SATBENCH: SATELLITE COORDINATION BENCHMARK BASED ON MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce SatBench, a realistic and flexible benchmark for coordinated satellite tasking in Earth observation (EO) missions. In this setting, multiple satellites potentially in different orbits must coordinate to capture images of ground targets under time constraints. While real-world satellites involve complex sensing and control dynamics, SatBench abstracts these into an agent-based learning framework with a standardized, user-friendly interface. This enables the application of reinforcement learning (RL) methods while preserving critical realism in the environment. SatBench supports a diverse set of configurable scenarios that capture varying EO requirements, including different satellite formations, target distributions, and prioritization schemes. We evaluate representative multi-agent reinforcement learning (MARL) algorithms across these scenarios, highlighting key challenges such as coordination under temporal coupling and scalability. SatBench aims to foster progress in both autonomous satellite coordination and the development of more robust, generalizable MARL methods. SatBench is available at `https://anonymous.4open.science/r/SatBench-43DB`.

1 INTRODUCTION

Modern satellite constellations are transforming the way we observe and monitor Earth's surface, with significant implications for disaster management (Santilli et al., 2018), agriculture (Sciddurlo et al., 2021), and urban planning (Song et al., 2025). The emergence of autonomous, coordinated constellations, comprising large fleets of Earth observation (EO) satellites operated by government and commercial actors such as SpaceX (McDowell, 2020) and BlackSky (Wagner et al., 2024), introduces new challenges for sequential decision-making under uncertainty, with the goal of achieving near real-time monitoring over dynamic regions of interest.

Manually designing coordination strategies for modern satellite constellations is increasingly impractical (Zilberstein et al., 2025). The tight coupling between sensing and mobility across multiple satellites introduces coordination challenges that are difficult to solve using fixed rules or expert-crafted heuristics. Multi-agent reinforcement learning (MARL) offers a compelling solution (Saeed et al., 2024; Holder et al., 2025), where each satellite is modeled as an agent that learns a decentralized coordination policy. These agents can adapt to dynamic EO tasks, make long-term decisions under partial observability, and respond to the behavior of other agents.

Conversely, satellite EO tasking also serves as a rich testbed for advancing MARL research. Many existing MARL benchmarks, such as MPE (Lowe et al., 2017), SMAC (Samvelyan et al., 2019), and MAMuJoCo (Peng et al., 2020), operate in stylized environments with simplified dynamics and coordination structures. In contrast, real-world satellite tasking scenarios involve high-dimensional action and observation spaces, delayed and sparse rewards, and geospatially grounded constraints (Wang et al., 2021c; Saeed et al., 2024). These complexities introduce new challenges into the current MARL algorithm design with strong robustness and higher scalability.

In this work, we introduce **SatBench**, a realistic and flexible benchmark for coordinated satellite tasking designed for MARL, as shown in Figure 1. SatBench simulates EO missions involving multiple satellites in diverse orbits and configurations, operating under partial observability with large yet constrained action spaces. It provides a standardized, easy-to-use interface for agent-environment interaction, together with configurable environments that capture a wide range of EO requirements, including variations in satellite formations, target distributions, and priority levels. This design allows

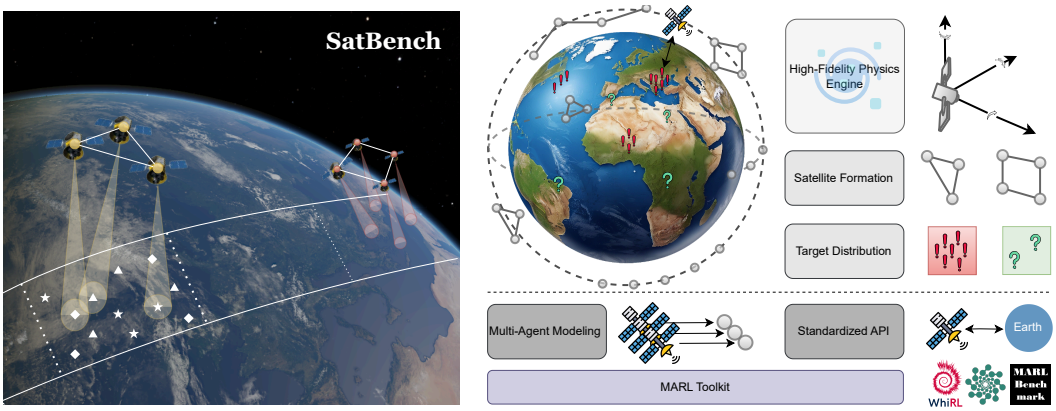


Figure 1: SatBench overview. The left panel illustrates EO satellites operating in coordinated formations over the Earth's surface. The right panel highlights the core components of SatBench: configurable satellite formations, target distributions, and a standardized API. These components enable the definition of diverse task scenarios and support seamless integration with MARL libraries for training and evaluation.

users to readily develop MARL frameworks for satellite EO missions without requiring specialized domain expertise, while still benefiting from the physical realism preserved by SatBench.

The contributions of this benchmark work are three-fold. First, we introduce a realistic multi-satellite simulator that models the physical dynamics and constraints relevant to EO missions, enabling coordination among satellites as a decentralized multi-agent system. Second, we design a suite of configurable, realistic EO mission scenarios that support structured evaluation. Third, we benchmark a range of MARL algorithms, identifying performance trends and coordination behaviors.

The remainder of this paper is organized as follows. Sec. 2 reviews related work on satellite simulation platforms and prior approaches to planning and learning for autonomous satellite operations. Sec. 3 describes the SatBench design, including its four-layer architecture of orbit environment, satellite agents, API, and MARL with a visualization interface. Sec. 4 reports experiments across six scenarios with task setups, results, and coordination findings. Sec. 5 and Sec. 6 discuss future directions, potential impacts, and conclusions.

2 Related Work

This section reviews prior work at the intersection of satellite simulation and task planning. We group the related literature into two categories: (1) existing simulators on space and satellite, and (2) diverse planning and learning frameworks that have been developed for satellite tasking.

2.1 Satellite Simulation Platforms for Space Missions

Simulation platforms play a central role in the development of satellite autonomy, evolving from simplified orbital tools to modular, physically grounded environments. In Table 1, we categorize the progression of these simulators across three dimensions: *Orbital dynamics*, *Satellite modeling*, and *Coordination features*.

Orbital dynamics. Early simulators such as EPOS (Abramson et al., 2002) enabled initial testing of mission planning logic but relied on static or kinematic orbital representations. SGP4 (Vallado et al., 2006) introduced realistic orbital propagation based on two-line elements, allowing for physically meaningful trajectory modeling. Later simulators, including Orekit (Maisonobe et al., 2010) and GMAT (Hughes et al., 2014), incorporated configurable propagators and perturbation models. More recent platforms such as Basilisk (Kenneally et al., 2020), STK (Wall, 2024) and LEOADCS (El wafi et al., 2024) provide high-fidelity orbital dynamics suitable for closed-loop simulations and onboard autonomy development.

Satellite modeling. In parallel, platform-level modeling has advanced from simple abstractions to rich dynamic representations. Early simulators like Orekit (Maisonobe et al., 2010) and GMAT (Hughes et al., 2014) support basic satellite properties such as attitude modes and ground station visibility.

Table 1: Comparison of satellite simulation platforms across orbital dynamics, satellite modeling, and coordination features. SatBench supports all categories, including formation control and configurable target distributions.

Simulator	Orbital Dynamics		Satellite Modeling		Coordination Features		
	Multi-Orbit	Physical Attr.	Multi-Sat	Sat-Dynamics	Formation	Target Dist.	Target Priority
EPOS (Abramson et al., 2002)	✗	✗	✓	Simplified	✗	Sparse	✓
SGP4 (Vallado et al., 2006)	✓	✓	—	—	—	—	—
Orekit (Maisonobe et al., 2010)	✗	✓	✗	Moderate	✗	Sparse	✗
GMAT (Hughes et al., 2014)	✓	✓	✓	Moderate	✗	Sparse	✗
DARTS (Jain, 2020)	✓	✓	✓	Realistic	Limited	—	—
Basilisk (Kenneally et al., 2020)	✓	✓	✓	Realistic	Limited	Sparse	✓
STK (Wall, 2024)	✓	✓	✓	Realistic	Limited	Sparse	✓
LEOADCS (El wafi et al., 2024)	✗	✓	✗	Realistic	✗	—	—
SatBench (Ours, 2025)	✓	✓	✓	Realistic	✓	**Sparse & Dense**	✓

DARTS (Jain, 2020), Basilisk (Kenneally et al., 2020) and STK (Wall, 2024) expanded this to include realistic actuator dynamics, energy management, and sensor control. LEOADCS (El wafi et al., 2024) offers similar physical fidelity in a computationally efficient package for single-satellite simulations. However, few platforms offer standardized interfaces for interactive or learning-driven control.

Coordination features. Coordinated satellite operations, such as collaborative imaging, formation control, and multi-target scheduling, require simulation features beyond individual dynamics. EPOS (Abramson et al., 2002) provided early support for multi-satellite testing with prioritized targets, though lacking physical realism. DARTS (Jain, 2020) added support for event-driven tasking and limited formation operations. Basilisk (Kenneally et al., 2020) and STK (Wall, 2024) enable scripted multi-satellite simulations, but lack native abstractions for complex task environments. These limitations pose challenges for benchmarking learning-based autonomy in coordination settings.

To address these gaps, our SatBench provides a configurable space simulation platform that supports multi-satellite physical realism alongside features essential for learning-based coordination. It integrates accurate orbital and satellite dynamics with abstractions for multi-satellite formations, dense target fields, and prioritized tasking. SatBench also exposes a structured interaction interface, enabling researchers to evaluate autonomous decision-making strategies in simulation environments that reflect real-world complexity.

2.2 Planning and Learning Methods for Satellite Tasking

Effective satellite autonomy relies not only on simulation fidelity but also on robust strategies for dynamic tasking with various targets. Here, we review the evolution of decision-making algorithms, from classical planning techniques to recent advances in learning-based coordination.

Traditional planning methods. Classical approaches to satellite tasking have relied on rule-based or optimization-based scheduling, particularly for single-satellite and static scenarios. For example, Lin et al. (2005) applied heuristic optimization to daily target imaging schedules, while Liang et al. (2021) used precedence rules to manage activity planning under resource constraints. To better handle dynamic conditions, Han et al. (2022) proposed a simulated annealing-based heuristic to prioritize high-value targets with minimal disruption. These methods are interpretable and efficient in deterministic environments, but they struggle to scale or adapt in uncertain, dynamic multi-satellite contexts. Nevertheless, they demonstrate the potential of automated heuristics as substitutes for fully human-crafted rules.

RL approaches. A range of data-driven methods have been explored to support adaptive satellite tasking and operation, including supervised learning and imitation learning techniques (Shirobokov et al., 2021; Ashith Shyam et al., 2021). Among learning-based methods, reinforcement learning (RL) has gained particular attention due to its ability to learn adaptive policies through trial-and-error interaction with the space environment. Early RL efforts on satellite operations focused on maximizing task completion under simplified dynamics (Chen et al., 2019; Huang et al., 2021), whereas more recent work incorporates realistic operation demands, such as power, actuator, and sensor (Herrmann & Schaub, 2023). RL has also been extended to multi-satellite problems (Lin et al., 2024), and dynamic planning techniques have enabled satellites to adapt to changing task target priorities in near real time (Li et al., 2025). These advances demonstrate RL's potential as a flexible approach for autonomous decision-making in complex space environments.

However, most existing RL-based approaches still focus on operation-level policy optimization with oversimplified dynamics, and rarely consider how policies can be optimized when inter-satellite

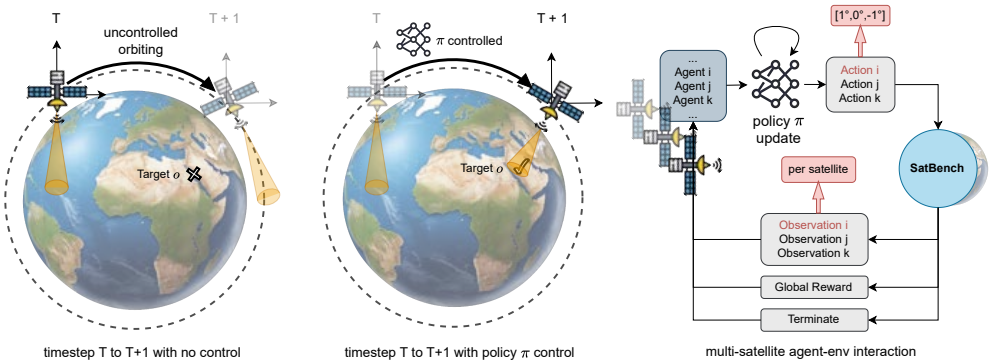


Figure 2: SatBench rollout loop: agents act based on local observations, and the environment simulates orbital dynamics, applies actions, and returns rewards for valid imaging.

cooperation is critical to mission success. To this end, multi-agent reinforcement learning (MARL) offers a promising framework, enabling distributed policies and cooperative behavior among satellites (Saeed et al., 2024; Lei et al., 2025). Realizing this potential requires realistic and extensible benchmarks, which motivates the design of SatBench.

3 SatBench: A Benchmark for Satellite Coordination via MARL

To address the identified gaps above, we design SatBench, a MARL environment modeling EO tasking as a cooperative decision-making problem. SatBench provides a physically grounded simulation platform that captures the structure and difficulty of realistic EO missions while offering a flexible API for MARL integration. SatBench consists of four modular layers: (1) a `space environment` that simulates orbital motion, satellite constraints, and ground target visibility; (2) an `agent modeling layer` that defines the observation, action, and reward interface for each satellite; (3) a `standardized API layer` that wraps the environment using PettingZoo (Terry et al., 2021) and Gymnasium (Towers et al., 2024) conventions; and (4) `plug-and-play compatibility` with widely used MARL libraries. The remainder of this section introduces these components in turn.

3.1 Space Environment: Modeling Orbital Tasking Constraints

SatBench simulates the physical substrate in which satellite coordination decisions unfold. It models a number of EO satellites tasked with imaging fixed ground targets under realistic physical constraints, including orbital trajectories, satellite dynamics, and target sensing range.

Orbit. Each satellite follows a deterministic trajectory defined by Classical Orbital Elements (COEs) (Stojanovski & Savransky, 2024), a standard 6-parameter representation that enables precise and flexible orbit propagation. This formulation supports a wide range of constellation configurations, including sun-synchronous, polar, and inclined orbits through direct control of orbital geometry. Compared to simplified alternatives like TLE+SGP4 (Vallado et al., 2006), which are optimized for short-term orbit prediction, COE-based propagation provides greater long-term numerical stability and tunability, making it more suitable for simulating realistic, temporally extended tasking scenarios.

Satellite. In SatBench, all satellites are equipped with a 3-degree-of-freedom (3-DoF) attitude control model, allowing them to steer their sensors using `pitch`, `yaw`, and `roll` commands (Crisp et al., 2018). These motions are subject to actuator dynamics that constrain angular velocity, acceleration, and control frequency. As a result, pointing changes occur gradually, reflecting the agility limits of real EO satellites such as Planet's FLOCK series (Foster et al., 2017). The left panel of Figure 2 illustrates this control process and its implications for sensing coverage. Satellites must continuously adjust their attitude to bring targets into view, often under motion constraints and coordination demands with other satellites. When the number of satellites exceeds two, their spatial configuration is also considered, a setting referred to as formation flying (Thangavel et al., 2024). SatBench models three commonly used formations: (1) *Clustered*, where all satellites share the same orbit and remain in close proximity; (2) *Trailing*, where satellites are spaced at greater distances along the same orbit, following one another; and (3) *Constellation*, where satellites are distributed across multiple orbits to achieve global coverage.

Target. A target is defined as a fixed ground location on the Earth's surface, specified by `latitude` and `longitude`, with optional attributes such as `priority`. In real-world EO missions, a target corresponds to an area of interest (AoI) with high value that requires monitoring (e.g., high-risk environmental zones (Zhang et al., 2022)). The distribution of targets can be configured from sparse to dense: sparse layouts reflect broad regional coverage, while dense layouts create local contention and coordination pressure (Gu et al., 2022).

The details of orbital setup, satellite dynamics, and target modeling can be found in **Appendix A**.

3.2 Agent Modeling Layer: Observation, Action, and Reward

SatBench models each satellite as an autonomous agent that perceives its local environment, makes control decisions, and receives feedback based on mission outcomes. As shown in Figure 2 right panel, the agent's behavior is shaped by three core components: its observations, the set of available actions, and the reward signals tied to successful EO tasking execution for capturing ground targets. These components reflect both the physical constraints of space operations and the decision-making challenges of coordinated satellite tasking, forming the foundation for the following learning pipelines.

Observation. Each agent observes a compact, task-relevant state composed of (i) its own satellite status, including orbital position, attitude, and angular velocity, and (ii) a filtered list of observable targets that fall within its sensing range. Visibility is determined by geometric constraints such as the minimum elevation angle, which align with real-world EO tasks constraints (Lv et al., 2022).

Action. The agent's action is defined as a three-dimensional real-time control vector, specifying the attitude adjustments applied to its current orientation. The rate of change along each axis is bounded by the satellite's physical constraint, known as the slew rate (Gorr et al., 2023). By default in SatBench, all satellites are limited to a maximum slew rate of 0.25° per second per axis, consistent with typical real-world satellite capabilities (Petermann et al., 2025). For example, if the current orientation is $[3^\circ, 5^\circ, 1^\circ]$ and the agent outputs an action of $[0.25^\circ, 0.25^\circ, -0.25^\circ]$ for 4 seconds, the accumulated orientation change will be $[1^\circ, 1^\circ, -1^\circ]$, resulting in a new orientation of $[4^\circ, 6^\circ, 0^\circ]$.

Reward. Rewards are granted only for successful, non-redundant imaging of valid targets. To qualify, a target must (i) be physically visible from the satellite's current orbit, (ii) lie within the field of view given the current attitude, and (iii) not have already been imaged by another agent. The reward is re-weighted by target priority, encouraging intelligent selection and inter-agent deconfliction.

We provide the detailed mathematical definitions and implementation of the agent-based observation, action, and reward design in **Appendix B**.

3.3 Standardized API and Integration for MARL Research

SatBench supports MARL through a standardized interface that allows researchers to run algorithms with minimal setup effort. The environment is designed to work with existing MARL toolkits, while retaining the flexibility for customizing the learning pipeline.

API compatibility. SatBench adheres to the PettingZoo (Terry et al., 2021) and Gymnasium (Towers et al., 2024) interfaces for agent-environment interaction in both single-agent and multi-agent settings. This standardization enables smooth integration into RL and MARL pipelines without the need for additional wrappers or conversion code. For MARL, the environment supports commonly used training features such as Centralized Training with Decentralized Execution (CTDE) (Schmidt et al., 2022), sequential and parallel agent rollouts, and parameter sharing across agents.

Training. SatBench is compatible with established MARL frameworks, including PyMARL (Samvelyan et al., 2019), EPyMARL (Papoudakis et al., 2021), and MARLlib (Hu et al., 2023). Users can launch training runs directly from the command line using built-in scenario configurations. The example below shows how to train a MAPPO (Yu et al., 2022) policy on a predefined task using EPyMARL:

```
$ python src/main.py --config=mappo --env-config=scenario1
```

Details on computing resources, hyperparameter settings, and additional training configurations are provided in **Appendix C**.

Usage example. SatBench can be easily instantiated for standalone evaluation or integration into larger training pipelines. The following code demonstrates how to initialize the environment and run a full episode rollout. Setting `render=True` will generate and save a visualization of the trajectory.

```python
from SatBenchMaps import make_satbench_env
env = make_satbench_env(map_name="scenario1",render=True)
env.reset()
done = False
while not done:
    action = env.action_space.sample()
    observation, reward, termination, truncation, info = env.step(action)
    if terminated or truncated:
        done = True
```

Customization and modular design. SatBench is built with a modular structure that separates the learning interface, simulation backend, and visualization logic. This design makes it easy to extend or replace components without modifying core files. Users can customize reward functions, observation spaces, environment arguments, or policy architectures through clean and isolated interfaces. This allows for rapid prototyping while preserving the integrity of the core environment, supporting both reproducible research and tailored experimentation. We illustrate how to design customized scenarios for future MARL research in **Appendix D**.

Source code and guide. We provide step-by-step instructions to install SatBench and run standard MARL experiments, along with the source code, in **Appendix E**.

4 EXPERIMENTS

To assess the utility of SatBench as both a benchmarking tool and a research testbed, we evaluate representative MARL algorithms across a suite of diverse satellite tasking scenarios. These experiments are carefully designed to probe the strengths and limitations of learning-based coordination in environments that closely mirror real-world conditions. By varying the number of satellites, orbital configurations, and spatial layouts of imaging targets, we aim to establish baseline performance and expose coordination challenges critical for advancing robust and generalizable MARL methods.

4.1 SCENARIO

To systematically evaluate coordination under varying spatial and temporal constraints, SatBench introduces six benchmark scenarios (see Table 2). Scenarios 1-4 are inspired by real-world satellite formation missions developed by the European Space Agency (Liu & Zhang, 2018; Shestov et al., 2021), while Scenarios 5-6 model constellations with more satellites, motivated by commercial Earth observation systems such as DigitalGlobe and Airbus (Denis et al., 2017). Collectively, these scenarios are designed to capture a progressive increase in Earth observation tasking complexity by varying the number of orbits, formation geometries, target distributions, and the total number of satellites and imaging tasks, thereby enabling rigorous evaluation of MARL algorithms under realistic and challenging coordination settings.

To be more specific, Scenarios 1 and 2 simulate close-proximity satellite operations within a single orbit, representing low-complexity cluster formations. Scenario 1 involves two satellites and 60 sparsely distributed targets, serving as a baseline with minimal sensing conflicts (Chen et al., 2019). Scenario 2 increases the target density to 70, requiring tighter coordination to avoid redundant observations and to manage limited actuation resources more effectively. Scenarios 3 and 4 employ a trailing formation with satellites allocated across two different orbits. Compared to Scenarios 1 and 2, these settings introduce spatial and temporal separation, shifting the coordination challenge from local interference avoidance to long-horizon planning (Wu et al., 2022). The number of targets is

Table 2: Six benchmark scenarios' configuration with diverse orbit number, satellite allocation, and target distribution settings.

Scenario	Num. of Orbits	Num. of Satellites	Formation Type	Num. of Targets	Target Distribution
Scenario 1 (1Orb_2Sat_Cl_60Tgt_Sp)	1	2	Cluster	60	Sparse
Scenario 2 (1Orb_2Sat_Cl_70Tgt_Dn)	1	2	Cluster	70	Dense
Scenario 3 (2Orb_4Sat_Tr_120Tgt_Sp)	2	4	Trailing	120	Sparse
Scenario 4 (2Orb_4Sat_Tr_140Tgt_Dn)	2	4	Trailing	140	Dense
Scenario 5 (3Orb_6Sat_Cn_240Tgt_Sp)	3	6	Constellation	240	Sparse
Scenario 6 (6Orb_6Sat_Cn_280Tgt_Dn)	6	6	Constellation	280	Dense

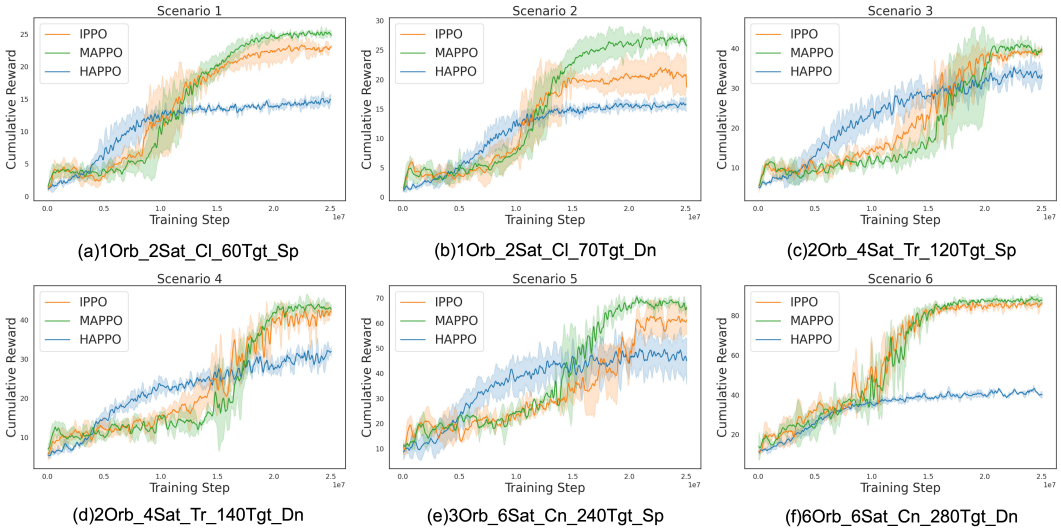


Figure 3: MARL training performance of IPPO, MAPPO, and HAPPO across SatBench scenarios (a)-(f), measured by cumulative reward. Solid lines indicate the mean reward over 4 random seeds, and shaded regions represent the standard deviation.

also increased, creating more overlap and requiring stronger inter-satellite cooperation. Scenarios 5 and 6 model the most realistic and most challenging systems: all satellites are distributed across more than three orbits, forming a large constellation (Chatterjee & Tharmarasa, 2022), with over 200 ground targets that are either sparsely or densely distributed and observable by six satellites. These scenarios reflect real-world commercial Earth observation networks that demand wide-area coverage. For MARL, they are the most challenging, as the observation tasks are more complex, while opportunities for inter-satellite coordination are urgently required during the mission.

4.2 Results and Analysis

We evaluate representative MARL algorithms, such as IPPO (de Witt et al., 2020), MAPPO (Yu et al., 2022), and HAPPO (Kuba et al., 2021b), across 6 SatBench scenarios, each trained for 25 million steps. Figure 3 presents MARL performance through the cumulative rewards per episode, which correspond to the total value of successfully imaged targets by satellites. To complement this metric, we also report the target completion rate (see Table 3), defined as the percentage of targets that are captured at least once by the end of an episode. The cumulative reward reflects each algorithm's ability to coordinate satellites in prioritizing high-value targets, while the target completion rate provides insight into overall coverage and coordination efficiency across varying scenario complexities.

Across all scenarios reported in Table 3, MAPPO consistently achieves the highest final task completion performance, benefiting from its centralized critic that facilitates coordinated task allocation. IPPO performs moderately, as its decentralized structure limits inter-agent awareness and coordination. HAPPO demonstrates rapid early learning but fails to maintain performance in later stages, particularly in complex settings. To further understand the observed performance differences, we analyse results through the lens of two core structural factors: satellite formation and target distribution.

Role of satellite formation. Different satellite formations demand varying levels of cooperation. Intuitively, satellites within the same cluster require stronger coordination skills, while those farther apart, such as in trailing configurations, require less. Our experimental results confirm this assumption: in clustered formations (Scenarios 1-2), explicit coordination is necessary due to overlapping FOVs. Here, MAPPO performs best with centralized training, IPPO underperforms due to limited inter-agent awareness, and HAPPO learns quickly at first but soon plateaus, indicating inefficiency in tightly coupled settings. In contrast, in trailing formations (Scenarios 3-4), where overlap between satellites is greatly reduced, the performance gap between MAPPO and the other algorithms narrows. Finally, in constellation formations (Scenarios 5-6), the wide separation minimizes coordination needs, making MAPPO and IPPO comparable.

Table 3: MARL training performance measured by task completion rate (%) across 6 SatBench scenarios. All values are reported as mean ± standard deviation over 4 random seeds. Best performance in each scenario is highlighted in bold.

Scenario	Algorithm	Training Timesteps				
		1M	5M	10M	15M	25M
Scenario 1	IPPO	5.12% ± 1.83%	11.07% ± 1.25%	40.16% ± 17.42%	62.41% ± 5.91%	77.33% ± 1.12%
	MAPPO	5.23% ± 1.61%	13.14% ± 3.93%	31.62% ± 9.73%	67.53% ± 4.31%	**83.42% ± 1.31%**
	HAPPO	3.65% ± 0.92%	23.08% ± 8.21%	41.35% ± 1.92%	46.82% ± 3.01%	49.02% ± 2.24%
Scenario 2	IPPO	5.01% ± 1.52%	10.93% ± 3.74%	24.71% ± 5.31%	57.93% ± 4.03%	59.61% ± 8.03%
	MAPPO	4.52% ± 1.72%	14.53% ± 5.01%	22.12% ± 3.43%	64.31% ± 8.43%	**73.42% ± 3.12%**
	HAPPO	3.41% ± 0.71%	15.94% ± 4.81%	39.13% ± 4.62%	42.81% ± 3.83%	44.53% ± 3.73%
Scenario 3	IPPO	9.33% ± 2.43%	16.53% ± 1.94%	23.63% ± 1.84%	35.72% ± 9.01%	**65.41% ± 1.03%**
	MAPPO	8.79% ± 0.82%	14.91% ± 2.21%	19.44% ± 2.63%	30.01% ± 11.43%	**66.21% ± 0.63%**
	HAPPO	8.22% ± 0.53%	21.73% ± 4.14%	41.51% ± 5.91%	46.19% ± 3.71%	56.23% ± 4.03%
Scenario 4	IPPO	8.53% ± 2.31%	17.63% ± 2.82%	21.64% ± 2.34%	30.71% ± 7.34%	59.82% ± 1.71%
	MAPPO	10.02% ± 1.53%	14.44% ± 1.54%	20.91% ± 2.83%	22.41% ± 12.21%	**61.83% ± 0.52%**
	HAPPO	7.72% ± 0.73%	21.73% ± 4.31%	33.41% ± 4.03%	39.12% ± 2.51%	44.72% ± 2.03%
Scenario 5	IPPO	8.91% ± 2.43%	10.81% ± 0.05%	19.63% ± 1.63%	26.53% ± 5.51%	51.93% ± 3.92%
	MAPPO	9.31% ± 0.62%	19.52% ± 0.82%	20.24% ± 2.21%	38.12% ± 10.42%	**55.41% ± 0.22%**
	HAPPO	7.23% ± 2.41%	20.21% ± 5.01%	31.63% ± 6.42%	37.21% ± 3.81%	36.34% ± 9.52%
Scenario 6	IPPO	8.34% ± 1.32%	22.71% ± 1.82%	39.12% ± 8.23%	59.43% ± 1.91%	**62.31% ± 1.03%**
	MAPPO	10.01% ± 5.03%	17.72% ± 5.92%	34.63% ± 9.73%	58.12% ± 4.23%	**62.61% ± 1.03%**
	HAPPO	8.41% ± 0.92%	19.43% ± 2.41%	25.23% ± 0.93%	27.91% ± 0.91%	28.73% ± 1.13%

Impact of target distribution. Target density significantly affects coordination complexity. In sparse scenarios (1, 3, and 5), lower contention allows decentralized methods like IPPO to perform well. In contrast, dense scenarios (2, 4, and 6) lead to more overlapping observations, where MAPPO consistently outperforms others by optimizing coverage and reducing redundancy. An exception is Scenario 6, where wide satellite separation limits interference despite high density, showing that formation geometry can outweigh target density in driving coordination needs.

Illustrative example. We visualize the best coordination performance, achieved by MAPPO in Scenario 1 in Figure 4. This snapshot illustrates how SatBench supports emergent coordination in realistic satellite tasking. Early in training (5M steps), satellites misalign with targets; by 15M steps, they begin orienting toward observable areas; and by 25M steps, coordinated behavior emerges, enabling successful capture of high-priority targets.

Additional experiments and demo. We report additional findings on single-satellite RL performance in simplified scenarios for low-complexity EO tasks, comparisons showing that value-based methods underperform relative to policy-based MARL methods, and evaluations of MARL performance on EO-constrained cooperative tasks, as described in **Appendix F**. For users interested in downstream scenarios, we provide a fire monitoring use case demonstrating how to access relevant satellite and target data, and how MARL achieves better coordination compared to traditional scheduling methods, as detailed in **Appendix G**.

5 Discussion, Impact, and Future Directions

SatBench is designed to bridge two communities: EO and MARL by providing a benchmark that is both operationally realistic and algorithmically revealing.

MARL for Space. From an EO perspective, the benchmark highlights the practical difficulty of translating RL successes into space systems. In real missions, task allocation is constrained by fixed orbital trajectories (Lv et al., 2022), limited maneuverability (Lim et al., 2025), sensor noise (Ditmar et al., 2012), environmental uncertainty such as cloud coverage (Han et al., 2022), and asynchronous dynamics across spacecraft (Chen & Luo, 2024). These factors indicate that decisions about "which target to capture" cannot be treated in isolation from "how and when to point." SatBench's continuous pointing formulation explicitly captures this coupling, requiring algorithms to manage the physical cost of retargeting while maximizing coverage and priority-weighted reward. This design ensures that performance gains in the benchmark correspond to improvements in operational feasibility for EO missions, particularly in applications such as disaster monitoring and resource management.

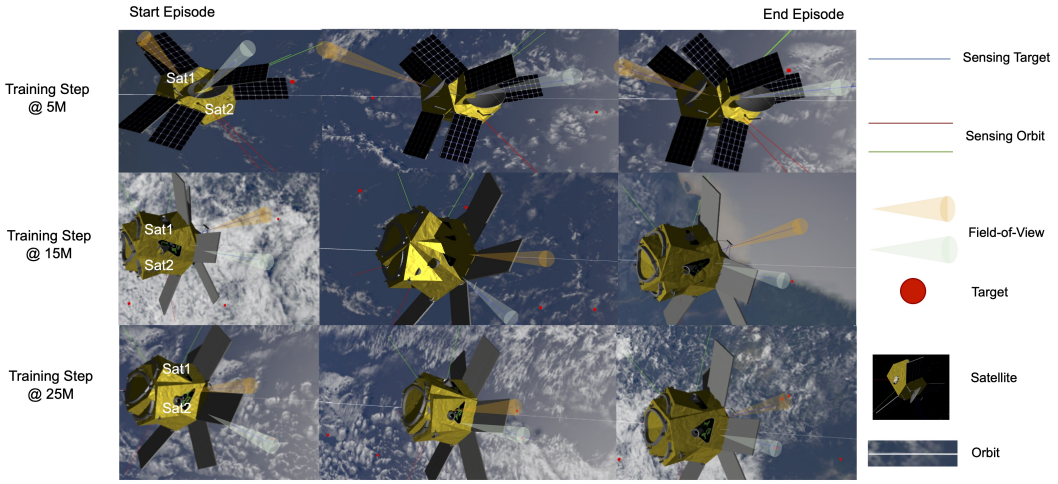


Figure 4: Snapshots of MARL training progression (at 5M, 15M, and 25M steps) for coordinating two satellites in Scenario 1 using MAPPO. The two satellites appear nearly overlapping due to their close physical proximity and parallax along the orbit, consistent with the cluster formation setting.

Next-Generation MARL. From a MARL perspective, SatBench exposes a structured set of challenges that align with open questions in the field. Coordinated satellite tasking involves heterogeneous agents with different orbital positions and capabilities (Wang et al., 2021a), limited communication bandwidth (Wang et al., 2022), and asynchronous actuation cycles that unfold over long horizons (Xiao et al., 2022). Taken together, these characteristics create natural testbeds for specialized MARL paradigms: robust MARL for handling model and environment uncertainty (Zhang et al., 2020), communication-aware MARL for bandwidth-limited coordination (Das et al., 2019), heterogeneous MARL for diverse agent roles (Wang et al., 2021b), and asynchronous MARL for staggered decision-making (Xiao et al., 2022). The widely used baselines such as IPPO, MAPPO, and HAPPO face systematic limitations under these conditions, not due to artificially imposed difficulty, but because the benchmark surfaces intrinsic coordination barriers, such as delayed credit assignment, inter-agent interference, and scaling with constellation size. By making these challenges reproducible, SatBench provides MARL researchers with a principled environment to evaluate and advance next-generation MARL algorithms.

Broad Impact. The broader contribution of SatBench lies in creating a shared problem formulation that is meaningful for both communities. For EO researchers, it offers a simulation environment where algorithmic improvements can be directly linked to operational constraints, enabling a clearer assessment of when learning-based autonomy offers advantages over traditional optimization approaches. For MARL researchers, it offers a domain where core methodological advances: credit assignment, communication, heterogeneity, and robustness can be stress-tested against realistic physical constraints rather than stylized toy environments. In this sense, SatBench is not only a benchmark for performance comparison but also a catalyst for cross-disciplinary progress, encouraging solutions that are simultaneously scientifically rigorous and practically impactful.

6 Conclusion

We introduced SatBench, a modular benchmark for evaluating multi-agent coordination in realistic Earth observation satellite tasking. By integrating physically grounded satellite dynamics, configurable scenario design, and a standardized MARL-compatible API, SatBench bridges the gap between real-world mission simulation and autonomous multi-agent decision-making. Through experiments, we demonstrated how formation geometry and target distribution shape coordination demands and directly influence MARL performance. Looking ahead, SatBench provides a foundation for advancing robust and generalizable MARL methods that operate under realistic physical constraints, with downstream impact on high-stakes EO applications such as climate monitoring, environmental surveillance, and rapid disaster response.

Reproducibility Statement

To support reproducibility, the appendices include all details necessary to replicate the SatBench experiments and results presented in this work. Appendix A describes the satellite orbital configuration, rotation control, formation flying methods, and ground target modeling. Appendix B details how satellites are modeled as RL agents, including observation space, action space, and reward structure. Appendix C provides hyperparameter settings, training pipelines, and environment rollout times for fair benchmarking. Appendix D outlines the modular architecture supporting communication, mean-field RL, heterogeneous satellites, noise and uncertainty injection, asynchronous control, and scenario customization. Appendix E provides the source code, installation instructions, training scripts, configuration file structure, visualization tools, and a comprehensive list of software dependencies. Together, these appendices provide a comprehensive and transparent framework enabling other researchers to fully replicate the experiments and independently validate SatBench claims and conclusions.

Ethics Statement

The SatBench research benchmarks MARL in the context of EO missions, relying solely on publicly available code repositories. It involves no human subjects, collects no personal data, and contains no sensitive information. Accordingly, it is not expected to produce negative ethical consequences, facilitate misuse, or cause societal harm.

References

Mark Abramson, David Carter, Stephan Kolitz, Michael Ricard, and Pete Scheidler. Real-time optimized earth observation autonomous planning. In *Proceedings of the NASA Earth Science Technology Conference*, pp. 9–12, 2002.

Emile Anand and Guannan Qu. Efficient reinforcement learning for global decision making in the presence of local agents at scale. *arXiv preprint arXiv:2403.00222*, 2024.

Emile Anand, Ishani Karmarkar, and Guannan Qu. Mean-field sampling for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2412.00661*, 2024.

RB Ashith Shyam, Zhou Hao, Umberto Montanaro, Shilp Dixit, Arunkumar Rathinam, Yang Gao, Gerhard Neumann, and Saber Fallah. Autonomous robots for space: Trajectory learning and adaptation using imitation. *Frontiers in Robotics and AI*, 8:638849, 2021.

Abhijit Chatterjee and Ratnasingham Tharmarasa. Reward factor-based multiple agile satellites scheduling with energy and memory constraints. *IEEE Transactions on Aerospace and Electronic Systems*, 58(4):3090–3103, 2022.

Ming Chen, Yuning Chen, Yingwu Chen, and Weihua Qi. Deep reinforcement learning for agile satellite scheduling problem. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 126–132, 2019.

Xin Chen and Zhiyong Luo. Asynchronous interference mitigation for leo multi-satellite cooperative systems. *IEEE Transactions on Wireless Communications*, 2024.

Nicholas Crisp, Peter Roberts, Steve Edmondson, Sarah Haigh, Claire Huyton, Sabrina Livadiotti, Vitor Toshiyuki Abrao Oiko, Katharine Smith, Stephen Worrall, J Becedas, et al. Soar-satellite for orbital aerodynamics research. In *69th International Astronautical Congress*, 2018.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted Multi-Agent Communication. In *International Conference on Machine Learning (ICML)*, pp. 1538–1546, 2019.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Gil Denis, Alain Claverie, Xavier Pasco, Jean-Pierre Darnis, Benoît de Maupeou, Murielle Lafaye, and Eric Morel. Towards disruptions in earth observation? new earth observation systems and markets evolution: Possible scenarios and impacts. *Acta Astronautica*, 137:415–433, 2017.

Zegang Ding, Pengnan Zheng, Han Li, Tianyi Zhang, and Zhe Li. Spaceborne high-squint high-resolution sar imaging based on two-dimensional spatial-variant range cell migration correction. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022.

Pavel Ditmar, João Teixeira da Encarnação, and Hassan Hashemi Farahani. Understanding data noise in gravity field recovery on the basis of inter-satellite ranging measurements acquired by the satellite gravimetry mission grace. *Journal of Geodesy*, 86(6):441–465, 2012.

Ilyas El wafi, Mohamed Haloua, Zouhair Guennoun, and Zakaria Moudden. A framework for developing an attitude determination and control system simulator for cubesats: Processor-in-loop testing approach. *Results in Engineering*, 22:102201, 2024.

Qiaoyun Fan, Chunyu Chen, Gangyi Wang, and Xinguo Wei. Parameters estimation of nutational satellite based on sun sensor. *IEEE Transactions on Instrumentation and Measurement*, 71:1–8, 2022.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Cyrus Foster, James Mason, Vivek Vittaldev, Lawrence Leung, Vincent Beukelaers, Leon Stepan, and Rob Zimmerman. Constellation phasing with differential drag on planet labs satellites. *Journal of Spacecraft and Rockets*, 55:1–11, 2017.

Ben Gorr, Alan Aguilar Jaramillo, Zida Wu, Wooyeong Cho, Kewei Cheng, Molly Stroud, Vinay Ravindra, Cédric H David, Huilin Gao, Yizhou Sun, et al. Multi-instrument flood monitoring with a distributed, decentralized, dynamic and context-aware satellite sensor web. In *IEEE International Geoscience and Remote Sensing Symposium*, pp. 4602–4605. IEEE, 2023.

Yi Gu, Chao Han, Yuhan Chen, Shenggang Liu, and Xinwei Wang. Large region targets observation scheduling by multiple satellites using resampling particle swarm optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 59(2):1800–1815, 2022.

Chao Han, Yi Gu, Guohua Wu, and Xinwei Wang. Simulated annealing-based heuristic for multiple agile satellites scheduling under cloud coverage uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(5):2863–2874, 2022.

Adam Herrmann and Hanspeter Schaub. Reinforcement learning for the agile earth-observing satellite scheduling problem. *IEEE Transactions on Aerospace and Electronic Systems*, 59(5):5235–5247, 2023.

Joshua Holder, Natasha Jaques, and Mehran Mesbahi. Multi agent reinforcement learning for sequential satellite assignment problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 26516–26524, 2025.

Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Xiaodan Liang, Zhihui Li, Xiaojun Chang, and Yaodong Yang. Marllib: A scalable and efficient multi-agent reinforcement learning library. *Journal of Machine Learning Research*, 24(315):1–23, 2023.

Yixin Huang, Zhongcheng Mu, Shufan Wu, Benjie Cui, and Yuxiao Duan. Revising the observation satellite scheduling problem based on deep reinforcement learning. *Remote Sensing*, 13(12), 2021.

Steven P Hughes, Rizwan H Qureshi, Steven D Cooley, and Joel J Parker. Verification and validation of the general mission analysis tool (gmat). In *AIAA/AAS Astrodynamics Specialist Conference*, pp. 4151, 2014.

Abhinandan Jain. Darts - multibody modeling, simulation and analysis software. In *Multibody Dynamics*, 2020.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

Patrick Kenneally, Scott Piggott, and Hanspeter Schaub. Basilisk: A flexible, scalable and modular astrodynamics simulation framework. *Journal of Aerospace Information Systems*, 17:1–13, 05 2020. doi: 10.2514/1.I010762.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2021a.

Jakub Grudzien Kuba, Ruiqing Chen, Munning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *International Conference on Learning Representations*, 2021b.

Guangchen Lan, Dong-Jun Han, Abolfazl Hashemi, Vaneet Aggarwal, and Christopher Brinton. Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis. In *The Thirteenth International Conference on Learning Representations*, 2024.

Chengjia Lei, Shaohua Wu, Yi Yang, Jiayin Xue, Dawei Chen, Pengfei Duan, and Qinyu Zhang. Joint partitioning, allocation, and transmission optimization for federated learning in satellite constellations via multi-task marl. *IEEE Transactions on Mobile Computing*, 2025.

Peiyan Li, Peixing Cui, and Huiquan Wang. Mission sequence model and deep reinforcement learning-based replanning method for multi-satellite observation. *Sensors*, 25(6), 2025.

Jun Liang, Yue he Zhu, Ya zhong Luo, Jia cheng Zhang, and Hai Zhu. A precedence-rule-based heuristic for satellite onboard activity planning. *Acta Astronautica*, 178:757–772, 2021.

Elliot Lim, Timothy Larson, Luke Johnston, Kate Johnston, Adam Garton, Phillip Schmedeman, Leo Langou, Karoline Hood, and William Koch. Exploring leading indicators of satellite maneuvers in geosynchronous orbit. In *IEEE International systems Conference (SysCon)*, pp. 1–5. IEEE, 2025.

Wei-Cheng Lin, Da-Yin Liao, Chung-Yang Liu, and Yong-Yao Lee. Daily imaging scheduling of an earth observation satellite. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(2):213–223, 2005.

Zhiyuan Lin, Zuyao Ni, Linling Kuang, Chunxiao Jiang, and Zhen Huang. Satellite-terrestrial coordinated multi-satellite beam hopping scheduling based on multi-agent deep reinforcement learning. *IEEE Transactions on Wireless Communications*, 2024.

Guo-Ping Liu and Shijie Zhang. A survey on formation control of small satellites. *Proceedings of the IEEE*, 106(3):440–457, 2018.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems*, 2017.

Mingsong Lv, Xuemei Peng, Wenjing Xie, and Nan Guan. Task allocation for real-time earth observation service with leo satellites. In *IEEE Real-Time Systems Symposium*, pp. 14–26. IEEE, 2022.

Luc Maisonobe, Véronique Pommier, and Pascal Parraud. Orekit: An open source library for operational flight dynamics applications. In *International Conference on Astrodynamics Tools and Techniques*, pp. 3–6, 2010.

Michael A Marshall, Richard G Madonna, and Sergio Pellegrino. Investigation of equatorial medium earth orbits for space solar power. *IEEE Transactions on Aerospace and Electronic Systems*, 58(3): 1574–1592, 2021.

Jonathan C. McDowell. The low earth orbit satellite population and impacts of the spacex starlink constellation. *The Astrophysical Journal Letters*, 892(2):L36, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Bei Peng, Tabish Rashid, C. S. D. Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Bohmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. In *Neural Information Processing Systems*, 2020.

Timon Petermann, Eric Jäger, Lisa Elsner, Dominik Pearson, Guido Dietl, and Klaus Schilling. A distributed hardware-in-the-loop testbed for attitude control of small communication satellites. In *2025 IEEE Space Hardware and Radio Conference*, pp. 8–11, 2025.

Thomy Phan, Fabian Ritz, Philipp Altmann, Maximilian Zorn, Jonas Nüßlein, Michael Kölle, Thomas Gabor, and Claudia Linnhoff-Popien. Attention-based recurrence for multi-agent reinforcement learning under stochastic partial observability. In *International Conference on Machine Learning*, pp. 27840–27853. PMLR, 2023.

Qingyu Qu, Kexin Liu, Xijun Li, Yunfan Zhou, and Jinhu Lü. Satellite observation and data-transmission scheduling using imitation learning based on mixed integer linear programming. *IEEE Transactions on Aerospace and Electronic Systems*, 59(2):1989–2001, 2022.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Amir K. Saeed, Francisco Holguin, Alhassan S. Yasin, Benjamin A. Johnson, and Benjamin M. Rodriguez. Multi-agent and multi-target reinforcement learning for satellite sensor tasking. In *2024 IEEE Aerospace Conference*, pp. 1–13, 2024.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, 2019.

Giancarlo Santilli, Cristian Vendittozzi, Chantal Cappelletti, Simone Battistini, and Paolo Gessini. Cubesat constellations for disaster management in remote areas. *Acta Astronautica*, 145:11–17, 2018.

Lukas M Schmidt, Johanna Brosig, Axel Plinge, Bjoern M Eskofier, and Christopher Mutschler. An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility. In *IEEE 25th International Conference on Intelligent Transportation Systems*, pp. 1342–1349. IEEE, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Giancarlo Sciddurlo, Antonio Petrosino, Mattia Quadrini, Cesare Roseti, Domenico Striccoli, Francesco Zampognaro, Michele Luglio, Stefano Perticaroli, Antonio Mosca, Francesco Lombardi, et al. Looking at nb-iot over leo satellite systems: Design and evaluation of a service-oriented solution. *IEEE Internet of Things Journal*, 9(16):14952–14964, 2021.

SV Shestov, AN Zhukov, Bernd Inhester, L Dolla, and M Mierla. Expected performances of the proba-3/aspiics solar coronagraph: simulated data. *Astronomy & Astrophysics*, 652:A4, 2021.

Maksim Shirobokov, Sergey Trofimov, and Mikhail Ovchinnikov. Survey of machine learning techniques in spacecraft control design. *Acta Astronautica*, 186:87–97, 2021.

Monika Singh, Chetan Sharma, Trapty Agarwal, and Meenakshi Shruti Pal. Forest fire prediction for nasa satellite dataset using machine learning. In *2022 10th International Conference on Reliability, Infocom Technologies and Optimization*, pp. 1–5. IEEE, 2022.

Minhyuk Song, Sungwon Han, Seungeon Lee, Donghyun Ahn, Jihee Kim, and Meeyoung Cha. Measuring fine-grained urban air temperature with satellite imagery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 28397–28404, 2025.

Zvonimir Stojanovski and Dmitry Savransky. The nonsingular estimator for exoplanet orbits: An unscented batch estimation method for direct imaging measurements. *The Astronomical Journal*, 168(1):40, 2024.

Jordan Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

Kathiravan Thangavel, Roberto Sabatini, Alessandro Gardi, Kavindu Ranasinghe, Samuel Hilton, Pablo Servidia, and Dario Spiller. Artificial intelligence for trusted autonomous satellite operations. *Progress in Aerospace Sciences*, 144:100960, 2024.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, KG Arjun, Rodrigo Perez-Vicente, Andrea Pierr'e, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *ArXiv*, abs/2407.17032, 2024.

David Vallado, Paul Crawford, Ricahrd Hujsak, and T.S. Kelso. Revisiting spacetrack report# 3. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, pp. 6753, 2006.

William C. Wagner, Christopher S. R. Neigh, David E. Shean, Paul M. Montesano, Tiangang Yin, and Ameni Mkaouar. Vegetation height stereo reconstruction with blacksky commercial frame camera imagery. In *IEEE International Geoscience and Remote Sensing Symposium*, pp. 2446–2450, 2024.

Alexis Wall. Systems tool kit (stk). In *Space System Architecture Analysis and Wargaming*, pp. 11–32. CRC Press, 2024.

Hao Wang, Zhi-yuan Wang, Ben-dong Wang, Zhong-he Jin, and John L Crassidis. Infrared earth sensor with a large field of view for low-earth-orbiting micro-satellites. *Frontiers of Information Technology & Electronic Engineering*, 22(2):262–271, 2021a.

Peng Wang, Hongyan Li, Binbin Chen, and Shun Zhang. Enhancing earth observation throughput using inter-satellite communication. *IEEE Transactions on Wireless Communications*, 21(10): 7990–8006, 2022.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning Roles to Decompose Multi-Agent Tasks. In *International Conference on Learning Representations (ICLR)*, 2021b.

Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. Agile earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal*, 15 (3):3881–3892, 2021c.

Annie Wong, Thomas Bäck, Anna V Kononova, and Aske Plaat. Deep multiagent reinforcement learning: challenges and directions. *Artificial Intelligence Review*, 56(6):5023–5056, 2023.

Guohua Wu, Qizhang Luo, Yanqi Zhu, Xinjiang Chen, Yanghe Feng, and Witold Pedrycz. Flexible task scheduling in data relay satellite networks. *IEEE Transactions on Aerospace and Electronic Systems*, 58(2):1055–1068, 2022.

Yuchen Xiao, Weihao Tan, and Christopher Amato. Asynchronous actor-critic for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4385–4400, 2022.

Jia Ye, Gaofeng Pan, and Mohamed-Slim Alouini. Earth rotation-aware non-stationary satellite communication systems: Modeling and analysis. *IEEE Transactions on Wireless Communications*, 20(9):5942–5956, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 2022.

Bing Zhang, Yuanfeng Wu, Boya Zhao, Jocelyn Chanussot, Danfeng Hong, Jing Yao, and Lianru Gao. Progress and challenges in intelligent remote sensing satellite systems. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:1814–1822, 2022.

Kaiqing Zhang, Tao Sun, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. Robust multi-agent reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 33:10571–10583, 2020.

Shun Zhou, Hongbo Pan, Tao Huang, and Ping Zhou. High accuracy georeferencing of gf-6 wide field of view scenes toward analysis ready data. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–12, 2023.

Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. Decentralized, decomposition-based observation scheduling for a large-scale satellite constellation. *Journal of Artificial Intelligence Research*, 82:169–208, 2025.

DECLARATION OF LLM USAGE

The core development of SatBench does not involve LLMs as essential, original, or non-standard components. GPT5 has only been used to assist in proofreading tasks, including polishing text, correcting grammar and spelling errors, and ensuring consistency in American English spelling.

A ENVIRONMENT SETUP

This section presents further details of the SatBench simulation environment. It outlines the configuration and propagation of satellite orbits, the modeling of satellite rotation control and formation flying method, and the representation and placement of ground targets on Earth.

A.1 ORBIT

An orbit is the curved trajectory that a satellite must follow to maintain its operational path. In SatBench, we adopt the Classical Orbital Elements (COEs) framework (Stojanovski & Savransky, 2024), a standard six-parameter model for representing orbital dynamics. Specifically, an orbit is described by the tuple $(a, e, i, \Omega, \omega, \nu_0)$, where each parameter defines a unique aspect of the orbit: $e \in [0, 1)$ is the eccentricity, defining the orbit's shape from circular ($e = 0$) to elliptical ($0 < e < 1$); $i \in [0°, 180°]$ is the inclination, representing the tilt of the orbital plane relative to Earth's equator. $\Omega \in [0°, 360°)$ is the right ascension of the ascending node, which specifies the orientation of the ascending node in the equatorial plane. $\omega \in [0°, 360°)$ is the argument of perigee, indicating the orientation of the orbital ellipse within its plane. $\nu_0 \in [0°, 360°]$ is the true anomaly at each epoch, specifying the satellite's initial position along its orbit at the start of the operation.

A.2 SATELLITE

SatBench models satellite dynamics through two key components: rotational control for attitude adjustments to support EO missions and satellite formation for coordinated constellation operations. For EO tasking, each satellite dynamically adjusts its orientation attitude using rotational control techniques to align with and capture ground targets, maintaining a field of view (FOV) (Zhou et al., 2023) within $\pm 15°$. For coordinated operations, SatBench simulates satellite formation flying (Thangavel et al., 2024), allowing multiple satellites to maintain relative positioning. This enables collaborative observation of shared targets or broader area coverage with an increased number of targets.

Rotation control. Each satellite in SatBench employs a rotation control method for attitude adjustments along three rotational axes-`pitch`, `yaw`, and `roll` (Crisp et al., 2018). At each time step, the satellite selects a desired pointing direction, and a proportional-derivative controller computes the corresponding control torque. This torque employs internal reaction wheels to reorient the satellite toward the target direction. To reflect practical agility constraints, the control torque is capped at $\boldsymbol{u}_{\max} = 0.2$ Nm, which limits the rotation rate to approximately $1°$ every 4 seconds. These parameters are chosen to ensure that satellite behavior in the simulation remains consistent with real-world physical capabilities.

Formation flying. SatBench supports three types of satellite formations for coordination, defined based on six orbital parameters as detailed in Appendix A.1. To construct a satellite formation flying, suitable orbital parameters must first be assigned to each satellite. This typically follows a common orbital baseline: an equatorial orbit (Marshall et al., 2021). An equatorial circular orbit is characterized by an inclination $i = 0°$, right ascension of the ascending node $\Omega = 0°$, and argument of perigee $\omega = 0°$, ensuring that the orbital plane lies on the Earth's equatorial plane. The semi-major axis and eccentricity are generally set to $a = 6798.3$ km and $e = 0.000691$, respectively. These default values can be modified to reflect different scenarios as needed. Based on this configuration, SatBench defines three types of satellite formations, as described below, with their corresponding orbital parameters summarized in Table 4.

Table 4: Defined orbital parameters for the three satellite formations in SatBench

Satellite	a	e	i	Ω	ω	ν_0	**Formation**
Sat1	6798.3	0.000691	0	0	0	75.1185°	Cluster
Sat2	6798.3	0.000691	0	0	0	75.12695°	Cluster
Sat1	6798.3	0.000691	3°	0	0	0	Trailing; Constellation
Sat2	6798.3	0.000691	3°	0	0	180°	Trailing; Constellation
Sat3	6798.3	0.000691	-3°	0	0	0	Trailing; Constellation
Sat4	6798.3	0.000691	-3°	0	0	180°	Trailing; Constellation
Sat1	6798.3	0.000691	2°	0	0	0	Constellation
Sat2	6798.3	0.000691	-2°	0	0	0	Constellation
Sat3	6798.3	0.000691	4°	0	0	0	Constellation
Sat4	6798.3	0.000691	-4°	0	0	0	Constellation
Sat5	6798.3	0.000691	6°	0	0	0	Constellation
Sat6	6798.3	0.000691	-6°	0	0	0	Constellation

- *Cluster*: Satellites share the same orbital parameters and are placed in a single orbit, differing only slightly in their true anomaly (ν_0), resulting in close spatial proximity and highly overlapping visibility regions, enabling dense coverage of a localized area.
- *Trailing*: Satellites occupy the same orbital plane but are significantly spaced apart in true anomaly (ν_0), introducing temporal diversity for observations of the same area over time.
- *Constellation*: Satellites are distributed across different orbital planes by varying the inclination (i), enabling broader geographic coverage and improved revisit frequency.

A.3 TARGET

A set of targets T is defined as objectives for satellite operation goals in Earth observation (EO) taking. In SatBench, each target $j \in T$ is represented by a fixed three-dimensional Cartesian position vector $\mathbf{p}_j \in \mathbb{R}^3$ in an Earth-centered, Earth-fixed coordinate system (Ye et al., 2021). This vector denotes the target's location on the Earth's surface. In addition, each target is associated with a scalar priority score $p_j \in [0, 1]$, where higher values indicate greater importance for EO tasking (a score of 1 represents the highest priority). SatBench supports two types of target distributions, as detailed below.

- **Sparse Distribution.** Targets are uniformly distributed along the satellite's ground track, with large spatial gaps between consecutive locations (e.g., on the order of hundreds of kilometers). Due to this wide separation, it is likely that only a single target falls within a satellite's FOV at any given time. Consequently, each satellite typically corresponds to capturing one target. In this scenario, the level of inter-satellite coordination is low.
- **Dense Distribution.** Multiple targets are grouped into compact areas positioned along the satellite's ground track. Within each group, targets are placed in close proximity, often resulting in multiple targets falling within a single FOV. This spatial concentration increases the likelihood of redundant observations when multiple satellites pass over the same region. As a result, dense target distributions demand a high level of coordination between satellites to avoid overlapping observation coverages.

B DEFINE SATELLITE AS AN AGENT

This section provides further details on how SatBench models each satellite as a reinforcement learning (RL) agent and coordinates them within a multi-agent reinforcement learning (MARL) framework. We define the satellite agent model by detailing its three core components: observation, action, and reward.

B.1 OBSERVATION

At each time step, a satellite agent receives a compact observation consisting of (i) its own satellite status and (ii) a filtered list of observable targets that fall within its sensing range.

Formally, each satellite agent i receives the observation:

$$o_i = \left[s_i,\ \{s_j\}_{j \in T_i^{\text{vis}}} \right],$$

where environmental state $s_i = [\mathbf{p}_i,\ \mathbf{d}_i,\ \boldsymbol{\omega}_i,\ \mathbf{v}_i,\ \boldsymbol{\omega}_{\text{rw},i}^{\max},\ \theta_i^{\text{fov}}]$ (Ye et al., 2021; Fan et al., 2022), including the satellite's orbital position $\mathbf{p}_i \in \mathbb{R}^3$, pointing direction $\mathbf{d}_i \in \mathbb{R}^3$, angular velocity $\boldsymbol{\omega}_i \in \mathbb{R}^3$, linear velocity $\mathbf{v}_i \in \mathbb{R}^3$, maximum reaction wheel speed $\boldsymbol{\omega}_{\text{rw},i}^{\max} \in \mathbb{R}^3$, and field-of-view half-angle $\theta_i^{\text{fov}} \in [0°, 180°]$. The set of targets is defined as T_i^{vis}. Each satellite agent also receives a target observation $s_j = [t_j^{\text{open}},\ t_j^{\text{close}},\ \angle_j,\ \mathbf{p}_j,\ p_j]$, where $(t_j^{\text{open}},\ t_j^{\text{close}})$ denotes the estimated access window during which the target is imageable, $\angle_j$ is the angle between the satellite's pointing direction and the target, $\mathbf{p}_j \in \mathbb{R}^3$ is the target's geodetic position, and p_j is its priority.

Only a subset of targets is visible to the satellite agent, which is determined by the following geometric constraints (Ding et al., 2022):

- the elevation angle is above $35°$;
- the slant range is less than 1000 km;
- the line of sight is not occluded by the Earth.

B.2 ACTION

The action of each satellite agent i is designed to adjust its orientation $\mathbf{a}_i \in A_i$ to align with a target on the ground, as detailed in Appendix A.2. In SatBench, this rotation control is modeled as an action vector over the frame:

$$\mathbf{a}_i = \delta \cdot [r_{\text{pitch}},\ r_{\text{yaw}},\ r_{\text{roll}}] \in A_i, \quad \text{with } r_{\text{pitch}}, r_{\text{yaw}}, r_{\text{roll}} \in \{-1,\ 0,\ +1\}$$

where δ is the fixed per-axis rotation increment. The full action set A_i includes all 27 possible combinations of actions across the three rotational axes. By default, δ is set to $1°$, which provides an approximation of continuous rotational control while preserving computational tractability. To modify δ and achieve finer or coarser control, simply modify the `ImagingFSWModel` class. This class controls the `angle_step` of each satellite's attitude control and is located at `SatBench/src/envs/bsk_rl/src/bsk_rl/sim/fsw`. Any such change must comply with the satellite's control torque limits, as detailed in A.2. A detailed example demonstrating how to modify `angle_step` of a satellite is illustrated below:

```
class SatBenchContinuousImagingFSWModel(ImagingFSWModel):
    ...
    @action
    def action_image(self, action: int) -> None:
        """Attempt to image a target at a location with attitude
        ↪  control.

        Args:
            action: Integer representing different rotations.
        """

        self.insControl.controllerStatus = 1
        self.dynamics.instrumentPowerSink.powerStatus = 1
        # Step 1: Define rotations
        angle_step = 1  # Rotate by 1° each time.
```

B.3 REWARD

A satellite agent receives a reward only when it successfully completes the rotation required to capture a target on the ground. The reward value is weighted by the priority score of the target, reflecting its relative importance for EO tasking.

Specifically, a target j is considered successfully captured at timestep t if there exists at least one satellite agent i such that:

1. $j \in T_{i,t}^{\text{vis}}$: the target is physically visible to satellite agent i
2. $j \notin T_t^{\text{done}}$: the target has not been successfully captured in any previous timesteps.

Once target j satisfies these conditions at time t, then $T_t^{\text{done}} = T_{t-1}^{\text{done}} \cup \{j\}$ means target j marked as successfully imaged and will be added to the completed set and its reward p_j is granted to all satellite agents. This reward at time t is then given by: $r_t = \sum_{j \in T_t^{\text{new}}} p_j$, where $T_t^{\text{new}} = T_t^{done} \setminus T_{t-1}^{done}$ is the set of targets captured at time t, and $p_j \in (0, 1]$ is the priority score of target j.

The overall objective for all satellite agents is to maximize the cumulative reward $\sum_t r_t$, which reflects the total prioritized coverage of targets within a single episode.

C MARL TRAINING

This section outlines the detailed experimental settings used to train and evaluate MARL algorithms in SatBench. All MARL training configurations are kept consistent across algorithms and scenarios to ensure fair comparisons.

C.1 HYPERPARAMETER SETTINGS

To ensure performance consistency, all training runs across different MARL algorithms follow an identical learning pipeline, with key hyperparameters summarized in Table 5. IPPO and MAPPO share most configuration settings, while HAPPO differs by employing a non-parameter-sharing strategy for its actor networks. Each algorithm is trained using 4 random seeds, with training metrics logged every 50,000 steps. Each episode simulates a complete satellite orbital period of 6,300 seconds.

Table 5: Key hyperparameters for MARL training in SatBench.

Category	**Parameter**	**IPPO**	**MAPPO**	**HAPPO**
Core Selection	Action Selector	soft_policies	soft_policies	soft_policies
	Runner	parallel	parallel	parallel
	Use RNN	True	True	True
	Buffer Size	32	32	32
	Batch Size	32	32	32
Observation Setup	Obs Agent ID	True	True	False
	Obs Last Action	False	False	False
	Obs Individual Obs	False	False	False
	Agent Output Type	pi_logits	pi_logits	pi_logits
Optimization & Training	Learning Rate	3e-4	3e-4	3e-4
	Entropy Coef	0.1	0.1	0.1
	Epochs per Update	5	5	5
	Clip ϵ	0.2	0.2	0.2
	Use Feature Norm	—	—	True
	Use Huber Loss	—	—	True
	Huber δ	—	—	10.0
	Value Loss Coef	—	—	1
	Standardise Rewards	True	True	True
	Gamma (γ)	0.99	0.99	0.99
	GAE Lambda (λ)	0.95	0.95	0.95

All experiments were conducted on a workstation equipped with an NVIDIA RTX 4090 GPU and an AMD EPYC 7662 CPU.

C.2 Environment Rollout Time

Understanding the computational scalability of a simulation environment is crucial, especially as the number of agents and task complexity increase. We analyze the environment rollout time per episode to demonstrate that SatBench remains computationally tractable, even for large satellite constellations. The experiments were conducted on an AMD Ryzen 7 7800X3D processor.

Table 6: Environment rollout time per episode in seconds, averaged over 3 runs per setting

Satellites	**Targets**			
	100	**200**	**300**	**400**
2	3.06 ± 0.05	3.10 ± 0.03	3.13 ± 0.07	3.05 ± 0.03
4	5.34 ± 0.15	5.36 ± 0.04	5.39 ± 0.09	5.34 ± 0.10
6	9.87 ± 1.44	9.79 ± 0.17	10.03 ± 0.09	10.07 ± 0.07
50	84.61 ± 3.24	83.30 ± 0.90	84.10 ± 1.10	83.68 ± 0.68

Table 7: MARL training time (in hours) across 6 SatBench scenarios.

Algorithm	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**Scenario 5**	**Scenario 6**
IPPO	19.9	17.8	40.6	37.8	84.8	72.4
MAPPO	17.2	13.7	41.6	38.3	106.2	79.1
HAPPO	23.8	20.4	45.1	39.6	87.8	81.4

Table 6 presents the environment rollout time per episode with varying numbers of satellites (agents) and targets (task density). A key finding is that the number of targets (columns 100 to 400) has a minimal impact on the environment step time. For any fixed number of satellites, the mean rollout time remains highly consistent across all target densities, indicating the environment's complexity is not heavily dominated by task density.

Conversely, the primary scaling factor is the number of agents (satellites). This increase in computational cost is due to the increased requirement for physical modeling, including the calculation of orbital propagation, inter-agent visibility checks, and individual satellite dynamics. Crucially, the data confirms that the overall computation time, driven by the number of satellites, scales approximately linearly. This pattern is also observed in Table 7, where the MARL training time scales linearly as the number of satellites increases. This linear scaling is evident when comparing the different constellation sizes, confirming that SatBench remains computationally tractable even as constellation size increases. This enables its practical use in both academia and industry.

D Customized Modular Design

This section details the implementation of several key features within the SatBench environment, highlighting the modules that allow researchers to perform advanced customization for testing and discovering different scenarios related to communication, uncertainty, heterogeneous agents and asynchronous control in MARL research.

D.1 Coordination Paradigms

D.1.1 Communication Module

SatBench incorporates a dedicated communication module located at `SatBench/src/envs/bsk_rl/src/comm`. This module provides built-in base classes, such as `FreeCommunication` and `LineOfSightCommunication`, designed to manage inter-agent communication constraints.

For researchers interested in communication-aware MARL or analyzing space systems communication, this component is readily extensible for custom designs and evaluation.

A simple example demonstrating paired communication using the `FreeCommunication` class is shown below, illustrating its function to generate all possible pairs for communication at every time step:

```
class FreeCommunication(CommunicationMethod):
    """Implements free communication between every satellite at every
    ↪  step."""

    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    def communication_pairs(self) -> list[tuple["Satellite",
    ↪  "Satellite"]]:
        """Return all possible communication pairs."""
        return list(combinations(self.satellites, 2))
```

D.2 Heterogeneous Satellites

The satellites in the constellation could have different FOVs, which result in different capabilities and are common in EO missions (Wang et al., 2021a). SatBench supports heterogeneous satellites with different FOVs in the `ImagingFSWModel` class, which is located at `SatBench/src/envs/bsk_rl/src/bsk_rl/sim/fsw`, by simply adjusting the `attErrTolerance` parameter. For example, to set up a heterogeneous satellite scenario with 2 different types of FOVs, simply create two different `ImagingFSWModel` classes:

```
class SatBenchContinuousImagingFSWModel15degree(ImagingFSWModel):
    ...
    class CustomImageTask(Task):
    ...
        def update(self):
            """check if a target is within FOV"""

            attErrTolerance = np.radians(15)

class SatBenchContinuousImagingFSWModel5degree(ImagingFSWModel):
    ...
    class CustomImageTask(Task):
    ...
        def update(self):
            """check if a target is within FOV"""

            attErrTolerance = np.radians(5)
```

Once the two FSW models are created, allocate and link them to satellites in the environment wrapper `Bsk_wrapper`, which is located at `SatBench/src/envs/bsk_rl/src/bsk_rl/Bsk_wrapper`. Specifically:

```
class CustomSatComposed1(sats.ImagingSatellite):
    ...
    dyn_type = CustomDynModel
    fsw_type = fsw.UniqueImagerFSWModel15

class CustomSatComposed2(sats.ImagingSatellite):
    ...
    dyn_type = CustomDynModel
    fsw_type = fsw.UniqueImagerFSWModel5
```

Appendix F.2 reports experiments conducted with heterogeneous satellites.

D.3 Noise and Uncertainty Module

The default noise-free observations setting in SatBench follows the standard precedent in many commonly used MARL environments (Lowe et al., 2017; Samvelyan et al., 2019), which prioritize deterministic baselines but allow for user-defined stochasticity via wrappers or configuration (Zhang et al., 2020; Phan et al., 2023). SatBench supports the injection of uncertainty through a standardized API, allowing researchers to study algorithmic robustness by injecting:

- **Observation Noise** (Phan et al., 2023): Satellite's observations can be corrupted, simulating noisy sensors (Ditmar et al., 2012).
- **Model Uncertainty** (Zhang et al., 2020): The reward signal from the environment can be made stochastic due to external factors, such as unpredictable cloud coverage (Han et al., 2022).

For example, observation noise can be implemented directly within the BSK wrapper (e.g., at `Satbench/src/envs/bsk_wrapper`) as follows:

```
def get_obs_agent(self, agent_id):
    default_obs =
    ↪  self.env._get_obs()[self.env.possible_agents[agent_id]]
    if self.noise_std > 0:
        noise = np.random.normal(loc=0.0, scale=self.noise_std,
        ↪  size=default_obs.shape)
        default_obs = default_obs + noise
    return default_obs
```

This design enables a comprehensive analysis of algorithm robustness under various uncertain conditions relevant to Earth Observation (EO) missions.

D.4 Asynchronous Control Support

SatBench supports asynchronous multi-agent settings through the `Bsk_wrapper` file, which specifies the `duration` of each satellite's decision step (`SatBench/src/envs/bsk\_rl/src/bsk\_rl/Bsk\_wrapper`). By configuring this parameter individually for different satellites, researchers can define customized temporal action intervals, thereby enabling realistic modeling of asynchronous coordination scenarios. For example, two satellites may be configured to execute actions at different time steps, such as 4 s and 8 s, respectively.

```
class CustomSatComposed1(sats.ImagingSatellite):
    ...
    action_spec = [
        act.Image(n_ahead_image=27,duration=4),
    ]

class CustomSatComposed2(sats.ImagingSatellite):
    ...
    action_spec = [
        act.Image(n_ahead_image=27,duration=8),
    ]
```

Such heterogeneous configurations allow for asynchronous action execution without requiring further modifications to the core SatBench architecture. Furthermore, federated learning approaches have demonstrated effectiveness in reinforcement learning with asynchronous settings(Lan et al., 2024).

D.5 Task-Assignment Inspired Approaches

SatBench can accommodate task-assignment-style planning, where agents are dynamically assigned targets and must decide how to image them over a long horizon. For instance, REDA (Holder et al., 2025) represents an RL-enabled greedy LP-style planner for discrete target allocation, which provides strong baseline performance for classic assignment problems. While such LP-style algorithms excel

at efficient discrete allocation, SatBench differs fundamentally: its focus is on continuous pointing and imaging control, where each satellite must continuously adjust its attitude to track targets, handle imaging constraints, and optimize long-horizon multi-agent coordination.

To support task-assignment-style strategies in SatBench, the core flight software module `ImagingFSWModel` (located at `SatBench/src/envs/bsk_rl/src/bsk_rl/sim/fsw`) must be substantially modified. The main difference is that instead of directly specifying continuous attitude commands, the agent now issues target-based actions, which the FSW translates into continuous pointing commands. Below is an example of how to achieve such action in SatBench:

```
def action_image(self, target_pos: Iterable[float], data_name: str) ->
↪  None:
    """Attempt to image a target at a specified target location.

    This example demonstrates converting a target-based action into
    continuous attitude commands. The FSW calculates the necessary
    pointing attitude to track the target and enable imaging.

    Args:
        target_pos: [m] Planet-fixed target position.
        data_name: Data buffer to store image data to.
    """
    self.insControl.controllerStatus = 1
    self.dynamics.instrumentPowerSink.powerStatus = 1
    self.dynamics.imagingTarget.r_LP_P_Init = r_LP_P
    self.dynamics.instrument.nodeDataName = data_name
    self.insControl.imaged = 0
    self.simulator.enableTask(self.LocPointTask.name +
    ↪  self.satellite.name)
```

D.6 Scenario Customization

SatBench supports scenario customization based on specific research needs through the `env_args` configuration block, which defines the key parameters for scenario setup (note that Appendix E.4 details the location and structure of the `env_args` block within the SatBench repository). Here, we provide a sample configuration snippet demonstrating how to build up a dense-target, two-satellite cluster scenario:

```
env_args:
  map_name: Scenario2
  Target_type: DenseTarget
  Num_targets: 70
  Target_density: 400000
  rewarder: UniqueImageReward
  Sat_orb_param: 2SatCluster.xlsx # Determine COEs for different
  ↪  formations
  Target_param: Default # Automatically generate targets along orbit
```

In this configuration, `map_name` specifies the scenario identifier. The `Target_type` field defines the spatial distribution pattern of ground targets, with supported options including `DenseTarget` and `SparseTarget`. The parameters `Num_targets` and `Target_density` control the total number of targets and their spatial concentration, respectively. The `rewarder` parameter determines the reward function; the default, `UniqueImageReward`, grants rewards only for non-redundant target observations. The `Sat_orb_param` field specifies an input file (e.g., `2SatCluster.xlsx`) that defines the satellites' orbital elements, allowing researchers to simulate various satellite formations and quantities. Finally, `Target_param` governs target generation, defaulting to automatic placement along the orbit.

This flexible configuration interface enables straightforward customization of SatBench benchmark scenarios. Once customized, SatBench automatically loads the specified configuration and initializes the corresponding simulation environment for MARL training.

For scalability of SatBench, mean-field MARL provides an alternative framework for scalable coordination in multi-agent systems, such as sampling-based mean-field variants that estimate this mean locally, enabling fully decentralized execution under partial observability (Anand & Qu, 2024; Anand et al., 2024). For EO tasks, such approaches could allow satellites to coordinate efficiently in medium- to large-scale constellations without relying on full global knowledge, improving scalability and training stability. SatBench's modular design supports such extensions, allowing the learning interface to be adapted for mean-field RL methods while keeping the simulation backend unchanged, facilitating future research on alternative coordination strategies without compromising fidelity or reproducibility.

E SatBench Source Code and Usage Guide

We release SatBench source code, including environment modules, scenario configuration files, and pipelines for training and evaluating MARL algorithms. The full repository is available at: `https://anonymous.4open.science/r/SatBench-43DB`.

E.1 Environment setup guide

This section provides a step-by-step guide for setting up the SatBench environment.

Step 1: Set Up the Python Environment. We recommend creating an isolated Conda environment to avoid dependency conflicts:

```
conda create -n SatBench python=3.10 -y
conda activate SatBench
```

Step 2: Install the Foundation Engine. Clone the Basilisk repository and install its dependencies:

```
git clone https://github.com/AVSLab/basilisk.git
sudo apt-get install swig
cd basilisk
pip install -r requirements_dev.txt
python3 conanfile.py
```

Step 3: Install SatBench and MARL Dependencies. Install SatBench in editable mode, and then install required MARL libraries:

```
cd ../../../SatBench/src/envs/bsk_rl
python -m pip install -e "." && finish_install
```

```
cd ../../../SatBench # Return to SatBench root directory
pip install -r requirements.txt
```

E.2 Training Scripts Guide

Training scripts are provided for MARL algorithms deployed in SatBench, using a unified command-line interface:

```
python src/main.py --config=algo --env-config=scenario
```

Example:

```
python src/main.py --config=mappo --env-config=Scenario1
```

E.3 DEMO VIDEO

We provide two complementary visualization modes that facilitate the interpretation of MARL strategies in SatBench. At a high-fidelity level, satellites can be rendered as realistic 3D models with dynamic fields of view, allowing users to directly observe how agents adjust their attitudes to bring targets into view (Figure 5). This mode highlights the physical feasibility of learned control policies and illustrates how local actions translate into realistic satellite maneuvers. At a more abstract level, we can visualize the mapping between satellites and targets (Figure 6), providing a concise overview of which agents image which targets at any given time. This abstraction makes it easier to interpret coordination patterns and evaluate the effectiveness of cooperative strategies. Together, these two modes offer both fine-grained and high-level perspectives, supporting comprehensive analysis of MARL behavior in EO scenarios.

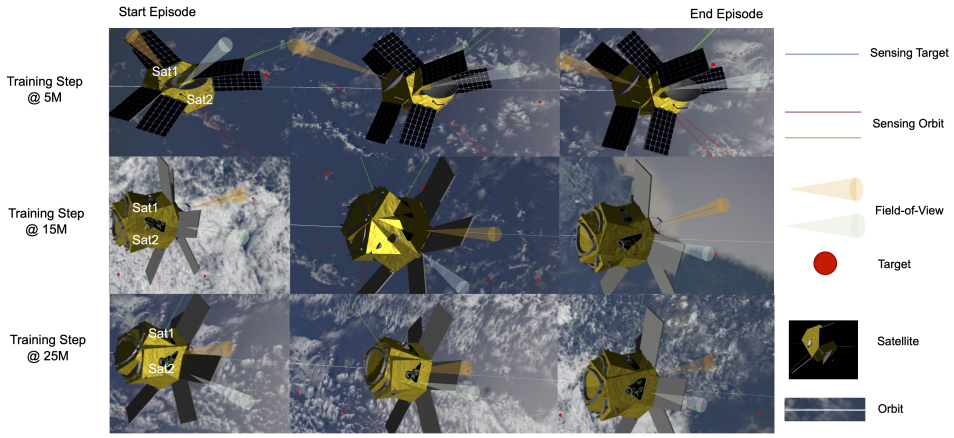


Figure 5: High-fidelity visualization of realistic satellite maneuvers with dynamic attitude and field-of-view changes, illustrating how agents adjust their orientation to image targets.

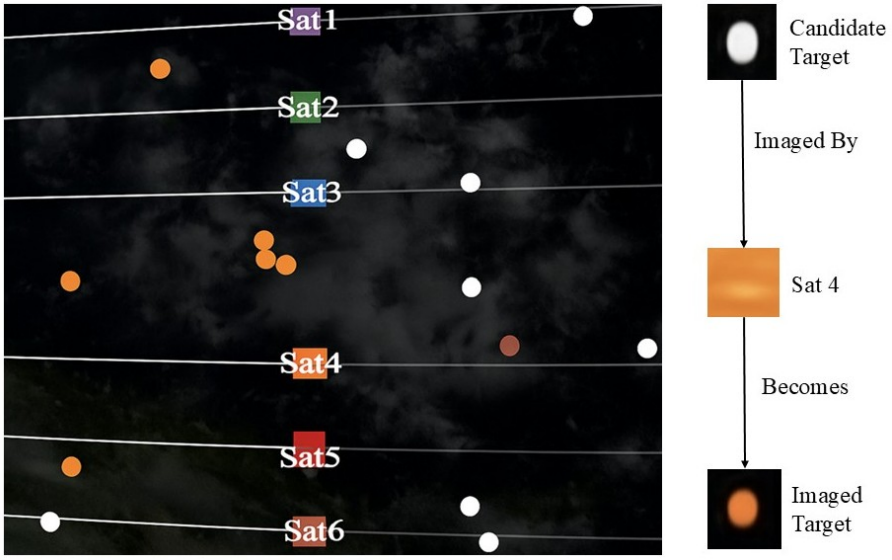


Figure 6: Abstract visualization of task allocation, showing which satellites successfully image which targets, enabling analysis of coordination patterns and coverage.

E.4 CONFIGURATION GUIDE

SatBench uses a modular configuration structure. Each component of the experiment environment, algorithm, and training setup is defined in a separate $YAML$ file:

- `src/configs/envs/`: Defines scenario-specific environment settings (e.g., number of satellites, orbit file, and target distribution).
- `src/configs/algo/`: Specifies algorithm-specific hyperparameters (e.g., learning rate and entropy).
- `src/configs/default/`: Contains launcher scripts that load the selected environment and algorithm configurations to start training.

Logging record and satellite visualization guide. SatBench supports both Weights & Biases (Wandb) and TensorBoard for logging during training and evaluation. Users can enable either option by setting the corresponding flag to `True` in the configuration file located at `src/configs/default/`.

SatBench also provides a rendering option for visualizing MARL-driven satellite behaviors. To enable this feature, set `render: True` in the appropriate configuration file under `src/configs/envs/`. When enabled, SatBench generates a `.viz` file that can be viewed using an external Vizard tool, which is available for download at `https://hanspeterschaub.info/bskFiles/Vizard_Windows64.zip` for Windows and `https://hanspeterschaub.info/bskFiles/Vizard_Linux.zip` for Linux OS.

E.5 LIST OF DEPENDENCIES

Table 8 lists the main open-source libraries used in SatBench and their licenses. All dependencies are installable via `pip` and specified in the environment file provided with the code.

Table 8: Key software licenses used in our implementation.

Software	License	License Link
numpy	BSD 3-Clause	https://numpy.org/doc/stable/license.html
scipy	BSD 3-Clause	https://github.com/scipy/scipy/blob/main/LICENSE.txt
matplotlib	BSD-compatible	https://matplotlib.org/stable/project/license.html
pandas	BSD 3-Clause	https://github.com/pandas-dev/pandas/blob/main/LICENSE
PyYAML	MIT	https://github.com/yaml/pyyaml/blob/main/LICENSE
torch	BSD 3-Clause	https://github.com/pytorch/pytorch/blob/main/LICENSE
torchvision	BSD 3-Clause	https://github.com/pytorch/vision/blob/main/LICENSE
gym	MIT	https://github.com/openai/gym/blob/master/LICENSE.md
Stable-Baselines3	MIT	https://github.com/DLR-RM/stable-baselines3/blob/master/LICENSE
wandb	MIT	https://github.com/wandb/wandb/blob/main/LICENSE
sacred	MIT	https://github.com/IDSIA/sacred/blob/master/LICENSE
basilisk	MIT	https://github.com/AVSLab/basilisk/blob/master/LICENSE
bsk_rl	MIT	https://github.com/AVSLab/bsk_rl/blob/main/LICENSE

F ADDITIONAL EXPERIMENTS AND RESULTS

We conducted three additional sets of experiments. First, we benchmarked two commonly used MARL baselines, COMA (Foerster et al., 2018) and QMIX (Rashid et al., 2020), to evaluate their performance under the same scenario settings. Second, we introduce EO-constrained cooperative tasks, which emphasize temporal synchronization and on-demand coordination. Third, we introduced single-satellite scenarios to assess how various single-satellite agents based on RL algorithms perform in settings where inter-agent coordination is not required. Together, these experiments provide a more comprehensive view of how RL and MARL methods handle the inherent difficulty of the SatBench environment and the challenges posed by different scenario configurations.

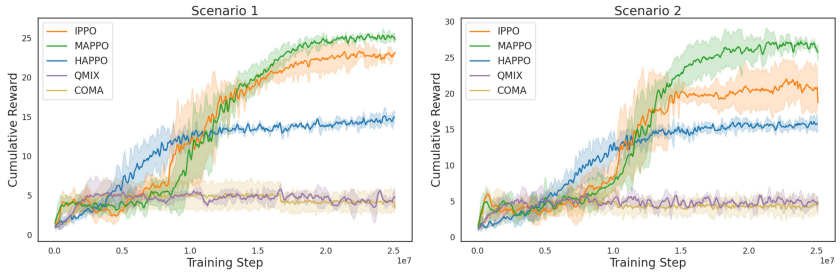
F.1 MARL Performance With Homogeneous satellites

SatBench supports a variety of MARL algorithms. In our main experiments, we focus on PPO-based methods in multi-agent settings, which have been widely shown to perform well in environments with continuous control, stochastic dynamics, and sparse rewards (Wong et al., 2023). These methods learn explicit action distributions and adaptively balance exploration and exploitation, making them particularly effective for long-horizon decision-making.

Although the action space in SatBench is technically discrete, it closely approximates continuous control due to fine-grained rotational increments across three attitude axes (pitch, yaw, and roll). As described in Appendix B.2, successfully capturing a ground target often requires a long sequence of coordinated actions, typically spanning 10 to 100 timesteps. This poses challenges for credit assignment and effective exploration in MARL, which policy gradient methods are generally better equipped to address.

To further validate this design choice, we additionally benchmark two widely used MARL baselines, QMIX (Rashid et al., 2020) and COMA (Foerster et al., 2018), on the simpler cluster scenarios (Scenarios 1 and 2). These experiments enable focused comparison under lower coordination complexity.

Results and Analysis. As shown in Figure 7 and Table 9, both QMIX and COMA consistently underperformed across all timesteps in Scenarios 1 and 2. Their completion rates remain low throughout training, with limited improvement beyond the early stages. For example, in Scenario 1, COMA peaks at 16.4% at 15M steps before declining, while QMIX fluctuates between 14% and 18% without a clear upward trend. A similar pattern is observed in Scenario 2, where both methods fail to exceed 16% even at 25M steps.



(a)1Orb_2Sat_Cl_60Tgt_Sp (b)1Orb_2Sat_Cl_70Tgt_Dn

Figure 7: Training performance of QMIX and COMA compared to IPPO, MAPPO, and HAPPO in Scenarios 1 and 2.

Table 9: MARL training performance measured by task completion rate (%) across the first two SatBench scenarios. All values are reported as mean ± standard deviation over 4 random seeds. Best performance in each scenario is highlighted in bold.

Scenario	Algorithm	Training Timesteps				
		1M	5M	10M	15M	25M
Scenario 1	IPPO	5.0 ± 1.8	11.0 ± 1.2	40.1 ± 17.4	62.4 ± 5.9	77.3 ± 1.1
	MAPPO	5.2 ± 1.6	13.1 ± 3.9	31.6 ± 9.7	67.5 ± 4.3	**83.4 ± 1.3**
	HAPPO	3.6 ± 0.9	23.0 ± 8.2	41.3 ± 1.9	46.8 ± 3.0	49.0 ± 2.2
	COMA	4.6 ± 2.9	14.5 ± 5.5	16.0 ± 5.4	16.4 ± 6.0	14.0 ± 3.5
	QMIX	3.1 ± 0.2	18.9 ± 5.0	17.2 ± 0.2	14.7 ± 0.8	15.7 ± 2.5
Scenario 2	IPPO	5.0 ± 1.5	10.9 ± 3.7	24.7 ± 5.3	57.9 ± 4.0	59.6 ± 8.0
	MAPPO	4.5 ± 1.7	14.5 ± 5.0	22.1 ± 3.4	64.3 ± 8.4	**73.4 ± 3.1**
	HAPPO	3.4 ± 0.7	15.9 ± 4.8	39.1 ± 4.6	42.8 ± 3.8	44.5 ± 3.7
	COMA	5.0 ± 0.9	14.8 ± 4.2	13.6 ± 5.7	11.5 ± 3.7	11.4 ± 2.6
	QMIX	3.4 ± 1.0	15.3 ± 3.5	15.4 ± 3.6	13.2 ± 0.9	12.1 ± 2.3

In comparison to PPO-based methods such as IPPO, MAPPO, and HAPPO, the learning progress of COMA and QMIX is significantly less stable. Moreover, their final performance is consistently lower, particularly in Scenario 2, where the denser target distribution imposes greater coordination demands among agents. Based on these observations, we chose not to evaluate COMA and QMIX in more complex scenarios, as their performance in simpler settings already indicates substantial limitations.

F.2 MARL Performance for EO-constrained Tasks

In SatBench, we also evaluate MARL policies on EO-constrained cooperative tasks that require coordinated imaging by multiple satellites. We instantiate four task scenarios by crossing two orbital setups with three sensor types that differ in responsiveness, resolution, and field of view. For each target τ, a strict temporal window $W_\tau = [t_\tau^-, t_\tau^+]$ (width Δt_τ) is specified; a capture counts as jointly successful only if at least two distinct satellites acquire valid imagery at timestamps $\{t_i\} \subseteq W_\tau$ with pairwise slack $\max_{i,j} |t_i - t_j| \leq \delta_\tau \leq \Delta t_\tau$. This design stresses temporal synchronization and on-demand coalition formation rather than mere single-satellite coverage. Throughout our additional experiments, the objective prioritizes coverage of unique targets: a reward is issued once per target upon its first joint success, and repeated or temporally misaligned acquisitions yield no additional return. Consequently, MARL policies must learn to avoid redundant shots, expand target coverage, and coordinate within the specified time window to earn a reward.

F.2.1 Experiment Setting

Orbital Motion and Observation Constraints. Each satellite agent in SatBench follows a deterministic Kepler orbit defined by Classical Orbital Elements (COEs) and is equipped with constrained pointing control. The orbital parameters used in different configurations are summarized in Table 10. All satellites share the same semi-major axis ($a = 6798.3$ km) and near-zero eccentricity, while inclinations differ to produce varied ground coverage across latitude bands.

- **Orbital motion:** Satellites move along pre-defined Kepler orbits with fixed COEs. Inclinations span $\{2°, 6°, 10°\}$ (prograde), yielding diverse visibility footprints globally.
- **Pointing actions:** Each satellite has 2-DoF control (pitch, yaw), with discrete increments $\{-5°, 0, +5°\}$ per axis (nine pointing directions), reflecting limited agility and control resolution of EO platforms.
- **Field of View (FOV):** Each sensor defines a fixed conical viewing area; full aperture angle depends on agent type (e.g., 10°–20°). A target is observable only if it lies within the projected view cone and exceeds a minimum elevation threshold $\theta_{\min}$ (e.g., 35°).

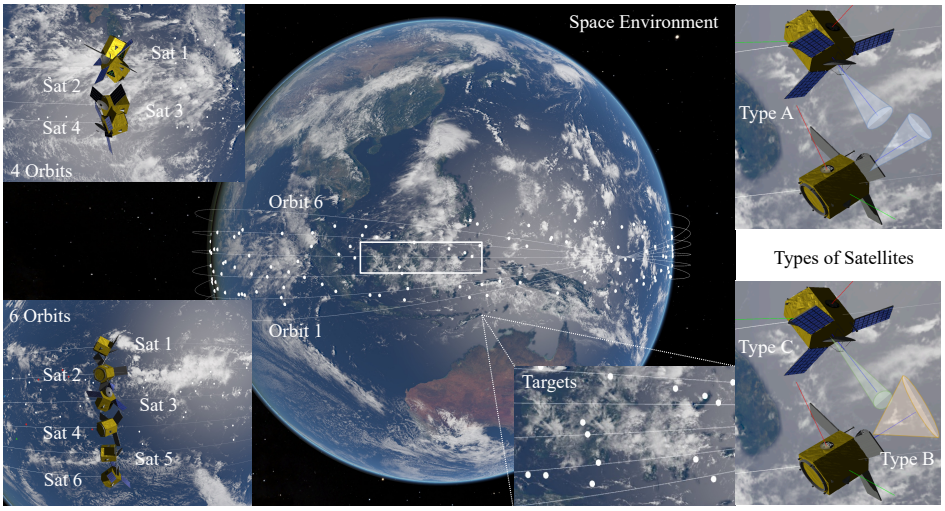


Figure 8: Experiment setting for EO-constrained cooperative tasks in SatBench.

Visibility model. A ground target g_k is *visible* to satellite i at time t if

$$g_k \in \mathrm{proj}_{\mathrm{FOV}_i}(t) \quad \text{and} \quad \mathrm{elev}(i, g_k, t) \;\geq\; \theta_{\min}.$$

Here $\mathrm{proj}_{\mathrm{FOV}_i}(t)$ is the FOV ground footprint, and the elevation can be written without approximation as

$$\mathrm{elev}(i, g_k, t) \;=\; \arcsin\!\left(\frac{(p_i(t) - p_k)\cdot \hat{n}_k}{\|p_i(t) - p_k\|}\right),$$

where $p_i(t)$ is the satellite ECEF position, p_k is the target ECEF position, and $\hat{n}_k$ is the target's local zenith unit vector. We define the binary visibility mask

$$\mathrm{vis}_{i,k}(t) = \begin{cases} 1, & \text{if } g_k \text{ is visible to } i \text{ at } t, \\ 0, & \text{otherwise.} \end{cases}$$

Table 10: Satellite COEs for EO-constrained cooperative tasks (two orbital setups). Angles in degrees.

Satellite	a (km)	e	i	Ω	ω	ν_0
4 Orbit setup						
Sat1	6798.3	0.000691	2	0	0	0
Sat2	6798.3	0.000691	6	0	0	90
Sat3	6798.3	0.000691	2	90	0	0
Sat4	6798.3	0.000691	6	90	0	90
6 Orbit setup						
Sat1	6798.3	0.000691	2	0	0	0
Sat2	6798.3	0.000691	6	0	0	90
Sat3	6798.3	0.000691	10	0	0	180
Sat4	6798.3	0.000691	2	90	0	0
Sat5	6798.3	0.000691	6	90	0	90
Sat6	6798.3	0.000691	10	90	0	180

Reward structure. This EO-constrained cooperative task is to image geographically distributed ground targets under a time-window constraint requiring *joint* observation. Each target g_k carries a priority $w_k \in [0.5, 1]$ and is considered completed only if it is successfully imaged by at least two distinct satellites within a temporal window of length Δt (e.g., 5 minutes) with pairwise slack $\leq \delta_\tau$. Define the *success event* for g_k at time t:

$$\mathrm{succ}_k(t) \;=\; \Big[\exists C \subseteq \mathcal{A},\, |C| \geq 2,\, \exists \{t_i\}_{i\in C} \subseteq [t{-}\Delta t,\, t] \text{ s.t. } \mathrm{vis}_{i,k}(t_i) = 1 \;\wedge\; \max_{i,j\in C} |t_i {-} t_j| \leq \delta_\tau\Big].$$

Let $\mathrm{done}_k(t)$ indicate whether g_k was completed before t. The global reward is

$$r_t \;=\; \sum_{k=1}^{K} w_k \mathbb{1}[\neg\, \mathrm{done}_k(t) \;\wedge\; \mathrm{succ}_k(t)]\,,$$

and a completed target is removed from future reward consideration.

Task configurations. In this additional experiment, SatBench considers four scenarios with increasing complexity, varying satellites, orbital planes, and sensing heterogeneity (Table 11).

Table 11: EO-constrained cooperative tasks.

Scenario	**Num. of Orbits**	**Num. of Satellites**	**Formation Type**	**Num. of Targets**	**Target Distribution**	**Agent Type**
EO-constrained Scenario 1	4	4	Constellation	100	Sparse	Homogeneous
EO-constrained Scenario 2	4	4	Constellation	100	Sparse	Heterogeneous
EO-constrained Scenario 3	6	6	Constellation	150	Sparse	Homogeneous
EO-constrained Scenario 4	6	6	Constellation	150	Sparse	Heterogeneous

Scenario notes. Unless otherwise stated, we instantiate three sensor types to reflect responsiveness–resolution–FOV trade-offs: Type A (baseline) uses a 15° full FOV with balanced slew responsiveness and medium ground resolution; Type B is narrow (10°), higher resolution and slightly slower to slew/settle; Type C is wide (20°), lower resolution but faster to slew/retask. All scenarios require a coalition size $m = 2$ and a time window $\Delta t = 5$ min with pairwise slack $\delta_\tau \leq \Delta t$.

EO-constrained Scenario 1. Four homogeneous satellites on four distinct orbits; all are Type A (15° FOV). This scenario isolates the *temporal synchronization* challenge without sensing heterogeneity: policies must align passes to achieve joint success while avoiding duplicate or off-window shots. Expected failure modes include near-simultaneous but non-overlapping timestamps and redundant reacquisitions of already-completed targets. It serves as a baseline for non-redundant scheduling and coverage expansion.

EO-constrained Scenario 2. Four satellites split into two sensing roles: agents $\{0, 2\}$ carry Type B (10° narrow, high-res) and agents $\{1, 3\}$ carry Type C (20° wide, fast-retask). Heterogeneity introduces *role specialization*: wide-FOV assets are effective for initial capture and timing alignment, while narrow-FOV assets refine coverage at higher resolution. Policies must learn when to let the wide sensor "anchor" the window and how to pair it with a narrow sensor to satisfy $m = 2$ *without* creating redundant duplicates. Typical pitfalls are overuse of wide sensors, causing starvation of narrow sensors, or narrow sensors missing the window due to slower slewing.

EO-constrained Scenario 3. Six homogeneous satellites across six orbits; all are Type A (15° FOV). Scaling up constellation size stresses *conflict resolution and load balancing*: while more opportunities exist for joint success, the risk of multiple agents targeting the same object simultaneously (and thus wasting shots) increases. Strong policies demonstrate coordinated target allocation, low synchronization slack, and improved success rate.

EO-constrained Scenario 4. Six satellites with mixed sensing: two Type A (15°), two Type B (10°), two Type C (20°). This is the most challenging setting, combining *scale* and *heterogeneity*. Effective behaviors include forming complementary coalitions (e.g., C + B or A + B) that meet the time window while expanding unique-target coverage. The policy must avoid redundant trios (>m) within the same window, mitigate timing mismatches between slow-to-slew narrow sensors and fast wide sensors, and balance workload across orbits. Failure patterns include persistent duplication on high-priority targets and systematic off-window acquisitions by one class of sensors.

Episode timing. Each episode spans 315 simulation steps. With a step size Δt_{step} (e.g., 20 s), this corresponds to ≈ 105 min, close to a full LEO orbital period at $a \approx 6798$ km, ensuring one complete cycle of target visibility/occlusion for fair, long-horizon coordination evaluation.

F.2.2 RESULTS

Table 12 reports the mean $\pm$ standard deviation over random seeds (higher is better). Two broad trends emerge. First, heterogeneous sensing and tighter coordination demands depress absolute scores relative to homogeneous counterparts (cf. the right columns of Table 12), indicating the difficulty of learning non-redundant, on-window coalitions. Second, among the three methods, HATRPO Kuba et al. (2021a) is the most robust across scales and heterogeneity, while HAPPO can edge out others only in the simplest setting.

On the homogeneous scenarios, HAPPO slightly outperforms in EO-constrained Scenario 1 (40.85 vs. 39.81/39.46 in Table 12), suggesting that in low-conflict regimes the alternating updates suffice to align two-satellite captures. However, when scaling to Scenario 3, HAPPO degrades markedly (24.59, a -39.8% drop from its Scenario 1 score), whereas HATRPO remains strong (37.56, only -5.7% vs. its Scenario 1) and surpasses MAPPO (33.38). The trust-region step thus appears to stabilize multi-agent coordination under increased contention for the same targets, reducing wasteful duplicates and missed windows, consistent with the gap between the HATRPO and HAPPO bars for Scenario 3.

On the heterogeneous scenarios, HATRPO consistently leads. In EO-constrained Scenario 2, it improves over MAPPO by 13.3% (19.74 vs. 17.43) and over HAPPO by 11.4% (19.74 vs. 17.72). In the more challenging EO-constrained Scenario 4, HATRPO remains best (30.58), with gains of 17.0% vs. MAPPO (26.14) and 4.9% vs. HAPPO (29.14). Notably, MAPPO exhibits high variance in EO-constrained Scenario 4 (± 6.43 in Table 12), indicative of instability in role assignment and

Table 12: Performance on EO-constrained cooperative tasks.

Scenarios	MAPPO	HAPPO	HATRPO
EO-constrained Scenario 1	39.46 ± 3.27	40.85 ± 0.37	39.81 ± 1.04
EO-constrained Scenario 2	17.43 ± 0.38	17.72 ± 0.26	19.74 ± 0.14
EO-constrained Scenario 3	33.38 ± 0.55	24.59 ± 1.12	37.56 ± 1.18
EO-constrained Scenario 4	26.14 ± 6.43	29.14 ± 0.11	30.58 ± 0.27

repeated, non-rewarding re-acquisitions; HATRPO maintains low dispersion (e.g., ± 0.27–1.18), suggesting steadier coalition formation and better avoidance of redundant shots. Overall, EO-constrained cooperative tasks favor methods that keep policy updates within a conservative region while exploiting a centralized critic for coordination benefits. HATRPO achieves the best balance of stability and performance across all but the simplest case; HAPPO is competitive only when conflicts are limited (EO-constrained Scenario 1); and MAPPO, while reasonable on average, is more sensitive to heterogeneity and scale. These outcomes align with our reward design, single-use per target, so effective policies expand unique-target coverage and synchronize captures within the window, rather than accumulating redundant acquisitions. The same pattern is numerically confirmed in Table 12.

To assess how coordination evolves under MAPPO in the EO-constrained Scenario 3 setting, Fig. 9 contrasts early (roughly 0–20% of training, t_{initial}) and late (70–100%, t_{end}) rollouts. Early on, several satellites keep their sensors off-nadir or outside target clusters, so the field-of-view footprints frequently miss candidate targets and many opportunities remain un-imaged. As training proceeds, agents learn basic nadir alignment and disperse more effectively across orbits, which increases first-hit coverage. By the late stage, MAPPO exhibits intermittent *cooperative* captures within the time window, but non-cooperative passes and duplicated attempts still occur, indicating reactive and partially uncoordinated behavior with fragmented spatial coverage relative to what the six-orbit layout could support.

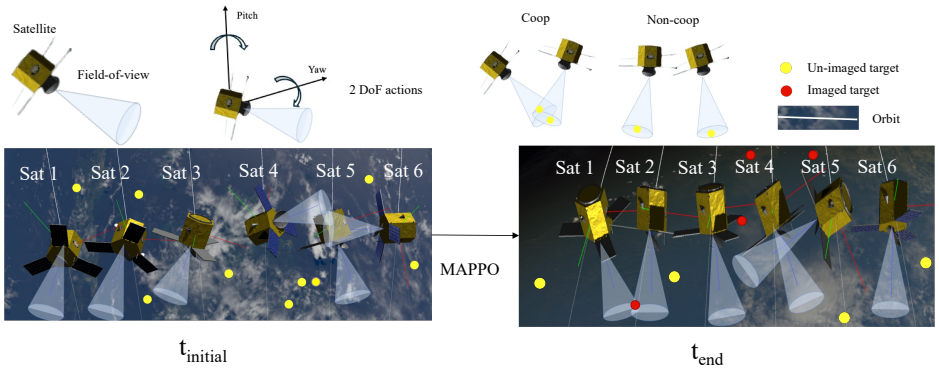


Figure 9: Visualization of MAPPO on Scenario 3 at early (t_{initial}) and late (t_{end}) training in Satbench.

F.3 RL Performance

F.3.1 RL Scenarios Definition

We design four RL scenarios derived from the original SatBench environment. RL-Scenarios 1 and 2 feature sparse target layouts with 30 and 60 targets, respectively, while RL-Scenarios 3 and 4 adopt dense layouts with 35 and 70 targets.

We also vary the priority of the target. In RL-Scenarios 1 and 3, all targets are assigned a uniform priority of 1.0, simplifying the mission objective to maximizing the number of targets captured. In contrast, RL-Scenarios 2 and 4 retain varying priority scores, requiring the satellite agent to weigh the relative value of each target when planning observations.

F.3.2 Results and Analysis

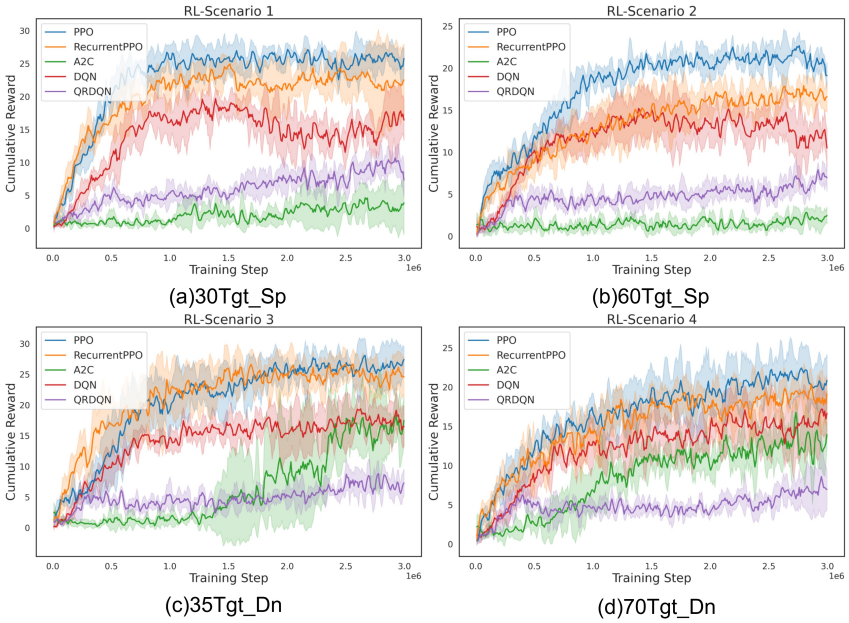


Figure 10: Training performance of PPO (Schulman et al., 2017), RecurrentPPO (Kapturowski et al., 2018), A2C (Mnih et al., 2016), DQN (Mnih et al., 2015) and QRDQN (Dabney et al., 2018) on four SatBench RL-scenarios. Lines show mean cumulative reward over 4 seeds; shaded regions indicate standard deviation.

Cumulative Reward. As shown in Figure 10, among all RL tested methods, PPO consistently achieves the highest cumulative rewards across both sparse and dense target distributions. Recurrent PPO closely follows, demonstrating strong performance across all scenarios. A2C performs reasonably well in the dense settings (RL-Scenarios 3 and 4). However, it struggles in sparse scenarios (RL-Scenarios 1 and 2), where capturing each target often requires long sequences of attitude adjustments. DQN and QRDQN underperform across all RL scenarios.

Task Completion Rate. As shown in Table 13, task completion rate trends are similar to the pattern of cumulative reward. PPO achieves the highest task coverage in all scenarios. Recurrent PPO also shows strong and stable results. A2C's performance is still not stable across different RL scenarios. Notably, DQN demonstrates early learning progress in sparse scenarios but plateaus or regresses beyond around 2 million steps (e.g., in RL-Scenario 2, DQN achieves a 43.1% completion rate but drops to 30.8% at 3 million steps), suggesting instability in long-horizon value estimation. QRDQN exhibits low final completion rates, consistent with its performance in the cumulative reward metric.

Scenario Difficulty Levels. The performance of the same algorithm across different scenarios varies significantly. RL-Scenario 1, which features sparse targets and uniform priority, proves to be the most tractable-agents simply aim to maximize coverage. In contrast, Scenarios 2 and 4 present greater challenges due to the additional complexity introduced by varying the priority of the target and the increased total number of targets.

Summary. Our provided additional results highlight the inherent difficulty of satellite rotation control in SatBench, even in the absence of inter-agent coordination. PPO and Recurrent PPO demonstrate strong learning performance across both sparse and dense target distributions, while value-based methods (DQN and QRDQN) and A2C struggle to achieve stable performance. The variation in performance across scenarios further confirms that target distribution and prioritization each introduce distinct challenges for single-agent policy learning.

Table 13: Single satellite agent performance on RL-Scenarios 1-4 measured by task completion rate (%) across training timesteps. All values are mean ± standard deviation over 4 seeds. Best performance is highlighted in bold.

Scenario	Algorithm	Training Timesteps			
		0.1M	1M	2M	3M
RL-Scenario 1	A2C	1.7 ± 2.1	5.3 ± 1.7	7.8 ± 5.4	14.2 ± 21.1
	DQN	5.0 ± 3.8	54.7 ± 15.4	51.7 ± 19.0	50.4 ± 14.5
	PPO	14.7 ± 2.9	81.7 ± 10.4	90.0 ± 4.7	**89.4 ± 5.8**
	QRDQN	5.8 ± 5.7	16.4 ± 8.0	21.1 ± 8.9	16.3 ± 7.4
	RecurrentPPO	21.9 ± 12.5	71.1 ± 11.1	69.4 ± 23.3	76.7 ± 20.5
RL-Scenario 2	A2C	1.4 ± 1.5	3.3 ± 4.1	8.2 ± 4.9	8.5 ± 4.1
	DQN	4.6 ± 3.8	41.8 ± 9.5	43.1 ± 18.7	30.8 ± 8.1
	PPO	19.7 ± 7.4	60.4 ± 2.8	64.0 ± 9.6	**68.3 ± 9.6**
	QRDQN	5.7 ± 7.5	13.6 ± 5.5	17.8 ± 5.6	21.0 ± 8.5
	RecurrentPPO	17.1 ± 7.4	42.8 ± 9.8	53.8 ± 13.0	55.3 ± 5.8
RL-Scenario 3	A2C	4.8 ± 3.6	3.8 ± 2.1	20.5 ± 20.6	47.9 ± 12.1
	DQN	4.5 ± 3.8	40.2 ± 10.8	40.7 ± 12.9	46.9 ± 9.9
	PPO	11.4 ± 6.1	48.3 ± 12.4	63.3 ± 8.1	**69.8 ± 8.2**
	QRDQN	2.1 ± 3.1	16.9 ± 12.6	15.5 ± 10.8	19.0 ± 10.0
	RecurrentPPO	18.6 ± 8.6	54.5 ± 14.5	61.4 ± 8.2	63.1 ± 13.3
RL-Scenario 4	A2C	3.8 ± 3.8	17.6 ± 7.4	24.6 ± 3.7	42.9 ± 6.6
	DQN	6.7 ± 4.9	37.1 ± 5.6	43.1 ± 7.3	41.3 ± 7.2
	PPO	8.6 ± 3.9	43.9 ± 7.7	48.2 ± 6.0	**56.0 ± 9.0**
	QRDQN	3.5 ± 2.7	10.7 ± 2.8	10.0 ± 3.0	17.5 ± 8.4
	RecurrentPPO	12.5 ± 6.3	39.0 ± 6.5	41.8 ± 2.9	51.2 ± 3.8

G USE CASE: SATBENCH FOR FIRE MONITORING

SatBench provides a physically grounded sandbox to prototype and evaluate satellite coordination strategies for wildfire monitoring. In SatBench, suspected or active fire hotspots are represented as geo-referenced targets; satellite assets evolve under orbital and attitude-slew constraints with bounded fields of view; and learning- or rule-based policies control sensor pointing and imaging to meet time-critical response objectives. This setup enables end-to-end assessment of fire-relevant performance metrics-such as time to first observation, revisit intervals, confirmation rates, and duplicate-imaging ratios-while fairly comparing centralized vs. decentralized coordination strategies under identical orbital and sensing conditions.

Building on this capability, we instantiate a FIRMS-informed fire-monitoring scenario (Singh et al., 2022). Using NASA-FIRMS near-real-time detections, we re-select 21 fire-prone target locations across multiple Australian states as persistent monitoring sites, and parameterize six real Earth-observation satellites as observing assets. We then evaluate SatBench tasking policies on latency-sensitive coverage objectives aligned with wildfire intelligence workflows.

We evaluate MAPPO across representative scenarios to validate SatBench as a realistic satellite coordination simulator and to enable a like-for-like comparison between learning-based and traditional approaches. Metrics include coverage and revisit adherence, mean response latency, task success rate, and overlap/conflict rate, as well as confidence improvement from targeted revisits. Results indicate that SatBench faithfully captures orbital geometry and visibility constraints, supporting an end-to-end pipeline from data-driven requirements (FIRMS targets) to satellite-executable tasks, and providing a unified, reproducible testbed for fair comparisons between MARL and industry-standard scheduling.

G.1 FIRE DATA SOURCE TO SATBENCH

SatBench models wildfire monitoring as coordinating satellites to service spatio-temporal `Target` objects. Each `Target` encodes a geographic location, a validity window, an intended revisit cadence, and a priority/confirmation policy. We derive these `Target`s from NASA's Fire Information for Resource Management System (FIRMS) through a reproducible, SatBench-native pipeline.

Given a scenario start time T_0, we: (i) query FIRMS near-real-time thermal anomaly feeds over a fixed look-back window $[T_0 - \Delta, T_0]$ within the area of interest (Australian states); (ii) apply quality and spatial filters (e.g., confidence threshold, land mask, AOI boundaries); (iii) aggregate detections into persistent fire sites via grid-based or density-based clustering and select the top-K sites (we use K=21) by persistence and radiative power; (iv) materialize SatBench `Targets` with attributes {`id`, latitude/longitude, $[t_{\min}, t_{\max}]$ validity, revisit rule (e.g., first confirmation within τ_1, follow-ups every τ_2), priority score, required confirmations}; (v) bind these `Targets` to a SatBench scenario that instantiates six real Earth-observation satellites as assets (ephemerides and sensor models defined elsewhere in this appendix); (vi) record data provenance and a random seed for full reproducibility.

The above steps express FIRMS detections directly in SatBench's tasking vocabulary: satellites receive executable `Targets` with timing, cadence, and priority semantics instead of raw thermal anomalies. This enables the end-to-end evaluation used in our wildfire study (e.g., time-to-first-observation, revisit compliance, overlap/conflict incidence, and confidence gains from targeted revisits) under the same orbital geometry and visibility constraints that SatBench enforces.

G.1.1 Fire Target in SatBench

We transform NASA-FIRMS *active-fire detections* into SatBench `target` objects. Each FIRMS record contributes a geo-referenced point (latitude/longitude) and timestamp that we map to a `target` with explicit spatial position and a validity window around the detection time. We optionally attach a revisit rule (e.g., initial confirmation within τ_1, follow-ups every τ_2), a priority score, and a required number of confirmations. This representation expresses human-readable fire-demand signals in the tasking vocabulary of SatBench: satellites do not consume raw thermal anomalies, but schedulable `targets` with timing and cadence semantics.

Within SatBench, spacecraft operate under realistic orbital geometry, attitude-slew limits, field-of-view bounds, and line-of-sight visibility. Using identical `target` sets and timing windows, we compare a learning-based method (MAPPO) (Yu et al., 2022) with an operations-research baseline (MILP) (Qu et al., 2022). We report SatBench-native metrics: coverage and revisit compliance, mean response latency, task completion rate, overlap/conflict incidence, and confidence gains from targeted revisits. This pipeline provides a unified and realistic setup that supports end-to-end studies from FIRMS-derived demand to executable satellite tasking within the benchmark.

G.1.2 Real-World Satellite Assets in SatBench

To ground the scenarios with realistic orbital geometry, we instantiate 6 spacecraft in sun-synchronous orbits using representative Keplerian elements at a common epoch. Table 14 lists the semi-major axis a, eccentricity e, inclination i, right ascension of the ascending node Ω, argument of perigee ω, and true anomaly ν_0 for each satellite used in our experiments.

Table 14: Optical satellite orbital parameters in Use Case of Fire Monitoring.

Satellite	a [km]	e	i [°]	Ω [°]	ω [°]	M [°]	rev/day
SKYSAT-C13	6798.30	0.000691	97.0113	86.2373	284.9298	175.1185	15.4882
SKYSAT-C11	6818.38	0.0005046	97.3746	168.1553	170.8700	189.2637	15.4199
FLOCK 4Q-34	6835.57	0.0010286	97.4281	114.0842	124.5557	235.6656	15.3618
Global-19	6812.76	0.0010458	42.0049	234.8969	257.1020	102.8651	15.4390
Global-18	6811.64	0.0005709	52.9867	94.5095	348.8911	11.1964	15.4428
Global-17	6811.92	0.0009131	41.9999	90.7625	66.3025	293.8772	15.4418

We ingest these elements into SatBench and propagate them to the experiment epoch (SGP4 for TLE-derived cases or a high-fidelity Kepler/J2 model otherwise) to obtain time-varying ephemerides. From these, we compute line-of-sight, look angles, and access windows to surface `targets`. Sensor characteristics (e.g., effective swath via `scan`/`track` proxies, day/night flags) are mapped to observation constraints; optional SAR/optical mode selection enables cloud-robust sensing studies. This process preserves the geometry and visibility conditions encountered in practice and enables like-for-like evaluation of scheduling policies within SatBench using real-world satellite kinematics.

Adding, removing, or modifying satellites only requires editing the corresponding configuration file at `SatBench/Satellites/LEO_satellite_orbit_data.xlsx`.

G.2 Experiment Settings

We evaluate MAPPO and a MILP baseline under scenarios that vary in the degree of dynamism in target priorities. Here, *dynamic priority* means that targets do not have fixed importance values across the experiment; instead, their priority scores are sampled from a specified interval. We consider two ranges: a high-priority interval [0.5,1] (Scenarios 1.x) and a full-range interval [0,1] (Scenarios 2.x). Please refer to Appendix B.3 for details. In the *Base* setting, three training episodes are generated where each target is assigned a single priority value at the beginning of the episode, which remains constant for all its revisit opportunities. This represents a lower level of dynamism, since priorities vary only across episodes but remain stable within an episode. By contrast, the *Base + condition* setting introduces a higher level of dynamism by resampling each target's priority at every revisit, allowing its importance to change over time within the same episode. These two configurations, therefore, represent different levels of dynamic tasking: cross-episode variability (Base) versus intra-episode variability (Base + condition). MAPPO is trained with multiple random seeds on the designated training episodes, while the MILP baseline is solved independently for each episode using the realized priority values. The performance metric is the cumulative episodic reward, defined as the sum of the priorities of all successfully completed targets, which directly reflects the effectiveness of each method in capturing high-value observations.

G.3 Results

As reported in Table 15, overall, MAPPO and the MILP baseline achieve comparable performance in the Base scenarios, where target priorities remain fixed within each episode. However, when the Base + condition setting is introduced, MILP performance declines, particularly in the 2.x scenarios with a wider range of target priorities, where the cumulative reward drops significantly. This highlights the advantage of MARL methods like MAPPO in adapting to dynamic tasking environments, as the learned policy can adjust to changing target importance during revisits, whereas MILP, which relies on static planning, is more sensitive to intra-episode variability and high priority variance.

Table 15: Comparison of MAPPO and MILP Rewards across Dynamic Tasking Scenarios (without revisit time).

Scenario	**Target Priority**	**MAPPO**	**MILP**
1.1 Base: dynamic priority for 3 episodes	[0.5–1]	24.167 ± 1.592	21.099 ± 1.734
1.2 Base + condition: different priority in each revisit	[0.5–1]	25.072 ± 2.093	20.288 ± 2.911
2.1 Base: dynamic priority for 3 episodes	[0–1]	15.483 ± 1.791	11.463 ± 1.538
2.2 Base + condition: different priority in each revisit	[0–1]	14.524 ± 2.197	6.428 ± 2.508