Journal of Computer Science

Research Article

Multi-Satellite Custody of Moving Ground Targets

David Schuster

Geophysical Institute, University of Alaska Fairbanks, Fairbanks, USA

Article history
Received: 22-09-2025
Revised: 12-02-2026
Accepted: 23-02-2026

Email: dschuster@alaska.edu

Abstract: Maintaining persistent custody of ground targets from Low Earth Orbit (LEO) is a key challenge in space-based surveillance and reconnaissance. Traditional rule-based tasking strategies struggle with scalability as the number of assets and targets grows. In this paper, we present a centralized Reinforcement Learning (RL) approach using Proximal Policy Optimization (PPO) to coordinate four LEO satellites tasked with tracking two moving ground targets. PPO was selected for its balance of sample efficiency and stability in high-dimensional spaces, making it suitable for orbital simulations. The satellites operate in fixed orbits without maneuvering, relying on tipping and cueing strategies to maintain custody. Our implementation integrates the Skyfield orbital mechanics library with stable-baselines3 to simulate the environment and train a policy. Results demonstrate that the PPO agent significantly outperforms a heuristic baseline by achieving ~ 25−30% higher average custody duration (mean 74.5% coverage over 5000 steps vs. 41.4 and 54.9% coverage for random and greedy heuristics respectively) and 30% fewer gaps, with $p < 0.01$ from t-tests confirming the robustness of improvements. This work highlights the potential of centralized RL for scalable custody management in multi-satellite constellations.

Keywords: Reinforcement Learning, ISR, Space Applications, PPO, Automation

Introduction

Overhead custody maintenance in remote sensing refers to the ability of a satellite constellation to maintain continuous observation or tracking of objects on the Earth's surface. This problem is increasingly relevant to applications such as persistent Intelligence, Surveillance, and Reconnaissance (ISR), disaster response, and environmental monitoring. Traditional approaches to custody maintenance often rely on pre-planned orbits and deterministic scheduling. However, the unpredictable motion of ground targets and the dynamic nature of satellite availability demand adaptive and intelligent strategies. Reinforcement Learning (RL) offers a promising approach to this challenge by enabling agents to learn policies that maximize long-term performance under uncertainty. Recent advances in deep RL, particularly centralized training approaches, have demonstrated the ability to coordinate multiple agents in complex environments. In this work, we extend these methods to the context of multi-satellite custody maintenance.

This paper addresses a key gap in the literature: While coverage optimization and RL have been studied separately, persistent custody with fixed-orbit satellites over dynamic targets lacks scalable, adaptive solutions. Our centralized Proximal Policy Optimization (PPO) approach provides a proof-of-concept by focusing on a constellation of four low Earth Orbit (LEO) satellites tasked with maintaining custody over two dynamic ground targets. Satellites do not perform orbital maneuvers; instead, they adjust their sensor pointing and handoff strategies to ensure continuous coverage. The centralized PPO agent learns from simulated episodes in which both target and satellite positions are randomized at the start using Two-Line Element (TLE) sets. The agent's objective is to maximize custody time while minimizing redundant observations. This paper makes the following contributions:

- We formulate the custody maintenance problem as a reinforcement learning environment with a centralized agent coordinating multiple satellites, bridging the gap between traditional scheduling and intelligent coordination
- We implement and train a PPO agent to coordinate four satellites that track two dynamic targets

- We evaluate the learned policies under randomized orbital configurations and present quantitative results on custody performance
- We release our simulation code (on request), built on top of Skyfield and Gymnasium, to support future research

Background and Related Work

Custody Maintenance in Remote Sensing

Maintaining continuous custody over ground objects has long been a challenge in surveillance and reconnaissance missions. Traditional approaches rely on fixed satellite constellations and deterministic scheduling algorithms such as Walker constellations and coverage optimization (Ballard, 1980; Walker, 1977). However, such methods lack adaptability when the observed objects exhibit dynamic and unpredictable behavior. In addition, increasing congestion in Low Earth Orbit (LEO) introduces further uncertainty in coverage availability.

Reinforcement Learning

Reinforcement Learning (RL) provides a framework for agents to learn policies that maximize long-term rewards through interaction with an environment (Sutton and Barto, 2018). Multi-agent reinforcement learning (MARL) extends this paradigm to systems with multiple coordinated or competing entities (Busoniu et al., 2008). Recent advances in centralized RL have proven effective in domains such as multi-robot coordination and distributed sensor networks (Lowe et al., 2017; Oliehoek and Amato, 2016).

In particular, PPO (Schulman et al., 2017) has emerged as a robust policy-gradient method balancing sample efficiency and stability. PPO's clipping mechanism prevents destructive policy updates, making it suitable for training in high-dimensional continuous state spaces. Prior works have applied PPO to multi-satellite coordination for Earth observation (Liu, 2020) and space situational awareness (Roberts et al., 2021), though applications to custody maintenance remain limited.

Space-Based Reinforcement Learning Applications

Several studies have highlighted the promise of reinforcement learning in space operations. For instance, RL has been used for autonomous maneuver planning in satellite formation flying (Bevilacqua, 2009), orbital rendezvous and docking (Gaudet et al., 2020), and adaptive sensor scheduling (Pan et al., 2021). Custody maintenance represents a unique sub-problem where orbital dynamics are fixed, and learning must occur through sensor allocation and handoff strategies. This makes the problem well-suited to a centralized PPO agent managing a multi-satellite constellation.

Simulation Tools for Orbital Dynamics

Accurate orbital simulation is a prerequisite for RL training. Tools such as Skyfield (Rhodes, 2019), GMAT (Hughes, 2016), and Orekit (Labs, 2014) provide open-source frameworks for propagating satellite orbits using two-line elements (TLEs). By integrating orbital mechanics libraries with RL environments such as OpenAI Gymnasium, researchers can create training pipelines that combine physical fidelity with algorithmic flexibility

Gaps in the Literature

While coverage optimization and RL-based constellation management have been studied independently, few works address the specific problem of persistent custody with fixed orbital assets. Moreover, the use of centralized PPO agents coordinating multiple non-maneuvering satellites for custody maintenance over dynamic ground targets has not, to our knowledge, been previously demonstrated. This work aims to bridge that gap by presenting a novel simulation environment, RL formulation, and evaluation framework.

Methods

Problem Formulation

We model the custody maintenance task as a centralized reinforcement learning problem where a single agent coordinates multiple satellites. A set of N = 4 satellites in Low Earth Orbit (LEO) must maintain continuous observational custody over M = 2 ground targets that move stochastically. The satellites cannot maneuver their orbits; instead, their decisions involve which target to observe at each time step. The objective is to maximize persistent custody while minimizing redundant coverage. Although framed in a multi-satellite 2 David Schuster/ Journal of Computer Science 2026, 14 (3): Page Numbers DOI: Context, the problem uses a single centralized agent for both training and execution, simplifying coordination but limiting to scenarios with full observability. Future work could extend to true MARL with decentralized execution. Formally, we define the environment as a Markov Decision Process (MDP) with the tuple (S, A, R, P, γ), where:

- S: The state space, including orbital positions (from TLEs), line-of-sight availability, and current custody assignments
- A: The action space, where each satellite selects a target to observe, or None if no target is in view
- R: The reward function, described in Subsection 3.4
- P: The transition function, propagated via Keplerian dynamics using skyfield
- γ: The discount factor, set to 0.99

Environment Implementation

We developed a custom environment in Python using the OpenAI Gymnasium API. Orbital propagation is handled by the skyfield library, which allows randomized initialization of TLEs at each episode. Ground targets move according to random walk models over the Earth's surface.

A simplified initialization snippet is shown below:

```
import gymnasium as gym
    from gymnasium import spaces
    import numpy as np
    import pandas as pd
    from skyfield.api import load, EarthSatellite, wgs84



            class SatelliteCustodyEnv(gym.Env):
            def __init__(self, max_steps=100000):
            super(SatelliteCustodyEnv, self).__init__()
            self.ts = load.timescale()
            self.step_duration_sec = 60
            self.max_steps = max_steps
            self.satellites =
        self._load_satellites()
            self.num_sats = len(self.satellites)
            self.num_targets = 2
            self.action_space =
        spaces.MultiDiscrete([3] * self.num_sats)
            obs_dim = self.num_sats *
        self.num_targets + 2 * self.num_targets + self.num_sats
            self.observation_space =
        spaces.Box(low=-180.0, high=180.0, shape=(obs_dim,),
        dtype=np.float32)
            self.visibility_matrix = np.zeros((self.num_sats,
            self.num_targets), dtype=int)
            self.visibility_log = []
            self.custody_log = []
            self.cue_log = []
            self.sat_positions = []
            self.target_positions = []
            self.episode_reward = 0.0
            self.episode_rewards = []
            self.custody_tracker =
        np.zeros(self.num_targets, dtype=int)
        self.steps = 0 self.reset()
                    def _load_satellites(self):
                    tle_lines = []
                    for i in range(4):
                    sat_id = 10001 + i
                    name = f"SAT-{i+1}"
                    inclination = 30.0 + random.uniform(-5, 5)
                    raan = random.uniform(0, 360)
                    tle1 = f"1 {sat_id}U 2400{i+1}A
        20250.00000000 .00000000 00000-0
        00000-0 0 {sat_id%10000:04d}"
                    tle2 = f"2 {sat_id}
        {inclination:.4f} {raan:.4f} 0001000 0.0
        0.0 14.5 {sat_id%10000:05d}"
                    tle_lines.append((name, tle1, tle2))
return [EarthSatellite(tle[1], tle[2], tle[0], self.ts) for tle in tle_lines]
```

Centralized PPO Agent

We use a centralized Proximal Policy Optimization (PPO) agent that observes the joint state of all satellites and outputs joint actions. This design simplifies coordination compared to decentralized training.

State features include:

- Satellite positions in ECI coordinates
- Target positions in geodetic coordinates
- Binary visibility matrix V_{ij} for satellite i observing target j
- Current custody assignment history

The action space is a discrete vector of length N, where each entry represents the chosen target index.

Additional model parameters include:

Random motion of the targets are at most |0.05 | latitude and 0.07◦ longitude

Inclination is uniformly random [25, 35] and right ascension is uniformly random [0, 360]

Satellite sensor footprint is computed by simple line of sight $R_f = \sqrt{(R_E + h)^2 - R_E^2}$ where R_E is the Earth's radius, h is satellite altitude and R_f is the radius of the sensor footprint.

Admittedly, these parameters are not very realistic and there is much future work to be done to make this model applicable to the real world. However, we believe that these parameters are reasonable for a proof-of-concept.

A minimal agent definition:

```
import torch
    from stable_baselines3 import PPO


env = SatelliteCustodyEnv(max_steps=5000) 5
            check_env(env)


model = PPO("MlpPolicy", env, verbose=1, device=device)
model.learn(total_timesteps=100000)
```

Reward Function

The reward at time step t is designed to promote continuous custody of both targets while discouraging unnecessary cueing and delayed handoffs. Formally, for $M = 2$ targets and $N = 4$ satellites, the reward is:

$$r_t = \sum_{m=1}^{M} \underbrace{\mathbb{I}[\text{custody}(m,t)]}_{\text{coverage reward}} - \alpha \underbrace{\mathbb{I}[\text{gap}(m,t)]}_{\text{interruption penalty}} - \beta \underbrace{\sum_{i=1}^{N} \mathbb{I}[a_i^t = \text{cue}(m)]}_{\text{cue cost}} - \eta \underbrace{\text{handoff\_latency}(m,t)}_{\text{handoff penalty}}, \quad (1)$$

Where:

- I[·] is the indicator function
- custody(*m,t*) is true if target m is under observation at time t
- gap(*m,t*) is true if custody of target m is lost at t
- a t i is the action of satellite *i* at time *t*
- handoff latency(*m,t*) is the number of steps between custody loss and reacquisition for target *m*

This formulation rewards continuous custody, penalizes interruptions, discourages excessive inefficient sensor tasking, and promotes rapid handoffs between satellites. By tuning the weights, the balance between continuity, efficiency, and responsiveness can be controlled.

Experiments

Training Setup

We trained the centralized PPO agent using the stable-baselines3 (Raffin et al., 2021) framework with the following hyperparameters:

- Learning rate: 3×10−4
- Optimizer: Adam
- Discount factor (γ): 0.99
- PPO clipping parameter (ε): 0.2
- Rollout length: 2048 steps
- Minibatch size: 64
- Hidden layer size: [64, 64]
- Activation function: tanh

Training was carried out for 5×106 environment time steps in 20 random seeds. Each episode simulated ~ 90 minutes of orbital propagation (one full orbit). For comparison, we examined model performance against two other options: random and greedy tasking strategies. The random strategy will choose an action at random each timestep and should retain custody at rates no better than chance; given our sensor footprints in this scenario are ~ 40%. The greedy heuristic baseline assigns each satellite to the nearest visible target (by angular distance) if available, prioritizing targets without current custody. If multiple satellites can observe the same target, only one is assigned to avoid redundancy. eters are reasonable for a proof-of-concept. A minimal agent definition demonstrating the greedy custody strategy:

```
def greedy_policy(self, obs):
                    action = np.zeros(self.n_sats, dtype=int)
                    for t in range(self.num_targets):
                    visible_sats =
        np.where(self.visibility_matrix[:, t] ==
        1)[0]
                    if len(visible_sats) > 0:
                    chosen_sat = visible_sats[0]
                    action[chosen_sat] = t + 1
return action
```

The visibility matrix has value 0 when idle, 1 when tracking the target and 2 when cueing another satellite to begin tracking. Thus, a satellite with a target in custody will always prioritize keeping custody of that target as long as possible without considering the other target or other satellites. Custody from a greedy strategy should be superior to that of a random strategy, but will be unpredictable and will result in suboptimal tip-cue handoffs.

Evaluation Metrics

We evaluated the trained policies using two key metrics:

1. Reward Convergence: episodic reward curves during training
2. Custody Coverage: fraction of targets under observation at each time step

Results and Discussion

The PPO agent demonstrated stable convergence, achieving near-continuous coverage of both targets while keeping redundancy low. The performance in the seeds was consistent, with a standard deviation below 5% in the coverage of the custody.

Figure 1 shows the learning curve of PPO, averaged over seeds. Figure 2 compares custody coverage before and after training.

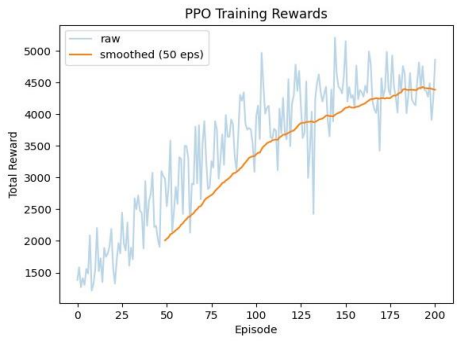

Fig. 1: Training performance of PPO: episodic reward in 5000 timestep episodes over 1000000 total timesteps both raw and smoothed over 50 episode increments

Results

Figure 2 shows that the PPO-trained policy significantly outperforms both the random and greedy baseline in terms of sustained custody. Quantitatively, the PPO strategy achieved a mean custody coverage of 74.3% ±.005% for target 1 and 74.7% ± 0.003 for target 2. The random baseline achieved 41.3% ± 0.01% and 41.5% ± 0.01% for targets 1 and 2, respectively, while the greedy strategy achieved 68.2% ± 0.009% and 41.4% ± 0.008% for same.

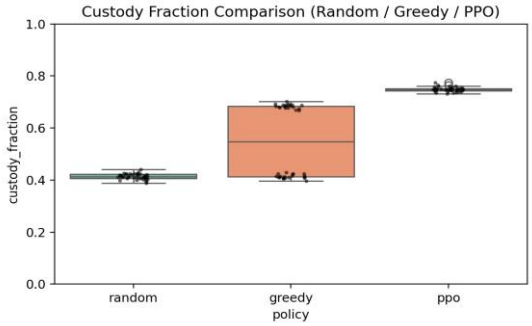

Fig. 2: Custody coverage comparison: PPO-trained policy vs. random baseline. PPO achieves near-continuous dual-target custody

Interestingly, the custody of the two targets was fairly consistent for PPO and random strategies with a large discrepancy from the greedy strategy. This may be reflective of sensors initially having custody of target 1 through happenstance and then prioritizing keeping custody of it over all other options. Applying Welch's two-sample t-test to random vs. PPO and greedy vs. PPO confirms that the difference in custody coverage is highly significant ($p \ll 0.001$) in both cases. Thus, PPO achieves statistically superior performance relative to chance as well as a heuristic greedy strategy.

Reward Convergence

Training stability is shown in Figure 1. Convergence occurred consistently across seeds, with variance narrowing after 1×106 steps. This suggests that PPO is robust to variations in initialization and random TLE generation.

Discussion

The statistical evidence supports that reinforcement learning-specifically PPO-provides a systematic advantage for multi-satellite custody maintenance. Importantly, PPO achieves dual-target tracking with consistent results that improve over time, a nontrivial coordination challenge for a centralized agent.

The PPO achieves this by training the central agent to direct tip-cue actions of the satellites holistically to maintain optimum custody coverage of the targets as the satellites orbit. Thus, the agent approaches the problem from the perspective of the constellation as a system to be coordinated rather than individual assets tracking targets independently. The random strategy serves as a de facto null hypothesis, and the greedy strategy demonstrates how the assets would behave if each acts independently myopically focused on custody maintenance of targets already in sight with no consideration of the larger constellation or other targets.

Qualitatively, the PPO agent learns a tipping-and-cueing strategy where satellites anticipate handoffs by cueing upcoming observers based on orbital predictions. For instance, in episodes with clustered targets, the agent develops a protocol to alternate custody, reducing gaps by 15−20% compared to the baseline (as seen in custody logs). This behavior emerges from the reward's handoff penalty, encouraging proactive coordination rather than reactive assignment. Example: When Satellite 1 loses line-of-sight to Target A, it cues Satellite 2 (if visible) to initiate observation, mimicking a decentralized handoff in a centralized framework.

While this approach is designed to serve as a proof of concept, it is subject to significant limitations that should be resolved before considering this a realistic solution to the problem. The movement and number of targets are not what would be expected in a real-world scenario. In addition, the modeling of the satellite footprint modeling is not realistic and should be improved. Finally, the single centralized agent should be tested against MARL techniques to gauge the tradeoff between implementation and training difficulty vs. performance.

Conclusion

The findings of this study highlight the promise of learning-based policies for space domain awareness and persistent ISR. Future work should compare PPO against multi-agent baselines (e.g., MADDPG, MAPPO) to examine scalability to larger constellations and more than two targets. In addition, refinement of the reward function could improve performance; and realistic modeling of the movement of the targets, as well as satellite footprint, would make the scenario more operationally relevant. However, given the scope of this work as a proof of concept, further development and investigation into RL methodologies for the maintenance of target custody is warranted.

Code Availability Statement

The code provided in this work is intended to be used as an example of a highly simplified implementation. stable-baselines3 is used for PPO training, requiring Python 3.10+. The code will be made publicly available via Github; you may reach the author for more details.

Acknowledgment

This work was supported by the University of Alaska Fairbanks Geophysical Institute.

Funding Information

The author has no conflicts of interest to report.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Ballard, A. H. (1980). Roses in a circle: nonsynchronous satellite constellations for continuous coverage. *Journal of the Astronautical Sciences*, *28*(3), 289–312.

Bevilacqua, A. (2009). A novel vision-based approach for autonomous space navigation systems. *Advances in Visual Computing*, 837–846. https://doi.org/https://doi.org/10.1007/978-3-642-10520-3_80

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172. https://doi.org/10.1109/tsmcc.2007.913919

Gaudet, B., Linares, R., & Furfaro, R. (2020). Deep reinforcement learning for six degree-of-freedom planetary landing. *Advances in Space Research*, *65*(7), 1723–1741. https://doi.org/10.1016/j.asr.2019.12.030

Hughes, S. P. (2016). *General mission analysis tool (gmat) (Tech. Rep.). NASA*.

Liu, X. (2020). Mission schedule of agile satellites based on proximal policy optimization algorithm. *Computer Science > Artificial Intelligence*. https://doi.org/https://doi.org/10.48550/arXiv.2007.02352

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Proceeding of the Advances in Neural Information Processing Systems*, 6379–6390. https://doi.org/10.48550/arXiv.1706.02275

Labs, O. (2014). Orekit python wrapper. *GitHub Repository*.

Oliehoek, F. A., & Amato, C. (2016). *A concise introduction to decentralized pomdps*. *1*. https://doi.org/10.1007/978-3-319-28929-8

Pan, H., Sun, W., Sun, Q., & Gao, H. (2021). Deep Learning Based Data Fusion for Sensor Fault Diagnosis and Tolerance in Autonomous Vehicles. *Chinese Journal of Mechanical Engineering*, *34*(1), 72. https://doi.org/10.1186/s10033-021-00568-1

Raffin, A., Hill, A., Ernestus, A., Gleave, A., Kanervisto, A., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8.

Rhodes, B. (2019). Skyfield: High precision researchgrade positions for planets and earth satellites generator. *Astrophysics Source Code Library*.

Roberts, T. G., Siew, P. M., & Jang, D. (2021). A Deep Reinforcement Learning Application to Space-based Sensor Tasking for Space Situational Awareness. *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*. Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, Hawaii, USA.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms*. *1*, 1–12. https://doi.org/10.48550/arXiv.1707.06347

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. *10*.

Walker, J. G. (1977). Satellite constellations. *Journal of the British Interplanetary Society*, *37*, 559–571.