Benchmarking Continuous-Control Reinforcement Learning for Free-Floating Space Manipulator Trajectory Tracking and Quantifying the Impact of PPO Hyperparameter Optimization

Firstname Lastname*
Department of Mechatronics / Robotics, University XYZ

Abstract

Free-floating space manipulators exhibit strong dynamic coupling between manipulator motion and spacecraft base drift. This paper studies continuous-control reinforcement learning (RL) for end-effector trajectory tracking in a physics-based Simulink/Simscape environment with integrated safety monitoring. We benchmark six RL agents (TRPO, TD3, SAC, PPO, Policy Gradient, and DDPG) on the same circular end-effector tracking task and evaluate performance using a unified set of trajectory, stability, smoothness, energy, and safety KPIs. Across the benchmark, PPO achieves the best overall trade-off between tracking accuracy, base stabilization, training stability, and wall-clock efficiency. Building on this result, we quantify the effect of systematic PPO tuning (network architecture and key hyperparameters). Compared to a default PPO configuration, the optimized PPO improves the mean episodic return magnitude by 80.7%, reduces the mean squared end-effector tracking error by 84.9% and the maximum tracking error by 70.3%, and stabilizes the base orientation (72.2% reduction in orientation error). Moreover, the optimized PPO eliminates collisions and joint-limit violations in evaluation while slightly reducing energy consumption (15.98%).

Keywords: space robotics, free-floating manipulation, reinforcement learning, PPO, safe simulation, trajectory tracking

1 Introduction

On-orbit servicing, assembly, and debris mitigation rely on robotic manipulators operating on free-floating platforms. In the absence of ground contact and strong damping, manipulator actions can induce significant base translation and rotation, which can degrade tracking accuracy and violate safety constraints. Reinforcement learning (RL) offers a model-free approach to synthesize feedback policies for nonlinear dynamics, but training must respect safety limits and should converge reliably.

This work focuses on (i) a controlled benchmark of multiple continuous-control RL algorithms on a unified free-floating trajectory tracking task, and (ii) an explicit quantification of the gains achievable through PPO optimization.

1.1 Contributions

- A unified benchmark of six continuous-control RL agents on the same free-floating trajectory tracking environment, evaluated using consistent trajectory, base-motion, smoothness, energy, and safety KPIs.
- A quantitative study of PPO optimization (network and hyperparameters), showing large improvements in learning stability and task performance, and elimination of safety violations in evaluation.

*Replace with your affiliation and email.

- An engineering-oriented setup with integrated safety logic (collision monitoring, torque bounds, and physical plausibility monitoring) suitable for reproducible RL experimentation.

2 Related Work

Classic texts cover RL fundamentals and MDP formulations [1]. For policy-gradient methods and trust-region style updates, see TRPO and PPO [2, 3]. Off-policy continuous control methods include DDPG, TD3, and SAC [4, 5, 6]. Free-floating space manipulator dynamics and momentum coupling are discussed in space robotics literature [7]. Domain randomization is a common tool to improve sim-to-real robustness [8].

3 Problem Formulation and Simulation Environment

3.1 Task: Circular End-Effector Tracking

The end-effector (EE) is commanded to follow a circular trajectory in the inertial world frame. A representative configuration uses a radius $r = 0.4\,\mathrm{m}$, duration $T = 8.5\,\mathrm{s}$, and angular rate $\omega = \pi/T$. The trajectory is generated at fixed simulation step $T_s = 0.01\,\mathrm{s}$, while the RL agent acts at $T_{s,\mathrm{agent}} = 0.1\,\mathrm{s}$ (actions are held between agent updates).

3.2 Observation and Action

We use a 23-dimensional observation vector

$$o = [e_p^\top, e_v^\top, q^\top, \dot{q}^\top, v_b^\top, \omega_b^\top, e_{\mathrm{ori}}^\top]^\top \in \mathbb{R}^{23}, \tag{1}$$

with $n_J = 4$ joints. The action is the joint torque command

$$a = \tau \in \mathbb{R}^4, \quad \tau \in [-2, 2]\ \mathrm{N\,m}\ \text{(per joint)}. \tag{2}$$

3.3 Reward and Episode Termination

A shaped reward combines tracking, base motion, actuation regularization, and a progress term:

$$\begin{aligned} \mathrm{cost} = {} & w_p \|e_p\|^2 + w_v \|e_v\|^2 + w_{\mathrm{ori}} \|e_{\mathrm{ori}}\|^2 + w_{\omega_b} \|\omega_b\|^2 + w_{v_b} \|v_b\|^2 \\ & + w_u \|\tau\|^2 + w_\Delta \|\tau - \tau_{\mathrm{prev}}\|^2, \end{aligned} \tag{3}$$

$$r = (\mathrm{progress} + \mathrm{bonus}) - \frac{\mathrm{cost}}{500}. \tag{4}$$

Episodes terminate on numerical issues, severe divergence, or safety violations (collision and joint-limit handling).

3.4 Safety and Consistency Monitoring

The environment includes:

- **Collision monitoring** via minimum distance $d_{\min}$ between selected bodies and an enforced safety margin $d_{\min} \geq d_{\mathrm{safe}}$ (typical $d_{\mathrm{safe}} = 0.02\,\mathrm{m}$).
- **Torque bounding** (saturation at actuator limits) and formal checking of bound satisfaction.
- **Physical plausibility monitoring** using total momentum behavior as a consistency check for free-floating dynamics.

Table 1: Optimized PPO settings used in this work.

Setting	Value
Agent sample time	$T_{s,\text{agent}} = 0.1\,\text{s}$
Experience horizon	1024
Mini-batch size	256
Epochs per update	10
Clip factor	0.2
Entropy loss weight	$1 \cdot 10^{-3}$
Advantage estimator	GAE ($\lambda = 0.95$)
Discount factor	$\gamma = 0.995$
Actor learning rate	$1 \cdot 10^{-3}$
Critic learning rate	$5 \cdot 10^{-4}$
Gradient threshold	1 (actor and critic)
Network architecture	2×128 (ReLU), Gaussian actor heads

4 Reinforcement Learning Methods

We benchmark six agents: TRPO, TD3, SAC, PPO, a vanilla Policy Gradient (PG) baseline, and DDPG. All agents are trained for $N_{\text{train}} = 1000$ episodes. For each algorithm, the best agent according to the moving average training reward is selected and evaluated over $N_E = 30$ episodes.

4.1 Optimized PPO Configuration

Given the benchmark results, PPO is selected for focused optimization. The optimized PPO uses explicit actor/critic networks: two shared fully connected layers with 128 units and ReLU activations, and Gaussian policy heads (mean and standard deviation). Key PPO options include an experience horizon of 1024, minibatch size 256, 10 epochs per update, clip factor 0.2, entropy weight 10^{-3}, GAE with $\lambda = 0.95$, discount factor $\gamma = 0.995$, actor learning rate 10^{-3}, critic learning rate $5 \cdot 10^{-4}$, and gradient thresholding.

5 Evaluation Metrics

We report:

- **Trajectory and safety KPIs (K1–K9):** mean episodic return, mean and maximum EE tracking error, base orientation error, collision rate, joint-limit violation rate, torque smoothness/jerk, base angular velocity, and energy.
- **Training KPIs (T1–T3):** reward variance late in training, variance of reward changes, and wall-clock training time.

6 Results

6.1 Benchmark: Agent Comparison

Tables 2 and 3 summarize the benchmark across six RL algorithms.

Key findings. PPO achieves the lowest mean tracking error (K2=0.0257) and strong base stabilization (K4=0.0667), while also being among the fastest to train (T3=00:07:40) and relatively stable (T2=0.2123). TRPO shows similar stability but worse tracking and longer

Table 2: Trajectory and safety KPIs for different RL agents (lower is better for K2–K4, K6–K9; higher is better for K1).

KPI	TRPO	TD3	SAC	PPO	PG	DDPG
K1 (mean episode return)	-2.2100	-5.3456	-6.2696	-1.0921	-5.3976	-6.0303
K2 (EE tracking MSE)	0.0680	0.1914	0.1908	0.0257	0.1638	0.0937
K3 (max EE tracking error)	0.6783	0.5341	0.7766	0.4833	1.0654	0.4758
K4 (mean base orientation error)	0.0694	0.0868	0.1687	0.0667	0.1302	0.2682
K5 (collision rate)	0	0	0	0	0	0
K6 (joint-limit violation rate)	0.0214	0	0	0.0505	0	0
K7 (torque smoothness / jerk KPI)	0.7498	0.9981	0.7536	0.6812	0.7125	0.8118
K8 (mean base angular speed)	0.0795	0.0206	0.1062	0.0948	0.0932	0.1078
K9 (mean energy)	3.7583	0.0893	9.0906	6.3251	5.3277	0.8185

Table 3: Training KPIs for different RL agents (lower indicates more stable training for T1/T2).

KPI	TRPO	TD3	SAC	PPO	PG	DDPG
T1 (late reward std)	0.6144	8.5168	9.6907	1.7756	13.1970	64.4764
T2 (reward-change std)	0.1703	1.5180	1.5301	0.2123	1.2325	3.5282
T3 (wall-clock time)	00:13:01	00:37:53	00:34:43	00:07:40	00:07:42	00:24:01

training time. TD3/SAC do not outperform PPO on tracking and require substantially more wall-clock time. DDPG reaches a competitive maximum tracking error (K3=0.4758) but exhibits highly unstable training (T1=64.4764).

6.2 PPO Optimization: Impact of Hyperparameters and Architecture

Figure 1 conceptually illustrates learning curves (replace with your exported plots). Quantitative improvements are summarized in Table 4.

Interpretation. The optimized PPO yields substantially smoother and more accurate trajectory tracking while simultaneously stabilizing the free-floating base. Importantly, safety violations observed in the default configuration (collisions and joint-limit violations) disappear in evaluation, indicating that the tuned policy better internalizes the safe operating envelope under the same environment constraints.

7 Discussion

7.1 Why PPO Performs Well Here

The benchmark indicates PPO provides a robust engineering trade-off: stable on-policy updates, bounded policy changes through clipping, and good wall-clock efficiency in this Simulink-based environment. While off-policy methods (TD3/SAC/DDPG) are often sample-efficient, their training variance and computational overhead can increase when the environment has expensive simulation steps and sensitive dynamics/safety termination.

7.2 Effect of PPO Hyperparameter Optimization

The results demonstrate that PPO performance is highly sensitive to the choice of horizon, minibatch regime, clipping, entropy regularization, and learning rates. The longer experience

Figure 1: Episodic reward over training for default PPO vs optimized PPO (placeholder).

Table 4: Default vs optimized PPO results (evaluation). Relative improvement is reported for error-like metrics as reduction; for K1 as reduction in |return|.

Metric	PPO (Default)	PPO (Optimized)	Relative improvement
K1 (mean episode return)	-2.23	-0.43	80.7% (lower $\|K1\|$)
K2 (EE tracking MSE)	0.0550	0.0083	84.9%
K3 (max EE tracking error)	0.9100	0.2700	70.3%
K4 (mean base orientation error)	0.0790	0.0220	72.2%
K8 (mean base angular speed)	0.0950	0.0500	47.4%
K5 (collision rate)	0.0670	0	eliminated
K6 (joint-limit violation rate)	0.0920	0	eliminated
K7 (torque jerk KPI)	0.4590	0.3510	23.5%
K9 (mean energy)	6.3250	5.3140	16.0%

horizon supports advantage estimation over a longer portion of the trajectory; minibatch/epoch settings improve data reuse without excessive gradient noise; and conservative critic learning plus gradient thresholding reduces instability.

7.3 Limitations and Future Work

Future work should include (i) domain randomization to address modeling uncertainty, (ii) multi-objective formulations balancing tracking accuracy, energy, and base motion, and (iii) hardware-in-the-loop or real-time testing for deployment realism.

8 Conclusion

We benchmarked six continuous-control RL algorithms for free-floating end-effector trajectory tracking with integrated safety logic. PPO provides the strongest overall performance among the tested agents. Further, targeted PPO optimization (architecture and hyperparameters) delivers major gains in learning stability, tracking accuracy, and base stabilization, while eliminating safety violations in evaluation and reducing energy consumption.

A Code Snippets (for Reproducibility)

A.1 Reward Function Skeleton (MATLAB)

Listing 1: Reward function structure (excerpt).

```
% Inputs: ep, ev, vbase, wbase, tau, tauPrev, e_ori
W.wp   = 150;   % position error
W.wv   = 25;    % velocity error
W.wori = 200;   % base orientation error
W.wwb  = 8;     % base angular speed
W.wvb  = 2;     % base linear speed
W.wu   = 0.02;  % torque magnitude
W.wd   = 0.06;  % torque changes

cost = W.wp*(ep'*ep) + W.wv*(ev'*ev) + W.wori*(e_ori'*e_ori) ...
     + W.wwb*(wbase'*wbase) + W.wvb*(vbase'*vbase) ...
```

```
     + W.wu*(tau'*tau) + W.wd*((tau-tauPrev)'*(tau-tauPrev));

reward = (prog + bonus) - cost/500;
```

A.2 Optimized PPO Options (MATLAB)

Listing 2: Optimized PPO options (excerpt).

```
agentOpts = rlPPOAgentOptions( ...
  'SampleTime', Ts_agent, ...
  'ExperienceHorizon', 1024, ...
  'MiniBatchSize', 256, ...
  'NumEpoch', 10, ...
  'ClipFactor', 0.2, ...
  'EntropyLossWeight', 1e-3, ...
  'AdvantageEstimateMethod', 'gae', ...
  'GAEFactor', 0.95, ...
  'DiscountFactor', 0.995);

agent.AgentOptions.ActorOptimizerOptions.LearnRate  = 1e-3;
agent.AgentOptions.CriticOptimizerOptions.LearnRate = 5e-4;
agent.AgentOptions.ActorOptimizerOptions.GradientThreshold  = 1;
agent.AgentOptions.CriticOptimizerOptions.GradientThreshold = 1;
```

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proc. ICML*, 2015.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.

[4] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.

[5] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. ICML*, 2018. (TD3)

[6] T. Haarnoja *et al.*, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proc. ICML*, 2018.

[7] K. Nanos and E. G. Papadopoulos, "On the dynamics and control of free-floating space manipulator systems in the presence of angular momentum," *Frontiers in Robotics and AI*, vol. 4, 2017.

[8] J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ IROS*, 2017.