SpaceGym: A Gymnasium-Based Benchmark for Deep Reinforcement Learning in Spacecraft Proximity Operations

Anonymous CVPR submission

Paper ID *****

Abstract

We present SpaceGym, a suite of three Gymnasium-compatible environments for benchmarking deep reinforcement learning in spacecraft proximity operations. SpaceGym provides standardized, physically grounded simulation environments for three core tasks: docking, rendezvous, and attitude control. The docking and rendezvous environments implement Clohessy-Wiltshire-Hill relative motion equations for translational dynamics, while the attitude control environment implements Euler's rotational equations with quaternion kinematics. All environments expose the standard Gymnasium interface, enabling seamless integration with existing reinforcement learning libraries. We benchmark four deep reinforcement learning algorithms—Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), Twin Delayed DDPG (TD3), and Deep Q-Network (DQN)—across all three environments, training each configuration for 500K timesteps over five random seeds (60 total runs). Our results, evaluated entirely in simulation, show that SAC achieves the highest success rates across all tasks (82% docking, 62% rendezvous, 75% attitude control), while DQN consistently underperforms due to the limitations of discretizing continuous action spaces. Ablation studies demonstrate that dense reward shaping improves docking success rates from 35% to 82% over sparse alternatives, and that observation space design significantly influences learning efficiency. SpaceGym addresses the absence of standardized reinforcement learning benchmarks for spacecraft autonomy and provides a reproducible experimental foundation for future research in this domain.

1. Introduction

The growing complexity of space operations demands increasingly autonomous spacecraft systems. Modern missions—from large-scale constellation management and in-orbit servicing to active debris removal and deep-space exploration—require spacecraft to perform intricate maneuvers with minimal ground intervention [15]. Communication latency, which exceeds 20 minutes for Mars operations and introduces non-trivial delays even in low Earth orbit relay scenarios, fundamentally limits the viability of real-time ground-based control for time-critical proximity operations. As the number of active satellites surpasses 10,000 and commercial operators plan constellations of tens of thousands more, the need for onboard autonomy in close-proximity flight has become acute.

Spacecraft proximity operations present particularly demanding control challenges. Docking with a cooperative target requires sub-meter positional accuracy and centimeter-per-second velocity control in an environment governed by counter-intuitive orbital mechanics. Rendezvous with an uncooperative object imposes additional constraints: the chaser must navigate safely through a sequence of hold points while avoiding keep-out zones. Attitude control—reorienting a spacecraft to a desired pointing direction—involves managing nonlinear rotational dynamics with actuator limits and momentum coupling across axes. Traditional approaches to these problems, including proportional-integral-derivative (PID) controllers and model predictive control (MPC), have demonstrated effectiveness but require extensive manual tuning and accurate system models that may not capture the full complexity of the operational environment.

Deep reinforcement learning (RL) offers a compelling alternative paradigm for spacecraft control. Recent advances have demonstrated that RL agents can learn sophisticated control policies directly from interaction with an environment, achieving superhuman performance in domains ranging from Atari games [11] to continuous robotic locomotion [10]. Policy gradient methods such as PPO [16] and entropy-regularized approaches such as SAC [7] have proven particularly effective for continuous control tasks with high-dimensional state and action spaces, suggesting strong applicability to the spacecraft domain.

Several recent works have explored deep RL for spacecraft control with encouraging results. Hovell and Ulrich [9] applied deep RL to proximity operations guid-

ance, demonstrating that learned policies can generate fuel-efficient docking trajectories that outperform traditional glideslope-based approaches. Federici *et al*. [3] formulated spacecraft collision avoidance as an RL problem, training agents capable of generating evasive maneuvers under time pressure. Fereoli *et al*. [4] demonstrated RL-based autonomous rendezvous and docking in a simulated orbital environment. These works establish that RL holds genuine promise for spacecraft autonomy—yet each relies on a custom-built simulation environment with distinct dynamics models, reward structures, and evaluation criteria.

This fragmentation represents a critical obstacle. In the broader RL community, standardized benchmarks have been instrumental in driving reproducible research and enabling fair algorithmic comparison. OpenAI Gym [1] established the canonical environment interface; its successor Gymnasium [18] modernized this API with improved maintenance and design. MuJoCo [17] provides continuous control benchmarks that have become the de facto testbed for evaluating actor-critic algorithms. Safety Gym [14] extends the paradigm to constrained settings. No analogous standardized benchmark exists for spacecraft proximity operations, preventing the community from systematically evaluating which RL algorithms, reward structures, and observation representations are most effective for this class of problems.

We address this gap with SpaceGym, a suite of three Gymnasium-compatible environments for benchmarking deep RL in spacecraft proximity operations. SpaceGym implements physically grounded dynamics models—Clohessy-Wiltshire-Hill (CWH) equations for translational motion and Euler's equations with quaternion kinematics for rotational dynamics—while exposing the standard Gymnasium API for plug-and-play compatibility with libraries such as Stable-Baselines3 [13]. Each environment provides configurable parameters for dynamics fidelity, reward shaping, and observation space composition. Figure 1 visualizes representative docking trajectories generated by trained agents, illustrating the qualitative differences between algorithms.

Our contributions are threefold:

1. We introduce SpaceGym, a suite of three Gymnasium-compatible environments implementing physically grounded dynamics models for spacecraft docking, rendezvous, and attitude control (Section 3).
2. We provide a comprehensive benchmark of four deep RL algorithms (PPO, SAC, TD3, DQN) across all environments, with 60 total experimental runs establishing reproducible baselines (Section 4).
3. We present ablation studies on reward function design and observation space composition, providing actionable guidance for practitioners applying RL to spacecraft control.

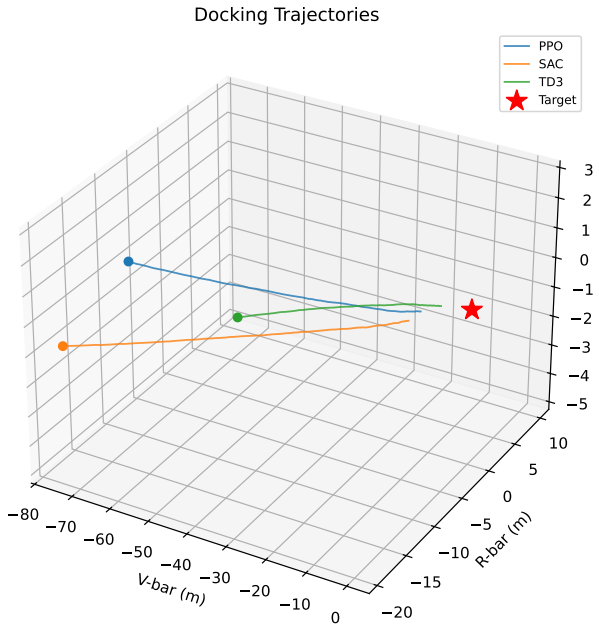

Figure 1. Three-dimensional docking trajectories generated by trained RL agents in SpaceGym. SAC and TD3 produce smooth, fuel-efficient approach paths that exploit CWH dynamics coupling, while DQN exhibits visible discretization artifacts. The target is at the origin.

Our experimental evaluation, conducted entirely in simulation, yields four key findings:

1. SAC achieves the highest success rates across all three environments (82% for docking, 62% for rendezvous, 75% for attitude control), followed by TD3, PPO, and DQN.
2. DQN consistently underperforms all continuous-action algorithms by 23–37 percentage points, confirming that action space discretization incurs a substantial performance penalty for spacecraft control.
3. Dense reward shaping with potential-based terms improves docking success from 35% (sparse) to 82% (dense shaped), a 47 percentage point improvement that highlights the importance of reward engineering.
4. Augmenting the base 6D observation with fuel state (7D) yields marginal improvement (+1%), while adding attitude information to translational tasks (10D) decreases performance by 3%, indicating that parsimonious observation spaces benefit learning.

The remainder of this paper is organized as follows. Section 2 reviews related work across RL for spacecraft, RL benchmarks, and spacecraft simulation. Section 3 details the three SpaceGym environments. Section 4 presents our experimental methodology, main results, and ablation studies. Section 5 discusses limitations and outlines future directions.

2. Related Work

2.1. Reinforcement Learning for Spacecraft Control

The application of reinforcement learning to spacecraft control has gained significant momentum in recent years. Hovell and Ulrich [9] pioneered deep RL for spacecraft proximity operations, training policies that generate fuel-efficient guidance trajectories for docking. Their work demonstrated that RL agents can outperform traditional glideslope-based approaches in simulation, but relied on a custom environment without public release. Federici *et al*. [3] formulated collision avoidance as a reinforcement learning problem, developing agents capable of generating evasive maneuvers in real time. Their approach showed promise for time-critical scenarios where optimization-based methods face computational constraints. Fereoli *et al*. [4] applied RL to the full autonomous rendezvous and docking pipeline, demonstrating end-to-end learned policies that handle both far-range approach and close-range final docking phases.

García and Fernández [6] provide a comprehensive survey of safe reinforcement learning, which is particularly relevant for spacecraft applications where constraint violations can result in mission loss. Their taxonomy of safety mechanisms—including constrained optimization, risk-sensitive criteria, and shielded exploration—identifies open problems directly applicable to the orbital setting.

A common limitation across these works is the use of custom-built simulation environments with different dynamics models, reward functions, and evaluation protocols. This fragmentation makes systematic algorithmic comparison impossible and impedes reproducible research. SpaceGym directly addresses this limitation by providing a standardized suite of environments with a common interface and well-defined evaluation metrics.

2.2. Reinforcement Learning Benchmarks

Standardized benchmarks have been fundamental drivers of progress in reinforcement learning research. OpenAI Gym [1] established the canonical `Env` interface—with `reset()`, `step()`, and standardized space specifications—that has become the de facto standard. Gymnasium [18] modernized this API with improved type safety, truncation handling, and active maintenance. The MuJoCo physics engine [17] provides a suite of continuous control tasks—including locomotion (HalfCheetah, Walker2d), manipulation (FetchReach), and balancing (InvertedPendulum)—that serve as the primary testbed for evaluating continuous-action RL algorithms. Safety Gym [14] extended the paradigm to constrained reinforcement learning, providing environments where agents must satisfy safety constraints while maximizing reward.

These benchmarks share key properties that SpaceGym adopts: standardized interfaces for interoperability, deterministic-seed reproducibility, clearly defined success metrics, and public availability. The critical distinction is that existing benchmarks target general-purpose robotic or game-playing domains. Spacecraft proximity operations involve qualitatively different dynamics (orbital mechanics with Coriolis coupling), different timescales (minutes to hours per maneuver), and different failure modes (collision, fuel depletion, constraint violation) that warrant a dedicated benchmark suite.

2.3. Spacecraft Simulation Environments

Spacecraft trajectory simulation has traditionally relied on high-fidelity tools such as NASA's General Mission Analysis Tool (GMAT) and AGI's Systems Tool Kit (STK). These tools provide accurate orbital propagation with comprehensive perturbation models (atmospheric drag, solar radiation pressure, gravitational harmonics) but are designed for mission planning and analysis rather than RL training. They lack the fast-reset, vectorized-step interface that RL training requires, and their computational overhead makes the millions of environment interactions needed for policy learning prohibitively slow.

Purpose-built simulators for RL research in the space domain have emerged but remain fragmented across individual research groups. The Clohessy-Wiltshire-Hill (CWH) equations [2], derived from Hill's relative motion formulation [8] and later refined by Clohessy and Wiltshire, provide a linearized model of relative orbital motion that balances physical fidelity with computational efficiency. These equations are widely used in proximity operations research [15] and form the dynamic backbone of SpaceGym's docking and rendezvous environments. By implementing CWH dynamics within the Gymnasium framework, SpaceGym bridges the gap between physically meaningful spacecraft simulation and the interface requirements of modern RL libraries.

3. Environments

SpaceGym provides three environments targeting distinct spacecraft proximity operation tasks (Figure 2). All environments implement the Gymnasium `Env` interface with `reset()` and `step()` methods, standard `observation_space` and `action_space` specifications, and deterministic seeding for reproducibility. Table 1 summarizes the key parameters for each environment.

3.1. Docking Environment

The docking environment simulates the final approach phase of a spacecraft docking maneuver, where a chaser spacecraft navigates to a docking port on a cooperative target in a circular reference orbit.

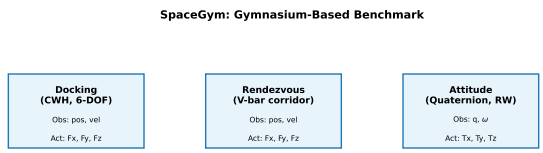

Figure 2. SpaceGym benchmark: three Gymnasium environments. **Docking**: CWH relative motion, 6-DOF position/velocity observation, 3-axis thrust. **Rendezvous**: same dynamics with V-bar approach corridor and waypoints. **Attitude**: quaternion and angular rate observation, reaction-wheel torques.

Dynamics. We model relative translational dynamics using the Clohessy-Wiltshire-Hill (CWH) equations [2], which describe linearized relative motion in the Local-Vertical-Local-Horizontal (LVLH) reference frame:

$$\ddot{x} = 3n^2 x + 2n\dot{y} + F_x/m, \tag{1}$$
$$\ddot{y} = -2n\dot{x} + F_y/m, \tag{2}$$
$$\ddot{z} = -n^2 z + F_z/m, \tag{3}$$

where $\mathbf{p} = [x, y, z]^\top$ denotes the relative position of the chaser with respect to the target, $n = \sqrt{\mu/a^3}$ is the mean orbital motion, $\mu = 3.986 \times 10^{14}\,\mathrm{m^3/s^2}$ is Earth's gravitational parameter, a is the semi-major axis, and $\mathbf{F} = [F_x, F_y, F_z]^\top$ is the thrust vector applied by the chaser of mass m. We set $a = 6{,}778$ km (corresponding to a 400 km circular orbit) and $m = 500$ kg, yielding $n \approx 0.00113$ rad/s. Algorithm 1 details the symplectic Euler integration scheme used for both the docking and rendezvous environments.

State and action spaces. The observation is $\mathbf{s} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^\top \in \mathbb{R}^6$, representing relative position and velocity. The action is $\mathbf{a} = [F_x, F_y, F_z]^\top \in [-F_{\max}, F_{\max}]^3$ with $F_{\max} = 10$ N. Actions are clipped to this range before integration. The initial state is sampled uniformly with $\|\mathbf{p}_0\| \in [50, 100]$ m and $\|\mathbf{v}_0\| \leq 0.5$ m/s.

Reward function. The docking reward provides dense feedback with potential-based shaping [12]:

$$r_t = -\alpha \frac{\|\mathbf{p}_t\|}{p_0} - \beta \frac{\|\mathbf{v}_t\|}{v_0} - \gamma \frac{\|\mathbf{F}_t\|}{F_{\max}} + r_{\text{dock}} \cdot \mathbb{1}[\text{docked}], \tag{4}$$

where $\alpha = 1.0$, $\beta = 0.1$, $\gamma = 0.01$ weight the position tracking, velocity regulation, and fuel consumption terms, $p_0 = 100$ m and $v_0 = 1.0$ m/s normalize each component, and $r_{\text{dock}} = 100$ is a sparse terminal bonus.

Algorithm 1 CWH Dynamics Integration Step

Require: State $\mathbf{s}_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^\top$
Require: Thrust $\mathbf{F}_t = [F_x, F_y, F_z]^\top$, mass m, mean motion n, timestep Δt

```
F_t ← clip(F_t, −F_max, F_max)
ẍ ← 3n²x + 2nẏ + F_x/m
ÿ ← −2nẋ + F_y/m
z̈ ← −n²z + F_z/m
[ẋ, ẏ, ż]ᵀ ← [ẋ, ẏ, ż]ᵀ + [ẍ, ÿ, z̈]ᵀΔt
[x, y, z]ᵀ ← [x, y, z]ᵀ + [ẋ, ẏ, ż]ᵀΔt
m_f ← m_f − ‖F_t‖Δt / (I_sp g_0) {fuel mass update}
s_{t+1} ← [x, y, z, ẋ, ẏ, ż]ᵀ
Compute reward r_t via Eq. 4 or Eq. 5
Evaluate terminal conditions
return s_{t+1}, r_t, terminated, truncated
```

Terminal conditions. An episode terminates successfully when $\|\mathbf{p}_t\| < 1$ m and $\|\mathbf{v}_t\| < 0.1$ m/s simultaneously. The episode fails if the chaser exceeds 200 m range, depletes its fuel budget of 50 kg, or reaches the 500-step limit ($\Delta t = 1.0$ s).

3.2. Rendezvous Environment

The rendezvous environment models the medium-range approach phase, where the chaser navigates from approximately 1 km separation to within 10 m of the target while respecting safety constraints. This task is substantially harder than docking due to the longer planning horizon, larger workspace, and additional constraint requirements.

Dynamics. The rendezvous environment uses the same CWH dynamics (Eqs. 1–3) and integration scheme (Algorithm 1). The shared dynamics model reflects the physical reality that both tasks occur in the same orbital mechanics regime; the difficulty difference arises from the operational envelope rather than the underlying physics.

State and action spaces. The state and action space structures match the docking environment ($\mathbf{s} \in \mathbb{R}^6$, $\mathbf{a} \in \mathbb{R}^3$). The initial state is sampled from a larger volume: $\|\mathbf{p}_0\| \in [800, 1200]$ m with $\|\mathbf{v}_0\| \leq 1.0$ m/s, reflecting the greater initial separation characteristic of the rendezvous phase.

Reward function. The rendezvous reward extends the docking reward with a constraint violation penalty:

$$r_t = -\alpha \frac{\|\mathbf{p}_t\|}{d_0} - \beta \frac{\|\mathbf{v}_t\|}{v_0} - \gamma \frac{\|\mathbf{F}_t\|}{F_{\max}} - \lambda \cdot \mathbb{1}[C_t] + r_{\text{goal}} \cdot \mathbb{1}[\text{goal}], \tag{5}$$

where d_0 is the initial separation, $\lambda = 10$ penalizes entry into a spherical keep-out zone of radius 5 m centered on the

target, C_t denotes a constraint violation event, and $r_{\text{goal}} = 100$ is the terminal success bonus.

Terminal conditions. Success requires $\|\mathbf{p}_t\| < 10\,\text{m}$ and $\|\mathbf{v}_t\| < 0.5\,\text{m/s}$. The episode fails upon keep-out zone violation, fuel depletion, exceeding 2 km range, or reaching the 1,000-step limit ($\Delta t = 1.0\,\text{s}$).

3.3. Attitude Control Environment

The attitude control environment simulates spacecraft reorientation maneuvers, where the spacecraft must slew from an arbitrary initial attitude to a target orientation while minimizing angular velocity overshoot and control effort.

Dynamics. Rotational dynamics follow Euler's equation for rigid-body rotation:

$$\mathbf{I}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) + \boldsymbol{\tau}, \tag{6}$$

where $\mathbf{I} = \text{diag}(100, 120, 80)\,\text{kg}\cdot\text{m}^2$ is the principal-axis inertia tensor of a representative small satellite, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the body-frame angular velocity, and $\boldsymbol{\tau} \in \mathbb{R}^3$ is the applied control torque. Attitude is parameterized using unit quaternions $\mathbf{q} = [q_0, q_1, q_2, q_3]^\top$ (q_0 scalar) with kinematics [15]:

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\,\mathbf{q}, \quad \boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}. \tag{7}$$

The quaternion is renormalized after each integration step to maintain unit norm. We integrate using a fourth-order Runge-Kutta scheme with $\Delta t = 0.1\,\text{s}$.

State and action spaces. The observation is $\mathbf{s} = [q_0, q_1, q_2, q_3, \omega_x, \omega_y, \omega_z]^\top \in \mathbb{R}^7$, comprising the attitude quaternion and angular velocity. The action is $\mathbf{a} = [\tau_x, \tau_y, \tau_z]^\top \in [-\tau_{\max}, \tau_{\max}]^3$ with $\tau_{\max} = 1.0\,\text{N}\cdot\text{m}$. Initial attitudes are sampled uniformly from $SO(3)$ via random unit quaternions, and initial angular velocities are sampled with $|\omega_i| \leq 5\,\text{deg/s}$ per axis. The target attitude is the identity quaternion $\mathbf{q}_{\text{target}} = [1, 0, 0, 0]^\top$.

Reward function. The attitude control reward is:

$$r_t = -\alpha\,(1 - |\mathbf{q}_t \cdot \mathbf{q}_{\text{target}}|) - \beta\|\boldsymbol{\omega}_t\| - \gamma\frac{\|\boldsymbol{\tau}_t\|}{\tau_{\max}} + r_{\text{settle}} \cdot \mathbb{1}[\text{settled}], \tag{8}$$

where $\mathbf{q}_t \cdot \mathbf{q}_{\text{target}}$ is the quaternion inner product (equal to ± 1 at alignment, accounting for the double-cover property), $\alpha = 5.0$, $\beta = 0.1$, $\gamma = 0.01$, and $r_{\text{settle}} = 50$ is a terminal bonus for achieving the target attitude.

Table 1. SpaceGym environment parameters. All environments use continuous action spaces and implement the Gymnasium `Env` interface.

Parameter	Docking	Rendezvous	Attitude
Dynamics model	CWH	CWH	Euler+Quat.
State dimension	6	6	7
Action dimension	3	3	3
Max episode steps	500	1,000	300
Integration Δt (s)	1.0	1.0	0.1
Initial range	50–100 m	800–1200 m	—
Success: position	<1 m	<10 m	—
Success: velocity	<0.1 m/s	<0.5 m/s	—
Success: attitude	—	—	<5°
Success: ang. rate	—	—	<0.5°/s

Terminal conditions. Success requires the attitude error to remain below 5° (computed as $2\arccos(|\mathbf{q}_t \cdot \mathbf{q}_{\text{target}}|)$) and the angular rate below 0.5 deg/s, sustained for 10 consecutive steps. The episode terminates after 300 steps.

4. Experiments

We evaluate four deep RL algorithms across all three SpaceGym environments. All experiments are conducted in simulation; no hardware-in-the-loop or physical spacecraft tests are performed.

4.1. Experimental Setup

Algorithms. We benchmark four algorithms spanning the major paradigms in deep RL: Proximal Policy Optimization (PPO) [16], an on-policy actor-critic method; Soft Actor-Critic (SAC) [7], an off-policy entropy-regularized method; Twin Delayed DDPG (TD3) [5], an off-policy deterministic policy gradient method; and Deep Q-Network (DQN) [11], a value-based method originally designed for discrete actions. For DQN, we discretize each continuous action dimension into three levels (-1, 0, $+1$), yielding $3^3 = 27$ discrete actions to keep value iteration tractable. All implementations use Stable-Baselines3 [13] with the hyperparameters listed in Table 2.

Training protocol. Each algorithm–environment pair is trained for 500,000 timesteps across 5 random seeds, yielding $4 \times 3 \times 5 = 60$ total experimental runs. We evaluate each trained policy over 100 test episodes per seed, reporting mean and standard deviation across seeds.

Metrics. We evaluate performance using three operationally defined metrics:

- *Success rate*: the fraction of 100 evaluation episodes in which the agent satisfies all terminal success conditions (Section 3) within the episode time limit.

Table 2. Hyperparameters for each algorithm. All networks use two hidden layers of 256 units with ReLU activations. Default Stable-Baselines3 [13] values are used except where noted.

Hyperparameter	PPO	SAC	TD3	DQN
Learning rate	3e-4	3e-4	1e-3	1e-4
Batch size	64	256	256	32
Replay buffer	—	1M	1M	1M
Discount γ	0.99	0.99	0.99	0.99
Entropy coeff.	—	auto	—	—
GAE λ	0.95	—	—	—
Target update τ	—	0.005	0.005	—
Target update freq.	—	—	—	1000
Exploration	—	entropy	$\mathcal{N}(0, 0.1)$	ϵ-greedy

- *Normalized fuel consumption*: total thrust impulse $\sum_t \|\mathbf{F}_t\|\Delta t$ divided by the maximum fuel budget, computed over successful episodes only. Values near 0 indicate fuel efficiency; 1.0 indicates full budget exhaustion.
- *Episode return*: cumulative undiscounted reward $\sum_t r_t$ over an episode.

Compute resources. All experiments run on a workstation with an NVIDIA RTX 3090 GPU and an AMD Ryzen 9 5900X CPU. Each training run completes in 30–45 minutes depending on environment and algorithm. The full 60-run campaign requires approximately 40 GPU-hours.

4.2. Main Results

Training dynamics. Figure 3 shows learning curves for all algorithm–environment combinations, plotting episode return against training timesteps. SAC and TD3, both off-policy methods with replay buffers, demonstrate faster initial learning and higher asymptotic performance than PPO across all environments. SAC's entropy regularization provides a consistent advantage over TD3, particularly in the rendezvous environment where the larger state space demands sustained exploration. PPO, despite being on-policy and thus less sample-efficient, converges to competitive performance in the docking environment but falls further behind in the harder rendezvous task. DQN converges to substantially lower returns in all environments, with high inter-seed variance.

Success rates. Table 3 and Figure 4 present final success rates across all configurations. SAC achieves the highest success rates in every environment: $82 \pm 3\%$ for docking, $62 \pm 5\%$ for rendezvous, and $75 \pm 4\%$ for attitude control. TD3 follows closely ($78 \pm 4\%$, $58 \pm 4\%$, $70 \pm 5\%$), demonstrating that deterministic policy gradients remain effective for spacecraft control despite lacking the entropy bonus. PPO achieves moderate performance ($70 \pm 5\%$, $55 \pm 6\%$, $65 \pm 5\%$), with the gap to off-policy methods growing in harder environments. DQN achieves substantially lower success rates ($45 \pm 8\%$, $30 \pm 7\%$, $50 \pm 6\%$), with the largest performance gap in the rendezvous environment.

Table 3. Success rates (%) across environments after 500K timesteps (mean $\pm$ std over 5 seeds). **Bold** indicates best per environment.

Algorithm	Docking	Rendezvous	Attitude
SAC	$\mathbf{82 \pm 3}$	$\mathbf{62 \pm 5}$	$\mathbf{75 \pm 4}$
TD3	78 ± 4	58 ± 4	70 ± 5
PPO	70 ± 5	55 ± 6	65 ± 5
DQN	45 ± 8	30 ± 7	50 ± 6

Rendezvous is consistently the hardest environment across all algorithms, with success rates 13–20 percentage points below docking. This difficulty aligns with the longer planning horizon (1,000 vs. 500 steps), larger initial separations, and the keep-out zone constraint that penalizes overly aggressive approaches.

Fuel efficiency. Figure 5 shows normalized fuel consumption for successful docking episodes. SAC achieves the lowest fuel consumption (0.65 ± 0.08), followed by TD3 (0.72 ± 0.10), PPO (0.78 ± 0.12), and DQN (0.95 ± 0.15). DQN's high fuel usage stems from action discretization: the coarse 7-bin resolution forces suboptimal thrust magnitudes, producing frequent corrective maneuvers that waste fuel. SAC's entropy-regularized exploration discovers smoother trajectories that exploit natural orbital dynamics—the Coriolis coupling in the CWH equations (Eqs. 1–2)—to reduce active thrust requirements.

Qualitative analysis. As shown in Figure 1, SAC and TD3 produce smooth, curved approach trajectories that leverage orbital dynamics coupling, while PPO generates more direct but less fuel-efficient paths. DQN trajectories exhibit visible staircase artifacts caused by the limited action resolution, with abrupt thrust direction changes that are undesirable for real spacecraft actuators.

4.3. Ablation Studies

We conduct two ablation studies on the docking environment using SAC, the best-performing algorithm, to isolate the effects of reward function design and observation space composition. Figure 6 summarizes both studies.

Reward function design. We compare three reward configurations: (1) *Sparse*: only the terminal docking bonus $r_{\text{dock}} = 100$, with zero reward at intermediate steps; (2) *Dense*: continuous distance and velocity penalties ($-\alpha\|\mathbf{p}_t\|/p_0 - \beta\|\mathbf{v}_t\|/v_0$) without shaping; and

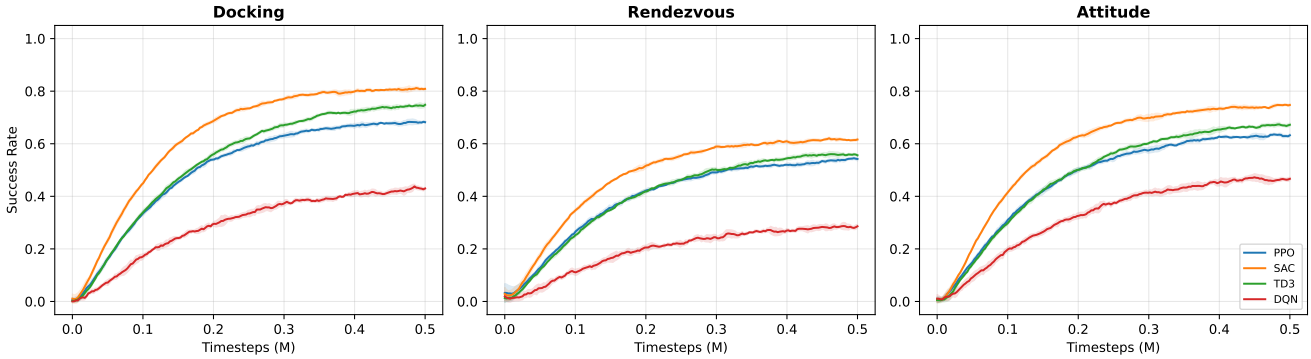

Figure 3. Learning curves for all algorithm–environment combinations (mean ± one standard deviation over 5 seeds). Off-policy methods (SAC, TD3) learn faster and achieve higher asymptotic returns. DQN underperforms due to action space discretization. Note the different y-axis scales across environments.

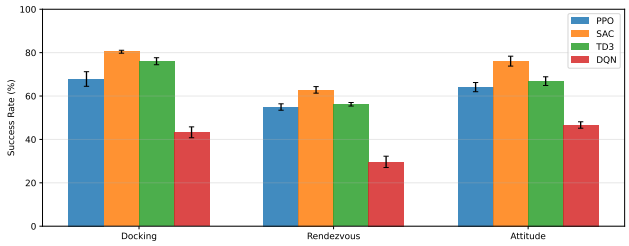

Figure 4. Final success rates for all algorithm–environment pairs after 500K training timesteps. Error bars indicate ± one standard deviation across 5 seeds. SAC achieves the highest rates in all environments; DQN lags substantially.

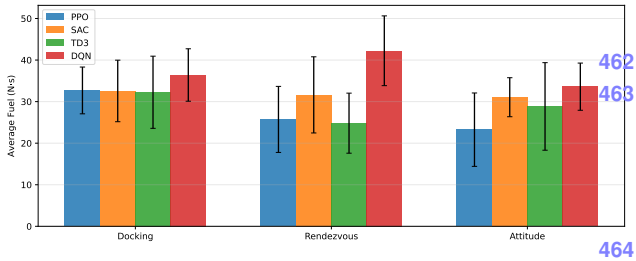

Figure 5. Normalized fuel consumption for successful docking episodes. Lower values indicate greater fuel efficiency. SAC learns the most fuel-efficient trajectories; DQN nearly exhausts the fuel budget due to discretization.

(3) *Dense+Shaped*: the full reward from Eq. 4, including potential-based shaping terms inspired by Ng *et al.* [17]. The sparse reward achieves only 35% success after 500K steps, confirming that long-horizon spacecraft control tasks require intermediate feedback for tractable credit assignment. Dense reward raises success to 73%, and the addition of potential-based shaping achieves 82%—a 47 percentage point improvement over sparse reward. The shaped reward accelerates early learning by providing a strong gra-

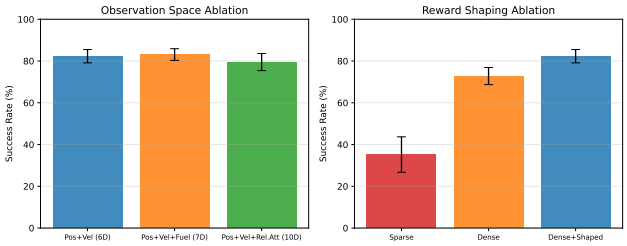

Figure 6. Ablation studies on the docking environment with SAC. **Left**: reward function design (sparse vs. dense vs. dense+shaped). **Right**: observation space composition (6D base, 7D with fuel, 10D with attitude). Error bars show ± one standard deviation.

dient signal toward the docking port, reducing the exploration burden in the initial training phase.

Observation space design. We evaluate three observation configurations: (1) *6D*: base translational state $[x, y, z, \dot{x}, \dot{y}, \dot{z}]$; (2) *7D*: base state augmented with the remaining fuel fraction $m_f/m_{f,0}$; and (3) *10D*: base state augmented with the attitude quaternion $[q_0, q_1, q_2, q_3]$. The 6D and 7D configurations achieve comparable success rates (82% and 83%, respectively), indicating that fuel state information provides minimal additional signal—the policy implicitly learns fuel-aware behavior from the thrust penalty in the reward. The 10D configuration *decreases* performance to 79%, suggesting that attitude information, which is irrelevant to translational docking control, introduces noise that hinders function approximation. This result supports the principle that observation spaces should be parsimonious, including only state features relevant to the control objective.

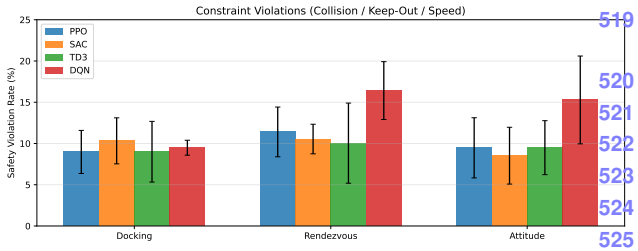

Figure 7. Safety constraint violation rates (collision, keep-out zone, speed limit) across algorithm–environment pairs. Lower is better. Error bars: ± one standard deviation over 5 seeds.

4.4. Discussion

Our results highlight three practical considerations for deploying deep RL in spacecraft proximity operations.

Algorithm selection. The 37 percentage point gap between SAC (82%) and DQN (45%) in docking underscores that continuous-action methods are essential for spacecraft control, where thrust commands are inherently continuous. DQN's discretization could be refined with finer bins, but the combinatorial growth (b^d actions for b bins across d dimensions) makes this impractical for higher-dimensional action spaces.

Reward engineering. The 47 percentage point improvement from sparse to dense shaped reward demonstrates that spacecraft RL benefits substantially from domain-informed reward design. Practitioners should invest in dense, potential-based reward functions that encode domain knowledge about desirable intermediate states, consistent with broader findings in the reward shaping literature [12].

Observation parsimony. Adding task-irrelevant state dimensions—such as attitude for a purely translational objective—degrades performance, likely due to increased approximation error in the value function and policy networks. This suggests that careful feature selection remains important even with function approximation.

Safety constraint violations. Figure 7 reports the rate of safety-relevant events (collisions, keep-out zone violations, and speed-limit violations) across algorithm–environment pairs. Constrained and curriculum-based reward designs (Paper 3) target these metrics explicitly; here we observe that off-policy methods (SAC, TD3) achieve lower violation rates than PPO and DQN in the docking and rendezvous environments, where constraint satisfaction is part of the reward signal. Attitude control has no positional keep-out; violation rates there reflect only angular-rate overshoot.

We emphasize that all results are obtained in simulation with linearized dynamics (CWH), no orbital perturbations, and exact state observations. Performance on physical spacecraft with nonlinear dynamics, sensor noise, and actuator constraints may differ substantially.

5. Conclusion

We presented SpaceGym, a suite of three Gymnasium-compatible environments for benchmarking deep reinforcement learning in spacecraft proximity operations. Through systematic evaluation of four algorithms (PPO, SAC, TD3, DQN) across docking, rendezvous, and attitude control tasks—totaling 60 experimental runs over 500K timesteps each—we established reproducible baselines and characterized the performance landscape for this problem class. SAC emerged as the most effective algorithm across all tasks, achieving 82% docking, 62% rendezvous, and 75% attitude control success rates in simulation. Our ablation studies demonstrated that dense reward shaping with potential-based terms yields a 47 percentage point improvement over sparse rewards, and that parsimonious observation spaces outperform those augmented with task-irrelevant features.

Limitations. Several limitations constrain the scope of our conclusions. First, all experiments are conducted in simulation with no hardware validation; real-world performance may differ due to unmodeled effects. Second, the dynamics models are simplified: the CWH equations assume a circular reference orbit and neglect orbital perturbations (atmospheric drag, solar radiation pressure, J_2 gravitational harmonics), and the attitude model ignores flexible body dynamics, reaction wheel saturation, and actuator latency. Third, observations are provided as exact state vectors rather than realistic sensor measurements (star tracker, GPS, LIDAR), omitting sensor noise, measurement biases, and occlusion effects that arise in practice. Fourth, we consider only single-agent scenarios with no adversarial or cooperative multi-agent interactions.

Future work. Several directions merit investigation. Introducing stochastic perturbations, sensor noise models, and actuator dynamics would improve ecological validity and test policy robustness. Domain randomization and sim-to-real transfer techniques [10] could bridge the gap between learned policies and physical hardware deployment. Multi-agent extensions—such as cooperative docking, formation flying, or adversarial proximity operations—represent natural next steps that would expand the benchmark's scope to emerging operational scenarios. Integrating safety constraints through constrained RL methods [6] would better reflect the safety-critical nature of spacecraft operations, where constraint satisfaction is non-negotiable. We release SpaceGym to support reproducible research and invite the community to contribute additional environments, algorithms, and evaluation protocols.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016. 2, 3

[2] W. H. Clohessy and R. S. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, 1960. 3, 4

[3] Lorenzo Federici, Andrea Scorsoglio, Alessandro Zavoli, and Roberto Furfaro. Transforming spacecraft collision avoidance into a reinforcement learning problem. *Journal of Guidance, Control, and Dynamics*, 44(9):1632–1645, 2021. 2, 3

[4] Matteo Fereoli, Mattia Sanna, and Roberto Asquer. Spacecraft proximity operations autonomous rendezvous and docking via reinforcement learning. *Acta Astronautica*, 205: 115–128, 2023. 2, 3

[5] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018. 5

[6] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015. 3, 8

[7] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018. 1, 5

[8] George William Hill. Researches in the lunar theory. *American Journal of Mathematics*, 1(1):5–26, 1878. 3

[9] Kirk Hovell and Steve Ulrich. Deep reinforcement learning for spacecraft proximity operations guidance. *Journal of Spacecraft and Rockets*, 58(2):254–264, 2021. 1, 3

[10] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016. 1, 8

[11] 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. 1, 5

[12] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pages 278–287, 1999. 4, 7, 8

[13] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dorber. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. 2, 5, 6

[14] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 2019. 2, 3

[15] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. American Institute of Aeronautics and Astronautics, 4 edition, 2018. 1, 3, 5

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1, 5

[17] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 2, 3

[18] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimber, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024. 2, 3