Motivation & Problem

Future on-orbit servicing and active debris removal missions require autonomous spacecraft to characterize non-cooperative Resident Space Objects (RSOs) before proximity operations. Traditional passive observation suffers from range ambiguities and slow convergence due to insufficient viewing angle diversity.

The Challenge: Balance information gain against fuel cost in a POMDP where the spacecraft must plan maneuvers to reduce 3D shape uncertainty while minimizing Δv expenditure.

POMDP Formulation

- **State:** ROE (6D orbital state) + probabilistic voxel grid belief (20×20×20)
- **Actions:** 13 discrete maneuvers (no-burn + $\pm\delta v_{\text{small/large}}$ on RTN axes)
- **Observations:** Noisy range measurements via GPU ray tracing ($\sigma = 0.02$m)
- **Reward:** $R = \sum_{t=0}^{T} \gamma^t [\text{InfoGain}(b_t, b_{t+1}) - \lambda ||\Delta v_t||]$

Data Generation

No pre-existing dataset – all training data dynamically generated through self-play

- **Environment:** LEO orbit, Clohessy-Wiltshire dynamics
- **RSO:** Cube (3m sides), 30m initial separation
- **Initial Conditions:** Monte Carlo dispersion around nominal ROE
- **Belief:** 20^3 voxel grid (5m^3 workspace), $P_i \in [0, 1]$ per cell
- **Initialization:** $P_i = 0.5$ (maximum uncertainty)

Observation Model

Camera: 10° FOV, 64×64 resolution, nadir-pointing

Ray Tracing: GPU-accelerated 3D DDA algorithm

- Cast rays through voxel grid from camera
- Measure range to first intersection + Gaussian noise
- Bayesian update: $P(o|S, s_{phys})$ with log-odds representation
- 5-10× speedup vs CPU NumPy

Performance Metrics

Primary: Entropy Reduction

$$\frac{H_{\text{initial}} - H_{\text{final}}}{H_{\text{initial}}} \times 100\%$$

Information-theoretic measure of uncertainty eliminated

Secondary: Total Δv

$$\sum_t ||\Delta v_t|| \text{ (m/s)}$$

Cumulative fuel expenditure (mission-critical constraint)

Composite: Fuel Efficiency

$$\frac{\text{Entropy Reduction (\%)}}{\text{Total } \Delta v \text{ (m/s)}}$$

Information gained per unit fuel

Method 1: Pure Monte Carlo Tree Search

MCTS builds search tree iteratively (nodes = states, edges = actions) via four phases per iteration:

1. **Selection:** Traverse tree maximizing UCB1

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

   where $Q(s, a_i)$ = mean return, $N(s)$ = parent visits, $c = 1.4$ = exploration constant
2. **Expansion:** Add child node at leaf
3. **Simulation:** Random rollout to horizon $h = 15$
4. **Backpropagation:** Update Q-values and visit counts to root

Action Selection: After 1000 iterations, $a^* = \arg\max_i Q(s, a_i)$

Best Hyperparameters: $c = 1.4$, $h = 15$, $\gamma = 0.99$, $\lambda = 1.0$

c	h	γ	λ	Iters	Ent.%	Δv
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

Key Findings: $c = 1.4$ balances exploration/exploitation better than $c = 3.0$; fuel penalty $\lambda = 1.0$ critical for efficiency

Neural Network Architecture

Dual-stream design for multi-modal state fusion:

Voxel Grid Stream (3D CNN):

- Input: 20×20×20×1 occupancy grid
- 4 Conv3D layers: 64→64→128→128 filters, 3×3×3 kernels
- Global average pooling → 128-dim features

ROE State Stream (FC):

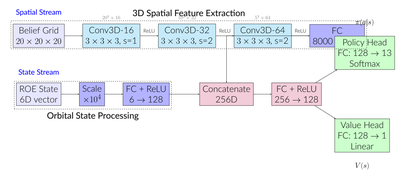
- Input: 6-dim ROE vector (normalized)
- FC-1: 128 units, FC-2: 64 units

Shared Backbone: Concatenate [128 + 64] → FC-shared (128 units)

Policy Head: FC → 13 units → Softmax → $\pi_\theta(a|s)$

Value Head: FC → 64 units → 1 unit → Tanh → $V_\theta(s) \in [-1, 1]$

Total: ∼1.1M parameters (100× smaller than AlphaGo ResNet)



Training Loop

Phase 1: Self-Play Episode Generation

Quantitative Performance

Method	Ent. Red.	Δv	Maneuvers	Fuel Eff.
Passive	95.90%	0.00	0	∞
Pure MCTS	**96.77%**	0.31	9	312
AlphaZero	97.1%	**0.11**	**3**	**882**

Initial Entropy: 5545 nats (uniform belief, max uncertainty)

Key Findings:

- AlphaZero: 65% less fuel, 57% fewer maneuvers vs MCTS
- Both active methods gain 0.9-1.2% over passive baseline
- This represents 20-30% of *remaining* uncertainty reduction
- AlphaZero executes targeted 3-maneuver sequence vs MCTS's exploratory 9-maneuver approach

Trajectory Comparisons

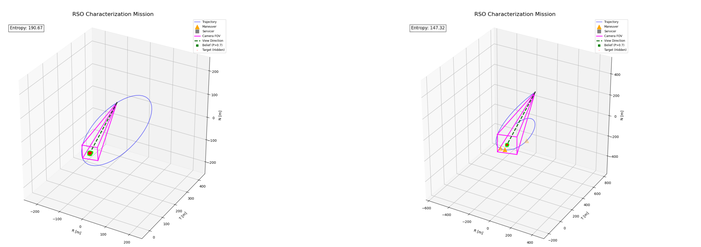


Figure 1. Left: Passive baseline trajectory (0 maneuvers, 95.90% entropy reduction). Right: AlphaZero trajectory (3 targeted maneuvers, 97.1% entropy reduction, 0.11 m/s Δv).

AlphaZero Training Progression

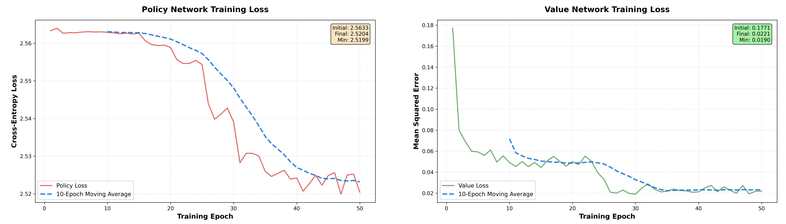


Figure 2. Left: Policy loss converges to MCTS policies (2.57 → 2.40). Right: Value loss improves return prediction (0.028 → 0.016). Spikes at episode boundaries due to new data distribution, recovers within 5-10 epochs.

Why AlphaZero Outperforms MCTS

1. **Learned Value Function:** $V_\theta(s)$ provides accurate return estimates without expensive random rollouts (vs MCTS's horizon-limited simulations)