

Coupled Local and Global World Models for Efficient First Order RL

Joseph Amigo^{*12}, Rooholla Khorrambakht^{*1},
Nicolas Mansard²³, Ludovic Righetti¹³

¹Machines in Motion Laboratory, New York University, USA

²LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France

³Artificial and Natural Intelligence Toulouse Institute, Toulouse, France

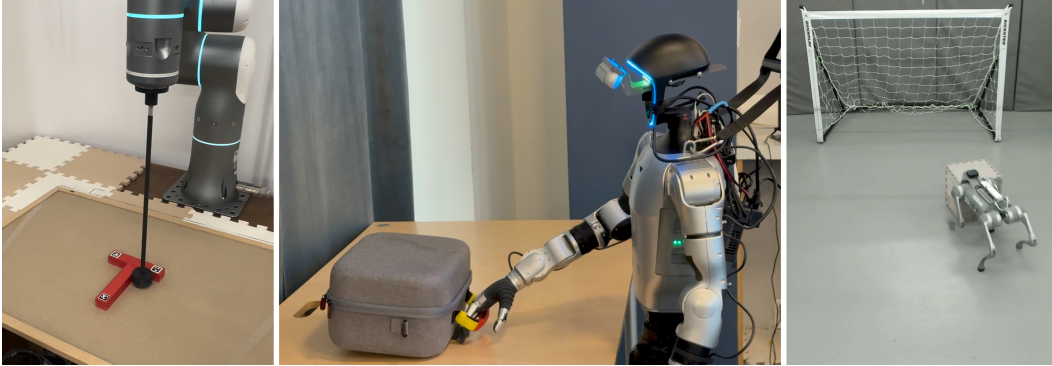


Figure 1: Real-world robotic tasks solved zero-shot by policies trained entirely from scratch within our learned world models: Push-T with a tabletop manipulator (left), Ego-Centric Grasp and Lift with a G1 humanoid (center), and Ego-Centric Push Cube with a Go2 quadruped (right).

Abstract:

World models offer a promising avenue for capturing complex environment dynamics where simulators face challenges. However, large-scale world models required for complex real-world settings are computationally expensive to adopt in popular RL approaches. In this paper, we introduce a novel first-order RL method that enables policy training via a novel decoupled first-order gradient (FoG) method: a large-scale world model generates accurate forward trajectories, while a lightweight latent-space surrogate approximates its local dynamics for efficient gradient computation. Our coupled local and global world model formulation allows high-fidelity forward dynamics alongside computationally efficient differentiation needed for model-based RL. Through real-world experiments across a range of robotic tasks with increasing complexity, we demonstrate tractable RL and zero-shot deployment thanks to our world-model-based formulation. We show significantly better sample efficiency compared to PPO in a canonical real-world PushT benchmark and observe similar patterns in more complicated ego-centric object manipulation and grasping tasks. Together, these results demonstrate that learning agents entirely inside data-driven world models is a feasible and promising pathway for solving hard-to-model tasks in image space without reliance on hand-crafted physics simulators.

Keywords: World Models, Reinforcement Learning, Robot Manipulation

1 Introduction

Recent breakthroughs in reinforcement learning (RL) for locomotion [1, 2, 3, 4, 5] have yielded highly agile policies, trained in parallelized GPU simulators and seamlessly deployed on real robots

^{*}equal contribution

for acrobatic tasks. Despite these gains, RL confronts substantial barriers in perceptual control and manipulation tasks. On the one hand, RL’s sample inefficiency prevents practical online learning directly from real-robot interactions. On the other hand, simulating manipulation tasks is inherently more difficult than locomotion, as the environment state changes with the agent’s action, adding an infinite degree of variability to the simulation problem, far beyond the ego-body dynamics and actuation modeling needed in locomotion. For instance, simulating the granular dynamics of gravel bags, pressurized deformations of an aluminum can while pulling its pull-tab, or the intricacies of folding a T-shirt, all projected onto the robot’s raw sensory space, even if computationally feasible in some cases, fall short in terms of alignment with reality and sim2real transferability.

Instead, training agents inside learned differentiable world models using first-order gradient (FoG) methods [6, 7, 8, 9] offers a promising pathway to resolve these issues. While previous FoG methods have predominantly relied on continuous online interaction within a physics simulator, applying this paradigm to learn dynamics directly from offline real-world data has the potential to effectively bridge the sim-to-real gap. Such a shift also simplifies modeling across varied environments through data acquisition rather than physics-based simulation. Furthermore, FoG methods inherently address RL sample inefficiency via gradient signals with reduced variance [9] relative to score-function based algorithms such as PPO [10] or SAC [11, 12].

However, the application of such methods in real-world robotics is severely constrained by forward simulation drifts [13, 14], which limit scalability to highly complex dynamical and visual effects common in real-world robotic settings. On the other hand, while recent diffusion world models have shown promising modeling capabilities [15], computing gradients through them at the scales required for training FoG RL policies is computationally expensive and often intractable.

To address this challenge, this paper builds on Decoupled forward-backward Model-based policy Optimization (DMO) [16], which separates forward simulation from backward differentiation, to make large-scale world models usable in FoG model-based RL (FoG-MBRL). Concretely, we propose a framework that maintains an offline trained large diffusion/flow-based model as a “global” simulator surrogate, alongside a secondary light-weight Recurrent State Space Model (RSSM)—acting as a “local” low-dimensional latent space world model that supplies stable, low-variance gradients without the need to backpropagate through the large global model. This decoupling resolves key obstacles to adopting FoG-MBRL alongside large-scale pixel-space world models, thereby opening the door to practical, efficient FoG-MBRL for real-world robotic problems. Through real-world experiments across a range of robotic tasks with increasing complexity, we empirically evaluate the capabilities of our approach in solving perceptual loco-manipulation tasks directly from real robot data.

2 Related Work

2.1 World Models and Policy Learning.

Model-Based Reinforcement Learning (MBRL) approaches are generally distinguished by their control strategy. A prevalent line of work employs the learned model for online planning via Model Predictive Control (MPC) [17, 18, 19] or Monte-Carlo Tree Search [20, 21, 22, 23]. While effective, these methods incur high computational costs at inference time. Conversely, policy optimization methods encode control into a neural network learned inside the world model. However, seminal works in this category, like DreamerV3 [7], typically rely on online interaction with a simulator. Simulator-free approaches like DayDreamer [24] learn directly from real-world interaction, but rely on Variational Auto-Encoders (VAEs) [25], which often exhibit limited expressivity and struggle to model complex distributions with high fidelity. In contrast, we target simulator-free learning using Diffusion Models, which offer superior modeling power but present unique policy optimization challenges.

2.2 Gradient Estimation for Policy Learning.

Optimization in MBRL generally falls into zeroth-order or first-order methods. Zeroth-order algorithms [8, 26, 27, 28, 29, 30] treat the dynamics as a black box. While robust to model inaccuracies, these methods are notably sample-inefficient. First-order Gradient (FoG) methods [9, 31] calculate analytic gradients via backpropagation, offering high data efficiency. However, standard FoG requires the dynamics derivatives to be computationally tractable, which excludes heavy-weight models. A promising solution is Decoupled forward-backward Model-based policy Optimization (DMO) [16], which utilizes a high-fidelity model for trajectory generation (forward) and a lightweight proxy model for gradient estimation (backward). While prior DMO works relied on the simulator for the forward pass, we extend this paradigm to the simulator-free setting in image space by using a “global” offline trained diffusion model. This allows us to leverage high-fidelity generation without incurring the prohibitive cost of backpropagating through the heavy diffusion network.

2.3 Diffusion for Efficient Control.

Recent works have successfully scaled World Models using diffusion parameterizations, such as DreamerV4 [28] and DIAMOND [32]. However, these methods typically rely on zeroth-order optimization, which requires a large number of samples. We argue that the combination of heavy-weight diffusion models and sample-inefficient zeroth-order optimization is computationally prohibitive, as it requires generating a vast number of expensive diffusion samples. Our method addresses this by employing sample-efficient FoG. By leveraging the decoupled architecture described above, we effectively distill the global diffusion priors into a local latent model, enabling fast analytic gradient updates and preventing the sampling bottleneck of diffusion-based zeroth-order RL methods.

3 Method

3.1 Problem Setting and Decoupling Framework

We consider an infinite-horizon discounted Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, r, \gamma, f)$, where \mathcal{S} and \mathcal{A} denote the state and action spaces, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$ is the dynamics. A policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ induces trajectories via $a_t = \pi_\theta(s_t)$, $s_{t+1} \sim f(s_t, a_t)$, and defines the objective $G(\theta) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.

First-order model-based RL (FoG-MBRL) computes policy gradients by differentiating through trajectories: $\nabla_\theta G(\theta) = \sum_{t=0}^{\infty} \gamma^t \left(\frac{\partial r(s_t, a_t)}{\partial s_t} \frac{ds_t}{d\theta} + \frac{\partial r(s_t, a_t)}{\partial a_t} \frac{da_t}{d\theta} \right)$, where the state and action sensitivities evolve as $\frac{ds_{t+1}}{d\theta} = \frac{\partial f(s_t, a_t)}{\partial s_t} \frac{ds_t}{d\theta} + \frac{\partial f(s_t, a_t)}{\partial a_t} \frac{da_t}{d\theta}$, $\frac{da_t}{d\theta} = \frac{\partial \pi_\theta(s_t)}{\partial \theta} + \frac{\partial \pi_\theta(s_t)}{\partial s_t} \frac{ds_t}{d\theta}$. In practice, the dynamics f is often approximated by a learned model \hat{f}_ϕ , used both for rollouts $\hat{s}_{t+1} = \hat{f}_\phi(\hat{s}_t, a_t)$ and for Jacobians $\partial \hat{f}_\phi / \partial (s, a)$, so gradients are evaluated along model-generated trajectories (\hat{s}_t) and are sensitive to compounding model error.

To address this, we adopt DMO [16], which decouples forward rollouts from backward gradient computation by using distinct models: a forward model f^\rightarrow to generate high-fidelity trajectories and a backward model $f_\phi^\leftarrow \equiv \hat{f}_\phi$ for gradient evaluation. Concretely, rather than evaluating the Jacobians of f_ϕ^\leftarrow at its own next state $s_{t+1}^\leftarrow = f_\phi^\leftarrow(s_t^\leftarrow, a_t)$, we evaluate them at the next state generated by the high-fidelity forward model $s_{t+1}^\rightarrow = f^\rightarrow(s_t^\rightarrow, a_t)$, i.e., $\left. \frac{\partial f_\phi^\leftarrow(s, a)}{\partial s} \right|_{(s_{t+1}^\rightarrow, a_{t+1})}$, and analogously for $\partial / \partial a$, to approximate the true dynamics derivatives in the policy gradient. This decoupling enables forward accuracy and backward tractability to be optimized independently.

3.2 Global World Model

Based on the complexity of each task and the available computational budget during each phase of our experiments, we adopt two different architectures for our global world model. For preliminary experiments on a canonical Push-T setup, we use a small-scale (4M parameters) diffusion-based

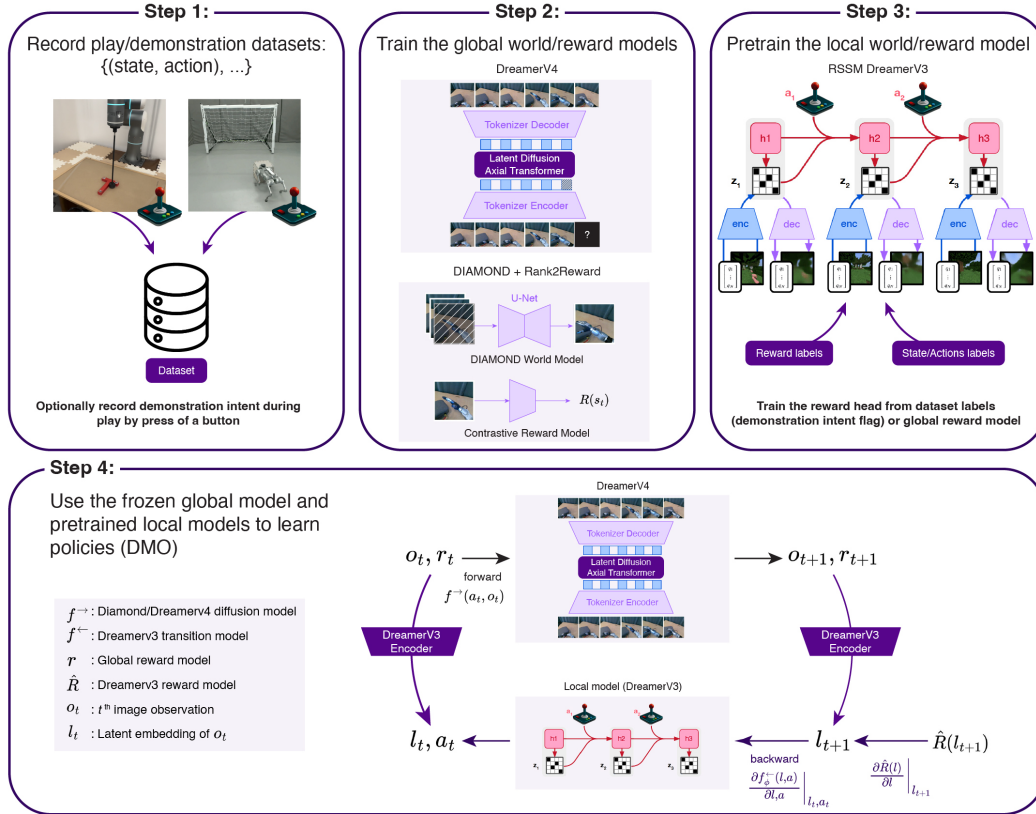


Figure 2: Overview of the proposed approach. Global world/reward models are learned from play/demonstration data (Steps 1 and 2). Then, local world/reward models are pre-trained (Step 3). Policy is optimized with the DMO algorithm (Step 4): forward simulation uses the diffusion-based global world model in pixel space, trained on real robot data, to produce high-fidelity rollouts. The backward pass uses gradients computed from a local world model operating in a low-dimensional latent space. The local world/reward model is further fine-tuned during policy optimization.

world model based on the DIAMOND [32] architecture due to long horizon prediction accuracy and efficient inference speeds. For all other tasks, we implemented a transformer-based latent diffusion world model, based on the recent DreamerV4 [15] architecture. We ensure efficient inference by adopting shortcut learning [33] (few diffusion steps) and causal diffusion forcing [34] (KV-cache-friendly real-time recurrent generation). We also integrate proprioceptive data as an additional modality to the tokenizer for our Humanoid G1 task.

3.3 Global Reward Model

Learning in pixel space requires image-based reward functions. We adopt two reward modeling strategies depending on the underlying global world model. For the Push-T experiments with the DIAMOND world model, we train a contrastive reward model on random play and, optionally, demonstration videos, following [35]. Specifically, we learn a goal-conditioned energy function $f_{\psi}(s_t | s_{\text{start}}, s_{\text{goal}})$ that assigns higher scores to states temporally closer to the goal.

For experiments using our transformer-based world model, we instead train a lightweight reward head that classifies whether a frame belongs to the task demonstration distribution. Labels are obtained during data collection via user-provided signals indicating task-relevant behavior (e.g., button presses during play).

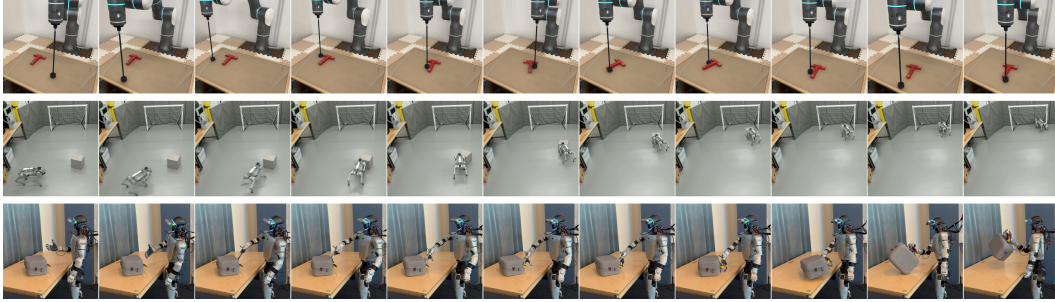


Figure 3: Real-world zero-shot deployment of our DMO policies trained in the world model.

3.4 Local Reward and World Models

For the backward model f_{ϕ}^{\leftarrow} , we adopt the recurrent state-space model (RSSM) architecture used in DreamerV3 [7], which acts as a joint local latent dynamics and reward model. The model is first pretrained offline on the play dataset, with the global reward model/head providing the reward ground truth. During online RL training, we continue fine-tuning f_{ϕ}^{\leftarrow} and its reward head using trajectories generated by the forward global model f^{\rightarrow} , ensuring that the RSSM maintains high local accuracy with respect to the current policy’s conditional action distribution. To avoid differentiating through pixels (for memory efficiency), an image encoder projects observations into a compact latent space, where f_{ϕ}^{\leftarrow} operates exclusively (the policy gradients do not flow through the encoder). In the DMO framework, unlike DreamerV3 or other MBRL frameworks, the transition model f_{ϕ}^{\leftarrow} requires to be accurate only over single forward steps, as Jacobians are evaluated at the encodings of precise forward-simulated images from the global model f^{\rightarrow} .

3.5 Data Collection and Training Protocol

We collect an unstructured high-entropy action-image pair *play dataset* on the real robot to train our global world model from scratch (reducing data needs through foundation-pretrained world models is the topic of future research). Such data does not require clean temporal segmentation of the sequence into clearly defined tasks and does not mandate frequent environment resets during data collection, making it less expensive and easier to collect and reuse from other tasks.

To ensure that the global world model accurately captures both the large workspace of the Go2 Push Cube task and the fine-grained contact dynamics between the robot head and the cube, as well as the complex multi-contact interactions involved in the high-DoF Humanoid G1 Grab Box task, we use two complementary refinement procedures. First, we qualitatively probe the world model through immersive exploration, using a Meta Quest 3 headset for Go2 Push Cube and a Pico headset with IMUs for Humanoid G1, to identify poorly modeled regions of the state space; we then collect additional data in those regions and fine-tune the model accordingly. This process is feasible because the world-model action space is expressed in terms of high-level actions that a human operator can reproduce. Second, we use RL to discover policies that exploit inaccuracies in either the world model or the reward model to obtain artificially high returns. We then deploy these exploiting policies on the real robot to automatically gather targeted data and fine-tune the model, thereby patching the weak regions that enabled the exploit.

4 Experiments

4.1 Tasks and Setups

To truly evaluate the potential of our approach to address the unmodeled complexities of real-world settings and its zero-shot sim2real transferability, we favor fewer, more representative real-world deployments over a large number of simulated tests that fall short of capturing the modeling complexities of real robotics tasks. Specifically, we consider three carefully chosen tasks and robot

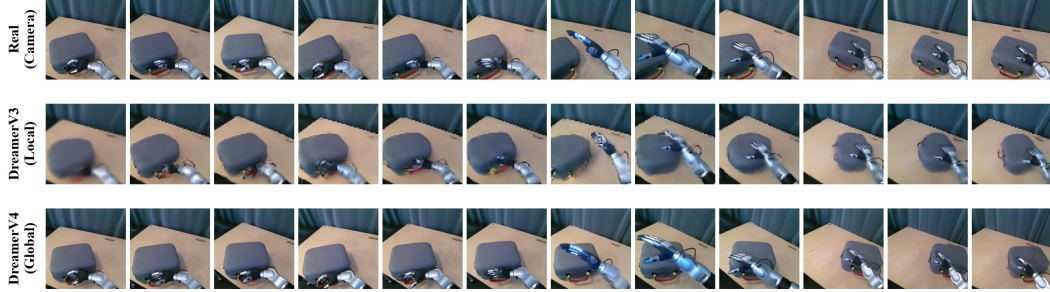


Figure 4: Unrolling of real and model-predicted trajectories on the G1 humanoid manipulation task. We compare the real camera observation (**top**), the local DreamerV3 RSSM trajectory (**middle**), and the global DreamerV4 diffusion trajectory (**bottom**) at time steps 0, 5, 10, . . . , 55 (5.5 s at 10 Hz), with both models initialized from the first real frame. The global diffusion model maintains high visual fidelity and accurate contact dynamics over the long horizon.

embodiments as introduced in the next subsections. It is important to note that, unlike other RL finetuning approaches that use world models [36, 37], we train our policies entirely within the world model from scratch, without any behavior-cloning guidance or regularization.

4.1.1 Push-T

We adopt the Push-T canonical task as an easier-to-model but non-trivial motion planning problem. Specifically, this task requires pushing a T-shaped object from a random pose on a flat 2D surface to the center of the board in a predefined orientation. The action is the tool’s vertical and horizontal velocity, and stacks of 4 consecutive frames are used as states, both provided at 5 Hz. We use a 7-DoF Flexiv Rizon-10S robot with an onboard task-space velocity controller to build our environment. We adopt a small 4M-parameter DIAMOND world model as the global forward model f^{\rightarrow} and train it on 4 hours of play data.

4.1.2 Ego-Centric Push Cube

In this task, we increase the difficulty by introducing partial observability, walking dynamics, and higher visual complexity. Specifically, a Unitree Go2 robot with ego-centric vision is tasked with pushing a cubical object into a soccer goal using its body. Body velocity commands to a low-level RL locomotion policy are taken as action and provided at 5 Hz. This task requires the policy and the world model to be robust to partial occlusions of the view by the object, maintain a short-term memory of where it was when it falls out of view, and handle the geometric complexity of a large room in image space. Thus, we train a 1.4B-parameter implementation of our transformer-based latent diffusion world model on 12 hours of data to serve as the forward model f^{\rightarrow} with a reward head.

4.1.3 Humanoid Grasp and Lift

Finally, we extend the previous experiments by considering grasping skill and a high-DoF Unitree G1 humanoid robot equipped with a BrainCo dexterous hand. The task is to grasp a box through its flexible handle and lift it above the table. The action space for the world model comprises full upper-body states (19-DoF) and lower-body locomotion (3-DoF) commands, while the policy is restricted to the right-arm and hip yaw, roll, and pitch (11-DoF). The humanoid is controlled by the NVIDIA Sonic [38] full-body tracking RL policy at 50 Hz, and receives body-tracking commands from our policy at 10 Hz. We adopt the same global forward and reward models as the Push Cube example and train them from scratch using 20 hours of play data.

4.2 Baselines and Ablations

We compare our method against two baselines: (1) PPO trained directly on image observations for both tasks, serving as a model-free RL benchmark; and (2) Action Chunking Transformer (ACT) [39], a well-known representative of Behavior Cloning (BC) policy.

We also include two ablations: the first, “No Diffusion”, replaces the DreamerV4 flow-matching world model for the forward pass with the DreamerV3 RSSM model and thus uses the same RSSM model for the forward and backward passes. The aim is to demonstrate that powerful models are indeed needed for the forward pass to accurately capture the complexity of real-world robotics scenarios. The second ablation, “No RSSM Finetuning”, evaluates the need to finetune the RSSM during RL training, providing the practitioner with additional insights into our design decisions.

4.3 Results

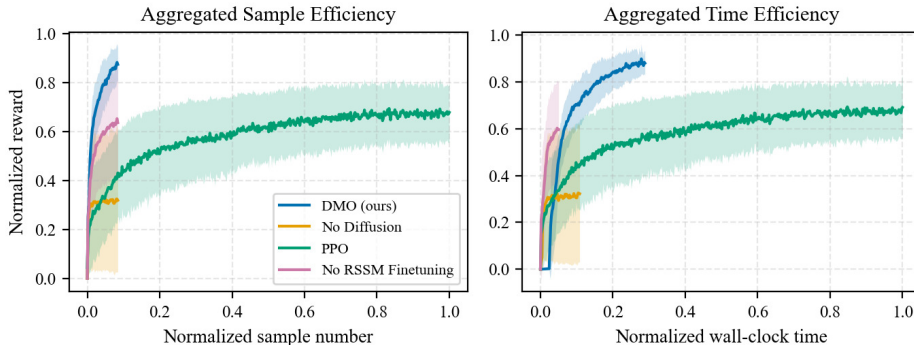


Figure 5: Aggregated performance metrics across all evaluation tasks. **Left:** Normalized reward relative to normalized sample number. **Right:** Normalized reward relative to normalized wall-clock time. Every experiment was run with 4 seeds.

The results in Fig. 5 demonstrate that DMO achieves superior sample and time efficiency compared to PPO across all tasks. Notably, the time efficiency gains shown in Fig. 5 validate our hypothesis that first-order policy gradients are more suitable than zeroth-order variants for computationally heavy world models (DMO vs PPO). Refer to Appendix A.3 for per-task efficiency results.

Furthermore, the poor performance of the “No Diffusion” ablation in Fig. 5 further underscores the necessity of a high-fidelity global world model for forward simulation of more complex tasks. As shown in Fig. 4, and in Appendix A.1, DreamerV3 generalizes less effectively than the diffusion world model. Nevertheless, using the local model for tractable gradient computation and, by decoupling, leveraging the diffusion model’s broad generalization, delivers significant improvements in both sample and time efficiency.

Table 1: Trajectory Quality Analysis. We report the mean \pm std over $N = 10$ randomly sampled evaluation episodes on the real robot for the Push Cube task. *Success Rate*: fraction of episodes reaching the goal; *Steps*: lower is better; *Straightness*: displacement/path length, \uparrow is straighter; *Curvature*: yaw change per meter, \downarrow is smoother. Metrics computed over successful episodes only.

Method	Success Rate (\uparrow)	Steps to Success (\downarrow)	Straightness Index (\uparrow)	Curvature (rad/m) (\downarrow)
DMO (Ours)	90%	177.2 ± 52.2	0.520 ± 0.266	0.692 ± 0.206
BC (ACT) ¹	60%	371.0 ± 94.0	0.330 ± 0.147	0.761 ± 0.151
PPO	40%	173.5 ± 20.3	0.233 ± 0.182	1.222 ± 0.306
No Diffusion	0%	—	—	—

Table 2: Success Rates Across Algorithms for Push-T (on the real robot, out of 10 Tries)

Algorithm	Successes / 10
DMO (Ours)	9/10
PPO	1/10 ²
No Diffusion	0/10

Table 3: Success Rates Across Algorithms for G1 Grab Box (on the real robot, out of 10 Tries)

Algorithm	Successes / 10
DMO (Ours)	8/10
BC (ACT)	7/10
PPO	1/10
No Diffusion	0/10

Table 2 reports the success rates for the Push-T real-robot experiment (Fig. 9) and confirms the advantage of our approach compared to the baselines. The “No Diffusion” ablation did not lead to the successful achievement of the task, and PPO only managed to solve the task once. We observe similar patterns in all other tasks, as shown in Table 1 and 3.

It is worth noting that for the Push Cube task, the forward simulation length for the PPO baseline had to be increased to 128 (as opposed to 64 for the DMO); otherwise, PPO fell into a suboptimal local solution where it repeatedly approaches and avoids the cube to collect rewards of 1. In the meantime, DMO succeeded in both settings and demonstrates robustness to hyperparameter choices.

Regarding the ACT baselines, we observed common BC problems such as not seeking long-term rewards (e.g., stopping in front of the goal in the push Cube task), not discovering solutions or failing at poorly covered regions, and being sensitive to the consistency of the demonstrations (demonstrator’s task-solving strategy).

Furthermore, the Push Cube experiment revealed distinct behavioral differences across methods, particularly in object localization, task completion, and control smoothness. In Table 1 we report *Straightness Index* (the ratio of Euclidean displacement to total path length, where 1.0 is optimal) and *Curvature* (average yaw change per meter) that verify this observation.

Finally, Fig. 3 shows snapshots of the deployed DMO policy across tasks. For more qualitative demonstrations, refer to the appendix.

5 Limitation

In this paper, we train our global world models from scratch and assume hand-labeled rewards for the reward head (Go2 Push Cube and G1 Grab Box tasks). Consequently, our play data must cover a wide range of physical interactions in image space, which challenges its adoption for more complex and long-horizon tasks. Our future work will focus on leveraging prior knowledge from internet-scale pre-trained video models to alleviate this limitation. Furthermore, the automated exploration and data collection scheme briefly explored under this project introduces a promising orthogonal solution worth further future investigation. This argument also applies to reward modeling, where leveraging VLM-based prior knowledge for reward labeling offers a promising solution for automating additional components within our system.

6 Conclusion

In summary, our framework bridges the limitations of simulator-centric RL by learning diffusion world models from real data, yielding visually realistic simulations that substantially reduce sim-to-real discrepancies in manipulation tasks. The decoupling mechanism leverages the comparative strengths of models—global diffusion for accurate, generalizable forward predictions and local la-

¹Under a relaxed success criterion (counting runs where the T briefly passes through the correct pose and position without stopping), PPO achieves 4/10 successes.

²For the BC baseline, success is defined as guiding the cube to the immediate pre-goal vicinity, as it never fully enters the goal.

tent models for tractable backward gradients—maintaining efficiency despite the high dimensionality of pixel inputs. Notably, our method enables learning policies directly from real robot data, using images as input, and transfers zero-shot with a high success rate on the real robot. Importantly, our experiments demonstrated practical improvements in task performance over pure behavior cloning. Our method provides a safe proxy for real-robot training, allowing scalable policy optimization without risking hardware damage or unsafe behaviors, paving the way for robust vision-based robotics in unstructured environments.

Acknowledgments

It was granted access to the HPC resources of IDRIS under the allocations AD011015316R1, A0201017555 and A0191016928 made by GENCI. This work was in part supported by the US National Science Foundation grants 2026479, 2222815, and 2315396, the French AI Interdisciplinary Institute ANITI. ANITI is funded by the France 2030 program (grant agreement ANR-23-IACL-0002). It was also supported in part by the AGIMUS project, funded by the European Union under GA no.101070165, and the ANR NERL project (grant ANR-23-CE94-0004).

A Appendix

A.1 Qualitative Comparison of World Model Rollouts

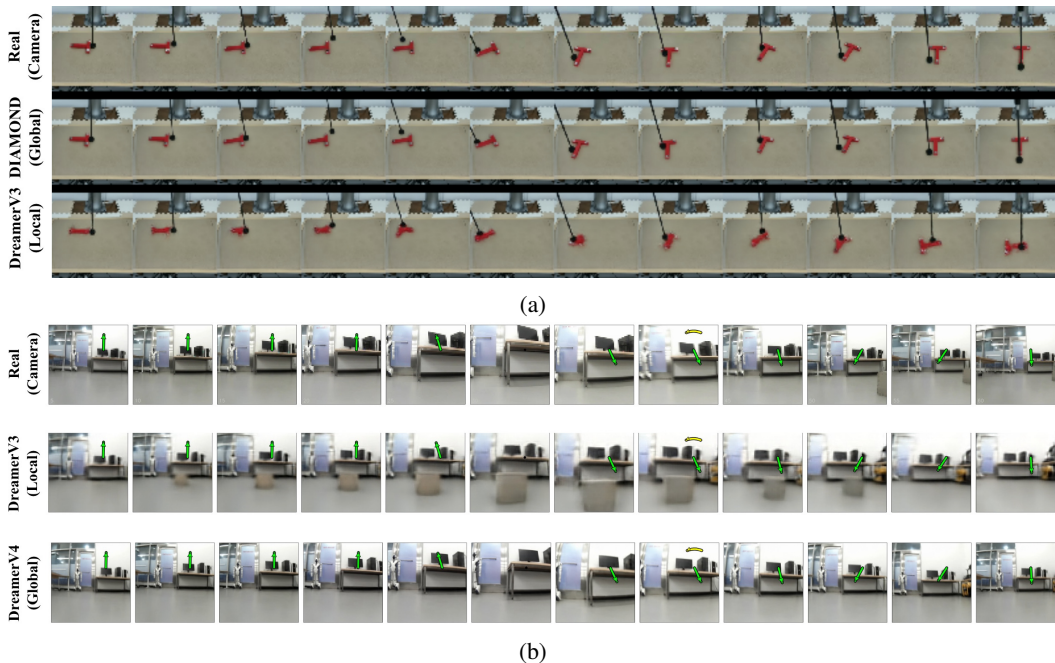


Figure 6: (a) Unrolling of real and model-predicted trajectories comparing the real trajectory (top), the diamond diffusion trajectory (middle), and the DreamerV3 trajectory (bottom) at time steps $0, 5, 10, \dots, 60$ (12 s). (b) Unrolling of real and model-predicted trajectories comparing the real trajectory (top), the DreamerV3 trajectory (middle), and the DreamerV4 diffusion trajectory (bottom) at time steps $0, 5, 10, \dots, 60$ (12 s), with both models initialized with zero context, and the cube initially occluded. Note that the local model violates object permanency by spawning the cube throughout the rollout.

A.2 Humanoid G1 Experiments: World Model Exploit Patching

In this section, we provide additional visual details regarding the world model patching process evaluated on the Unitree G1 humanoid robot. The task involves the robot using a dexterous hand to grasp and lift a box equipped with a semi-soft handle.

As discussed in the main text, the reinforcement learning (RL) policy initially discovered exploits within the dynamics of the original world model. To address this, we deployed the exploiting policy directly on the real robot to collect targeted patching data during these exploitation attempts. This real-world data was then utilized to patch and refine the world model.

Figure 7 illustrates the rollouts of the world model before and after this patching process, demonstrating how the updated model successfully corrects the physical inconsistencies that were previously exploited by the RL policy.

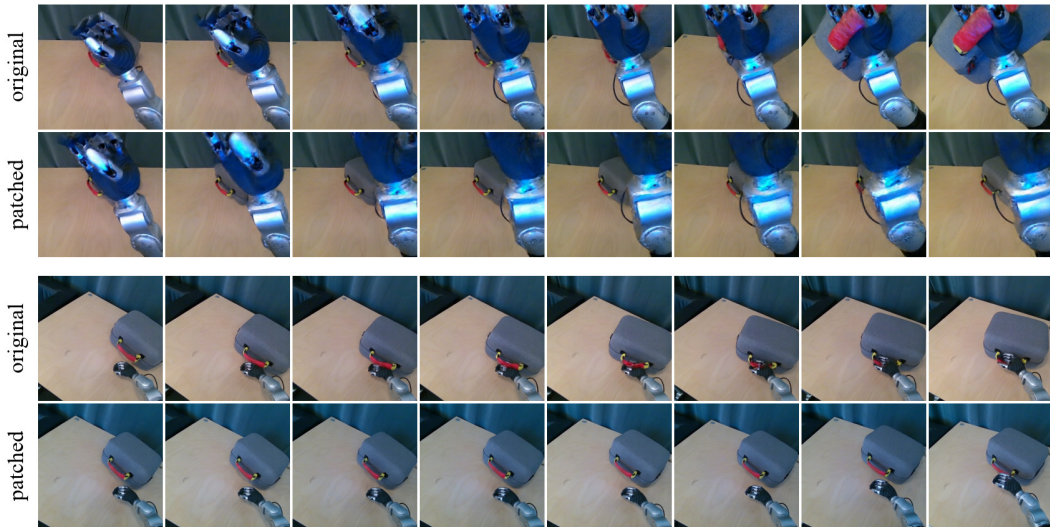


Figure 7: Visual comparison of the world model before and after patching on the Unitree G1 Grab Box task. Rows labeled *original* display the unpatched world model where the RL policy discovered physical exploits to artificially maximize reward. In the first rollout (**top two rows**), the policy exploits the model by moving the hand into a lifting pose, causing the box to spontaneously teleport into its grasp. In the second rollout (**bottom two rows**), the hand remains distant but pulls the box toward itself using an “invisible force.” Rows labeled *patched* show the corrected world model rollouts after being fine-tuned with real-world data collected by deploying these exploiting policies, effectively eliminating the physical hallucinations.

A.3 Extended Quantitative Results: Per Task Sample and Time Efficiency

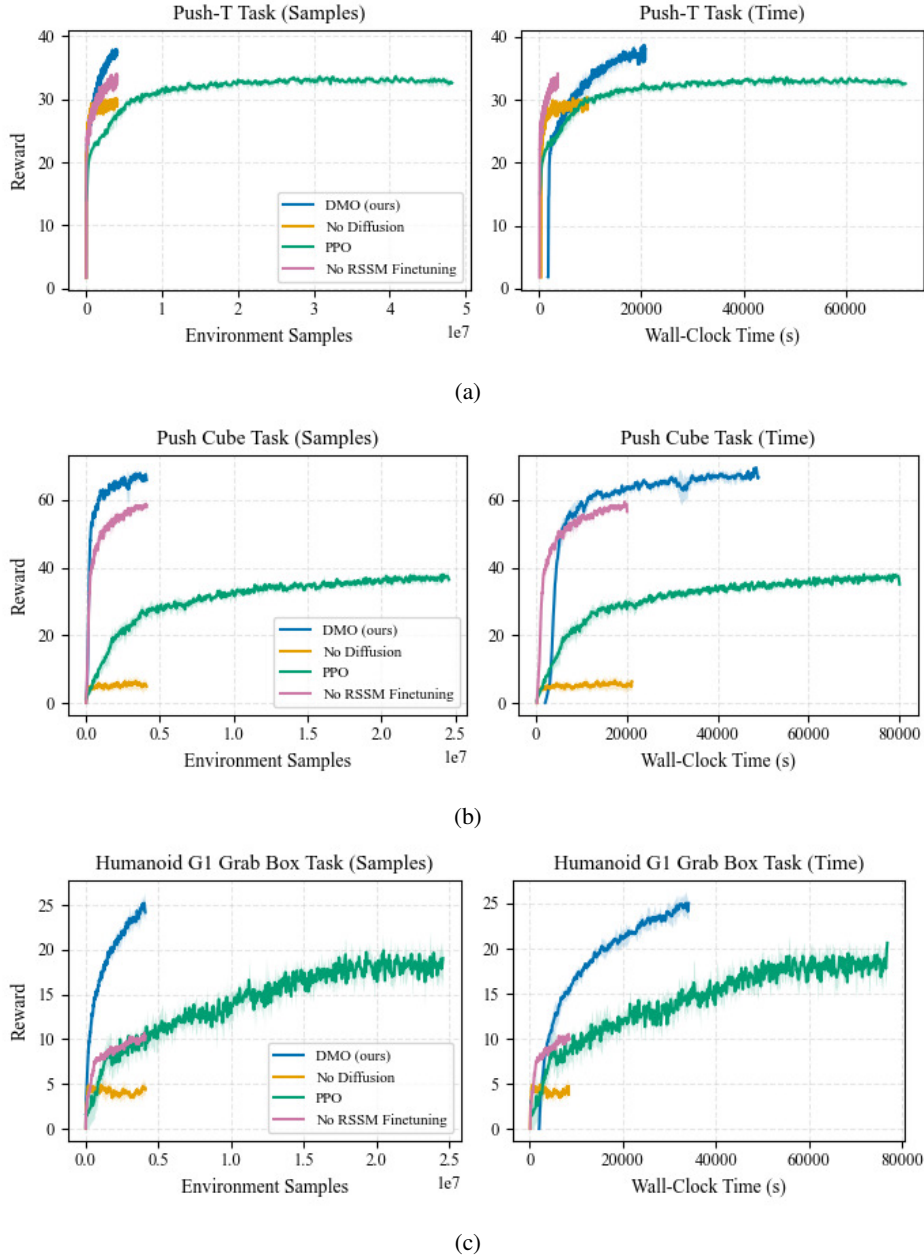


Figure 8: (a) Efficiency comparison on the Push-T task: sample efficiency of DMO (8M samples) versus PPO (40M samples) and the No Diffusion and No RSSM Finetuning ablations (left), and corresponding time efficiency comparison (right). (b) Efficiency comparison on the Push Cube task: sample efficiency of DMO (4M samples) versus PPO (25M samples) and the No Diffusion and No RSSM Finetuning ablations (left), and corresponding time efficiency comparison (right). (c) Efficiency comparison on the G1 Grab Box task: sample efficiency of DMO (4M samples) versus PPO (25M samples) and the No Diffusion and No RSSM Finetuning ablations (left), and time efficiency comparison between DMO, PPO, No Diffusion, and No RSSM Finetuning (right). Every experiment was run with 4 seeds.

A.4 The Algorithm

Given the described global and local models, we present our complete procedure in Algorithm 1. To ensure stable policy updates, we use the SAPO version of the DMO algorithm, which incorporates the Soft Analytic Policy Optimization (SAPO) objective [40]. Specifically, we optimize the policy parameters θ to maximize the entropy-regularized expected return:

$$\mathcal{L}_\pi^{\text{DMO-SAPO}}(\theta) := \mathbb{E}_{\tau \sim \pi_{\theta, f}} \left[\sum_{h=1}^{H-1} \gamma^h (r(s_h, a_h) + \alpha \mathcal{H}_\pi[a_h | s_h]) + \gamma^H V_\psi^{\pi_\theta}(s_H) \right], \quad (1)$$

where $\mathcal{H}_\pi[a_h | s_h]$ denotes the continuous Shannon entropy of the action distribution and α is the temperature parameter.

The critic network $V_\psi^{\pi_\theta}$ is learned via SGD by minimizing the temporal difference error against λ -returns:

$$\mathcal{L}_V(\psi) := \sum_{h=1}^{H-1} \left\| V_\psi^{\pi_\theta}(s_h) - \hat{V}(s_h) \right\|_2^2, \quad (2)$$

where:

$$\begin{aligned} V_h(s_t) &:= \sum_{n=t}^{t+h-1} \gamma^{n-t} r(s_n, a_n) + \gamma^{t+h} V_\psi^{\pi_\theta}(s_{t+h}) \\ \hat{V}(s_t) &:= (1 - \lambda) \left[\sum_{h=1}^{H-t-1} \lambda^{h-1} V_h(s_t) \right] + \lambda^{H-t-1} V_H(s_t) \end{aligned} \quad (3)$$

When unrolling the DreamerV4 global world model, we employ KV Caching to significantly reduce inference time. To initialize the context of the DreamerV4 world model during RL training, we do not start from a single frame. Instead, we sample a short history window of length $L_{\text{init}} = 32$ from the real trajectory dataset and pre-fill the key-value (KV) caches of the Transformer. This provides the model with sufficient temporal context to resolve potential ambiguities (e.g., momentary occlusions of the object) before the policy begins its interaction.

Algorithm 1 DMO-SAPO with global and local world models

Offline training of the diamond or dreamerV4 diffusion world model f^{\rightarrow} and global reward world model r

Offline pretraining of local dreamerV3 world model f_{ϕ}^{\leftarrow} (including its reward head \hat{R}_{ϕ})

for epoch = 1 to N **do**

 # Backward model learning

for model mini epoch **do**

$(o, a, o') \sim \mathcal{B}$, where \mathcal{B} is the replay buffer

$\phi \leftarrow \phi + \alpha_{\phi} \nabla_{\phi} \mathcal{L}_{f^{\leftarrow}}(\phi)$, where $\mathcal{L}_{f^{\leftarrow}}$ is the dreamerV3 model learning loss

end for

 # Actor and critic learning

$total_reward \leftarrow 0$

for h = 1 to H **do**

$l_h \leftarrow \text{encoder}(o_h)$

$a_h \leftarrow \pi_{\theta}(l_h)$

$o_{h+1} \leftarrow f^{\rightarrow}(o_h, a_h)$ (where f^{\rightarrow} is the diamond or the dreamerV4 diffusion global world model)

$r_h \leftarrow r(o_{h+1}, a_h)$ (where r is the global reward model, only used to learn the local reward and the value function)

$\hat{R}_h \leftarrow \hat{R}_{\phi}(l_{h+1}, a_h)$

$total_reward \leftarrow total_reward + r_h - \hat{R}_h$

end for

 Compute $\mathcal{L}_{\pi}^{\text{DMO-SAPO}}(\theta)$ from $total_reward$ and $V_{\psi}^{\pi_{\theta}}(l_{H+1})$

 Compute $\mathcal{L}_V(\psi)$ from (l_1, \dots, l_H)

 Use $\left. \frac{\partial f_{\phi}^{\leftarrow}(l, a)}{\partial l} \right|_{(l_{t+1}, a_{t+1})}$ and $\left. \frac{\partial f_{\phi}^{\leftarrow}(l, a)}{\partial a} \right|_{(l_{t+1}, a_{t+1})}$ to approximate $\left. \frac{\partial f(o, a)}{\partial o} \right|_{(o_{t+1}, a_{t+1})}$ and

$\left. \frac{\partial f(o, a)}{\partial a} \right|_{(o_{t+1}, a_{t+1})}$ during the following backward pass.

$\nabla_{\theta} \mathcal{L}_{\pi}^{\text{DMO-SAPO}}(\theta) \leftarrow \text{backward}(\mathcal{L}_{\pi}^{\text{DMO-SAPO}}(\theta))$

$\theta \leftarrow \theta + \alpha_{\theta} \nabla_{\theta} \mathcal{L}_{\pi}^{\text{DMO-SAPO}}(\theta)$

$\psi \leftarrow \psi + \alpha_{\psi} \nabla_{\psi} \mathcal{L}_V(\psi)$

end for

A.5 Additional Real-World Deployments and Behaviors

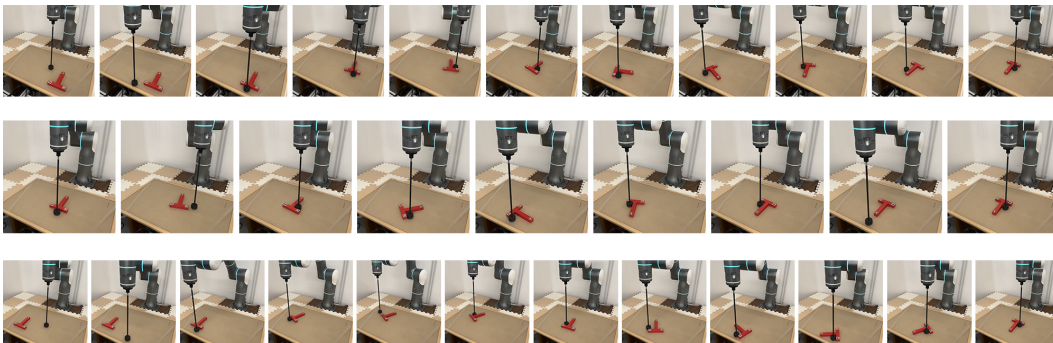


Figure 9: Three real-robot Push-T trajectories executed by the policy learned with our approach.

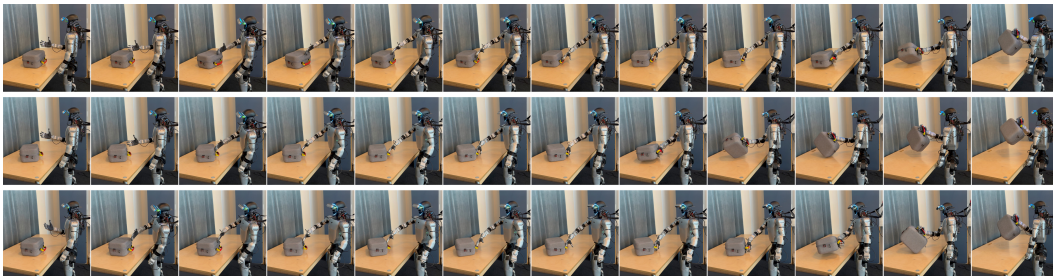


Figure 10: Three real-robot G1 Grab Box trajectories executed by the policy learned with our approach.

As shown in Fig. 11, DMO exhibits an emergent active search strategy when the object is out of view. Specifically, the agent retreats to the goal area and executes a full 360° yaw rotation to scan the environment. In contrast, the BC policy (first row of Fig. 11) simply moves backward when the cube is lost. This blind retraction is brittle (if the cube is located behind the robot, the agent drags it farther away) and reflects the poor search strategy in the demonstration subset of the dataset. However, RL leverages random exploration during training to discover more effective behaviors.

Then, as mentioned in the previous section and shown in Fig. 13, the BC (ACT) policy consistently stops before fully pushing the cube into the goal, whereas DMO successfully completes the task by leveraging sparse successful examples with high terminal rewards.

Finally, we evaluated robustness to dynamics distribution shifts by replacing the low-level locomotion controller with a variant with a different response behavior and gait style (no Raibert heuristics in reward). As shown in Fig. 12, without collecting new data or retraining the world models, the robot successfully pushes the cube into the goal. While PPO failed to generalize due to its higher entropy and less smooth action output, both DMO and BC maintained their performance. This result addresses a common critique of model-based methods—that models must be retrained for every specific robot configuration. Instead, this experiment demonstrates that the DMO framework correctly discovers stabilizing and smooth feedback strategies that are robust to variations in low-level dynamics. This finding implies that pretrained general-purpose world models may still produce useful policies for downstream tasks. A deeper investigation of this axis is the topic of our future research.

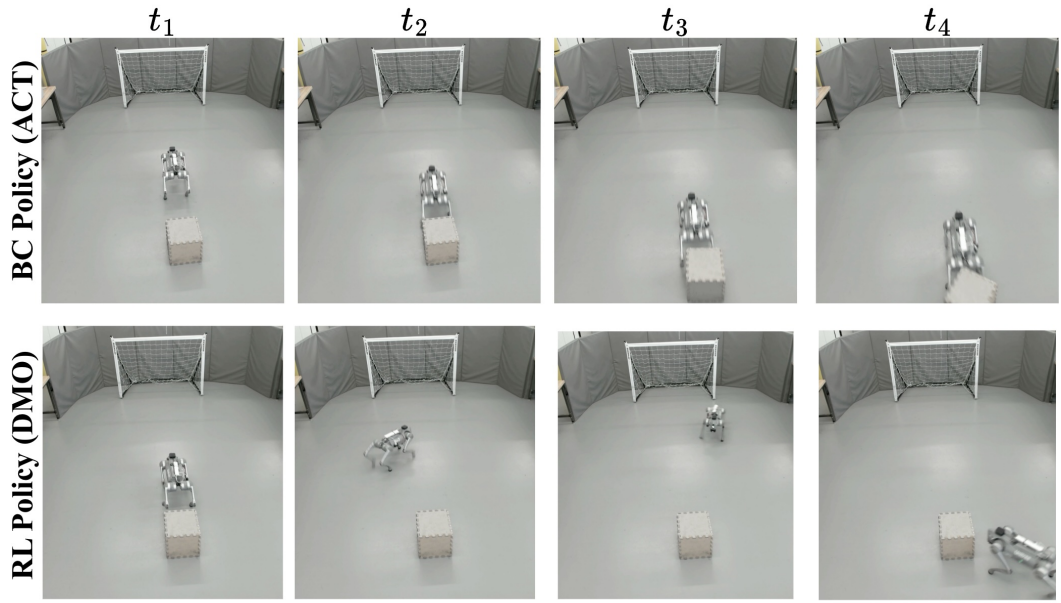


Figure 11: Comparison of search strategies under partial observability. **Top:** When the cube is out of view, the BC policy (ACT) executes a naive backward retreat, which fails to locate the object if it is directly behind. **Bottom:** The DMO policy exhibits an emergent active search behavior, rotating to visually scan the environment and successfully localize the target.

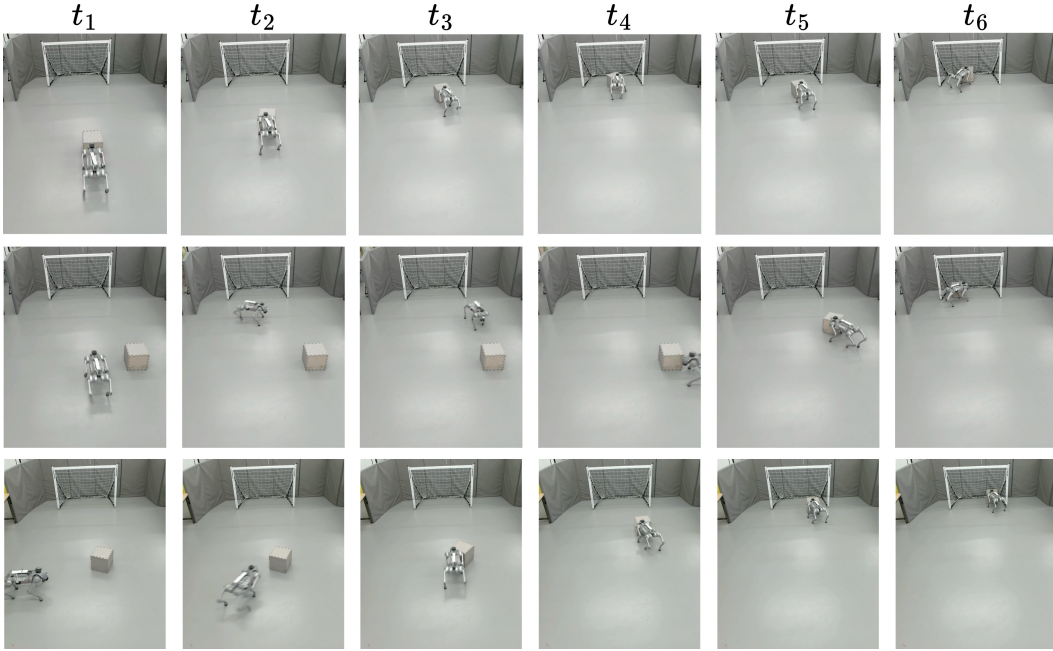


Figure 12: Real-world execution of the DMO-learned policy for the Go2 Push Cube task when using a different low-level policy than seen during training. **Top:** Standard successful push trajectory. **Middle:** Emergent active search behavior; when the object is initially out of view (or lost), the agent retreats and scans the environment to localize the cube before pushing. **Bottom:** Successful approach and push from a different initial configuration.

A.6 Implementation Details and Hyperparameters

Table 4: **Hyperparameters.** Settings for PPO and DMO-SAPO across both tasks, along with the specific World Model architectures used in our framework. DIAMOND is used for Push-T, while DreamerV4 is used for both Push Cube and G1 Grab Box.

RL Algorithms (All Tasks)			Global World Model (DreamerV4)		
Parameter	PPO	DMO-SAPO	Parameter	Push Cube	G1 Grab Box
Image Size (Input)	64×64	64×64	Image Size (Generation)	128×128	256×256
Horizon (H)	64	16	Patch Size	16	16
Gamma (γ)	0.99	0.99	Enc/Dec Model Dim	768	1024
Lambda (λ)	0.95	0.95	Dynamics Model Dim	1536	1536
Mini-batch Size	4096	256	Num Latents / Bottleneck	256 / 16	256 / 32
Mini-epochs	4	8	Enc/Dec/Dyn Layers	12 / 12 / 24	16 / 16 / 24
Optimizer	Adam	Adam	Enc/Dec/Dyn Heads	12 / 12 / 24	16 / 16 / 24
Optimizer (β_1, β_2)	(0.9, 0.999)	(0.7, 0.95)	Diffusion Steps	4	4
LR Schedule	Linear	Linear	Max Context Length	96	96
Learning Rate (Actor)	$5e-4$	$2e-3$	Per Rank Batch Size	1	1
Learning Rate (Critic)	$5e-4$	$5e-4$	Compute	24 H200	64 H100
Learning Rate (Entropy)	-	$5e-3$	Gradient Accum. Steps	10	4
Entropy Coef / Target	0.01	-6.0	Effective Batch Size	240	256
Grad Norm Clip	0.5	1.0	Global World Model (DIAMOND - Push-T Only)		
Replay Buffer Size	-	10^6	Parameter	Value	
Num Actors	256	128	Image Size (Generation)	64×64	
Network	CNN + MLP	MLP [256, 128, 64]	Diffusion Steps	3	
Activation	ReLU	SILU	Conditioning Steps	4	
Compute	1-4 H100/H200	1-4 H100/H200	UNet Channels	[64, 64, 64, 64]	
Local World Model (RSSM - DreamerV3 Style)			UNet Depths	[4, 4, 4, 4]	
Parameter	Push-T	Push Cube & G1 Grab Box	Attn Depths	[0, 0, 0, 0] (None)	
Learning Rate	$1e-4$	$1e-4$	Cond Channels	256	
Recurrent State Size	1024	4096	Sigma Distribution	$\mu = -0.4, \sigma = 1.2$	
Stochastic/Discrete Size	32	32	Optimizer	AdamW ($\beta = 0.999$)	
Dense Units	640	1024	Learning Rate	$1e-4$	
Channels Multiplier	48	96	Batch Size	2 (Accum Grad 4 = 8)	
MLP Layers	3	5	Compute	1 RTX4090	
KL Scales (Dyn/Rep/Free)	0.5 / 0.1 / 1.0	0.5 / 0.1 / 1.0			
Reward Bins / Vmin / Vmax	101 / -10 / 10	101 / -10 / 10			

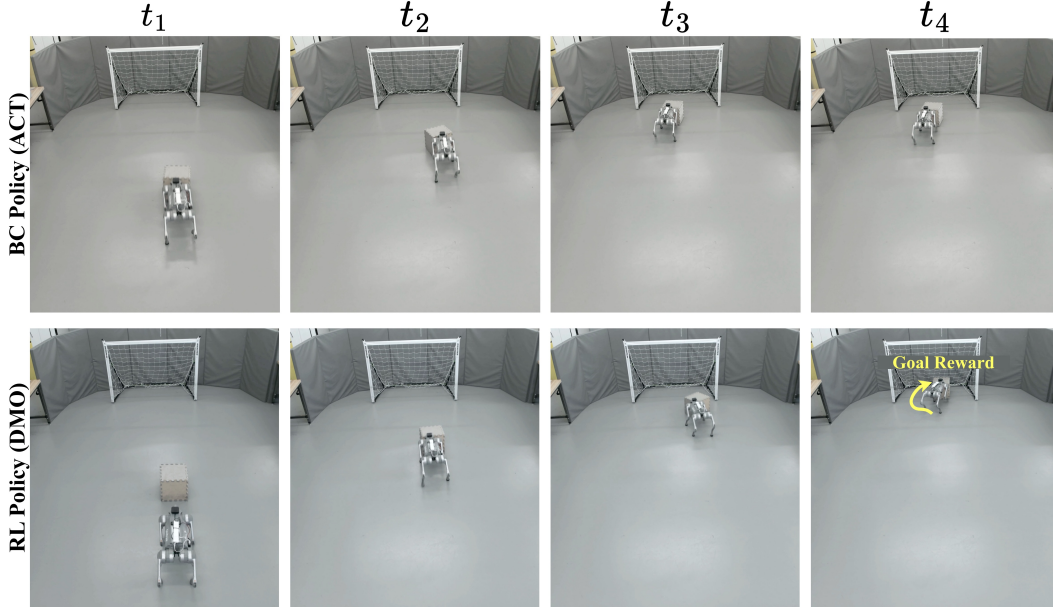


Figure 13: Task completion comparison. **Top:** The Behavior Cloning (ACT) policy successfully approaches the cube but stops pushing before the object enters the goal, reflecting the sub-optimal demonstration distribution. **Bottom:** Our RL Policy (DMO) generalizes beyond the demonstrations, learning to push the cube fully into the net to maximize the task reward.

A.7 CEM-MPC with the Global World Model

We additionally implemented a Cross-Entropy Method (CEM) model-predictive controller for the Go2 Push Cube task using the same global world model and reward model employed during RL training. This experiment was designed to assess whether the learned global model could be used directly for online planning, without policy distillation.

In order to make planning as efficient as possible, we applied several aggressive optimizations. First, we reduced the number of diffusion steps from 4 to 1. Second, we used only a single outer-loop CEM refinement iteration for each action executed on the robot. Third, we exploited the DreamerV4 dynamics model in chunk-diffusion mode, enabled by the diffusion-forcing-style training procedure, which supports both autoregressive one-step diffusion and joint diffusion over multiple future steps.

Despite these optimizations, online planning remained prohibitively expensive for real-time control. In practice, achieving real-time inference still required at least 16 H200 GPUs for 128 parallel rollouts. These results, therefore, support our central motivation for policy learning: while large world models can provide accurate forward predictions, direct test-time planning with them remains too computationally demanding, and memorizing the resulting behavior into a policy, as enabled by DMO, remains necessary.

References

- [1] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, 2024.
- [2] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88), 2024.
- [3] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

- [4] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard. Cat: Constraints as terminations for legged locomotion reinforcement learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13303–13310, 2024.
- [5] E. Chane-Sane, J. Amigo, T. Flayols, L. Righetti, and N. Mansard. Soloparkour: Constrained reinforcement learning for visual locomotion from privileged experience. In *Conference on Robot Learning (CoRL)*, 2024.
- [6] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.
- [7] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.
- [8] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.
- [9] I. Clavera, V. Fu, and P. Abbeel. Model-augmented actor-critic: Backpropagating through paths. In *International Conference on Learning Representations (ICLR)*, 2020.
- [10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning (ICML)*, 2018.
- [12] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2019.
- [13] N. Lambert, K. Pister, and R. Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.
- [14] C. Xiao, Y. Wu, C. Ma, D. Schuurmans, and M. Müller. Learning to combat compounding-error in model-based reinforcement learning, 2019.
- [15] D. Hafner, W. Yan, and T. Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.
- [16] J. Amigo, R. Khorrambakht, E. Chane-Sane, N. Mansard, and L. Righetti. First order model-based rl through decoupled backpropagation. In *Conference on Robot Learning (CoRL)*, 2025.
- [17] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.
- [18] N. Hansen, X. Wang, and H. Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.
- [19] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.
- [20] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- [21] Y. Niu, Y. Pu, Z. Yang, X. Li, T. Zhou, J. Ren, S. Hu, H. Li, and Y. Liu. Lightzero: A unified benchmark for monte carlo tree search in general sequential decision scenarios. *Advances in Neural Information Processing Systems*, 36, 2024.

- [22] Y. Pu, Y. Niu, J. Ren, Z. Yang, H. Li, and Y. Liu. Unizero: Generalized and efficient planning with scalable latent world models. *arXiv preprint arXiv:2406.10667*, 2024.
- [23] R. Khorrambakht, J. Ortiz-Haro, J. Amigo, O. Mostafa, D. Dugas, F. Meier, and L. Righetti. Worldplanner: Monte carlo tree search and mpc with action-conditioned visual world models, 2025.
- [24] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.
- [25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.
- [26] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations (ICLR)*, 2018.
- [27] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.
- [28] D. Hafner, W. Yan, and T. Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.
- [29] C. Li, A. Krause, and M. Hutter. Uncertainty-aware robotic world model makes offline model-based reinforcement learning work on real robots, 2025.
- [30] A. Jordana, J. Zhang, J. Amigo, and L. Righetti. An introduction to zero-order optimization techniques for robotics. *arXiv preprint arXiv:2506.22087*, 2025.
- [31] I. Georgiev, V. Giridhar, N. Hansen, and A. Garg. Pwm: Policy learning with multi-task world models. *arXiv preprint arXiv:2407.02466*, 2024.
- [32] E. Alonso, A. Jelley, V. Micheli, A. Kanervisto, A. Storkey, T. Pearce, and F. Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37, 2024.
- [33] K. Frans, D. Hafner, S. Levine, and P. Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.
- [34] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37, 2024.
- [35] D. Yang, D. Tjia, J. Berg, D. Damen, P. Agrawal, and A. Gupta. Rank2reward: Learning shaped reward functions from passive video. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2813. IEEE, 2024.
- [36] X. Zhang, Z. Xu, Y. Tao, Y. Wang, Y. She, and A. Boularias. Learning visual feature-based world models via residual latent action, 2026. URL <https://arxiv.org/abs/2605.07079>.
- [37] A. K. Sharma, Y. Sun, N. Lu, Y. Zhang, J. Liu, and S. Yang. World-gymnast: Training robots with reinforcement learning in a world model, 2026. URL <https://arxiv.org/abs/2602.02454>.
- [38] Z. Luo, Y. Yuan, T. Wang, C. Li, S. Chen, F. Castañeda, Z.-A. Cao, J. Li, D. Minor, Q. Ben, X. Da, R. Ding, C. Hogg, L. Song, E. Lim, E. Jeong, T. He, H. Xue, W. Xiao, Z. Wang, S. Yuen, J. Kautz, Y. Chang, U. Iqbal, L. Fan, and Y. Zhu. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *arXiv preprint arXiv:2511.07820*, 2025.

- [39] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.
- [40] E. Xing, V. Luk, and J. Oh. Stabilizing reinforcement learning in differentiable multiphysics simulation. *International Conference on Learning Representations (ICLR)*, 2025.