Evaluating Hybrid Rollout for Efficient Robotic Simulation

Keegan Crasto[1]

Received October 26, 2025
Accepted May 28, 2026
Electronic access June 30, 2026

This work investigates a hybrid rollout framework for improving the efficiency of policy evaluation and rollout execution in robotic simulation. The approach reduces reliance on expensive physics-based simulation by integrating a learned dynamics model that predicts intermediate state transitions, while periodically synchronizing with the real environment to correct accumulated error. Experiments were conducted in the PandaReach-v3 environment using PPO, SAC, and a grouped reinforcement policy optimization (GRPO) controller. Results show that hybrid rollout achieves significant runtime improvements, with speedups of up to 3.34x compared to pure physics-based execution while maintaining high short-horizon task acquisition across a wide range of synchronization intervals, although large synchronization intervals produced degradation in long-horizon trajectory stability. Analysis of long-horizon behavior shows that learned models exhibit accumulated prediction error ("model drift") under recursive rollout, but periodic correction stabilizes control and prevents divergence. These findings demonstrate that accurate short-horizon prediction combined with intermittent correction is sufficient to replace a large fraction of simulator interaction, providing a simple and scalable approach for accelerating simulation and rollout execution in robotic environments while preserving control performance.

Keywords: Reinforcement Learning, Hybrid Rollout, Model-Based Reinforcement Learning, World Models, Simulation Efficiency, Robotics, PyBullet, PandaReach, PPO, SAC, GRPO

Introduction

Robots can be trained to perform tasks by repeatedly interacting with virtual environments and learning from trial and error, similar to how humans improve through practice. These virtual environments, known as simulators, allow robots to safely practice millions of actions before being deployed in the real world. In practice, reinforcement learning allows a robot to gradually improve its behavior by receiving feedback from the environment after each action. Actions that help achieve the task objective are rewarded, while unsuccessful behaviors are discouraged over repeated interactions.

Reinforcement Learning (RL) has emerged as a powerful framework for enabling robots to learn complex behaviors through trial-and-error interactions within simulated environments[1–4]. In robotic control tasks, RL allows agents to autonomously discover policies that optimize long-term performance, offering significant advantages over traditional control strategies that rely on explicit system modeling or manual programming[1,4]. However, despite these advantages, one of the primary bottlenecks in RL-based robotic training lies in the heavy computational demand of physics-based simulation[5]. Training an RL policy often requires millions of environment interactions, and each step in a high-fidelity simulator such as PyBullet or MuJoCo incurs significant computational cost due to real-time physics calculations[5–7]. As a result, long training durations become a major obstacle, especially for researchers constrained by limited computational resources.

To address this issue, several studies have explored approaches that improve the efficiency of simulation-driven RL training. These include methods such as domain randomization[8], adaptive randomization[9], and model-based reinforcement learning (MBRL)[10–12]. While these methods successfully enhance generalization and data efficiency, they do not directly accelerate the runtime execution of physics simulations. The computational bottleneck therefore persists at the simulator level. Achieving faster simulations without compromising accuracy remains an open challenge in the field.

This research introduces a hybrid rollout framework designed to bridge this gap by combining the accuracy of physics-based simulation with the efficiency of a learned world model. The approach integrates a multilayer perceptron (MLP) world model to predict intermediate environment transitions, executing N-1 steps using the learned model before synchronizing with the true physics environment at the Nth step[13,14]. This periodic correction mechanism ensures that cumulative prediction error, referred to as model drift, remains bounded, where model drift denotes the deviation between predicted and true states as errors accumulate over repeated predictions.

Intuitively, hybrid rollout replaces expensive physics up-

[1] *R.N. Podar School, India*

© The National High School Journal of Science 2026

dates with fast model predictions, followed by periodic correction using the true simulator. We hypothesize that replacing a portion of physics- based simulation steps with learned model predictions, combined with periodic synchronization, can significantly reduce runtime while maintaining comparable task performance.

The method was implemented and evaluated on the PandaReach-v3 environment from the Panda Virtual Gym, a standard robotic control benchmark.

Experimental results demonstrate that the proposed hybrid rollout achieves substantial computational acceleration, reaching up to 3.34x faster execution compared to pure PyBullet rollouts, while maintaining high short-horizon task acquisition across a wide range of synchronization intervals, although long-horizon trajectory stability degraded at larger synchronization intervals. Beyond efficiency, the results highlight a clear tradeoff between speed and stability: longer intervals yield greater speedup but increase the risk of model drift [10,13,15]. These findings indicate that learned dynamics can partially replace physics-based simulation without major loss in accuracy, representing a step toward more scalable and efficient reinforcement learning for robotic systems [2].

Related Work

Related Work and Research Gap

Reinforcement learning (RL) relies heavily on simulation to provide the large-scale interactions required for robotic policy training while avoiding hardware damage and safety risks [1,4,5,16]. However, policies trained in simulation often suffer from the sim-to-real gap, where performance degrades when transferred to physical robots due to inaccuracies in simulator dynamics, simplified robot models, and unrealistic environmental assumptions [5,8,16]. Improving simulation fidelity and reducing instability during transfer therefore remain major challenges in robotics reinforcement learning. Model-based reinforcement learning (MBRL) methods improve efficiency by learning transition dynamics models that reduce dependence on expensive real-world interaction [10,11,15]. Techniques such as PETS and MBPO use probabilistic neural dynamics models, uncertainty estimation, and short branched rollouts to reduce compounding prediction error while improving sample efficiency [10,13]. Other approaches including domain randomization, adaptive randomization, SimOpt, RCAN, and GraspGAN improve robustness and sim-to-real transfer by exposing policies to diverse environments or aligning simulated and real-world observations [8,9,17–21]. However, these methods primarily focus on improving transferability, robustness, or training efficiency rather than directly reducing the computational cost of simulator execution itself.

The proposed framework addresses this gap by reducing the number of expensive physics simulator calls through learned model prediction while maintaining stability using periodic synchronization with the real simulator. Although the method shares conceptual similarities with Dyna-style architectures [14], important differences exist. Dyna primarily interleaves real and model-generated experience to improve learning efficiency, often using tabular or planning-based updates, whereas the proposed approach uses a neural MLP world model together with deterministic periodic synchronization to constrain accumulated rollout error.

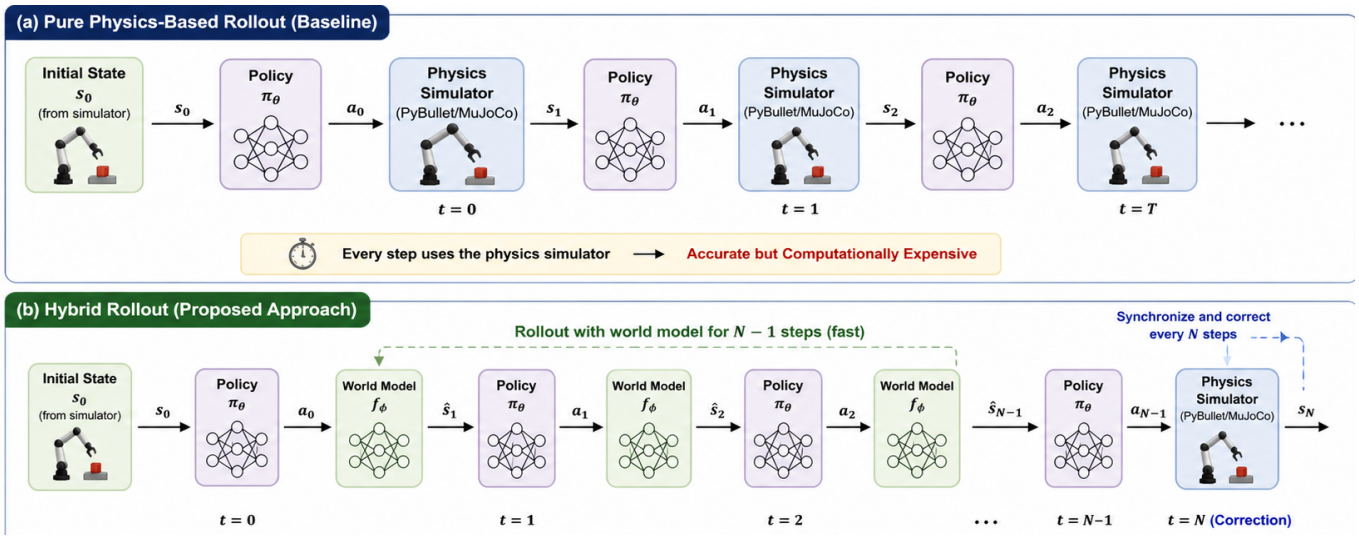

Fig. 1 Hybrid rollout framework. The policy selects actions, the world model predicts intermediate states, and the physics simulator is used periodically to correct drift.

© The National High School Journal of Science 2026

The framework is also closely related to Model-Based Policy Optimization (MBPO)[13], which uses short branched model rollouts and probabilistic ensemble dynamics models to improve policy-training sample efficiency. In contrast, this work uses a single deterministic MLP world model with direct recursive rollout and focuses on reducing runtime computational cost during simulator execution rather than improving policy optimization efficiency. Periodic synchronization with the physics simulator is used to stabilize long recursive rollouts and limit accumulated model drift during execution.

Methods

Task Environment and Problem Formation

All experiments were conducted using the PandaReach-v3 environment from Panda Gym[22], a robotic manipulation benchmark based on the Franka Emika Panda robotic arm. The objective of the task is to control the robot end-effector so that it reaches a randomly generated target position within the workspace. This environment was selected because its continuous control setting, goal-conditioned observations, and realistic robot kinematics make it well suited for comparing conventional reinforcement learning methods (PPO and SAC), grouped-action reinforcement learning (GRPO), and model-based hybrid control approaches.

Each episode begins with a default arm configuration. The simulator then samples a target position within a bounded three-dimensional workspace. At every timestep, the agent outputs continuous motion commands that move the end-effector toward the goal. Episodes terminate when the target is reached or when the maximum timestep limit is exceeded. Observations contain structured simulator variables, including the current end-effector position, the achieved goal, and the desired goal. For world-model training, only the Cartesian end-effector coordinates were retained as the system state. This reduced representation captures the variables most relevant to reaching while lowering prediction complexity. Task success was defined using the standard PandaReach criterion[22]: the Euclidean distance between the end-effector and the target must be less than 0.05 meters. Let x_t denote the end-effector position and g denote the target position. Success occurs when:

$$||x_t - g||_2 < 0.05 \tag{1}$$

where $||\cdot||_2$ denotes the Euclidean norm.

RL Policy Training (PPO, SAC, and GRPO)

To evaluate the proposed hybrid rollout framework across multiple reinforcement learning paradigms, three independent controllers were trained in the PandaReach-v3 environment: Proximal Policy Optimization (PPO)[12], Soft Actor-Critic (SAC)[23], and Grouped Relative Policy Optimization (GRPO). PPO was selected as a strong on-policy baseline with stable clipped policy updates, SAC as a widely used off-policy algorithm with strong continuous-control performance and sample efficiency, and GRPO as a grouped candidate-action controller in which multiple actions are sampled and ranked before execution. This enabled evaluation of whether the hybrid rollout framework generalizes across on-policy, off-policy, and grouped decision-making approaches. All agents were trained using the same task environment, identical success criterion, and the same goal-conditioned observation structure. PandaReach-v3 provides dictionary observations containing the current state, achieved goal, and desired goal. To process this structured input, PPO, SAC, and GRPO used a MultiInputPolicy architecture, enabling each controller to jointly reason over robot state and target location when selecting continuous Cartesian actions.

For PPO and SAC, the learned control mapping can be expressed as:

$$(state, goal) \rightarrow action \tag{2}$$

where the action corresponds to continuous end-effector displacement commands.

For GRPO, the controller generates a group of k candidate continuous actions which are scored relative to the goal before one action is selected for execution:

$$(state, goal) \rightarrow \{a_1, a_2, \ldots, a_k\} \rightarrow a^* \tag{3}$$

where k denotes the candidate group size and a^* denotes the selected grouped action.

Training all three controllers under matched conditions enabled direct comparison between conventional policy optimization methods, entropy-regularized off-policy control, and grouped-action decision-making within the proposed hybrid rollout framework.

PPO Hyperparameter Sweep

Before final training, a structured PPO hyperparameter sweep was conducted to identify a stable high-performing configuration suitable for world-model dataset generation and hybrid rollout benchmarking. The sweep varied learning rate, discount factor (γ), and rollout horizon (n_steps).

Configurations were evaluated using learning rates of 1×10^{-4}, 3×10^{-4}, and 1×10^{-3}, discount factors of 0.95, 0.98, and 0.99, and rollout horizons of 1024, 2048, and 4096. All runs used the same clipping coefficient, entropy coefficient, and training budget of 200,000 timesteps. Evaluation was based primarily on task success rate, with mean episodic reward used as a secondary metric.

The lowest learning rate (1×10^{-4}) was suboptimal, achieving only a 10% success rate. In contrast, configurations using

© The National High School Journal of Science 2026

3×10^{-4} consistently achieved stable convergence and 100% task success, while variations in γ and rollout horizon produced comparatively smaller performance differences. The strongest PPO configuration used:

- learning rate = 3×10^{-4}
- $\gamma = 0.98$
- n_steps = 1024

This configuration provided the best balance between convergence speed, policy stability, and final task performance, and was selected for all subsequent PPO experiments.

Final PPO Training

After hyperparameter selection, a final PPO controller was trained for 500,000 timesteps using the chosen configuration.

The resulting policy achieved: Success rate = 100%, Mean episode reward = -1.680 ± 0.859 across 300 evaluation episodes, Mean episode length = 2.680 ± 0.859 steps.

These results correspond to the standardized deployment evaluation protocol used throughout the Results section and indicate that the final PPO agent solved the PandaReach-v3 task consistently and efficiently, reaching the target in only a small number of control steps. The trained PPO controller was subsequently used for PPO world-model generation and hybrid rollout evaluation.

SAC Hyperparameter Sweep

A parallel hyperparameter sweep was conducted for SAC using the same training budget and evaluation procedure. Because SAC is an off-policy algorithm, rollout horizon is not a primary parameter. Instead, learning rate, discount factor (γ), and batch size were varied.

Several SAC configurations were evaluated using learning rates of 1×10^{-4}, 3×10^{-4}, and 1×10^{-3}, discount factors of 0.95, 0.98, and 0.99, and batch sizes ranging from 128 to 512. All runs were trained for 200,000 timesteps using identical replay buffer size, target smoothing coefficient, and update schedule.

Unlike PPO, SAC achieved a 100% success rate across all tested configurations, demonstrating high robustness. Selection was therefore based on mean episodic reward, identifying the optimal configuration as: Learning rate = 1×10^{-3}, γ = 0.98, batch size = 256.

This setup was used for all subsequent SAC training and hybrid rollout evaluations.

Final SAC Training

Using the selected configuration, a final SAC controller was trained for 500,000 timesteps with periodic evaluation checkpoints.

The resulting policy achieved: Success rate = 100%, Mean episode reward = -1.603 ± 0.770 across 300 evaluation episodes, Mean episode length = 2.603 ± 0.770 steps.

These results correspond to the standardized deployment evaluation protocol used throughout the Results section and indicate that the final SAC agent solved the PandaReach-v3 task reliably and efficiently, reaching the target in only a small number of control steps. The trained SAC controller was subsequently used for SAC world-model generation and hybrid rollout evaluation.

GRPO Hyperparameter Selection and Controller Development

GRPO was implemented as a custom grouped-action extension of the Stable-Baselines3 PPO algorithm, and its final configuration was determined through iterative empirical tuning rather than a formal hyperparameter sweep. The objective of this stage was to develop a stable high-performance grouped controller for PandaReach-v3 suitable for subsequent world-model training and hybrid rollout experiments.

The Stable-Baselines3 PPO implementation was used as the optimization backbone because it provides stable clipped policy updates, generalized advantage estimation (GAE), rollout buffers, and actor-critic policy networks. Rather than changing PPO's learning objective, the controller logic was modified so that grouped candidate actions were used during decision-making.

In standard PPO, one action is sampled from the policy distribution and executed at each timestep. In GRPO, multiple candidate actions were sampled from the stochastic actor:

$$\{a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(k)}\} \tag{4}$$

where K is the group size.

Each candidate action was ranked using a goal-directed heuristic based on predicted progress toward the PandaReach target:

$$Score_i = -||\hat{x}_{t+1}^{(i)} - g||_2 - \lambda ||a_t^{(i)}||_2 \tag{5}$$

where $\hat{x}_{t+1}^{(i)}$ is the predicted next end-effector position, g is the target position, and λ is a weighting coefficient controlling the penalty applied to unnecessarily large actions.

The predicted next position was estimated using:

$$\hat{x}_{t+1}^{(i)} = x_t + (\text{STEP\_SCALE} \times a_t^{(i)}) \tag{6}$$

STEP_SCALE was fixed at 0.05, corresponding approximately to the normalized Cartesian displacement scaling used in PandaReach-v3. The value was selected empirically during preliminary experimentation to provide stable first-order approximation of end-effector motion during grouped candidate ranking, and remained fixed across all experiments.

Instead of always selecting only the single best candidate, a stochastic top-k selector was introduced. Candidate actions were ranked, the top three were retained, and one of these top

© The National High School Journal of Science 2026

candidates was randomly selected for execution to improve exploration diversity and reduce premature convergence. Preliminary experiments evaluated top-k values of 1, 3, and 5. Using k = 1 produced more deterministic behavior and reduced exploration diversity, while k = 5 increased stochasticity and action variance without consistent improvement in task success. Empirically, k = 3 provided the best balance between exploration diversity, rollout stability, and consistent task acquisition in PandaReach-v3, and was therefore selected for all subsequent experiments.

Only the selected action was executed in the environment and stored inside PPO's original rollout buffer. PPO's clipped surrogate objective, critic regression, entropy regularization, and GAE therefore remained unchanged, with modifications occurring only at the action-generation and action-selection stage.

Several training settings were refined during development, including learning rate, rollout horizon, batch size, update epochs, and grouped candidate size. Group sizes between 4 and 32 and learning rates between 1×10^{-3} and 1×10^{-5} were evaluated empirically. Smaller groups reduced exploration diversity, while larger groups increased computational overhead without consistent performance improvement. Higher learning rates reduced stability, whereas smaller optimizer steps and longer rollout collection windows improved training consistency. The PPO clip range ε remained fixed at 0.2 following the standard Stable-Baselines3 PPO configuration.

Here, K denotes the total candidate group size, while k denotes the number of top-ranked candidates retained before stochastic selection.

The final GRPO configuration used MultiInputPolicy with learning rate = 1×10^{-4}, n_steps = 4096, batch size = 512, n_epochs = 10, $\gamma = 0.98$, GAE $\lambda = 0.95$, entropy coefficient = 0.01, clip range = 0.2, value function coefficient = 0.5, group size K = 16, and top-k selection using random choice among the best 3 candidates.

Training was conducted for 100,000 timesteps using separate training and evaluation environments. The controller was evaluated every 10,000 timesteps over 25 episodes, and the best-performing checkpoint was automatically saved.

Final evaluation showed a strong difference between deterministic actor output and full grouped inference. Using only the actor mean action produced low success, while enabling grouped candidate selection achieved 100% task success with mean episode reward of -4.420 ± 2.961 across 300 evaluation episodes.

These results indicate that the learned actor primarily served as a proposal generator, while grouped ranking performed the final real-time control optimization. The resulting GRPO controller was subsequently used for all world-model and hybrid rollout experiments.

Comparative Interpretation

The PPO, SAC, and GRPO development stages revealed three distinct optimization patterns. PPO showed greater sensitivity to conservative learning rates, with very low learning rate settings underperforming within the fixed training budget. SAC was more robust across the tested hyperparameter grid, consistently achieving strong performance and stable off-policy learning behavior. GRPO followed a different optimization pattern because its performance depended not only on optimizer settings, but also on grouped-controller design choices such as candidate group size, rollout horizon, and top-k action selection strategy.

Final experiments showed that the deterministic actor alone performed poorly, while the full grouped inference controller achieved 100% success. This indicates that GRPO performance emerged from the interaction between learned policy proposals and grouped online action selection.

After final training, all three methods produced high-performing reaching controllers suitable for downstream comparison. PPO and SAC achieved perfect task success through direct policy inference, while GRPO achieved perfect task success when grouped candidate selection was enabled during evaluation. This ensured that subsequent hybrid rollout comparisons were not confounded by weak controller quality.

Role in the Overall Pipeline

The final PPO, SAC, and GRPO controllers served as the fixed control components for the remainder of the study. Each controller was paired with its corresponding learned world model and evaluated under identical synchronization benchmarks.

This created a controlled comparative framework: Policy $\in$ {PPO, SAC, GRPO}, World Model = policy-aligned learned dynamics model, Synchronization Interval N = variable

This design isolates whether hybrid rollout performance depends primarily on controller type, world-model accuracy, or synchronization strategy. Including GRPO strengthened the comparison by introducing a controller family based on grouped candidate search rather than direct action output. PPO represented an on-policy baseline, SAC represented an off-policy entropy-regularized baseline, and GRPO represented grouped-action control with online candidate ranking.

Conclusion

Three reinforcement learning controllers were developed for PandaReach-v3 using PPO, SAC, and GRPO. PPO and SAC underwent structured hyperparameter sweeps to select robust final configurations, while GRPO underwent iterative controller refinement as a grouped-action extension of PPO.

Following final training, PPO and SAC achieved perfect task success through standard policy inference, while GRPO achieved perfect task success using full grouped candidate-action evaluation. These controllers formed the foundation

© The National High School Journal of Science 2026

for all subsequent world-model learning and hybrid rollout experiments, enabling direct comparison between on-policy, off-policy, and grouped-action control within the proposed framework.

World Model Learning Framework

To reduce reliance on computationally expensive physics simulation, a learned dynamics model was introduced to approximate environment transitions. Instead of querying the simulator at every timestep, the model predicts state transitions directly from state–action inputs, enabling efficient hybrid rollout [16,24].

Policy-Aligned Dataset Generation

Training data was generated by rolling out trained PPO, SAC, and GRPO policies in the environment. Separate datasets were constructed for each controller to match the state–action distributions encountered during deployment.

For PPO and SAC, trajectories were collected directly from policy execution. For GRPO, actions were selected using grouped inference, but only the executed action was recorded. Each transition consisted of the current end-effector position, executed action, and resulting state change:

$$(x, y, z, a\_x, a\_y, a\_z, \Delta x, \Delta y, \Delta z) \tag{7}$$

Modeling transitions as state differences rather than absolute next states improved learning stability for small-step robotic motion.

PPO and SAC datasets were generated using 1000 rollout episodes with a maximum horizon of 125 steps, producing approximately 125,000 transition tuples per controller. GRPO data was generated using 300 grouped-inference rollout episodes, producing approximately 1,600 executed transition tuples due to the controller's low average episode length. No data augmentation techniques were applied.

State and Action Representation

A reduced Cartesian representation was used consisting only of end-effector position and control input:

$$x\_input = [x, y, z, a\_x, a\_y, a\_z] \tag{8}$$

$$y\_output = [\Delta x, \Delta y, \Delta z] \tag{9}$$

This first-order transition formulation is sufficient for PandaReach-v3, where dynamics are locally smooth and dominated by position control [16].

Neural Network Architecture

The dynamics model was implemented as a fully connected multilayer perceptron (MLP) to approximate the transition function, a common approach in learning-based dynamics modeling [16].

The architecture consisted of:

- input dimension: 6
- two hidden layers of size 256 with ReLU activation
- output dimension: 3

Preliminary experiments compared hidden layer sizes of 128, 256, and 512 units. The 128-unit configuration produced increased validation error, weaker prediction accuracy, and reduced rollout stability, particularly on GRPO-generated transitions, while 512 units provided only marginal improvement relative to increased computational cost. Empirically, the 256-unit architecture provided the best balance between prediction fidelity, rollout stability, and computational efficiency, and was therefore selected for all subsequent experiments.

The model was trained as a supervised regression problem using:

- Mean Squared Error (MSE) loss
- Adam optimizer (learning rate = 1e-3)
- batch size = 256
- 50 training epochs

These optimization settings were selected based on stable convergence behavior during preliminary experimentation and consistency with commonly used configurations in model-based reinforcement learning.

Input features were normalized using dataset statistics, while outputs were left unnormalized to preserve physical scale. The world model predicts state transitions (delta prediction) rather than absolute future states, improving optimization stability and prediction behavior for physical systems [13].

This supervised formulation is widely used in model-based reinforcement learning for approximating transition functions [24].

Summary of Model Configuration

Table 1 Model Configuration

Component	Specification
Input	(x,y,z,a_x,a_y,a_z)
Output	(Δx, Δy, Δz)
Architecture	MLP (256,256)
Activation	ReLU
Loss	MSE
Optimizer	Adam (1e-3)
Batch Size	256
Epochs	50

© The National High School Journal of Science 2026

Consistency Across Controllers

The model architecture and training procedure were kept identical across PPO, SAC, and GRPO datasets, ensuring that downstream performance differences are attributable to differences in the underlying data distributions rather than model design.

GRPO produced more diverse transitions due to its grouped action selection mechanism, increasing the complexity of the learned dynamics despite using the same modeling framework.

Summary

The proposed world model framework combines policy-aligned data collection with a consistent supervised learning pipeline, enabling fair comparison across controllers while ensuring that the learned dynamics remain representative of each policy's behavior during hybrid rollout.

Hybrid Rollout Control Framework

The hybrid rollout framework reduces the computational cost of environment interaction by replacing a subset of simulator steps with predictions from a learned dynamics model. This follows the general paradigm of model-based reinforcement learning, where learned models approximate environment transitions to reduce computational cost[24].

Hybrid Rollout Formulation

In standard RL, the policy interacts with the simulator at every timestep. In contrast, the hybrid rollout framework alternates between real environment transitions and model-predicted transitions. At each timestep, the system performs either a real simulator step or an imagined step using the learned world model, thereby reducing simulator calls while preserving trajectory evolution[16].

Synchronization Mechanism (N-Step Rollout)

The transition between real and imagined steps is governed by an interval N: if (t mod N = 0), a real environment step is performed; otherwise, the system performs an imagined step. Thus, only every N-th step queries the simulator, while the remaining steps rely on the model. This periodic real step acts as a correction mechanism to prevent long-term trajectory divergence[16].

State Propagation Using the World Model

During imagined steps, the next state is predicted using the current state s_t and action a_t. The model predicts the state change:

$$\Delta s_t = f\theta(s_t, a_t) \tag{10}$$

The next state is then computed as:

$$s_{t+1} = s_t + \Delta s_t \tag{11}$$

This formulation is consistent with world model training and enables iterative rollout for improved stability[10,13].

Observation Construction for Imagined Steps

Instead of querying the simulator, a synthetic observation is constructed: the end-effector position is replaced with the predicted state and the achieved goal is updated, while the desired goal remains constant. This ensures the policy operates on a consistent observation space regardless of whether the step is real or imagined.

Policy Execution

For PPO and SAC, policy execution remains unchanged. At each timestep, the observation (real or synthetic) is passed to the policy, which outputs a continuous action applied to either the simulator or world model. No modification is made to the policy architecture or inference procedure.

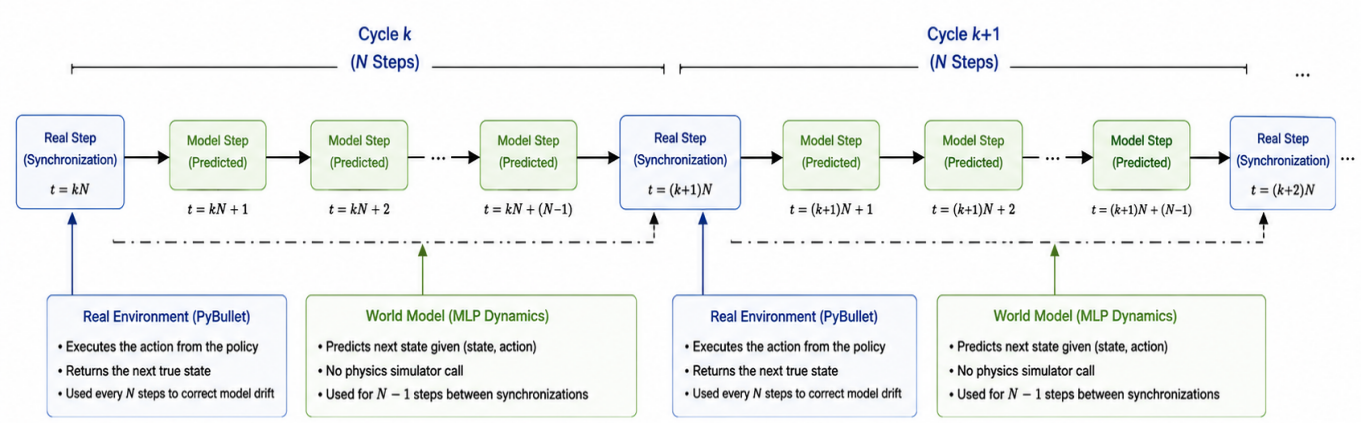

Fig. 2 Hybrid rollout with periodic synchronization. Every N-th step uses the real simulator, while intermediate steps use the learned world model.

GRPO Action Selection

In GRPO, action selection is performed using grouped candidate evaluation rather than direct policy output. At each timestep:

1. multiple candidate actions are sampled from the policy distribution
2. each candidate is evaluated based on predicted goal progress
3. the top-performing candidates are identified
4. one action is selected from this subset

Candidate evaluation uses a short-horizon approximation: predicted_next_state = current_state +(STEP_SCALE × action).

STEP_SCALE was fixed at 0.05 based on the typical magnitude of end-effector state changes observed in PandaReach-v3, providing a simple first-order approximation for evaluating candidate actions.

Each candidate is scored as:

$$\begin{aligned} \text{Score} = &- \text{distance}(\text{predicted\_next\_state}, \text{goal}) \\ &- \lambda ||\text{action}|| \end{aligned} \tag{12}$$

where λ controls the penalty applied to excessively large actions.

This selection strategy introduces additional diversity in executed actions, influencing the transition distributions observed by the world model.

Baseline Configuration

A pure environment baseline was defined using N = 1, where all transitions were executed directly through the PyBullet simulator without learned model prediction. This baseline served as the primary reference for evaluating runtime efficiency and task performance.

To isolate whether computational gains could be achieved simply by reducing simulator updates, an additional coarse-timestep physics baseline was evaluated using frame-skip values of $\{1, 2, 5, 10, 20\}$ across PPO, SAC, and GRPO controllers. In this baseline, the simulator executed repeated physics steps using the same action over multiple frames without world-model prediction.

Results showed that coarse-timestep execution substantially degraded task performance and trajectory quality compared to hybrid rollout. PPO success decreased from 100 percent at frame skip = 1 to 46 percent at frame skip = 20, while SAC decreased from 100 percent to 80 percent. GRPO performed poorly across all tested frame skips, remaining between 2 percent and 5 percent success. Runtime evaluation further showed that coarse timestep execution achieved only limited acceleration, reaching at most approximately 1.26× speedup despite substantial performance degradation.

In contrast, the proposed hybrid rollout framework achieved substantially larger acceleration while preserving significantly stronger short-horizon task acquisition performance across large synchronization intervals. These findings suggest that the observed efficiency-performance improvements arise from learned model-based transition substitution rather than naive reduction in simulator update frequency alone.

The repeated-action coarse-timestep baseline was selected because it represents a standard frame-skip approximation for reducing simulator interaction frequency without introducing additional policy inference overhead. Re-querying the policy at every physics step would substantially reduce the intended computational savings and therefore would not provide a directly comparable reduced-simulation baseline.

Summary

The hybrid rollout framework integrates learned dynamics with real environment interaction through periodic synchronization. By reducing simulator calls while maintaining trajectory accuracy, the framework enables efficient evaluation of PPO, SAC, and GRPO controllers under a unified control loop.

Evaluation Metrics

This section defines the metrics used to evaluate task performance, world model accuracy, and computational efficiency. These metrics were applied consistently across PPO, SAC, and GRPO to enable fair comparison.

Table 2 Comparison between coarse-timestep execution and hybrid rollout at synchronization interval N=20

Controller	Method	Interval	Success Rate	Speedup
PPO	Coarse timestep	20	0.46	1.10×
PPO	Hybrid rollout	20	1.00	2.63×
SAC	Coarse timestep	20	0.80	1.26×
SAC	Hybrid rollout	20	1.00	3.34×
GRPO	Coarse timestep	20	0.05	1.14×
GRPO	Hybrid rollout	20	1.00	2.39×

© The National High School Journal of Science 2026

Task Performance Metrics

Task performance was evaluated based on the ability of the end-effector to reach a target position in Cartesian space. A rollout was considered successful if the Euclidean distance between the achieved state s and goal g satisfied:

$$d = ||s - g|| < 0.05 \tag{13}$$

Two success criteria were evaluated. Any-time success measures whether the end-effector reached within 0.05 m of the goal at any timestep during the episode, while final-state success measures whether the end-effector remained within 0.05 m of the goal at the final evaluation timestep.

Any-time success captures short-horizon task acquisition, while final-state success measures long-horizon trajectory stability under recursive rollout. Success rate was computed as the fraction of rollouts satisfying these conditions. Mean episodic reward, episode length, and final goal distance were also recorded to measure trajectory quality and convergence efficiency.

Prediction Accuracy Metrics

World model accuracy was evaluated by comparing predicted state transitions with true environment transitions.

Mean Squared Error (MSE) measured absolute prediction error, while the coefficient of determination (R^2) measured how well predictions explained variance in the true dynamics. R^2 was computed separately for each spatial axis (x, y, z) to capture directional prediction fidelity.

These metrics are standard for evaluating learned dynamics models in model-based reinforcement learning [24].

Computational Efficiency Metrics

Computational performance was evaluated using runtime-based metrics. Inference time was measured as the wall-clock time required to execute a rollout of fixed length X. Speedup was defined relative to the pure environment baseline:

Speedup = (runtime of pure rollout) / (runtime of hybrid rollout)

A speedup greater than 1 indicates improved efficiency due to reduced simulator usage.

Summary

The evaluation framework combines task success, prediction accuracy, and computational efficiency metrics to provide a comprehensive assessment of hybrid rollout performance across controllers.

Experimental Design

This section describes the procedures used to evaluate (i) one-step prediction accuracy, (ii) long-horizon stability of learned dynamics, and (iii) hybrid rollout performance. All experiments were conducted in PandaReach-v3 using PPO, SAC, and GRPO controllers under consistent evaluation protocols.

World Model Validation Protocol

One-step prediction accuracy was evaluated using the Predicted Delta vs. Actual Delta test.

Transitions were collected under deployment conditions using trained controllers (PPO/SAC: deterministic inference; GRPO: grouped selection). For each configuration: episodes = 250, steps per episode = 5, minimum delta threshold = 0.005.

For each transition, true and model-predicted state changes were computed from the same state–action pair. Near-stationary transitions ($||\Delta|| < 0.05$) were filtered to reduce noise.

Prediction accuracy was evaluated using MSE and axis-wise R^2 values (X, Y, Z), providing a measure of local transition fidelity.

Table 3 Summary of evaluation metrics used for assessing task performance, prediction accuracy, and computational efficiency.

Summary of Metrics		
Category	**Metric**	**Purpose**
Task Performance	Success Rate	Task completion reliability
	Mean Reward	Control quality
	Episode Length	Efficiency of convergence
	Goal Distance	Continuous performance measure
Prediction Accuracy	MSE	Absolute prediction error
	R^2 (x, y, z)	Variance explained per axis
Efficiency	Inference Time	Runtime cost
	Speedup	Efficiency gain vs baseline

© The National High School Journal of Science 2026

Long-Horizon Imagination Testing

To evaluate multi-step stability, two complementary experiments were conducted.

Step Count vs. Euclidean Drift

This experiment measures how prediction error accumulates over repeated imagined steps. trials = 100, horizon = 30 steps.

Each trial begins with identical real and imagined states. The same action is applied to both the real environment and world model at every step. Drift is computed as:

$$drift(t) = ||s\_real - s\_imagined|| \tag{14}$$

Drift was recorded across timesteps and averaged over trials to capture long-horizon error accumulation.

Imagination-Only Rollout Evaluation

This experiment evaluates whether the controller can operate using only model-predicted states. trials = 1000, horizons: H $\in \{5, 10, 20, 30\}$, success threshold = 0.05 m.

Starting from the true initial state, rollouts were executed entirely using the world model without further environment interaction. At each step, actions were selected using synthetic observations derived from predicted states.

Performance was measured using: initial and final goal distance, distance reduction, success rate.

This experiment evaluates the usability of the learned model as a standalone simulator.

Hybrid Rollout Benchmark Setup

The hybrid rollout framework was evaluated by varying synchronization frequency between real environment interaction and model prediction.

Each configuration was defined by:

- rollout length: X $\in \{50, 100, 200, 500\}$
- synchronization interval: N $\in \{1, 2, 5, 10, \ldots, 100\}$

At each timestep, real or imagined transitions were applied based on N. PPO and SAC used direct action inference, while GRPO used grouped action selection.

Each configuration was evaluated over multiple episodes, recording: success rate, cumulative reward, episode length, final goal distance, runtime Performance was compared against the pure environment baseline (N = 1), with runtime measured using wall-clock timing over complete rollouts.

Summary

The experimental design consisted of three stages: one-step validation to assess local model accuracy, long-horizon testing to evaluate error accumulation, and hybrid rollout benchmarking to measure efficiency and control performance. Together, these experiments provide a structured evaluation of learned dynamics models and their integration into hybrid reinforcement learning systems.

Table 4 Summary of experimental stages and their corresponding evaluation objectives.

Experiment Type	Purpose
One-step validation	Local transition accuracy
Drift evaluation	Error accumulation over time
Imagination-only rollout	Model usability as simulator
Hybrid benchmark	Efficiency vs performance tradeoff

Implementation Details

This section summarizes the software environment, implementation choices, and reproducibility protocols used in this study.

Software and Frameworks

All experiments were implemented in Python using Gymnasium for environment interfacing, Panda-Gym for the PandaReach-v3 robotic manipulation environment, PyTorch for neural network modeling and training, and Stable-Baselines3 (SB3) for PPO and SAC baselines. GRPO was implemented as a custom extension within the Stable-Baselines3 framework while maintaining compatibility with its policy and training interfaces.

GRPO Implementation Details

The GRPO controller was implemented by modifying the standard PPO policy execution pipeline to incorporate grouped action sampling. At each decision step, multiple candidate actions were sampled from the policy distribution, evaluated using a goal-directed scoring function, ranked, and one action was randomly selected from the top-K subset.

The scoring function used a short-horizon approximation of goal progress based on a scaled action step and included an action-magnitude penalty. This grouped selection mechanism was used consistently during training, evaluation, hybrid rollout, and imagination-based experiments.

Hardware Setup

All experiments were conducted on a local machine equipped with an Intel Core i7-10750H CPU running at 2.60 GHz, 64 GB RAM, and an NVIDIA GeForce GTX 1650 GPU with 4 GB VRAM. Environment simulation was executed primarily on CPU, while neural network training and inference used optional GPU acceleration through PyTorch when available.

© The National High School Journal of Science 2026

Reproducibility

All experiments were conducted in PandaReach-v3 using identical evaluation protocols across PPO, SAC, and GRPO. Deterministic policy inference was used for PPO and SAC during evaluation, while GRPO used a fixed grouped action selection procedure with predefined parameters.

Dataset generation, world model training, and evaluation scripts were kept consistent across all controllers to ensure fair comparison under identical experimental conditions. GRPO grouped candidate evaluation used a fixed STEP_SCALE = 0.05 across all experiments.

Summary

The implementation follows a unified framework combining Stable-Baselines3 baselines with a custom GRPO extension, enabling consistent experimentation across multiple reinforcement learning paradigms. All components were designed to ensure comparability, reproducibility, and compatibility within the hybrid rollout framework.

Reproducibility Details

The world model was implemented as an MLP that takes the current state and action as input and predicts the corresponding state transition. The model was trained using supervised learning on transition data collected from PPO, SAC, and GRPO policy rollouts in PandaReach-v3.

Training used Mean Squared Error (MSE) loss, the Adam optimizer (learning rate = 1×10^{-3}), batch size = 256, and 50 training epochs. All evaluation experiments used fixed random seeds and identical synchronization configurations across controllers to ensure fair comparison.

Success-rate evaluation was performed across 300 episodes per configuration, while runtime benchmarks were averaged across 10 independent runs. All experiments followed the same rollout horizon, synchronization procedure, and evaluation methodology described throughout the paper, enabling reproducibility of the reported results.

Results and Discussion

Baseline Controller Performance under Real Environment Interaction

This section evaluates PPO, SAC, and GRPO controllers on PandaReach-v3 under standard environment interaction. All methods were evaluated over 300 episodes using deterministic inference for PPO and SAC, and both deterministic and grouped inference for GRPO.

PPO and SAC Performance

Both PPO and SAC achieved perfect task success, with success rates of 1.00 across 300 episodes. PPO obtained a mean episodic reward of -1.680 ± 0.859, while SAC achieved -1.603 ± 0.770. The low reward variance indicates consistent behavior across episodes for both methods.

These results confirm that PPO and SAC learn stable and reliable policies for PandaReach-v3. Their similar performance suggests that both on-policy and off-policy methods can achieve near-optimal behavior for this relatively low-dimensional control task[24].

GRPO Policy Performance

GRPO was evaluated using deterministic actor output and grouped action selection. Under deterministic inference, GRPO achieved a success rate of 0.21 with mean reward -42.05 ± 16.39, indicating unstable and ineffective task execution when relying solely on the policy mean action.

In contrast, grouped action selection achieved a success rate of 1.00 with mean reward 4.420 ± 2.961, representing substantial improvement in both reliability and task performance.

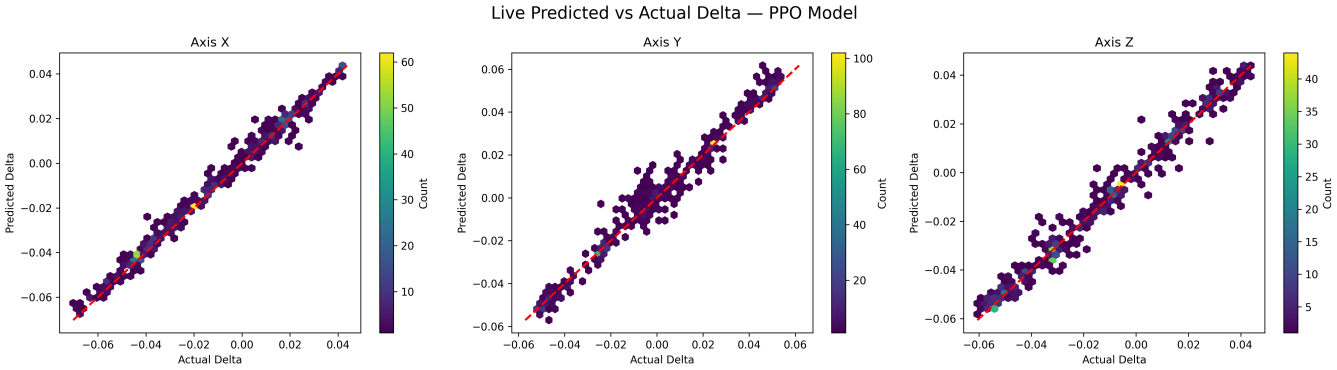

Fig. 3 Predicted vs. Actual Delta for the PPO world model across X, Y, and Z axes. The predictions closely align with the diagonal, indicating near-perfect one-step transition accuracy ($R^2 \approx 0.98 - 0.99$) with minimal error.

© The National High School Journal of Science 2026

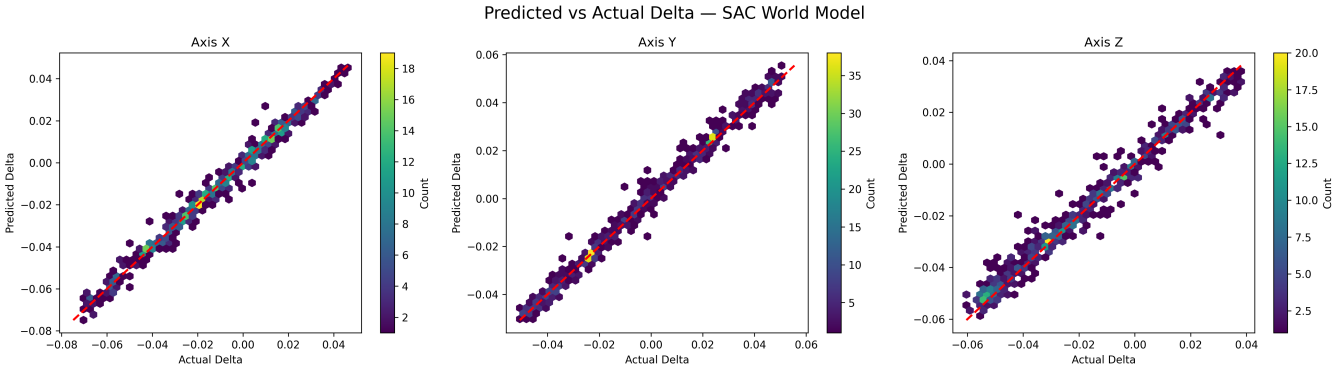

Fig. 4 Predicted vs. Actual Delta for the SAC world model across X, Y, and Z axes. The predictions exhibit strong alignment with the diagonal, demonstrating high one-step prediction fidelity ($R^2 \approx 0.97 - 0.99$) across all axes.

Deterministic vs Grouped GRPO

The large performance gap between deterministic and grouped GRPO highlights a fundamental difference in how the policy should be interpreted. The deterministic actor output alone does not reliably solve the task; however, the learned policy distribution contains useful action directions that can be exploited through sampling-based candidate selection[24].

This demonstrates that GRPO does not function as a standard deterministic actor policy. Instead, it operates as a sampling-based controller whose performance depends on evaluating multiple candidate actions rather than executing a single direct policy output.

Comparative Discussion

PPO and SAC produce policies whose deterministic outputs are directly suitable for deployment, resulting in stable and consistent performance. In contrast, GRPO relies on an additional grouped action-selection mechanism to achieve comparable task success.

Although grouped GRPO achieves the same success rate as PPO and SAC, it exhibits lower reward efficiency, indicating that while the controller reliably reaches the goal, its trajectories are less optimal.

Summary

PPO and SAC achieve near-identical and optimal performance on PandaReach-v3 under deterministic control. GRPO, while ineffective under deterministic inference, achieves equivalent success rates when combined with grouped action selection. These results establish grouped GRPO as the appropriate evaluation mode for subsequent world-model and hybrid rollout experiments. All baseline controller statistics reported in Sections ”RL Policy Training (PPO, SAC, and GRPO)” and ”Baseline Controller Performance under Real Environment Interaction” correspond to the same standardized deployment evaluation protocol conducted over 300 episodes.

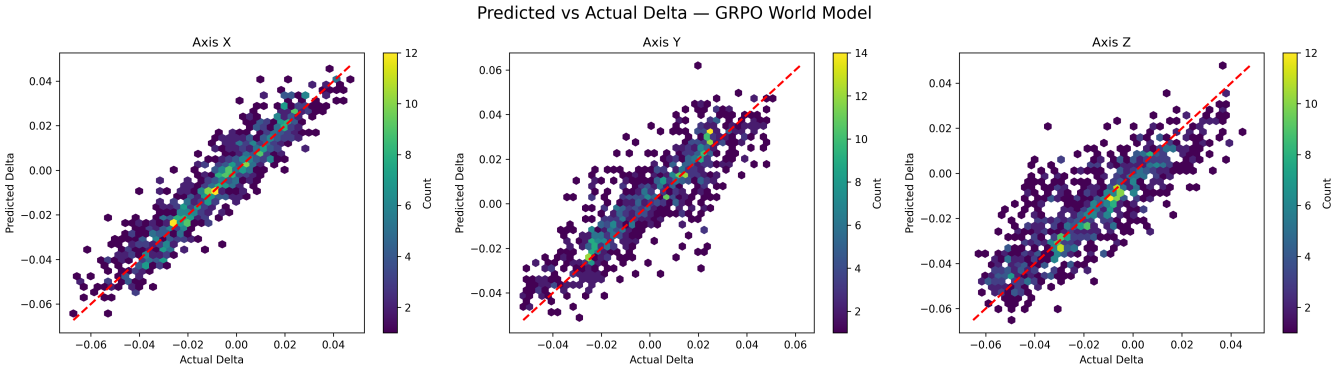

Fig. 5 Predicted vs. Actual Delta for the GRPO world model across X, Y, and Z axes. The predictions follow the general diagonal trend but show increased dispersion, particularly along the Y and Z axes, reflecting comparatively lower but still robust accuracy.

© The National High School Journal of Science 2026

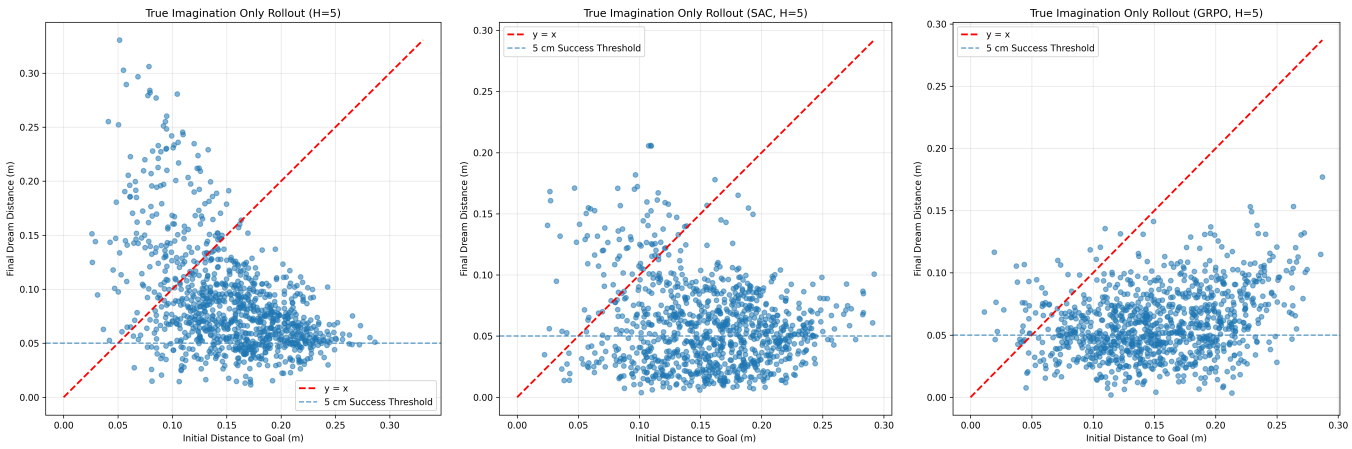

Fig. 6 Imagination-only rollout performance at H = 5 for PPO, SAC, and GRPO. All controllers demonstrate effective short-horizon control, with consistent reduction in distance to the target.

World Model Prediction Accuracy

This section evaluates the one-step transition accuracy of learned world models trained using PPO, SAC, and GRPO-generated datasets. Since hybrid rollout replaces real environment interaction with model-predicted transitions, accurate short-horizon dynamics modeling is critical for limiting recursive error accumulation.

One-Step Prediction Performance

The Predicted Delta vs. Actual Delta test measured how accurately each world model predicts the change in end-effector position after a single action. Prediction quality was evaluated using coefficient of determination (R^2) and Mean Squared Error (MSE) across the X, Y, and Z axes.

The PPO world model achieved near-perfect predictive performance: X-axis, R^2 = 0.9903 and MSE = 0.000008; Y-axis, R^2 = 0.9847 and MSE = 0.000015; Z-axis, R^2 = 0.9831 and MSE = 0.000013. Figure 3 shows predicted and actual deltas tightly aligned along the diagonal with minimal dispersion, confirming extremely high one-step prediction fidelity.

The SAC world model similarly achieved near-perfect accuracy: X-axis, R^2 = 0.9874 and MSE = 0.000009; Y-axis, R^2 = 0.9905 and MSE = 0.000008; Z-axis, R^2 = 0.9775 and MSE = 0.000015. Figure 4 shows strong diagonal alignment with very low spread across all axes, indicating highly accurate modeling of local dynamics.

The GRPO world model achieved strong but comparatively lower predictive performance: X-axis, R^2 = 0.8709 and MSE = 0.000065; Y-axis, R^2 = 0.7391 and MSE = 0.000129; Z-axis, R^2 = 0.6896 and MSE = 0.000159. Figure 5 shows noticeably greater dispersion, particularly along the Y and Z axes, reflecting reduced precision relative to PPO and SAC while still maintaining strong alignment with true dynamics.

Axis-Wise Analysis

Across all controllers, prediction accuracy varied across spatial dimensions. PPO exhibited highly consistent performance across X, Y, and Z axes, while SAC showed similarly strong alignment with slightly greater dispersion. GRPO exhibited the strongest alignment along the X-axis, with progressively larger variance along the Y and Z axes, indicating increased modeling difficulty for more nonlinear vertical motion.

Comparative Discussion

The observed differences in predictive performance between PPO/SAC and GRPO can largely be attributed to differences in the underlying data distribution. PPO and SAC generate smooth, policy-consistent trajectories, producing highly structured and predictable state transitions. This is reflected in Figures 3 and 4, where predicted values closely follow the ideal diagonal with minimal variance.

In contrast, GRPO uses grouped action selection, where multiple candidate actions are sampled and evaluated before execution. This produces a more diverse and adaptive transition distribution, increasing modeling complexity. Figure 5 correspondingly exhibits greater dispersion in predicted deltas, particularly along the Y and Z axes.

This highlights a trade-off between structured trajectories that are easier to model (PPO/SAC) and more diverse trajectories that are harder to predict but potentially more expressive (GRPO).

Implications for Hybrid Rollout

These results confirm that all three world models provide sufficiently accurate one-step predictions for hybrid rollout. The near-perfect accuracy of PPO and SAC suggests minimal short-term prediction error, while the GRPO model, de-

spite lower one-step R^2 values, still demonstrates strong alignment with true dynamics and remains suitable for integration into the hybrid control framework. Importantly, one-step prediction accuracy alone does not determine whether a world model is suitable for hybrid rollout. Practical usability depends on recursive error accumulation, controller robustness to state inaccuracies, and synchronization frequency during deployment. Although the GRPO model achieved lower one-step accuracy than PPO and SAC, subsequent long-horizon experiments showed that hybrid rollout remained effective at moderate synchronization intervals despite increased drift accumulation.

These results suggest that there is no single universal R^2 threshold for usable hybrid rollout performance; instead, rollout stability emerges from the interaction between local prediction accuracy, recursive error growth, and periodic correction frequency. Furthermore, the structured nature of prediction errors observed in Figures 3–5 suggests that errors are not purely random, which is important for maintaining stability during multi-step imagined rollouts.

Summary

The Predicted Delta vs. Actual Delta test demonstrates that PPO and SAC world models achieve near-perfect one-step prediction accuracy, while the GRPO world model achieves slightly lower but still robust performance. Figures 3–5 confirm strong alignment between predicted and actual transitions across all controllers, supporting the use of these models in subsequent hybrid rollout experiments.

Long-Horizon Prediction Stability

While Section "World Model Prediction Accuracy" demonstrated strong one-step prediction accuracy, this does not guarantee stability under recursive rollout. To evaluate long-horizon behavior, pure imagination rollouts were conducted for horizons H $\in \{5, 10, 20, 30\}$, where control was performed entirely using model-predicted states. Figures 6–8 show imagination-only rollout behavior for PPO, SAC, and GRPO at horizons H = 5, 10, and 30 respectively.

Imagination-Only Rollout Performance PPO

PPO exhibited limited robustness under imagination-based control. At H = 5 (Figure 6), mean distance decreased from 0.154 m to 0.088 m (reduction = +0.066 m), achieving 18.2% success. However, performance degraded rapidly beyond short horizons. At H = 10 (Figure 7), mean final distance increased to 0.534 m (reduction = −0.380 m), indicating divergence, while at H = 30 (Figure 8), mean final distance reached 41.639 m with success rates remaining below 10%. These results show that PPO cannot sustain stable recursive imagination beyond very short horizons.

SAC demonstrated stronger short-horizon performance than PPO. At H = 5 (Figure 6), mean distance decreased from 0.156 m to 0.058 m (reduction = +0.098 m), achieving 46.0% success. However, similar degradation occurred as horizon increased. At H = 10 (Figure 7), mean final distance increased to 0.424 m (reduction = −0.270 m), while at H = 30 (Figure 8), mean final distance reached 30.097 m. Despite stronger initial performance, SAC exhibited the same long-horizon failure pattern.

GRPO demonstrated substantially improved robustness under imagination-based control. At H = 5 (Figure 6), mean distance decreased from 0.1525 m to 0.0594 m (reduction = +0.0931 m), achieving 41.4% success. Performance further improved at H = 10 (Figure 7), where final distance decreased to 0.0565 m, reduction increased to +0.0995 m, and success

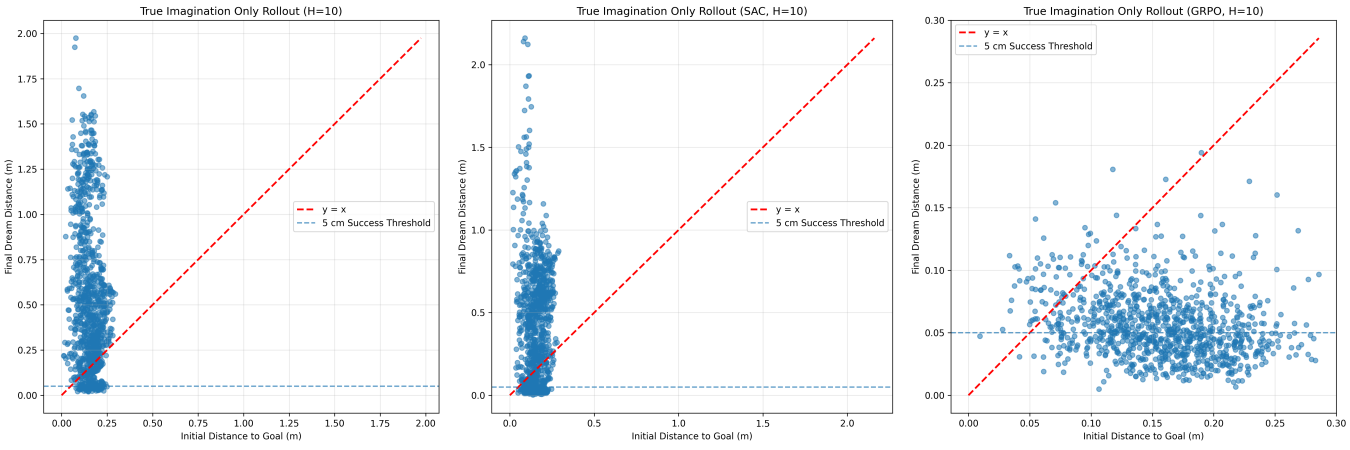

Fig. 7 Imagination-only rollout performance at H = 10 for PPO, SAC, and GRPO. PPO and SAC begin to degrade due to compounding prediction error, while GRPO maintains strong performance and achieves improved task completion.

© The National High School Journal of Science 2026

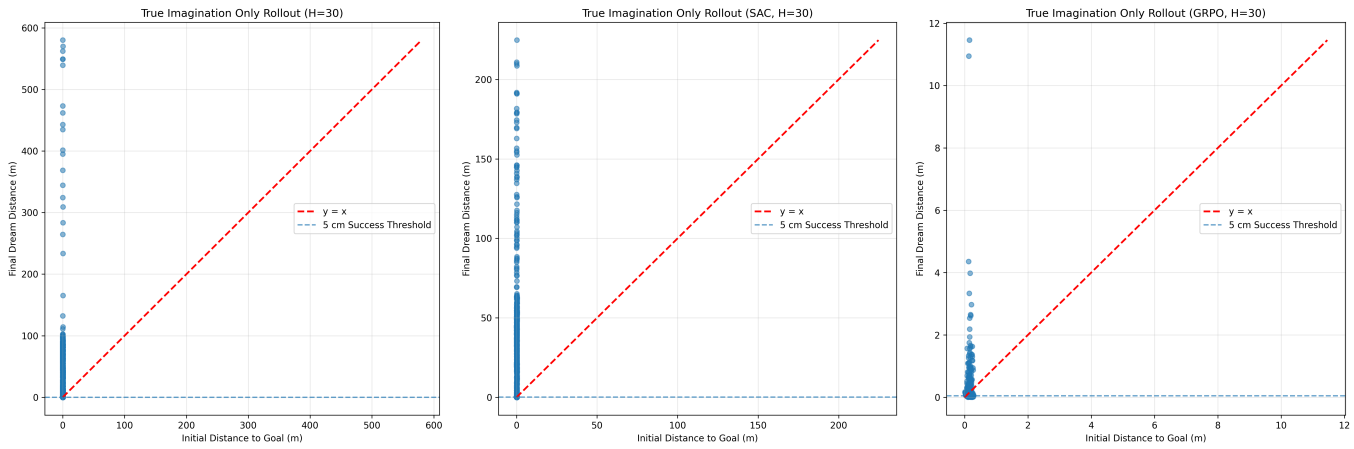

Fig. 8 Imagination-only rollout performance at H = 30 for PPO, SAC, and GRPO. PPO and SAC exhibit severe divergence, whereas GRPO degrades more gradually and retains partial task effectiveness.

rate increased to 47.1%. This suggests that grouped action selection benefits from additional decision steps before model error becomes dominant.

At H = 20, GRPO remained functional with final distance = 0.0894 m, reduction = +0.0632 m, and success rate = 33.7%. At H = 30 (Figure 8), degradation remained gradual rather than catastrophic, with final distance = 0.1813 m, reduction = −0.0283 m, and success rate = 31.6%. Unlike PPO and SAC, GRPO retained partial task effectiveness even under long recursive imagination.

Comparative Analysis

Figures 6–8 illustrate the transition from stable to unstable behavior as imagination horizon increases. PPO and SAC achieved strong short-horizon performance but diverged rapidly beyond H $\geq$ 10. In contrast, GRPO degraded more gradually, with improved performance between H = 5 and H = 10 suggesting that grouped action selection benefits from additional decision steps before accumulated model error dominates behavior.

Even at H = 20, GRPO maintained positive distance reduction, while at H = 30 it retained non-trivial success rates. This indicates that grouped action selection provides greater robustness to recursive prediction error than direct deterministic policy execution.

Implications for Hybrid Rollout

These results show that the learned world models are reliable for short imagination horizons but unstable under long recursive rollout. This behavior, visible in Figures 6–8, directly motivates the hybrid rollout framework, where short model-based rollouts are periodically corrected using real environment interaction.

Notably, SAC pure imagination rollouts diverged at H = 10, while subsequent hybrid rollout experiments maintained 100% task success at synchronization intervals as large as N = 45. This apparent contradiction arises because hybrid rollout periodically re-synchronizes predicted trajectories with the true environment, preventing recursive prediction error from compounding indefinitely. In pure imagination, prediction error accumulates continuously without correction, eventually causing catastrophic divergence. In contrast, periodic real-environment correction constrains accumulated drift and restores trajectory alignment before instability becomes dominant.

Summary

Pure imagination control is effective only over short horizons. PPO and SAC exhibit rapid instability beyond H = 10, while GRPO demonstrates improved robustness due to grouped action selection. These findings establish the necessity of periodic correction and motivate the proposed hybrid rollout framework.

Step Count vs. Euclidean Drift

To isolate recursive prediction error accumulation, a Step Count vs. Euclidean Drift experiment was conducted. Starting from identical initial states, the same action sequence was applied to both the real environment and learned world model. Drift at timestep t is defined as:

$$drift(t) = ||s\_real(t) - s\_model(t)|| \tag{15}$$

This represents a strict open-loop evaluation in which the policy acts only on imagined states without mid-rollout correction.

© The National High School Journal of Science 2026

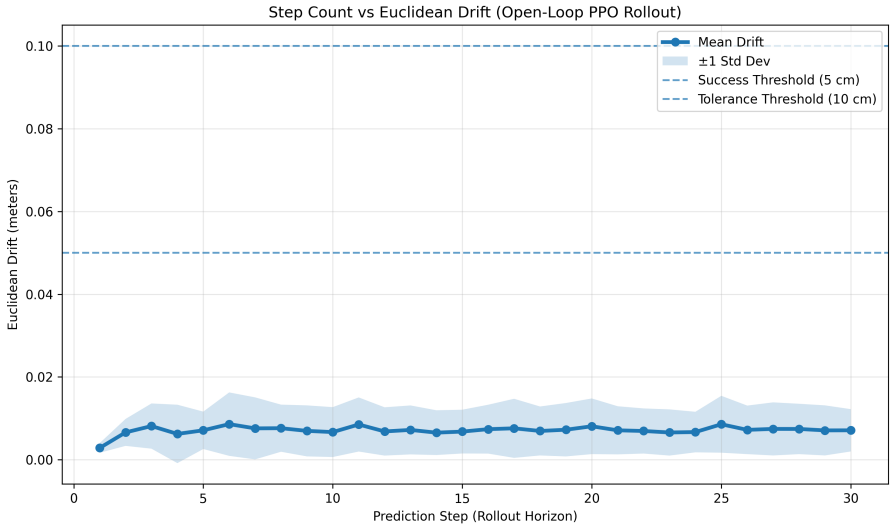

Fig. 9 Step count vs. Euclidean drift for the PPO world model. Drift stabilizes at 0.006–0.009 m, remaining well below the 0.05 m threshold.

As shown in Figure 9, the PPO world model exhibited bounded drift with three phases: initial drift ≈ 0.003 m (steps 1–3), small early growth (steps 2–6), and a plateau between 0.006–0.009 m across steps 6–30. Drift remained far below both the 0.05 m success threshold and 0.10 m tolerance throughout the full horizon, indicating stable long-horizon prediction without runaway divergence.

Figure 10 shows similar bounded behavior for SAC: initial drift ≈ 0.0025 m, early growth to ≈ 0.0054–0.0072 m, and a stable plateau between 0.005–0.008 m over steps 4–30. Drift again remained well below both threshold values, confirming high-fidelity long-horizon prediction under repeated rollout.

In contrast, Figure 11 shows progressive drift accumulation for GRPO. Drift increased from 0.0073 m to 0.0259 m within the first three steps, crossed 0.05 m near step 12, and reached 0.1558 m by step 30. Standard deviation also increased substantially with horizon, from 0.0035 m at step 1 to 0.2035 m at step 30, indicating widening trajectory variance. Drift remained below 0.10 m until approximately step 22, after which larger prediction errors became increasingly common.

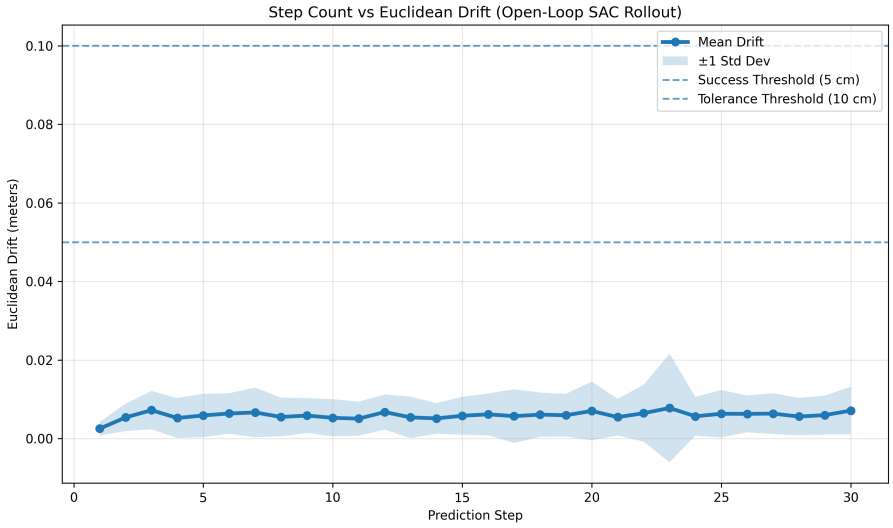

Fig. 10 Step count vs. Euclidean drift for the SAC world model. Drift stabilizes at 0.005–0.008 m, remaining well below the 0.05 m threshold.

© The National High School Journal of Science 2026

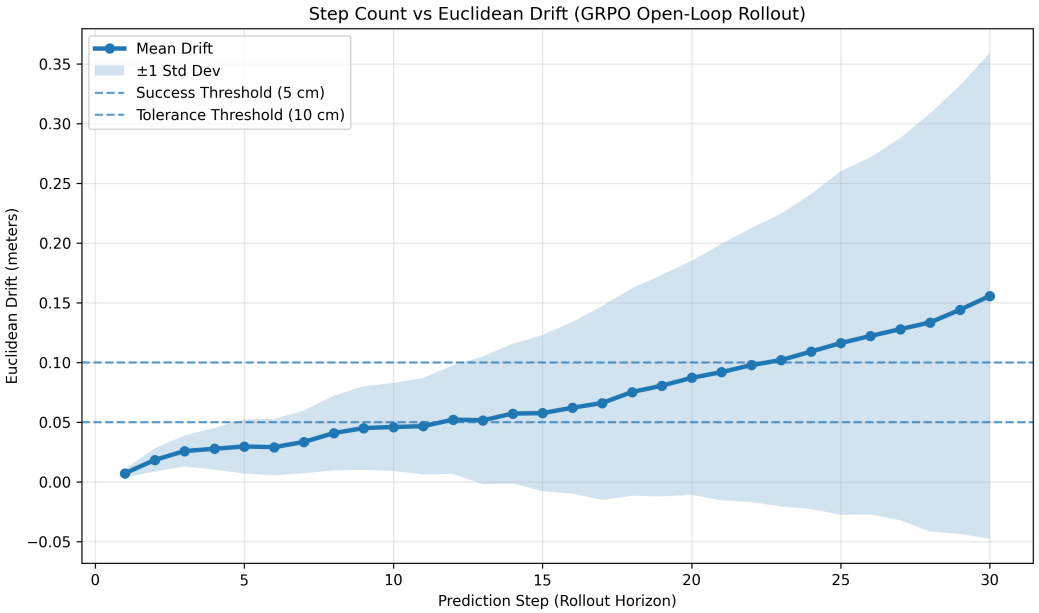

Fig. 11 Step count vs. Euclidean drift for the GRPO world model. Drift increases from 0.007 m to 0.156 m, crossing 0.05 m at $\sim$ step 12.

Interpretation

These results clarify the behaviors observed in Figures 6–8. Despite bounded millimeter-scale drift, PPO and SAC still exhibited degraded long-horizon imagination performance, suggesting that repeated recursive rollout amplifies even small state inaccuracies through the controller's closed-loop decision process. A formal sensitivity analysis of policy response to state perturbation was beyond the scope of this study.

GRPO exhibited gradual centimeter-scale drift growth, but grouped action selection provided partial robustness, allowing useful performance at medium horizons (approximately 10–20 steps) before accumulated error became dominant. One possible explanation is that grouped candidate selection provides partial robustness to moderate state inaccuracies by evaluating multiple alternative actions before execution, although formal robustness analysis was not performed.

Overall, drift behavior differed qualitatively between controllers: PPO and SAC exhibited bounded non-divergent error, while GRPO showed monotonic drift growth with increasing variance.

Summary Extension

The drift experiment shows that PPO and SAC maintain bounded millimeter-scale error over 30 steps, while GRPO accumulates centimeter-scale error, reaching approximately 15.6 cm by step 30. Practical safe imagination horizons are therefore approximately $\leq$10–12 steps across all methods, with GRPO remaining usable up to roughly 20 steps before significant degradation. Together with the imagination-only rollout results, these findings support short model rollouts with periodic synchronization as the effective operating regime for hybrid control.

Hybrid Rollout Performance

This section evaluates the hybrid rollout framework under deployment conditions by analyzing computational efficiency, task success preservation, and the combined efficiency–performance trade-off across PPO, SAC, and GRPO controllers using varying synchronization intervals N.

Synchronization Interval vs Inference Time

Computational efficiency was evaluated using fixed-horizon timing benchmarks with rollout lengths T $\in \{50, 100, 200, 500\}$, ensuring runtime differences arose solely from synchronization frequency. All timing measurements were averaged across 10 independent benchmark runs and are reported as mean ± standard deviation. All reported runtime and speedup values correspond to the same standardized benchmark evaluation protocol.

As shown in Figure 12, inference time decreases consistently as N increases across all controllers.

This occurs because learned model inference requires only a lightweight MLP forward pass, whereas each PyBullet simulation step requires substantially more expensive dynamics computation. Benchmark evaluation showed that a single MLP inference step required approximately 0.083 $\pm$ 0.004 ms

© The National High School Journal of Science 2026

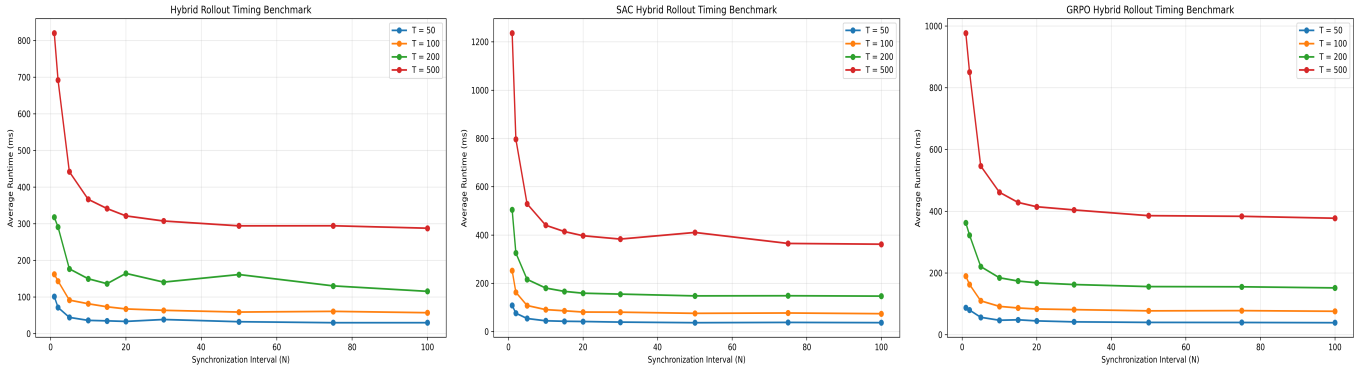

Fig. 12 Synchronization interval vs inference time for PPO, SAC, and GRPO. Runtime decreases as N increases, with diminishing returns at higher values.

for GRPO, 0.087 ± 0.002 ms for SAC, and 0.087 ± 0.003 ms for PPO. In contrast, a full PyBullet simulation step required approximately 1.531 ± 0.013 ms, 1.542 ± 0.009 ms, and 1.536 ± 0.028 ms respectively, making physics simulation approximately 18× slower than learned model inference across all controllers.

Although individual physics steps were substantially slower than learned model inference, overall rollout acceleration was lower due to additional fixed overheads including policy inference, synchronization logic, observation construction, and controller execution.

For PPO at T = 500, runtime decreased from 796.87 ± 17.52 ms (N = 1) to 303.20 ± 9.41 ms (N = 100), achieving 2.63× speedup. SAC achieved stronger acceleration, decreasing from 1239.46 ± 29.92 ms to 371.04 ± 10.32 ms, corresponding to 3.34× speedup. GRPO also achieved substantial gains despite grouped action computation, decreasing from 910.60 ± 19.31 ms to 381.66 ± 8.95 ms, corresponding to 2.39× speedup.

Across all controllers, the largest improvements occurred when increasing N from 1 to approximately 10–30. Beyond this range, runtime reduction saturated due to fixed overheads such as policy inference and neural network evaluation. Because speedup values were derived directly from repeated runtime measurements, the corresponding runtime standard deviations indicate that speedup remained stable across benchmark trials.

Synchronization Interval vs Success Rate

The impact of reduced simulator interaction on task reliability was evaluated using success rate across 300 episodes per configuration.

As shown in Figure 13, all controllers maintained near-perfect any-time success across a wide range of synchronization intervals. However, additional analysis using a stricter final-state success criterion revealed substantial degradation in long-horizon trajectory stability at larger synchronization intervals.

For PPO, any-time success remained near 100 percent across large synchronization intervals, but final-state success degraded substantially as N increased. At N = 10, final-state success was only 18 percent and decreased further at larger synchronization intervals as recursive prediction error accumulated. Runtime measurements remained highly consistent across synchronization settings; for example, at T = 500 and N = 100, runtime was 303.20 ± 9.41 ms while maintaining near-perfect task success.

SAC similarly maintained near-perfect any-time success across large synchronization intervals, but final-state success decreased from 45 percent at N = 10 to 25 percent at N = 100, indicating progressive long-horizon instability despite successful short-horizon target acquisition. Runtime measurements again remained highly consistent, with runtime at T = 500 and N = 100 measured at 371.04 ± 10.32 ms while maintaining 100 percent success.

GRPO also maintained near-perfect any-time success across all tested synchronization intervals. However, final-state success remained low across all evaluated horizons, ranging from approximately 10–16 percent, while final goal distance increased substantially at large synchronization intervals. Runtime measurements remained stable across repeated evaluations; for example, at T = 500 and N = 100, runtime was 381.66 ± 8.95 ms with corresponding 2.39× speedup relative to the pure-environment baseline.

Despite preserved any-time success, reward values degraded substantially at very large N. Additional trajectory analysis showed that large synchronization intervals produced substantial increases in final goal distance despite maintained task completion, indicating degraded trajectory quality and accumulated recursive prediction error during long imagined rollouts. For example, PPO reward decreased from −1.05 at N = 10 to −1060 at N = 100.

© The National High School Journal of Science 2026

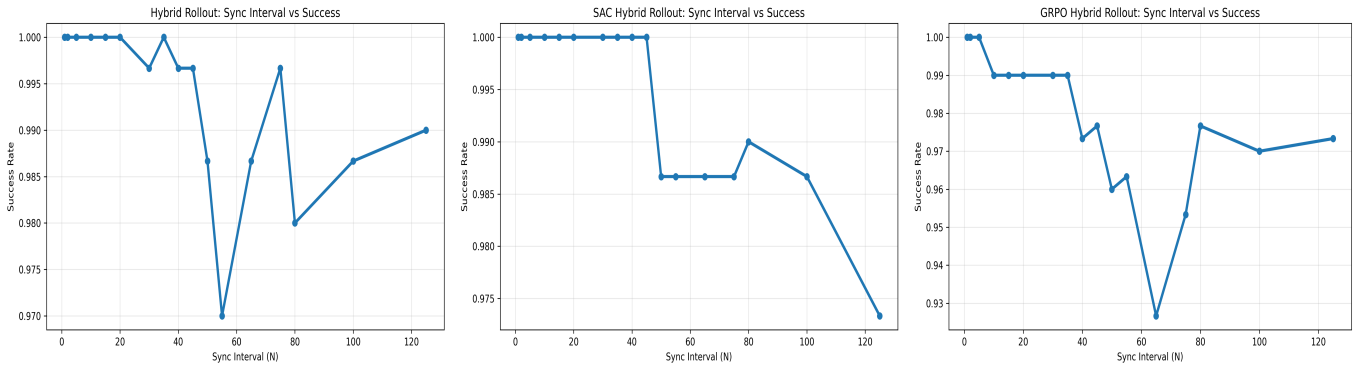

Fig. 13 Synchronization interval vs success rate for PPO, SAC, and GRPO. Success remains near 100% across most intervals, with minor degradation at very high N.

The discrepancy between any-time success and final-state success explains the severe reward degradation observed at large synchronization intervals. In many episodes, the controller briefly reached the target region during rollout execution, satisfying the any-time success criterion, but subsequently drifted away due to accumulated recursive prediction error during imagined rollout. This same phenomenon was observed at larger synchronization intervals such as N = 125, where controllers often briefly entered the target region before diverging during subsequent imagined rollout.

One important observation is that pure imagination rollout and hybrid rollout behave fundamentally differently despite both relying on recursive model prediction. Section "Imagination-Only Rollout Performance PPO" showed that SAC imagination-only rollouts diverged at H = 10, whereas hybrid rollout maintained near-perfect any-time success at synchronization intervals as large as N = 45. These results are not contradictory because pure imagination constitutes a fully open-loop process in which all future policy decisions depend entirely on recursively predicted states, allowing small prediction errors to compound continuously through the controller's feedback process.

In contrast, hybrid rollout periodically re-synchronizes with the true environment, preventing long-term state error accumulation. Although imagined states may drift locally between corrections, each synchronization step reprojects the trajectory back onto true system dynamics before divergence becomes catastrophic.

Importantly, the any-time success metric evaluates whether the controller reaches the target region at any point during rollout rather than requiring long-horizon trajectory stability. As shown in Table 7, final-state success degraded substantially at large synchronization intervals despite preserved any-time success, indicating that recursive prediction error still accumulated during long imagined rollout. Thus, periodic correction does not eliminate model drift, but constrains it sufficiently to preserve short-horizon task acquisition while substantially reducing simulator interaction.

These results indicate that periodic synchronization helps maintain stable control under hybrid rollout, although larger synchronization intervals can still produce accumulated model drift during long imagined rollouts.

Trajectory Stability Analysis

To evaluate whether successful episodes at large synchronization intervals reflected genuine goal-directed behavior, qualitative trajectory analysis was performed across multiple synchronization settings. Low and moderate synchronization intervals generally preserved stable motion toward the target,

Table 5 Any-Time Success vs Final-State Success Under Hybrid Rollout

Controller	N	Any-Time Success	Final-State Success	Mean Final Distance
PPO	10	1.0	0.18	0.441
PPO	100	1.0	0.16	4.92×10^7
SAC	10	1.0	0.45	0.058
SAC	100	1.0	0.25	2.47×10^5
GRPO	10	1.0	0.10	0.480
GRPO	100	1.0	0.10	4.83×10^6

© The National High School Journal of Science 2026

while larger synchronization intervals produced increasingly unstable trajectories due to accumulated model drift during recursive imagined rollout. This instability was also reflected quantitatively through substantial increases in final-distance measurements at high synchronization intervals despite maintained any-time success.

These observations were consistent with the drift accumulation analysis presented earlier, where cumulative prediction error increased progressively with recursive rollout depth across PPO, SAC, and GRPO controllers. GRPO exhibited monotonic drift growth with increasing variance, while PPO and SAC showed smaller but persistent recursive prediction error accumulating over longer imagined horizons. Together, these results indicate that successful task completion alone does not necessarily imply stable or efficient control behavior under long imagined rollout. Instead, periodic synchronization remains necessary to constrain accumulated prediction error and preserve stable long-horizon trajectory execution.

Final Hybrid Rollout Benchmark: Efficiency vs Performance

The final benchmark integrates runtime and success rate into a unified evaluation of the hybrid rollout trade-off.

As shown in Figures 14–16, hybrid rollout achieves substantial runtime reduction while maintaining high task success across all rollout horizons. For PPO (Figure 14), runtime decreases from 796.87 $\pm$ 17.52 ms to 303.20 $\pm$ 9.41 ms at T = 500, corresponding to 2.63× speedup while maintaining near-perfect any-time success across most synchronization intervals.

For SAC (Figure 15), runtime decreases from 1239.46 ± 29.92 ms to 371.04 ± 10.32 ms at T = 500, corresponding to 3.34× speedup while maintaining near-perfect any-time success across most synchronization intervals.

For GRPO (Figure 16), runtime decreases from 910.60 ± 19.31 ms to 381.66 ± 8.95 ms, corresponding to 2.39× speedup while maintaining near-perfect any-time success.

Across all controllers, the most effective operating region lies approximately within N = 10–30, where speedups of roughly 2.2×–3.3× are achieved while maintaining near-perfect success and controlled model error. At larger synchronization intervals, runtime gains begin to saturate while trajectory quality degrades due to accumulated recursive prediction error.

Comparative Analysis Across Controllers

Figures 12–16 reveal consistent trends across PPO, SAC, and GRPO, alongside meaningful controller-specific differences. Comparison against the coarse-timestep physics baseline showed that simply reducing simulator update frequency substantially degraded task performance, whereas hybrid rollout maintained high task success at comparable acceleration levels. These results suggest that the observed efficiency–performance improvements arise from learned model-based substitution of intermediate transitions rather than naive reduction in simulator update frequency alone.

PPO achieved strong acceleration while maintaining stable performance across moderate synchronization intervals, with consistently low runtime variance and reliable control under periodic synchronization.

SAC achieved the largest speedups due to its higher baseline end-to-end rollout runtime under the evaluation pipeline used in this study. Although PyBullet physics-step cost is approximately controller-independent, total rollout runtime

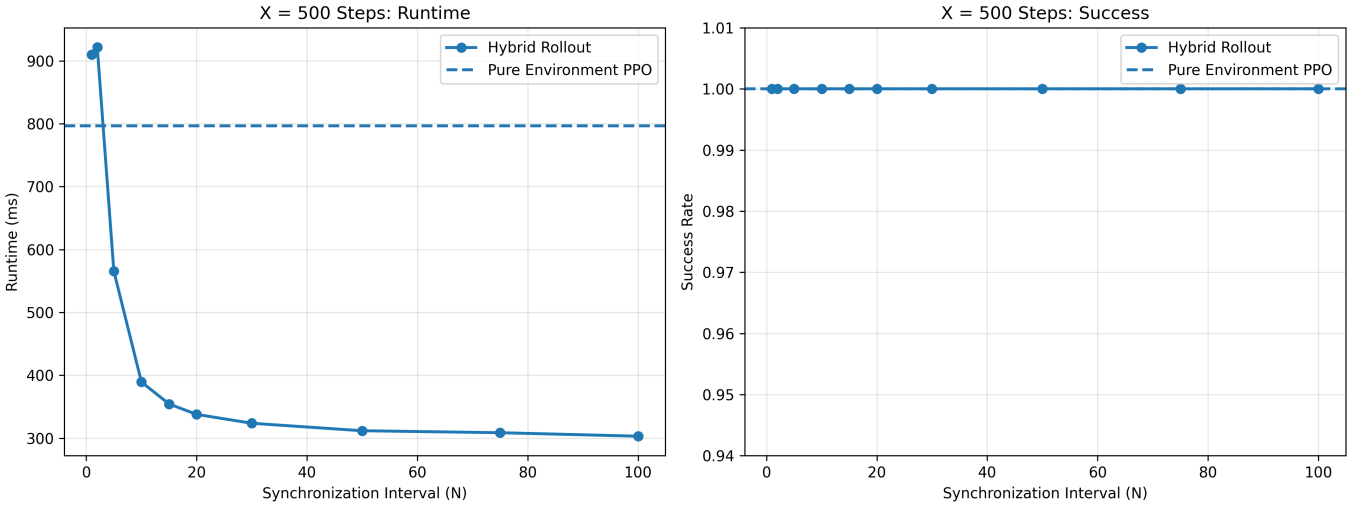

Fig. 14 Final hybrid rollout benchmark for PPO showing runtime efficiency versus task success across different synchronization intervals. Significant runtime reduction is achieved while maintaining near 100 percent success.

© The National High School Journal of Science 2026

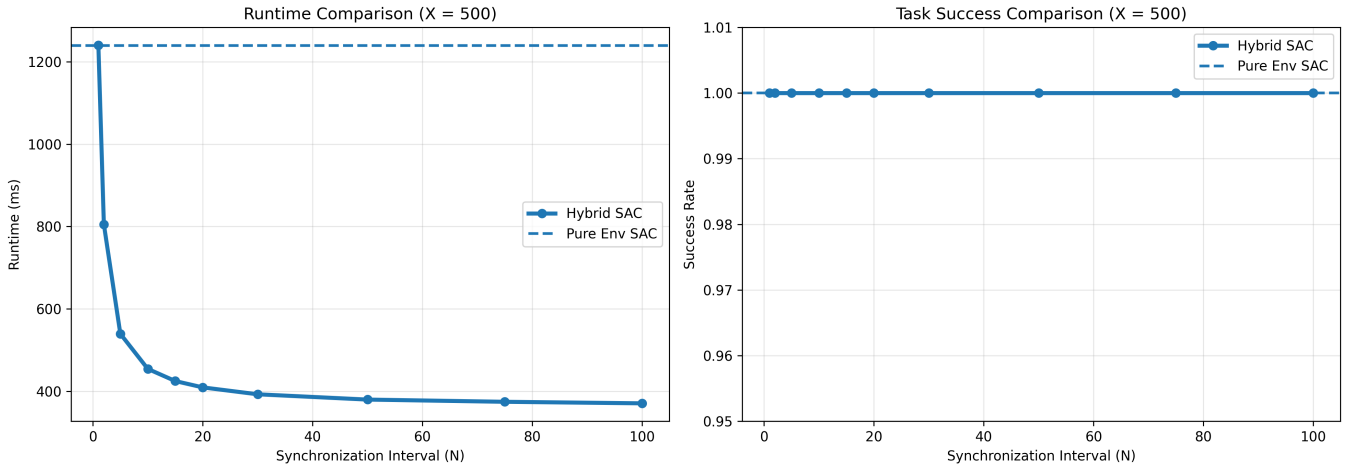

Fig. 15 Final hybrid rollout benchmark for SAC showing runtime efficiency versus task success across different synchronization intervals. Higher speedups are achieved compared to PPO while maintaining near-perfect any-time success

also includes controller inference, observation processing, synchronization logic, and framework overhead. The larger SAC speedups therefore reflect higher overall execution cost rather than higher physics simulation cost alone. SAC also maintained low runtime variance and strong robustness under sparse synchronization.

GRPO achieved slightly lower speedups (approximately 2.3×–2.4×) due to grouped action computation, but maintained strong robustness under sparse synchronization with consistently low runtime variance across repeated benchmark runs. These differences reflect the interaction between controller structure and model error. PPO and SAC rely on direct policy outputs, while GRPO's grouped action selection provides partial robustness under imperfect state estimates.

Statistical Comparison Across Controllers

Although the primary objective of this work was to analyze efficiency–stability trends under hybrid rollout rather than establish formal controller superiority, additional statistical testing was performed to compare runtime efficiency and long-horizon stability across PPO, SAC, and GRPO.

At N = 100, Welch's t-tests showed statistically significant runtime differences between all controller pairs: PPO vs SAC ($p < 0.001$), PPO vs GRPO ($p < 0.001$), and SAC vs GRPO ($p < 0.001$). These results support the observed differences in computational efficiency and grouped-action overhead across controllers.

Nonparametric Mann–Whitney U tests were additionally performed on final-distance distributions at N = 100. SAC and PPO exhibited significantly different long-horizon stability behavior ($p < 0.001$), while SAC and GRPO also differed significantly ($p < 0.001$). In contrast, PPO and GRPO final-distance distributions were not significantly different ($p = 0.74$), indicating similarly severe instability under extremely long imagined rollout horizons.

Summary

The hybrid rollout framework substantially reduced computational cost while preserving strong short-horizon task acquisition performance. Across PPO, SAC, and GRPO, speedups of up to 3.34× were achieved while maintaining high any-time success across most synchronization intervals. However, additional analysis showed that long-horizon trajectory stability degraded at large synchronization intervals due to accumulated recursive prediction error.

These results support the central claim of this work: learned world models can replace a large fraction of expensive simulator interactions when combined with periodic synchronization, enabling more efficient robotic control. Although experiments were conducted only in PandaReach-v3, the results demonstrate the effectiveness of hybrid rollout for reducing simulator interaction while preserving task performance within this benchmark setting.

Conclusion

This work addressed the computational limitations of physics-based simulation in reinforcement learning by proposing a hybrid rollout framework combining real environment interaction with learned world model prediction. The approach reduces reliance on expensive simulator calls by replacing intermediate transitions with learned model predictions and periodically synchronizing with the true environment to correct accumulated error.

© The National High School Journal of Science 2026

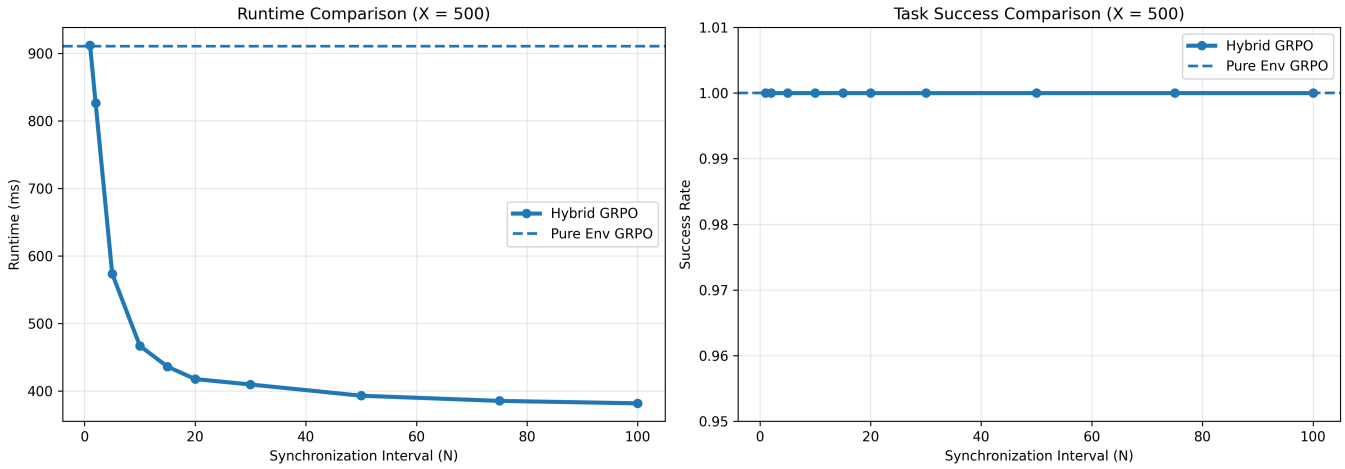

Fig. 16 Final hybrid rollout benchmark for GRPO showing runtime efficiency versus task success across different synchronization intervals. Moderate speedups are achieved with strong robustness and near-perfect success.

Experimental results across PPO, SAC, and GRPO controllers demonstrated substantial improvements in computational efficiency while preserving task performance. Speedups of up to 3.34× were achieved while short-horizon task acquisition remained highly reliable across most synchronization intervals. Additional trajectory analysis showed that long-horizon stability degraded at large synchronization intervals due to accumulated recursive prediction error. Long-horizon evaluation further demonstrated that although pure imagination rollout diverges under recursive prediction error accumulation, periodic synchronization effectively prevents catastrophic instability and enables stable control.

These results highlight a central insight: accurate short-horizon prediction combined with intermittent correction is sufficient to replace a large fraction of simulator interaction without requiring a perfectly accurate dynamics model. Hybrid rollout therefore provides a practical and scalable strategy for accelerating rollout execution in robotic environments.

Overall, this work demonstrates that learned dynamics models can be integrated into control pipelines in a computationally efficient and practically reliable manner. The proposed framework provides a simple yet effective approach for improving simulation efficiency and enabling faster rollout execution in simulation-driven robotic learning environments.

Several limitations should also be noted. First, the world model used only Cartesian end-effector position and action as input without explicit velocity or force information. While this simplified representation was sufficient for PandaReach-v3, more complex environments involving contact dynamics, momentum transfer, or high-speed manipulation would likely require richer state representations including velocity and force-related variables. Second, all experiments were conducted within a single benchmark environment, and future work should evaluate whether the framework generalizes to higher-dimensional and contact-rich robotic tasks.

Future Work

Several directions could further extend this work. First, evaluation on more complex and higher-dimensional robotic tasks would provide a stronger test of scalability, particularly in environments involving contact dynamics, multi-object manipulation, or longer-horizon control.

Second, improving long-horizon prediction stability remains a key challenge. Although periodic synchronization mitigates drift, more advanced dynamics models or uncertainty-aware prediction methods may further reduce recursive error accumulation during imagined rollout.

Third, alternative architectures such as recurrent networks or transformer-based world models may better capture temporal dependencies and improve prediction accuracy.

Finally, extending the framework to real-world robotic systems would represent an important step toward practical deployment. Integrating hybrid rollout into physical robotic control pipelines could enable faster and more computationally efficient decision-making in real environments.

Future work may also incorporate larger-scale statistical evaluation across synchronization intervals and controller families to further quantify efficiency and stability differences under hybrid rollout.

© The National High School Journal of Science 2026

References

1 T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2015.https://arxiv.org/pdf/1509.02971.

2 A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in Proc. Conf. Robot Learn. (CoRL), 2019.https://arxiv.org/pdf/1909.11652.

3 V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.https://faculty.washington.edu/minster/bio_inspired_robotics/research_articles/mnih_atari_deep_reinforcement_learning_nature2015.pdf.

4 J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," Int. J. Robot. Res., vol. 32, no. 11, pp. 1238–1274, 2013.http://www.cs.cmu.edu/~jeanoh/16-785/papers/kober-ijrr2013-rl-in-robotics-survey.pdf.

5 J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," arXiv preprint arXiv:1804.10332, 2018.https://arxiv.org/pdf/1804.10332.

6 E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2012, pp. 5026–5033.doi:10.1109/IROS.2012.6386109.

7 E. Coumans and Y. Bai, "PyBullet: A Python module for physics simulation for games, robotics and machine learning," GitHub repository, 2016.https://pybullet.org.

8 J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2017.https://arxiv.org/pdf/1703.06907.

9 Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2019.https://arxiv.org/pdf/1810.05687.

10 K. Chua, R. Calandra, R. T. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2018.https://arxiv.org/pdf/1805.12114.

11 A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2018.https://arxiv.org/pdf/1708.02596.

12 J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.https://arxiv.org/pdf/1707.06347.

13 M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2019.https://arxiv.org/pdf/1906.08253.

14 R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in Proc. Int. Workshop Mach. Learn., 1990.https://papersdb.cs.ualberta.ca/~papersdb/uploaded_files/paper_p160-sutton.pdf.stjohn.

15 M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in Proc. Int. Conf. Mach. Learn. (ICML), 2011.https://aiweb.cs.washington.edu/research/projects/aiweb/media/papers/tmpZj4RyS.pdfhttps://aiweb.cs.washington.edu/research/projects/aiweb/media/papers/tmpZj4RyS.pdf.

16 W. Zhao, W. Mou, J. Xu, et al., "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," arXiv preprint arXiv:2009.13303, 2021.https://arxiv.org/pdf/2009.13303.

17 F. Muratore, F. Treede, M. Gienger, and J. Peters, "Domain randomization for simulation-based policy optimization with transferability assessment," in Proc. Conf. Robot Learn. (CoRL), 2018.https://proceedings.mlr.press/v87/muratore18a/muratore18a.pdf.

18 K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta reinforcement learning for sim-to-real domain adaptation," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2020.https://arxiv.org/pdf/1909.12906.

19 S. James et al., "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019.https://arxiv.org/pdf/1812.07252.

20 K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2018.https://arxiv.org/pdf/1709.07857.

21 K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2019.https://arxiv.org/pdf/1902.03701.

22 Q. Gallouedec, N. Cazin, E. Dellandréa, and L. Chen, "panda-gym: Open-source goal-conditioned environments for robotic learning," arXiv preprint arXiv:2106.13687, 2021.https://arxiv.org/pdf/2106.13687.

23 T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," arXiv preprint arXiv:1801.01290, 2018. https://arxiv.org/pdf/1801.01290.

24 R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

© The National High School Journal of Science 2026