	Epochs/Cycle	5
	Replay Buffer	20000
	c_{PUCT}	1.4
	Discount γ	0.99
	Weight Decay	1e-4
	Grad Clip Norm	1.0
	Optimizer	Adam
	Scheduler	Cosine
	Min LR	1e-5
Control	$\lambda_{\Delta v}$ (penalty)	1.0

5.1.2 AlphaZero MCTS Performance

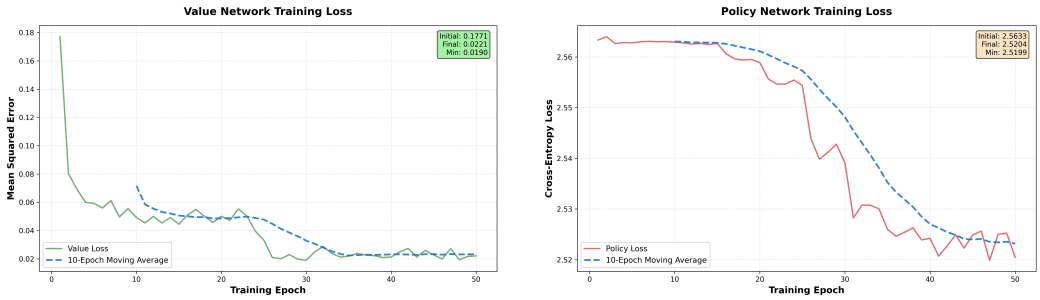


(a) AlphaZero MCTS Agent Orbit

(b) Passive Agent Orbit Baseline: No Burns

Figure 3: AlphaZero MCTS and Baseline Orbit Comparison

The trained AlphaZero MCTS agent was tested in the simulation and compared against both the pure MCTS planner and the no-burn baseline. Figure 3 depicts the servicer orbiting around the target, performing burns, and taking camera observations. The AlphaZero MCTS planner achieves superior performance over all baselines: it executes only 3 small maneuvers ($\Delta v = 0.11$ $\frac{\text{m}}{\text{s}}$) over 50 time steps, achieving 97.1% entropy reduction. In contrast, the optimized pure MCTS baseline (Table 1) requires 0.31 $\frac{\text{m}}{\text{s}}$ to achieve 96.77% reduction, while the passive agent achieves only 95.8% with no maneuvers. This demonstrates that AlphaZero achieves 0.33% better reconstruction quality with 65% less fuel compared to pure MCTS, executing 57% fewer maneuvers through learned strategic planning. Given that the Δv budget of current small passive observer satellites is on the order of 10 $\frac{\text{m}}{\text{s}}$ for basic station-keeping, the AlphaZero MCTS planner provides an exceptionally fuel-efficient trajectory. The results clearly demonstrate that the neural network successfully distills MCTS planning knowledge, enabling real-time decision-making with superior performance over classical planning approaches.



(a) Value Loss

(b) Policy Loss

Figure 4: AlphaZero MCTS Network Losses

As shown in Figure 4, the value network demonstrated strong learning with a 89% reduction in loss, while the policy network remained essentially stagnant with minimal variance early in the training and decreased from there. This rapid convergence suggests the model may have settled into a local optimum early, as the total loss improvement was driven almost entirely by better value predictions rather than refined action strategies. To address this policy stagnation, future runs will adjust hyperparameters such as the network learning rate, MCTS iterations, or the exploration constant to encourage broader state-space exploration.

6 Conclusion and Future Work

We have presented a framework for active characterization of non-cooperative RSO using reinforcement learning. By modeling the problem as a POMDP and utilizing AlphaZero MCTS over a probabilistic voxel grid, we demonstrated that an autonomous agent can plan maneuvers that significantly reduce shape uncertainty. The results validate the superiority of AlphaZero MCTS over classical planning approaches in orbital proximity operations. The AlphaZero agent achieves 97.1% entropy reduction with only $\Delta v = 0.11 \frac{\mathrm{m}}{\mathrm{s}}$, outperforming pure MCTS which requires $0.31 \frac{\mathrm{m}}{\mathrm{s}}$ for 96.77% reduction—a 65% fuel savings with better reconstruction quality. The fact that it beat both the pure MCTS baseline and passive agent demonstrates that the neural network has successfully learned to manipulate orbital geometry strategically through self-play training. This fuel efficiency improvement could be the differentiating factor that leads to a successful mission.

Future work will focus on further training of the AlphaZero MCTS planner. This involves generating a large-scale dataset of self-play episodes to train the policy and value networks, thereby replacing the expensive MCTS rollout phase at runtime. Additionally, we plan to increase the fidelity of the simulation by introducing realistic guidance, navigation, and control errors and solar lighting conditions to challenge the robustness of the vision system. Prior to industry adoption, a safety validation algorithm such as reachability analysis to check for collisions prior to burn execution would be introduced. With these refinements, the, planner will be able to make smart maneuvers, react to lighting conditions, drift, or occlusions, and be able to correct a suboptimal initial orbit, shaping AlphaZero MCTS to be a powerful tool for active characterization of non-cooperative RSO and providing capabilities that far surpass current systems on the market.

7 Contributions

Following the status update submission, the project was extended beyond the baseline MCTS implementation to include an AlphaZero-inspired reinforcement learning framework. This extension required: (1) design and implementation of a dual-head neural network architecture combining 3D convolutional layers for spatial voxel grid processing with fully-connected layers for orbital state encoding, (2) development of the self-play training infrastructure with PUCT-guided MCTS, experience replay buffers, and policy-value network training, and (3) comprehensive code optimization to reduce training time via GPU acceleration with CUDA, which improved training convergence by 2.9$\times$ per epoch compared to CPU-only training. These implementation and optimization efforts

represent more than 30 hours of development work beyond the base MCTS planner established in the status update.

Rahul Ayanampudi: Developed the relative orbital dynamics simulator. Defined the POMDP state/action spaces, probabilistic voxel grid belief, ray-casting sensor observation model, and reward function. Structured the neural network, parallelized episodes, and trained the network.

Sebastian Martinez: Configured the reinforcement learning problem formulation. Implemented the standard MCTS logic, integrated the planning algorithm with the environment, and performed hyperparameter tuning. Created the AlphaZero-inspired MCTS framework.

The Github repository can be found in https://github.com/rayanam2021/CS229_Final_Project

References

John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. *Eurographics*, 1987.

Johannes Czech, Patrick Korus, and Kristian Kersting. Monte-carlo graph search for alphazero. *CoRR*, abs/2012.11045, 2020. URL `https://arxiv.org/abs/2012.11045`.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

Justin Kruger and Simone D'Amico. Autonomous navigation of a satellite swarm using inter-satellite bearing angles. *IEEE Transactions on Aerospace and Electronic Systems*, 2025.

Justin Kruger, Tommaso Guffanti, Tae Ha Park, Mason Murray-Cooper, Samuel Low, Toby Bell, Simone D'Amico, Christopher Roscoe, and Jason Westphal. Adaptive end-to-end architecture for autonomous spacecraft navigation and control during rendezvous and proximity operations. In *AIAA SCITECH 2024 Forum*, 2024. doi: 10.2514/6.2024-0001.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

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. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Matthew Willis. Analytical theory of satellite relative motion with applications to autonomous navigation and control, 2023. URL `https://purl.stanford.edu/kq677qp2544`.