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

Stanford CS229 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`

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 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.

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.

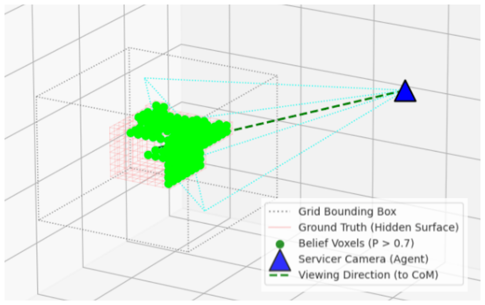


Figure 1: Probabilistic Voxel Belief Grid

The dataset is generated dynamically in a self-play loop, as explained in section 4.3.

4 Methods

Our approach uses Monte Carlo Tree Search (MCTS) as a classical planning algorithm to generate high-quality action sequences and compute target policies. Then, we adopt an AlphaZero-inspired self-play reinforcement learning framework where a neural network learns to approximate the planning results, enabling fast online decision-making.

4.1 Reward Structure

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 total entropy of the probabilistic grid, and λ is a weighting factor penalizing fuel consumption (Δv).

4.2 Monte Carlo Tree Search (MCTS)

At each decision step, MCTS builds a search tree (Figure 4) where nodes represent states and edges represent maneuvers.

UCB1 Selection and Expansion: The tree search is guided by the Upper Confidence Bound (UCB1) formula, balancing exploitation of high-value actions with exploration of rarely visited states:

$$\text{UCB1}_i = Q(s, a_i) + c\sqrt{\frac{\ln N(s)}{N(s, a_i)}} \tag{2}$$

where c is the exploration constant, $Q(s, a_i)$ is the estimated mean return for action a_i at state s, $N(s)$ is the total visits to state s, and $N(s, a_i)$ is the visits to action a_i from s.

Rollout Policy: At leaf nodes, the algorithm performs a rollout using a random rollout policy, where actions are sampled uniformly from the action space. This baseline approach provides a conservative estimate of future returns without domain-specific heuristics

Backpropagation and Best Action Selection: After a fixed budget of iterations and tree depth, the best root action is selected greedily $a^* = \arg\max_i Q(s, a_i)$. The visit count distribution $\pi_{MCTS}(a|s) = N(s, a)/\sum_j N(s, a_j)$ represents an improved policy and serves as a training target for supervised learning.

4.3 Reinforcement Learning Framework

An AlphaZero-inspired framework [Silver et al., 2018] is adopted, where a neural network learns to approximate the MCTS planning results through iterative self-play, alternating between two phases: (1) using MCTS guided by the current network to generate training data, and (2) training the network on the data to improve future planning. 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: 3D convolutions extract spatial features from the voxel grid, while fully-connected layers encode the ROE vector. After concatenation, a shared trunk outputs $f_\theta(s) = (\mathbf{p}, V_\theta(s))$, where the policy head produces $\pi_\theta(a|s) = \text{softmax}(\mathbf{p})$, action probabilities that guide tree search via the PUCT exploration bonus (Appendix 7.2.2), and the value head outputs $V_\theta(s)$, predicting expected discounted return to bootstrap leaf node evaluations when confidence is sufficient. The neural network was programmed with PyTorch.

The training procedure is done in two phases. In the planning phase, starting from an initial state, MCTS with PUCT selection generates episodes by performing tree search at each step t. The policy network guides exploration toward promising actions, while the value network replaces expensive rollouts at leaf nodes when its confidence exceeds a threshold. After search completes at each step, we store the tuple $(s_t, \pi_{MCTS}(s_t), z_t)$ where $\pi_{MCTS} = N(s, a)/\sum_{a'} N(s, a')$ is the visit-count distribution and z_t is the observed discounted return. Multiple episodes populate a replay buffer.

In the training phase, the network is updated by sampling mini-batches and minimizing $\mathcal{L}(\theta) = (z - V_\theta(s))^2 - \pi_{MCTS}(a|s)^\top \log \pi_\theta(a|s) + ||\theta||^2$, where the value loss trains the value head to predict returns, the policy loss trains the policy head to match MCTS policy via cross-entropy, and L2 regularization prevents overfitting. As the network improves, the expensive rollout simulations at leaf nodes are gradually replaced with fast network predictions $V_\theta(s_{leaf})$, which significantly accelerates MCTS planning in the self-play loop.

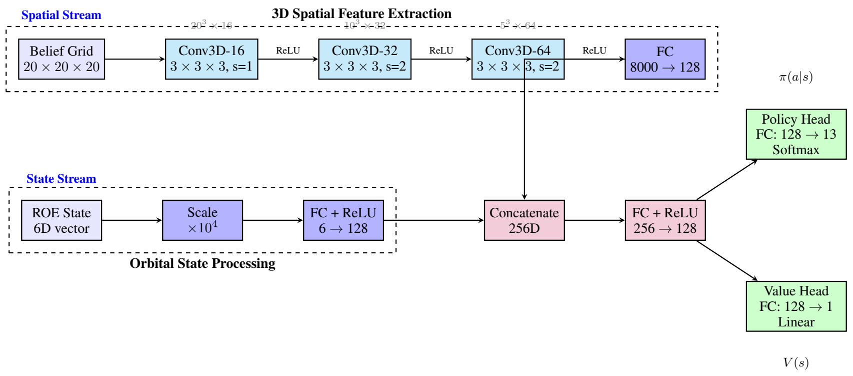

Spatial Stream
3D Spatial Feature Extraction
$20^3 \times 16$
$10^3 \times 32$
$5^3 \times 64$
Belief Grid $20 \times 20 \times 20$
Conv3D-16 $3 \times 3 \times 3$, s=1
ReLU
Conv3D-32 $3 \times 3 \times 3$, s=2
ReLU
Conv3D-64 $3 \times 3 \times 3$, s=2
ReLU
FC $8000 \rightarrow 128$
$\pi(a|s)$
Policy Head FC: $128 \rightarrow 13$ Softmax
State Stream
ROE State 6D vector
Scale $\times 10^4$
FC + ReLU $6 \rightarrow 128$
Concatenate 256D
FC + ReLU $256 \rightarrow 128$
Orbital State Processing
Value Head FC: $128 \rightarrow 1$ Linear
$V(s)$

Figure 2: AlphaZero Policy-Value Network Architecture

5 Experiments and Results

5.0.1 MCTS Hyperparameter Tuning

Hyperparameter sweeps were conducted to validate MCTS performance. The primary metrics are: (1) **Entropy Reduction (%)**, measuring reconstruction completeness as $(H_{\text{initial}} - H_{\text{final}})/H_{\text{initial}} \times 100\%$, and (2) **Total Δv (m/s)**, the cumulative fuel expenditure across the episode.

Experiment 1: Baseline Sweep (500 Iterations) Sweeps done over horizon $h \in \{10, 20\}$, exploration constant $c \in \{1.4, 2.0\}$, discount factor $\gamma \in \{0.9, 0.95, 0.99\}$, and fuel penalty $\lambda_{\Delta v} \in \{0.01, 0.1\}$, as shown in Table **??**

Table 1: Experiment 1: Top 4 configurations by entropy reduction

c	h	γ	$\lambda_{\Delta v}$	Entropy Red. (%)	Total Δv (m/s)
1.4	20	0.99	0.010	96.60	0.75
1.4	20	0.99	0.100	96.58	0.79
1.4	10	0.95	0.010	73.52	0.71
1.4	10	0.95	0.100	73.52	0.71

Deeper planning ($h = 20$, $\gamma = 0.99$) achieves $\approx 97\%$ reconstruction with 0.75-0.79 m/s, while shallow planning ($h = 10$, $\gamma = 0.95$) achieves only 73.5% with 0.71 m/s using the same exploration constant ($c = 1.4$).

Experiment 2: Optimized Sweep (1000 Iterations) To test whether increased exploration can compensate for reduced planning depth, we fixed $h = 10$ and $\gamma = 0.99$ (high discount but shallow horizon), varying $c \in \{1.4, 2.0, 3.0\}$ and $\lambda_{\Delta v} \in \{0.5, 1.0, 5.0\}$ with 1000 MCTS iterations.

Table 2: Experiment 2: Configurations ranked by entropy reduction

c	$\lambda_{\Delta v}$	Entropy Red. (%)	Total Δv (m/s)
3.0	0.5	96.37	1.10
3.0	1.0	94.96	1.17
1.4	1.0	94.60	1.36
2.0	1.0	93.39	1.40

Aggressive exploration ($c = 3.0$) with increased computational budget recovers near-optimal performance (96.37%) despite shallow planning, demonstrating that exploration breadth can compensate for reduced lookahead depth. However, this requires 47% more fuel (1.10 vs 0.75 m/s) than Experiment 1's deep planning approach.

Key Insights Experiments reveal a fundamental tradeoff between planning depth and exploration breadth. Deep planning ($h = 20$) achieves $\approx 97\%$ 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 ($c = 1.4$, $h = 20$, $\gamma = 0.99$, $\lambda = 0.01$) achieves 96.60% entropy reduction with 0.75 m/s.

5.1 AlphaZero MCTS Planner

5.1.1 Simulation Parameters and Neural Network Hyperparameter Tuning

As defined in Appendix 5.1.1, the simulation parameters are identical those used in the previous MCTS experiments to enable direct comparison. Similar hyperparamter tuning sweeps were performed for the AlphaZero MCTS planner. The optimal hyperparameters are shown in Table 3.

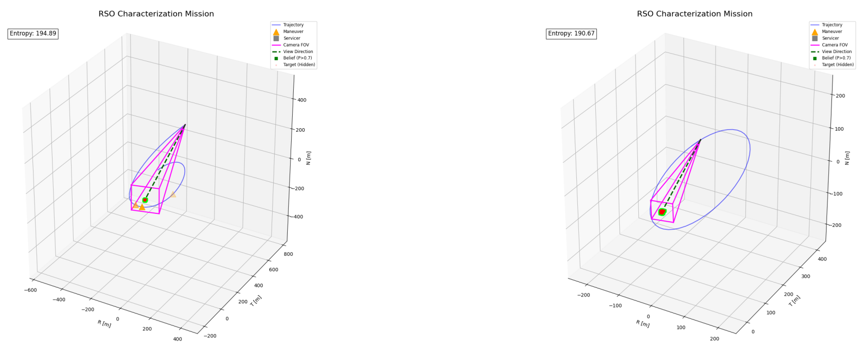
The overall parameter configuration balances four key considerations: physical realism (matching operational satellite missions in LEO with realistic sensor characteristics), computational tractability

Table 3: Training and Control Hyperparameters

Category	Parameter	Value
AlphaZero Training	Batch Size	64
	Learning Rate	5×10^{-4}
	MCTS Iterations	100
	Epochs per Cycle	5
	Replay Buffer Size	20000
	c_{PUCT}	1.4
	Discount Factor γ	0.99
	Weight Decay	1×10^{-4}
	Gradient Clip Norm	1.0
	Optimizer	Adam
	LR Scheduler	Cosine Annealing
	Min Learning Rate	1×10^{-5}
Control	$\lambda_{\Delta v}$ (penalty)	1.0

(enabling training completion within available resources), learning complexity (creating a non-trivial problem through partial observability, long horizons, and sparse rewards), and adherence to established practices (following hyperparameter conventions from the AlphaZero literature).

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

After several iterations of the batched episode self-play loop with the optimal hyperparameters for the MCTS and neural network exhibited acceptable non-increasing loss as shown in Appendix 7.2.4, the trained AlphaZero MCTS agent was tested in the simulation and compared against the standard MCTS planner and the no-burn baseline. Figure 3 depicts the servicer orbiting around the target space, performing burns, and taking camera observations. As exhibited, the AlphaZero MCTS planner successfully outperforms the passive agent by taking only 3 small maneuvers ($\Delta v = 0.11\ \frac{\text{m}}{\text{s}}$) out of the 50 time steps, decreasing the entropy by approximately 1.1% of the agent's initial entropy. The entropy progression of the AlphaZero MCTS episode can be found in Appendix 7.2.3. For reference, the standard MCTS implementation yielded an entropy reduction improvement of approximately 0.23% with $\Delta v = 0.24\ \frac{\text{m}}{\text{s}}$). 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, both the standard MCTS and AlphaZero MCTS planners provide a reasonable trajectory that boosts performance. The experiment results clearly demonstrate that the active characterization enabled by the AlphaZero MCTS planner represents a significant advancement over traditional passive observers.

6 Conclusion and Future Work

We have presented a framework for active characterization of around 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 use of AlphaZero MCTS in orbital proximity operations as it successfully outperformed the passive agent with minimal $\Delta v = 0.11\ \frac{\text{m}}{\text{s}}$ by decreasing the entropy by approximately 1.1% of the agent's initial entropy. The fact that it beat the baseline proves it has learned to manipulate the orbit geometry to its advantage. This improvement could be the differentiating factor that leads to a successful mission.

Future work will focus on further training of the AlphaZero MCTS. 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 bad initial orbit, shaping AlphaZero MCTS to be a powerful tool for active characterization of non-cooperative RSO providing capabilities that far surpasses current systems on the market.

7 Appendices

7.1 Simulation Parameters

Table 4: Environment and Model Configuration Parameters

Category	Parameter	Value
Simulation	Max Horizon	5
	Num Steps per Episode	50
	Time Step (s)	120.0
Orbit	μ_{Earth} (km^3/s^2)	398600.4418
	Semi-major Axis a_{chief} (km)	7000.0
	Eccentricity e_{chief}	0.001
	Inclination i_{chief} (deg)	98.0
	RAAN Ω_{chief} (deg)	30.0
Camera	Field of View (deg)	10.0
	Sensor Resolution (pixels)	64×64
	$P(\text{hit}\mid\text{occupied})$	0.95
	$P(\text{hit}\mid\text{empty})$	0.001
Initial ROE (meters)	δa	0.0
	$\delta\lambda$	200.0
	δe_x	100.0
	δe_y	0.0
	δi_x	50.0
	δi_y	0.0
Monte Carlo Perturbations (meters)	Num Episodes	65
	δa bounds	± 0.0
	$\delta\lambda$ bounds	± 20.0
	δe_x bounds	± 10.0
	δe_y bounds	± 10.0
	δi_x bounds	± 10.0
	δi_y bounds	± 10.0
Network	Hidden Dimension	128
	Num Actions	13
	Grid Dimensions	$20 \times 20 \times 20$

Parameter Justification Summary

- **Simulation**: 50-step episodes with $\Delta t = 120\,\mathrm{s}$ span approximately one orbit. MCTS horizon of 5 balances lookahead with computational cost.
- **Orbit**: $7000\,\mathrm{km}$ LEO sun-synchronous orbit ($i = 98°$, $e = 0.001$) represents typical Earth observation missions (consistent lighting) with near-circular trajectories.
- **Camera**: $10°$ FOV and 64×64 resolution match embedded satellite constraints; asymmetric noise model ($P(\text{hit}|\text{occ}) = 0.95$, $P(\text{hit}|\text{emp}) = 0.001$) enforces partial observability.
- **Initial ROE**: Nominal $\delta\lambda = 200\,\mathrm{m}$, $\delta e_x = 100\,\mathrm{m}$, $\delta i_x = 50\,\mathrm{m}$ with $\pm$10–20 m perturbations across 65 episodes tests generalization.
- **Monte Carlo Perturbations**: The expensive computations limited the number of episodes that could be run. However, the AlphaZero MCTS approach has high sample efficiency and accuracy since the network provides a high-quality evaluation based on pattern recognition from training. It essentially gives the MCTS "intuition," making the value estimates (Q-values) much closer to the ground truth (optimal play). Hence, the AlphaZero MCTS planner needs fewer episodes to yield a good policy. The perturbation magnitudes were chosen based on domain knowledge.
- **Network**: 128 hidden units and 20^3 voxel grid balance expressiveness with computational tractability. 13 discrete actions enable MCTS search.
- **Control**: $\lambda_{\Delta v} = 1.0$ provides equal weighting of entropy reduction and fuel cost.

The training process employs a batch size of 64 samples, providing stable gradient estimates while fitting within typical memory constraints. The Adam optimizer uses an initial learning rate of 5×10^{-4}, a conservative value that prevents catastrophic forgetting in the continual learning setting while allowing effective policy updates. This learning rate is gradually reduced via cosine annealing to a minimum of 1×10^{-5} over the training duration, enabling fine-tuning without completely halting adaptation to new data. Weight decay regularization ($\lambda = 1 \times 10^{-4}$) and gradient clipping (max norm = 1.0) provide additional stability and prevent overfitting to specific trajectories.

Each MCTS search performs 100 iterations, balancing search quality with computational cost (approximately 1–2 seconds per search on modern CPUs) and exploring roughly 10 actions to depth 2–3. The exploration constant $c_{\text{PUCT}} = 1.4$ follows the standard AlphaZero formulation, balancing exploitation of promising moves with exploration of unvisited nodes. 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.

The discount factor $\gamma = 0.99$ appropriately weights long-term rewards for the 50-step episodes ($\gamma^{50} \approx 0.605$), ensuring that terminal rewards significantly influence early-episode decisions while preventing the myopic behavior that would result from lower discount factors. The fuel cost penalty $\lambda_{\Delta v} = 1.0$ provides equal weighting between information gain (entropy reduction) and fuel consumption in the reward function, establishing a neutral baseline for subsequent tuning studies.

The configuration strikes a balance between physical realism (orbital mechanics, sensor models), computational constraints (training time, memory), and learning theory (exploration, generalization, stability).

7.2 MCTS

7.2.1 MCTS Tree Example

7.2.2 PUCT Selection

An enhancement from AlphaZero is the use of the Polynomial Upper Confidence Trees (PUCT) formula for action selection during tree search, which directly incorporates the neural network's policy prior. During tree traversal, actions are selected according to:

$$a^* = \arg\max_a \left[Q(s,a) + c \cdot \pi_\theta(a|s) \cdot \frac{\sqrt{N(s)}}{1 + N(s,a)} \right] \tag{3}$$

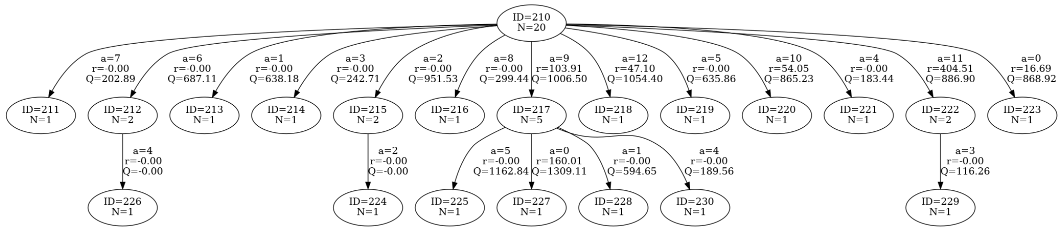


Figure 4: MCTS Tree of Depth 2

where $\pi_\theta(a|s)$ is the policy network's predicted probability for action a. This modification allows the network to guide exploration toward promising actions earlier in the search, improving sample efficiency. When the network is untrained or not available, the system falls back to standard UCB1.

7.2.3 Entropy Progression

Each episode of the simulation has a non-increasing entropy progression plot. However, some runs have more entropy reduction than others. An example of an AlphaZero MCTS episode entopy progression is shown in Figure 5.

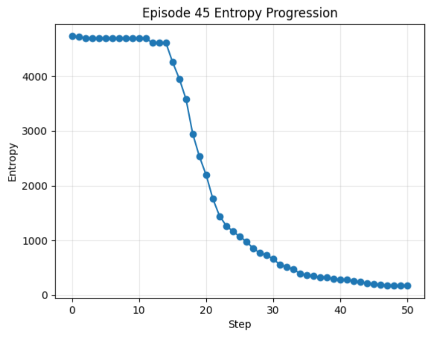


Figure 5: AlphaZero MCTS Episode Entropy Progression Example

7.2.4 Policy and Value Network Loss

Although there is no trivial baseline to compare the network loss to, the policy and value loss decrease and plateau below the threshold. Initially, when the network knows nothing, it assigns equal probability $(1/13)$ to every action, making the cross entropy loss $-ln(\frac{1}{13}) = 2.5649$. As shown in Figure **??**, the policy loss decreases from the random policy loss and the value loss decreases, as expected.

7.3 Hardware

The parallelized simulation episodes with 13 workers and network training was performed on an Intel Core i7 processor. The training time for a single batch of 13 episodes with the parameters listed in Tables 3 and 4 was approximately 6 hours. The largest bottleneck is the expensive ray tracing with 64x64 sensor resolution. Adjusting the maximum iterations, horizon, number of simulation time steps, and number of episodes also directly affect runtime.

8 Contributions

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.

References

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

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`.