Simulation Distillation: Pretraining World Models in Simulation for Rapid Real-World Adaptation

Jacob Levy*[1] Tyler Westenbroek*[2] Kevin Huang[2] Fernando Palafox[1] Patrick Yin[2]
Shayegan Omidshafiei[3] Dong-Ki Kim[3] Abhishek Gupta†[2] David Fridovich-Keil†[1]
[1]University of Texas at Austin [2]University of Washington [3]FieldAI
https://sim-dist.github.io

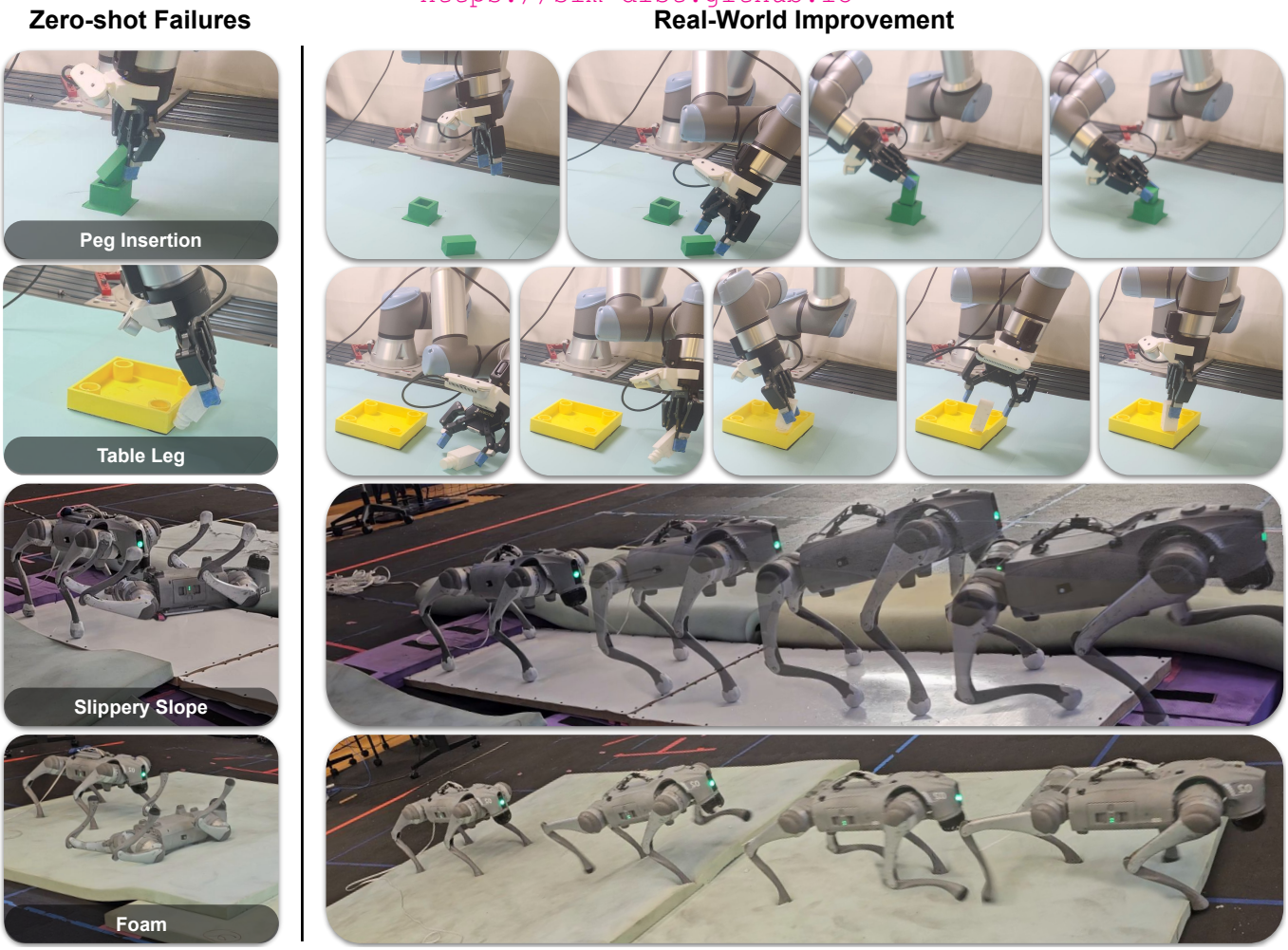

Fig. 1: Failures of zero-shot sim-to-real policies (left). Our framework `SimDist` rapidly overcomes the dynamics gap and improves performance with minimal real-world interaction. We demonstrate substantial gains in task execution on both precise manipulation and quadrupedal locomotion tasks with only 15-30 minutes of real-world data, substantially outperforming baselines.

***Abstract*—Robot learning requires adaptation methods that improve reliably from limited, mixed-quality interaction data. This is especially challenging in long-horizon, contact-rich tasks, where end-to-end policy finetuning remains inefficient and brittle. World models offer a compelling alternative: by predicting the outcomes of candidate action sequences, they enable online planning through counterfactual reasoning. However, training action-conditioned robotic world models directly in the real world requires diverse data at impractical scale. We introduce `Simulation Distillation` (`SimDist`), a framework that uses physics simulators as a scalable source of action-conditioned robot experience. During pretraining, `SimDist` distills structural priors from the simulator into a world model that enables planning from raw real-world observations. During real-world adaptation, `SimDist` transfers the encoder, reward model, and value function learned in simulation, and updates only the latent dynamics model using real-world prediction losses. This reduces adaptation to supervised system identification while preserving dense, long-horizon planning signals for online improvement. Across contact-rich manipulation and quadruped locomotion tasks, `SimDist` rapidly improves with experience, while prior adaptation methods struggle to make progress or degrade during online finetuning. Project website and code: https://sim-dist.github.io**

*Equal contribution, †Equal advising

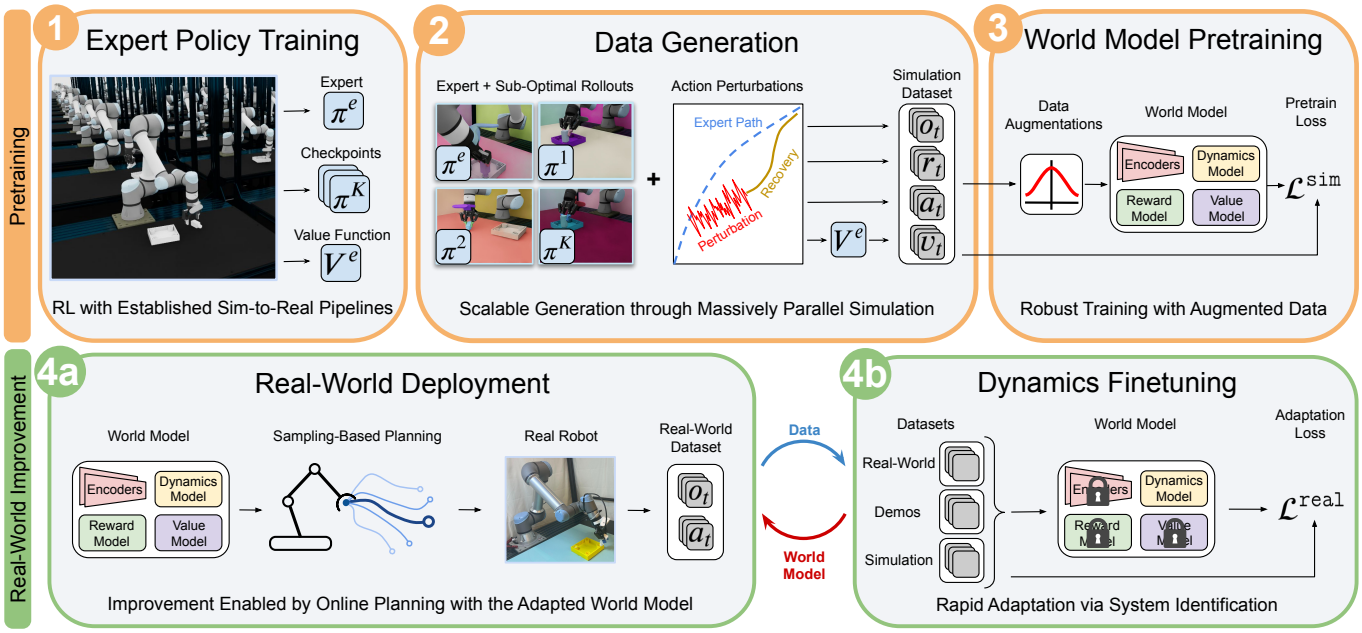

Fig. 2: `SimDist` overview. 1) An expert policy, policy checkpoints, and a value function are trained in simulation using privileged state. 2) Large-scale training data are generated by combining expert and sub-optimal policies with contiguous action perturbations, yielding diverse trajectories with dense reward and value supervision. 3) A planning-oriented latent world model is pretrained on this data, learning representations, dynamics, rewards, and values from raw observations. 4a) At deployment, the learned representation and dense reward and value models are transferred to the real robot to enable planning with the latent dynamics. 4b) Real-world data is then used to finetune only the dynamics via supervised system identification, with representations, rewards, and values frozen. Deployment and finetuning are iterated, enabling rapid and stable real-world adaptation.

I. Introduction

Robotic systems must effectively adapt their behavior using limited interactions in new environments. This adaptation data is often of mixed quality, spanning demonstrations, failed attempts, exploratory actions, and rollouts from previous policies. An effective adaptation algorithm should preserve useful priors from pretraining while extracting maximal information from each new in-domain sample. This is especially important in long-horizon, contact-rich tasks, where small errors compound and success requires reasoning through many possible futures.

We argue that world models [3, 59], rather than monolithic end-to-end policies [19, 20, 25], provide the right abstraction for leveraging prior experience to improve efficiently in new environments. Existing end-to-end reinforcement learning methods [2, 12] often collapse when finetuning policies in new domains [52, 42, 43], indicating catastrophic forgetting of pretraining priors [57]. This reflects a limitation of off-policy model-free finetuning: task representations, reward and value estimates, dynamics, and action selection are tightly entangled, so adaptation updates the entire decision-making process end-to-end while solving difficult long-horizon credit assignment problems. In contrast, world model architectures [14, 15] typically *modularize decision making* by learning separate networks for environment prediction and credit assignment. This separation allows new environment data to refine the model of action consequences without overwriting the broader decision-making structure learned during pretraining. Online planning can then convert these improved predictions into better behavior by evaluating counterfactual futures beyond those directly observed in the robot's data.

However, learning a world model suitable for planning requires action-conditioned robot data with diverse coverage at a scale that is prohibitively expensive to collect purely in the real world. Planning algorithms [47] sample candidate trajectories beyond the optimal data distribution and must *discriminate* action sequences that lead to success from those that lead to failure over long horizons. This requires dynamics predictions, return estimates, and state representations that generalize over mistakes, corrections, and varying contact sequences. *How can we obtain the data coverage and supervision required to train these models at scale?*

We argue that simulation, despite the sim-to-real gap, provides an ideal setting for bootstrapping the components of a world model that are required for effective real-world decision making. Beyond cheap, scalable interaction data, privileged simulator state enables supervision that is difficult to obtain at scale in the real world. Simulator rewards can train dense reward models that provide informative planning signals from raw perception. Expert policies and critics learned by existing student-teacher RL pipelines [52, 29] provide scalable labels for action priors and long-horizon value estimation. Together, these rich sources of supervision encourage the world-model encoder to learn robust state representations that capture the features required for effective decision making [15]. However, a natural concern remains: if a simulator does not exactly replicate the real world, won't a world model inherit the same biases, leading to poor real-world performance?

Our key insight is that these components need not be perfectly calibrated to the real world to enable effective planning

during deployment. Indeed, reward and value models only need to define an accurate *ranking* over real-world states to enable the planner to distinguish promising futures from poor ones. This is a weaker and more transferable requirement than estimating exact returns [52]. For example, in our peg insertion task, the model need only rank states where the peg is closer to the hole, better aligned, or partially inserted above states farther from success. The dynamics model, in contrast, ties actions to future states and is particularly sensitive to the dynamics gap. However, with a properly initialized model, this gap can be corrected efficiently using simple, supervised finetuning on real-world data.

Concretely, we introduce `Simulation Distillation` (`SimDist`), a framework for bootstrapping world models in simulation and efficiently adapting in the real world with online planning and dynamics adaptation. During simulation pretraining, we systematically generate diverse rollouts by perturbing expert action sequences to expose the model to mistakes, recoveries, and failed attempts. At deployment, we freeze the state encoder, reward model, and value function learned in simulation, and update only the latent dynamics model using real-world prediction losses. The frozen reward and value heads provide immediate long-horizon planning signals, enabling a relatively short-horizon planner to improve performance as dynamics predictions become more accurate. We provide extensive ablations showing that the encoder, reward model, and value function transfer robustly across the sim-to-real gap, and that broad simulation pretraining enables the dynamics model to generalize to new trajectories from limited real-world data. Altogether, `SimDist` sidesteps the long-horizon credit assignment and bootstrapping problems that make real-world RL unstable and data-hungry by moving this bootstrapping to simulation. Across four real-world tasks spanning quadruped locomotion and contact-rich manipulation, `SimDist` reliably improves with additional experience, while baseline methods for online finetuning struggle to make meaningful progress and can even degrade during adaptation.

We summarize our contributions as follows:

1) We introduce `SimDist`, a world-model framework for sim-to-real transfer that reduces real-world adaptation to supervised dynamics learning while reusing reward, value, and representation priors learned in simulation.

2) We instantiate `SimDist` on two contact-rich manipulation tasks and two quadruped locomotion tasks in the real world, achieving reliable autonomous improvement with only 15–30 minutes of real-world data and substantially outperforming existing adaptation strategies.

II. Related Work

Real-World Reinforcement Learning. A growing body of work studies reinforcement learning on real-world robotic systems [35, 32, 52, 43, 45, 46, 31, 24, 13, 28]. However, both model-free and model-based approaches remain challenging to apply reliably in low-data regimes and typically require sophisticated regularization. Efficient model-free methods aggressively reuse off-policy data and rely on frequent critic

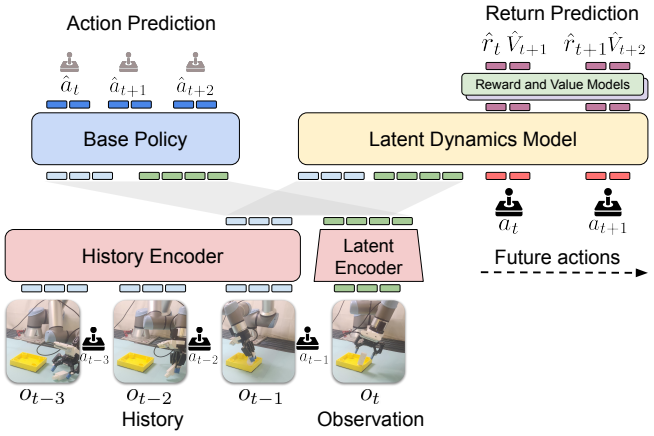

Fig. 3: World model architecture. The most recent observation is encoded into a latent representation while a history encoder processes a history of observations and actions. These jointly condition a transformer-based latent dynamics model that predicts future latent trajectories under candidate action sequences. Transformer-based reward and value heads evaluate predicted trajectories to produce reward and value sequences, while a base policy head predicts action chunks used to warm-start sampling-based planning.

updates [2, 17, 35, 7], often leading to value overestimation and unstable learning. Prior work mitigates these issues using conservative value estimation [27], policy constraints [30, 56, 44], or critic ensembles [6, 12, 18]. Model-based methods instead learn dynamics and reward models [14, 15] to reason about unseen trajectories, but must carefully avoid exploiting model inaccuracies. Common strategies include uncertainty-aware dynamics [9, 32, 23] and explicitly penalizing out-of-distribution predictions [54, 55, 11]. Rather than constraining learning, we show that bootstrapping a world model in simulation provides the coverage necessary to enable generalization beyond a small real-world dataset.

Adaptation with Physics-based Models. Many lines of work leverage approximate physics-based models for adaptation and control, but typically rely on simplified, low-dimensional state representations that are brittle in partially observed, contact-rich settings. Classical adaptive control [1, 41, 21] and model-predictive control [37, 10] approaches use highly simplified dynamics, abstracting away complex contacts and interactions. Neural physics engines [50, 40] combine structured system identification with residual learning to adapt high-fidelity simulators using real-world data, but often assume access to object poses, contact labels, or reliable state estimates that degrade under partial observability. Closely related work learns world models from mixtures of simulated and real data [33, 48] or transfers value functions from simulation to guide real-world learning [45, 52], but similarly depends on low-dimensional state observations. We instead propose distilling simulator structure from raw perception.

Generative World Models for Robotics. Recent work trains large video models on internet-scale data to learn broad physical priors for robotics [51, 3, 39], sometimes augmented with simulation data [51]. Translating these predictions into executable robot actions typically requires expert demonstrations, either by planning in video space and using inverse

models to recover actions [49, 22], or by combining video prediction with behavior cloning [59, 34, 4]. While effective for reproducing demonstrated behaviors, these methods are typically trained on narrow, expert-like action distributions and remain constrained by the real world data. In contrast, `SimDist` does not predict pixels and learns a task-oriented latent world model with reward and value heads over a broad distribution of low-level robot actions generated in simulation, enabling planning to improve beyond the real-world data.

III. Preliminaries

Problem Setting. We consider the problem of controlling a robotic system operating in the real world under partial observability and unmodeled dynamics. Specifically, we assume the dynamics are of the form $s_{t+1} \sim p(\cdot|s_t, a_t)$ where $s_t \in \mathcal{S}$ is the underlying state of the system and $a_t \in \mathcal{A}$ is the robot action. The underlying state is not directly observable in the real world as the robot only has access to raw observations $o_t \in \mathcal{O}$. We model the control problem as a partially observable Markov decision process (POMDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p, r, \gamma)$, with user-specified reward function $r(s_t, a_t, s_{t+1})$ and discount factor $\gamma \in (0, 1]$, and seek to maximize the discounted return $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\right]$. In general, dense informative reward functions are difficult to evaluate directly in the real world, given the difficulty in measuring the underlying state of the system. To render this problem more tractable, we make the following assumption:

Assumption 1. *We assume access to an approximate physics-based simulator $s_{t+1} \sim p_{\texttt{sim}}(\cdot \mid s_t, a_t)$ that provides privileged access to the underlying state s_t.*

Planning-Oriented Latent World Models. We build on common planning-oriented latent world models from works such as [15, 14]. These approaches learn reward and value models that operate directly on raw observations, enabling planning in the real world using the algorithms outlined below. The primary novelty of `SimDist` lies in our systematic framework for sim-to-real pre-training and adaptation. We structure our world model according to (1), depicted in Fig. 3:

$$\begin{aligned}
&\text{Latent representation:} && z_t = E_\theta(o_t) \\
&\text{History representation:} && h_t = C_\theta(o_{t-H:t-1}, a_{t-H:t-1}) \\
&\text{Latent dynamics:} && \hat{z}_{t+1:t+T} = f_\theta(z_t, a_{t:t+T-1}, h_t) \\
&\text{Reward Prediction:} && \hat{r}_{t:t+T-1} = R_\theta(\hat{z}_{t:t+T}, a_{t:t+T-1}) \\
&\text{Value Prediction:} && \hat{v}_{t+1:t+T} = V_\theta(\hat{z}_{t:t+T}) \\
&\text{Base Policy:} && \hat{a}_{t:t+H} = \pi_\theta(z_t, h_t).
\end{aligned} \tag{1}$$

Here E_θ encodes a latent representation z_t of the state s_t from raw observation o_t; h_t is an encoding of history over a window of H timesteps; $\hat{z}_{t+1:t+T}$ is a predicted sequence of T future latent states generated by the learned model f_θ; and $\hat{r}_{t:t+T-1}$ and $\hat{v}_{t+1:t+T}$ are reward and value estimates. We discuss key architecture decisions in Section IV-C.

Sampling-Based Planning. We control the robot in the real world with Model Predictive Path Integral (MPPI) [47] control, a sampling-based Model Predictive Control (MPC) method. At each control step, we sample a batch of candidate future action sequences and evaluate them using the world model (1), extracting the predicted future reward sequence $\hat{r}_{t:t+T-1}$ and terminal value $\hat{v}_{t+T}$, which are combined to compute the trajectory return $\mathcal{R}(a_{t:t+T-1}) = \gamma^T \hat{v}_{t+T} + \sum_{s=t}^{t+T-1} \gamma^{s-t} \hat{r}_s$. MPPI then computes the control action to execute by importance-weighting sampled trajectories according to their predicted returns. Similar to TD-MPC [15], we warm-start sampling by seeding a subset of candidate action sequences with noise-corrupted outputs of $\hat{a}_{t:t+T-1}$ from the base policy π_θ.

IV. Simulation Distillation for Efficient Real-World Adaptation

We now introduce `Simulation Distillation`, a framework for distilling physical priors and task structure into a world model in a form that enables rapid real-world adaptation with online planning and dynamics adaptation.

A. Pretraining on Simulated Data

`SimDist` builds on sim-to-real data-generation pipelines that use privileged, state-based expert policies to collect large-scale datasets paired with raw perception [53]. While prior work primarily uses this data for imitation, planning-based adaptation places stronger demands on the learned model. A planner will actively search for high-value action sequences and exploit model errors wherever coverage is weak, so the model must remain reliable far beyond the expert and real-world data distributions. Dynamics predictions must generalize from few real-world samples, while reward and value models must distinguish good and bad outcomes under off-policy actions. To provide this coverage, we deliberately inject sub-optimal actions during simulation rollouts, providing coverage over mistakes, recoveries, and failed attempts.

Expert Policy Training. We first train a state-based expert policy $\pi^e(s_t)$ with reinforcement learning using existing sim-to-real pipelines [53, 29]. We also save the optimal state-based value function $V^e(s_t)$ and intermediate policy checkpoints $\{\pi^k\}_{k=1}^K$ learned during training to be used for value supervision and diverse, sub-optimal data generation.

Generating Diverse Trajectories and Dense Supervision. We generate diverse simulation rollouts (Fig. 2) by alternating between an expert policy π^e and a set of sub-optimal policies $\{\pi^k\}_{k=1}^K$, and by periodically injecting random action perturbations over short temporal windows. This generates diverse failure and recovery behaviors beyond the nominal expert manifold, broadening coverage over dynamically feasible state-action trajectories and enabling the world model to reliably predict counterfactual outcomes. This results in a dataset $\mathcal{D}_{\texttt{sim}} = \{(o_t, a_t, r_t, v_t)\}_{t=0}^N$, where rewards r_t are computed from privileged simulator state and value targets v_t are provided by the expert value function $V^e(s_t)$. Crucially, this data-generation process is massively parallelizable and exploits the full training artifact of the simulator—including expert policies, intermediate checkpoints, and value functions—to produce rich supervision at scale. In addition, we randomize perceptual observations extensively to ensure robust transfer of the encoder. See Appendix A for details.

Algorithm 1 SimDist

// Pretraining
Perform RL, saving expert policy π^e, policy checkpoints $\{\pi^k\}_{k=1}^K$, and learned value function V^e
Generate dataset $\mathcal{D}_{\texttt{sim}}$ in simulation per Section IV-A
while not converged **do**
    Draw segments $\{(o_t, r_t, v_t, a_t)\}_{t=i-H}^{i+T} \sim \mathcal{D}_{\texttt{sim}}$
    Update θ, minimizing $\mathcal{L}_t^{\texttt{sim}}(\theta)$ at each time step
// Iterative Finetuning
Initialize $\mathcal{D}_{\texttt{real}}$ with offline real-world data if available
for J iterations **do**
    Collect real-world rollouts $\{(o_t, a_t)\}_{t=0}^M$ with MPPI and world model (1), then add to $\mathcal{D}_{\texttt{real}}$
    while not converged **do**
        Draw segments $\{(o_t, a_t)\}_{t=i-H}^{i+T} \sim \mathcal{D}_{\texttt{real}}$
        Freeze $C_\theta, E_\theta, R_\theta, V_\theta, \pi_\theta$
        Update f_θ, minimizing $\mathcal{L}_t^{\texttt{real}}(\theta)$

World Model Pretraining. We pretrain the world model (1) on $\mathcal{D}_{\text{sim}}$ by applying the following loss to predictions made at each time step t:

$$\mathcal{L}_t^{\texttt{sim}}(\theta) = \sum_{i=0}^{T} \Bigg(\underbrace{\|\hat{z}_{t+i+1} - \texttt{sg}(E_\theta(o_{t+i+1}))\|_2^2}_{\text{latent dynamics}} + c_1 \underbrace{(\hat{r}_{t+i} - r_{t+i})^2}_{\text{reward}} + c_2 \underbrace{(\hat{v}_{t+i+1} - v_{t+i+1})^2}_{\text{value}} + c_3 \underbrace{\mathbb{1}_e(a_{t+i}) \|\hat{a}_{t+i} - a_{t+i}\|_2^2}_{\text{behavior cloning}} \Bigg), \tag{2}$$

where sg is the stop-grad operator, constants $c_{1:3}$ are determined by normalizing over the range for each target, and $\mathbb{1}_e(a_t) = 1$ if a_t came from the uncorrupted expert policy and $\mathbb{1}_e(a_t) = 0$ otherwise. We apply various data augmentations discussed in Appendices B and C to prevent overfitting to simulated observations an ensure that the model learns robust, transferable representations. In contrast to typical online MBRL approaches [15, 14] which are computationally intensive and attempt to bootstrap behavior from scratch, we offload behavior generation to a privileged expert. This reduces pretraining to optimizing a simple, stationary objective, without requiring temporal-difference learning.

Robust Representations Without Reconstruction. Predicting rewards and values from z forces the encoder to capture the task-specific features required for effective planning [15]. In contrast, many world-model approaches use pixel-level reconstruction objectives, arguing that they produce more robust and generalizable latent representations [3, 14]. We find reconstruction unnecessary, and in some cases, harmful, for two reasons. First, our diverse data-generation procedure exposes the model to an extremely broad range of state-action pairs, spanning initial conditions, object configurations, contact modes, expert behaviors, failures, and recoveries. This diversity forces the model to learn a robust representation without the added computational cost of reconstructing pixels. Second, sim-to-real transfer requires extensive visual randomization. Pixel reconstruction would therefore pressure the latent state to encode randomized texture, lighting, and rendering artifacts that are deliberately varied and irrelevant to the task, which can hurt transfer [58].

B. Real World Transfer and Efficient Dynamics Adaptation

Our key insight is that global task structure is largely invariant to low-level sim-to-real dynamics gaps. For example, in Peg Insertion (Fig. 1), a meaningful latent state captures the locations of the peg and hole, while the value function encodes distance to the goal and motions leading to successful insertion. This structure persists across the sim-to-real gap, even though the low-level actions required to realize these behaviors differ between domains. Exploiting this decomposition, SimDist finetunes only the environment-specific dynamics model while freezing the encoder, reward, and value functions. Planning with frozen reward and value models enables immediate improvement as dynamics predictions become more accurate, without requiring reward or value bootstrapping in the real world. Because the world model can be trained on arbitrary trajectories, SimDist naturally supports off-policy learning and can incorporate prior data such as demonstrations. Importantly, the adaptation remains relatively short horizon and local, since no long-horizon bootstrapping is needed.

Dynamics Adaptation Loss. When updating the dynamics model we apply the following loss at each time t:

$$\mathcal{L}_t^{\texttt{real}}(\theta) = \textstyle\sum_{i=0}^{T} \|\hat{z}_{t+i+1} - \texttt{sg}(E_\theta(o_{t+i+1}))\|_2^2, \quad \text{with } C_\theta, E_\theta, R_\theta, V_\theta, \pi_\theta \text{ frozen, } f_\theta \text{ finetunable.} \tag{3}$$

Because the encoder is frozen, it provides consistent latent targets $E_\theta(o_{t+i+1})$ throughout finetuning, avoiding the need to bootstrap a representation as in (2). This also anchors the adapted dynamics to the same latent representation used by the frozen reward and value models, rather than drifting away from the representation R_θ and V_θ were trained to evaluate.

Iterative Improvement. The overall pipeline for SimDist is shown in Fig. 2 and in pseudo-code in Algorithm 1. SimDist autonomously improves in the real-world by repeatedly collecting M on-policy rollouts under the planner, adding this data to the real-world data set $\mathcal{D}_{\texttt{real}}$, then finetuning f_θ to minimize prediction losses as in (3). Notably, because system identification can effectively learn from any real-world trajectories, SimDist is a simple off-policy reinforcement learning strategy which can easily incorporate diverse data sources such as demonstrations or play data into $\mathcal{D}_{\texttt{real}}$.

Remark 1. *In contrast to standard MBRL frameworks [14, 15], which must jointly bootstrap latent representations, value functions, and policies from scarce in-domain data,* SimDist *offloads these challenging objectives to simulation, where diverse data is cheap and plentiful. As a result, we can reduce real world adaptation to supervised finetuning of the dynamics.*

C. World Model Design Decisions

In order to successfully improve behavior, the planner must sample numerous off-policy rollouts and accurately model returns. The following decisions were necessary to accelerate inference and enable reliable real-time decision making.

Minimal History Representation. Observation histories $o_{t-H:t}$ can contain high-dimensional inputs such as images that are costly to process. To reduce inference cost, we split observations $o_t = (o_t^p, o_t^e)$ into proprioceptive and exteroceptive components and feed only $(o_{t-H:t}^p, a_{t-H:t}, o_t^e)$ into the history encoder C_θ. Using only the most recent high-dimensional observation substantially reduces planning latency and, empirically, improves training stability by reducing context length.

Chunked Prediction for Planning. Autoregressive world models [15, 14, 5] require sequential unrolling over the planning horizon, which bottlenecks parallelism when evaluating numerous rollouts. Inspired by [48], our latent dynamics model f_θ predicts T future states in a single forward pass using a transformer with cross-attention between the encoded history tokens and a candidate action sequence, together with a causal mask. This chunked prediction fully exploits GPU parallelism, dramatically improving planning throughput.

Sequence-to-Sequence Return Modeling. Prior work often decodes rewards and values with per-timestep MLPs applied independently to each predicted latent state [15, 14, 5]. We instead use transformer-based reward and value models that attend over the entire predicted latent trajectory $\hat{z}_{t:t+T}$. This aggregates information across the entire trajectory, yielding more accurate return estimates (see Section V-D).

V. Experiments

We evaluate the ability of `SimDist` to adapt in the real world on the four manipulation and quadruped tasks depicted in Fig. 1 and carefully ablate key design decisions. Additional details can be found in the Appendix. We aim to answer: 1) does `SimDist` outperform existing online RL methods and behavior cloning baselines? 2) what factors enable the planner to effectively improve performance with minimal real-world data? 3) which components of the architecture and pretraining procedure are crucial for the performance of `SimDist`?

A. Robotic Systems and Tasks

Manipulation System and Tasks. Manipulation experiments are conducted on a UR5e robot. Actions are six-dimensional relative end-effector pose targets and a binary gripper action, and observations include joint states and three 224×224 RGB images from wrist-mounted, overhead, and side-view cameras. Each image is encoded with a pretrained ResNet-18, fused with proprioception, and mapped to a 64-dimensional latent state z. Training uses 100k trajectories (see Appendix B for details on data mixture). We use history and prediction horizons $H = T = 5$ and control the robot at $5\,\mathrm{Hz}$. Expert policies are trained with [53]. We consider two precise assembly tasks, with initial conditions drawn from a Narrow ($2\,\mathrm{cm} \times 2\,\mathrm{cm}$) or Wide ($35\,\mathrm{cm} \times 35\,\mathrm{cm}$) grid:

1) **Peg Insertion.** We construct a peg insertion task similar to the $16\,\mathrm{mm}$ square peg task from [38]. This task requires picking the peg, aligning it with the hole, and insertion.
2) **Table Leg.** We follow [16], wherein a table leg must be picked, aligned, and threaded into a hole on a table.

Quadruped System and Tasks. We conduct quadrupedal experiments on a Unitree Go2, with actions given as position targets for the 12 joints. Observations include proprioception and a local terrain height map, encoded using a CNN and fused with low-dimensional observations via an MLP to produce the latent state z. The policy π_θ, reward R_θ, and value V_θ are additionally conditioned on desired base forward, lateral, and yaw velocities. Pretraining uses 100M simulated trajectories. We use history and prediction horizons $H = T = 25$ and plan at $50\,\mathrm{Hz}$ on a laptop with an RTX 4090M. See Appendix C for details. We consider two tasks:

1) **Slippery Slope.** The robot traverses straight over two panels inclined at 3.0° and 5.7°, respectively. The panels are covered with PTFE (Teflon), and the robot's feet are wrapped with thermoplastic, creating extremely low-friction contacts. A trial is successful if the robot traverses $1.82\,\mathrm{m}$, clearing both panels. We conduct five trials per commanded forward speed at 0.1, 0.3, and $0.5\ \mathrm{m\,s^{-1}}$.
2) **Foam.** The robot traverses two $5\,\mathrm{cm}$ thick overlapping memory foam pads whose compliant dynamics are not modeled in simulation. A trial is successful if the robot traverses $3.00\,\mathrm{m}$ to clear the foam. We conduct five trials per commanded forward speed at 0.2, 0.7, and $1.2\ \mathrm{m\,s^{-1}}$.

B. Real-World Learning Methods.

We evaluate `SimDist` against state-of-the-art real-world RL methods (and behavior cloning baselines on manipulation tasks). All RL methods use the same encoders as `SimDist`, but with an MLP policy head (no action chunking).

Manipulation. For each task, we collect 20 real-world demonstrations via teleoperation. We evaluate two variants of `SimDist`: (i) `SimDist`, which adapts only the latent dynamics model using on-policy rollouts (Algorithm 1), without access to the demonstrations and (ii) `SimDist`+BC, which additionally finetunes the base policy π_θ using expert action labels from the teleoperated data. In both cases, the dynamics model is updated after every 20 real-world episodes. We compare against model-free RL baselines for offline-to-online RL, RLPD [2] and IQL [26], trained from sparse rewards to reflect common manipulation settings where dense rewards are difficult to specify; these baselines update after every episode. We also include SGFT-SAC [52], a model-free method that transfers a simulated value function, isolating the benefit of full world-model adaptation beyond value transfer alone. All online RL methods are given access to the 20 pre-collected demonstrations. Finally, we compare against behavior cloning baselines—Diffusion Policy [8] and $\pi_{0.5}$ [20]—trained with 100 real-world demonstrations, as well as variants trained on both real-world demonstrations only and co-trained on these demonstrations and the simulated dataset used by `SimDist`. The task set-up is identical to the corresponding tasks from

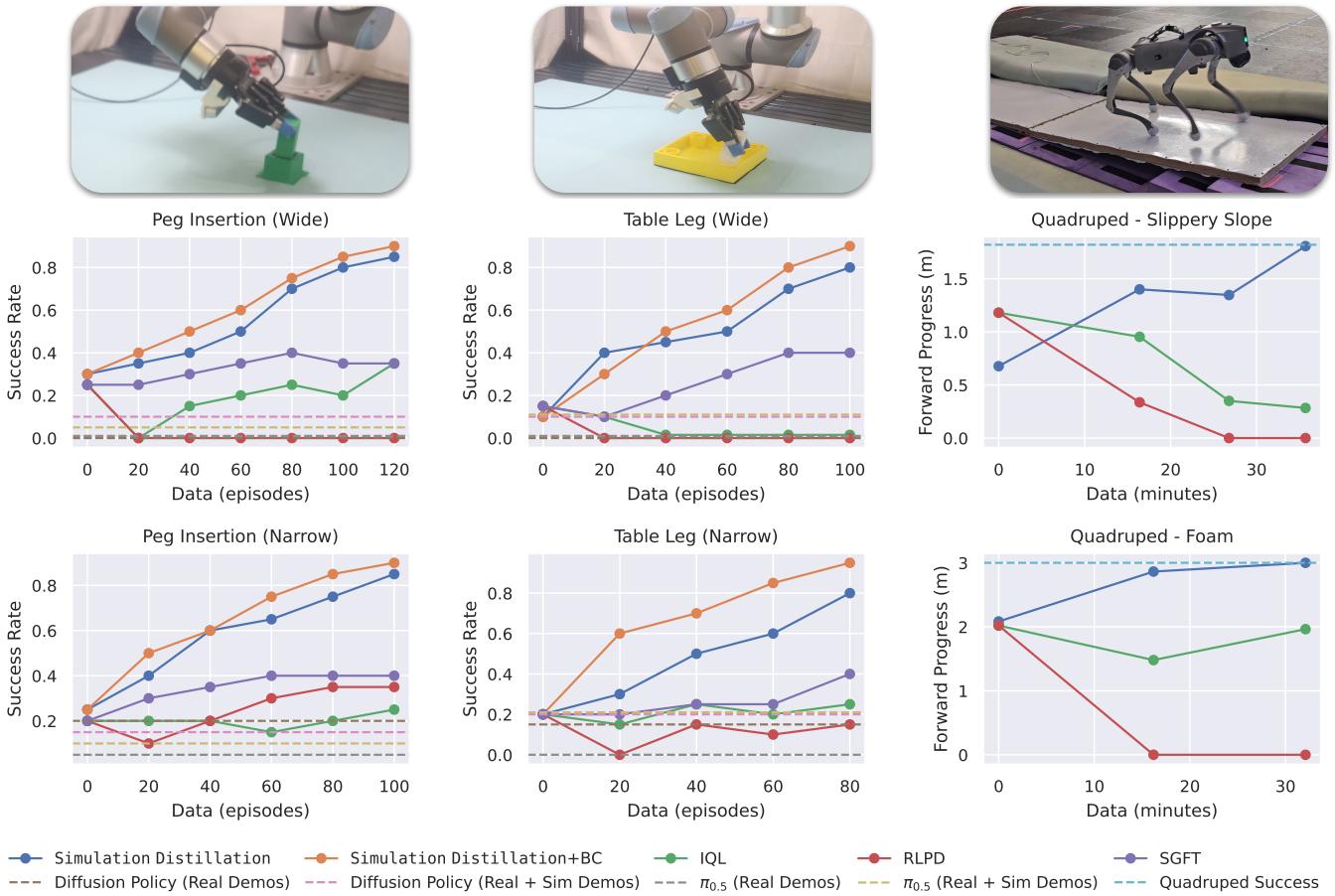

Fig. 4: Real-world results. Success rate for two manipulation tasks, computed over 20 trials, and average forward progress for two quadruped locomotion tasks, averaged across all 15 trials (3 speeds, 5 trials each), as a function of real-world finetuning data. SimDist exhibits rapid and consistent improvement with limited data by finetuning only the latent dynamics model while planning with frozen reward and value models. In contrast, direct policy finetuning with the baselines shows limited or no improvement under the same data budgets.

[52], however, we depart from this work by defining a rollout to be successful if it completes the task within 45 seconds.

Quadruped. For quadruped locomotion, we compare SimDist against the off-policy algorithm RLPD [2] and the offline-to-online finetuning method from IQL [26], whose value function is pretrained in simulation. All methods use the same learned reward model from SimDist, allowing us to isolate the effect of different adaptation strategies. We do not have the ability to collect demonstrations for this system, and thus do not compare methods which require them.

C. Real World Improvement Results

Figure 4 summarizes our results. Across all tasks, SimDist consistently outperforms prior approaches, achieving substantially higher success rates with far greater sample efficiency than online RL baselines, while autonomously improving well beyond the performance of behavior cloning methods. Across the board SimDist typically reaches scores around $2\times$ higher than any baseline. Standard RL finetuning methods frequently exhibit catastrophic forgetting, with performance collapsing during adaptation, whereas SimDist makes steady, monotonic progress throughout training, due to its ability to side step long-horizon credit assignment and improve performance with simple supervised learning. SGFT avoids catastrophic collapse by transferring value functions from simulation, but remains significantly more sample inefficient than SimDist, which can efficiently improve by leveraging the world model to make numerous counterfactual predictions about trajectories the robot has not directly experienced. Finally, we observe that providing SimDist with demonstrations only boosts performance, highlighting how SimDist can naturally absorb heterogeneous, mixed quality sources of data.

For the two manipulation tasks, the performance gap between SimDist and baselines widens as the task is made more difficult by expanding the ranges of initial conditions from the narrow to wide distribution. This underscores the benefit of broad simulation pretraining, which enables SimDist to retain structural priors from the simulator and reliably improve performance over the entire state-space with limited real-world data. This effect is illustrated in Fig. 7, which visualizes successful and failed initial conditions for SimDist and Diffusion Policy on Peg Hard, revealing the substantially greater robustness of policies learned by SimDist.

Finally, Fig. 6 provides additional insight into how SimDist improves performance by plotting the number of success/min achieved during training. SimDist monotonically improves throughput by $\sim 1.5\times$–$2\times$ over zero-shot

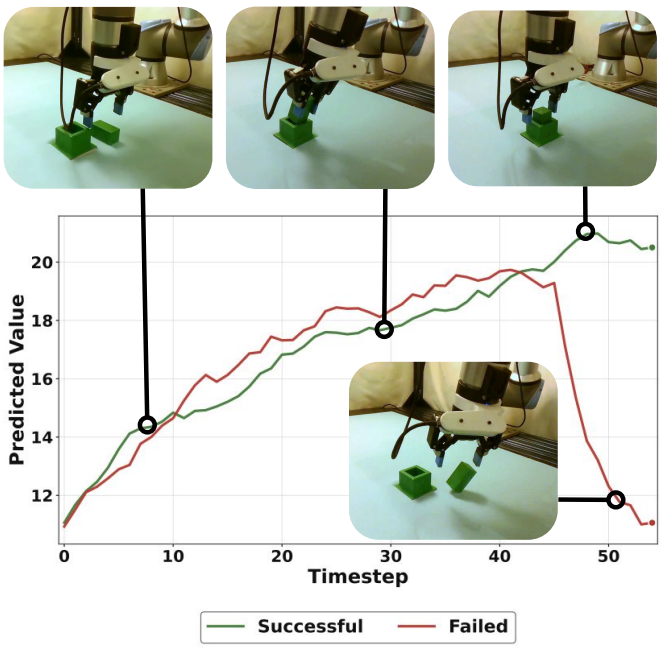

Fig. 5: Value predictions from SimDist along successful and failed real-world Peg trajectories starting from the same initial condition. The predicted values track task progress and clearly distinguish successful from failure.

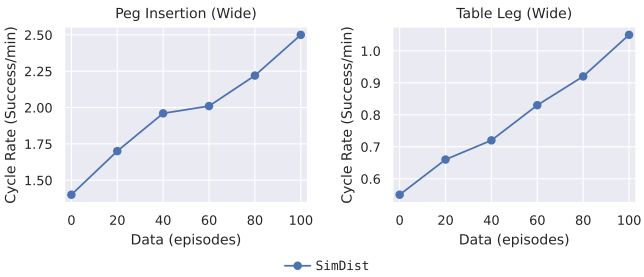

Fig. 6: Throughput of manipulation policies throughout training. SimDist reliably improves the velocity of successful task completions.

performance.

D. Analyzing Real-World Predictions and Effects on Planning

The planner can only improve behavior if it can reliably distinguish trajectories with high and low returns. This requires both accurate dynamics predictions and successful transfer of reward and value models. We examine value transfer in Fig. 5, which plots predicted values over time for successful and failed trajectories. For the successful rollout, predicted value increases consistently over time, while the value drops sharply when the robot drops the peg during the failed rollout. Thus the transferred encoder E_θ and value function V_θ *reliably discriminate between successful and failed trajectories*.

Next in Fig. 9 we compare ground truth camera observations for the peg task compared to images reconstructed from the corresponding latent encoding $z_t = E_\theta(o_t)$ generated by the frozen encoder. The images are generated by an auxiliary probe trained on all available real world data, and we re-iterate that the world model itself is not trained with the reconstruction loss. Even though the encoder is not trained

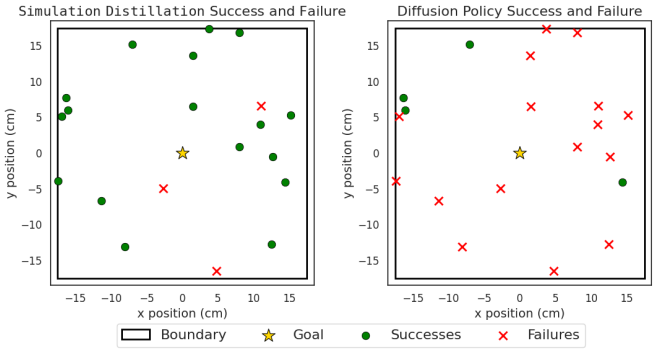

Fig. 7: Scatter-Plot showing successful and failed attempts at solving the Peg Wide task for Diffusion Policy (Right) and the final trained policy for SimDist(Left). The broad coverage of pretraining data for SimDist enables efficiently learning policies which are far more robust than baselines.

to explicitly reconstruct pixels, the the broad diverse pre-training of SimDist forces the encoder to learn a robust representation which is able to capture the underlying state of the real-world scene, enabling the reconstructions portrayed.

We next evaluate dynamics prediction accuracy on the Quadruped Slippery Slope task in Fig. 8a. We roll a held-out real-world trajectory through the world model and compute the latent dynamics loss (3) at each timestep, yielding an average loss of 0.076 for the pretrained model and 0.019 after finetuning. To visualize the impact of this improvement, we decode predicted latent states into predicted front-left foot positions in Fig. 8c. Before finetuning, the model incorrectly predicts stable contact on the PTFE surface and fails to anticipate slip; after finetuning, it accurately predicts future slippage, closely matching the real trajectory (Fig. 8b). This illustrates how broad simulation pretraining enables the dynamics model to generalize to real-world trajectories outside the training set.

Finally, improved dynamics predictions directly reshape planning behavior. As shown in Fig. 8d, trajectory samples generated with the finetuned model reflect the altered contact dynamics and lead the planner to select plans that account for real-world slip. In contrast, plans derived from the pretrained model are qualitatively inconsistent with the true dynamics. Together, these results show that finetuning corrects latent dynamics errors in a way that meaningfully changes the planner's trajectory distribution, explaining the rapid real-world performance gains observed in Fig. 4.

E. Ablating Key Design Decisions

Unfreezing World Model Components. We first ablate unfreezing world-model components during real-world adaptation, with results in Fig. 10. Unfreezing the encoder causes complete performance loss, as frozen reward and value heads receive latents outside their training distribution. Unfreezing the value function, following [11], reintroduces long-horizon credit assignment from limited real-world data and causes catastrophic forgetting. We omit reward-model unfreezing, since accurate dense reward labels are generally unavailable in the real world, underscoring a key advantage of SimDist: reward transfer without real-world labels.

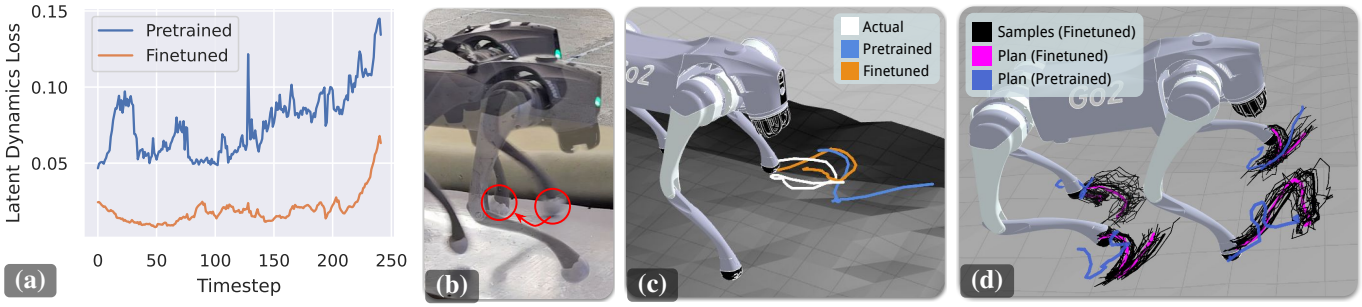

Fig. 8: **(a)** Finetuning drastically lowers dynamics prediction loss on a quadruped Slippery Slope trial. **(b)** Frames showing the front left foot slipping during the trial. **(c)** Foot-trajectory predictions from the world model at the same instant: the finetuned model correctly anticipates future slippage, while the pretrained model fails to do so. **(d)** Visualization of sampling-based planning. Candidate action sequences are evaluated with the world model. Sampled trajectories are shown from the finetuned dynamics model, along with the resulting optimal plans under the finetuned and pretrained models. The finetuned model produces plans that account for real world dynamics mismatch, while the pretrained model generates qualitatively different plans.

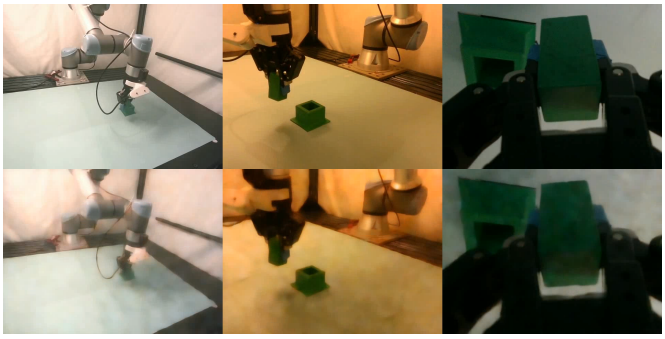

Fig. 9: Real world camera observations and images reconstructed from the corresponding encoded latent state z_t using an auxiliary encoder. We reiterate pixel reconstruction is not used as a training objective for the model.

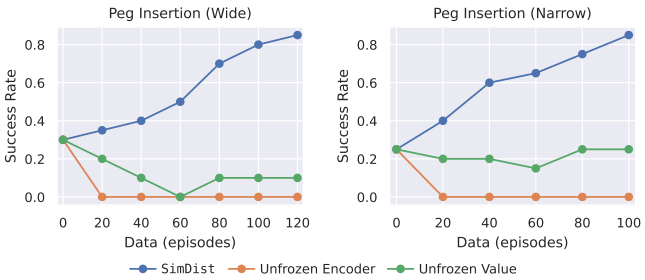

Fig. 10: Unfreezing world-model components during real-world adaptation.

We run the following ablations in simulation, with results in Table I (see Appendix D for details).

Data Scale and Diversity. We first study the effect of dataset scale by reducing simulation rollouts to 10% and 50% of the full pretraining data. Performance drops sharply as data is reduced for both systems. We next evaluate the role of data diversity by comparing pretraining on expert-only trajectories to our mixed datasets with equal data volume, and observe that expert-only data leads to a substantial performance drop. Together, these results highlight the importance of obtaining large, diverse datasets to provide the coverage needed to pretrain world models which can be used for effective planning.

Reward and Value Transformers. Next, we ablate the use of sequence-to-sequence transformers for reward and value prediction, replacing them with per-timestep MLP decoders as in [14, 15]. This consistently degrades performance across tasks. We attribute this to the inability of per-step models to capture trajectory-level structure, which is essential for accurately evaluating candidate action sequences during planning.

TABLE I: Ablation results in simulation, reporting success rates for manipulation tasks and average state-based reward per episode for the quadruped.

	Peg Insertion (SR)	Table Leg (SR)	Quadruped (Reward)
`SimDist`	0.90	0.85	22.78
50% data	0.72	0.61	22.73
10% data	0.06	0.02	19.38
Expert Data Only	0.10	0.05	16.68
MLP Reward+Value Models	0.82	0.60	19.47
Raw Obs. Reconstruction	0.32	0.21	23.34

Reconstruction Loss. We study the effect of adding an observation reconstruction loss to the training objective, a common design choice in MBRL [14, 3]. While reconstruction slightly improves quadruped performance, it leads to a steep performance drop for manipulation, consistent with the concern that pixel reconstruction pressures the latent state to encode task-irrelevant details (Section IV-A).

VI. Conclusion

We introduced `SimDist`, a framework for sim-to-real adaptation that uses the modular structure of world models to target the dynamics gap between simulation and reality. Across two precise manipulation tasks and two quadruped locomotion tasks, `SimDist` improves more efficiently and reliably than prior RL finetuning methods, which often collapse or fail to make meaningful progress. However, freezing reward and value models can cap performance when the transferred value function saturates or no longer distinguishes high-performing real-world trajectories. Closing the gap to near-perfect success may require selectively updating value functions in addition to dynamics. `SimDist` also does not train on internet-scale video or richer sensing modalities, and still depends on simulation coverage broad enough to support reliable planning.

Acknowledgments

The authors thank Trey Smith, Brian Coltin and the members of the WEIRD Lab at UW for their insights and feedback. This work was supported by a NASA Space Technology Graduate Research Opportunity under award 80NSSC23K1192 and the generous support of FieldAI.

References

[1] Karl Johan Åström. Adaptive control. In *Mathematical System Theory: The Influence of RE Kalman*, pages 437–450. Springer, 1995.

[2] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[3] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[4] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[5] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[6] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

[7] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[9] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[10] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018. doi: 10.1109/IROS.2018.8594448.

[11] Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world, 2023. URL https://arxiv.org/abs/2310.16029.

[12] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[13] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[14] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[15] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pages 8387–8406. PMLR, 2022.

[16] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[17] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022.

[18] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xCVJMsPv3RT.

[19] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. Pi*0.6: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[20] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[21] Petros A Ioannou and Jing Sun. *Robust adaptive control*, volume 1. PTR Prentice-Hall Upper Saddle River, NJ, 1996.

[22] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

[23] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2021. URL https://arxiv.org/abs/1906.08253.

[24] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors,

Proceedings of The 2nd Conference on Robot Learning, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/kalashnikov18a.html.

[25] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[26] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[27] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020. URL https://arxiv.org/abs/2006.04779.

[28] JB Lanier, Kyungmin Kim, Armin Karamzade, Yifei Liu, Ankita Sinha, Kat He, Davide Corsi, and Roy Fox. Adapting world models with latent-state dynamics residuals, 2025. URL https://arxiv.org/abs/2504.02252.

[29] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5 (47):eabc5986, 2020.

[30] Kun Lei, Huanyu Li, Dongjie Yu, Zhenyu Wei, Lingxiao Guo, Zhennan Jiang, Ziyu Wang, Shiyu Liang, and Huazhe Xu. Rl-100: Performant robotic manipulation with real-world reinforcement learning. *arXiv preprint arXiv:2510.14830*, 2025.

[31] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

[32] Jacob Levy, Tyler Westenbroek, and David Fridovich-Keil. Learning to walk from three minutes of real-world data with semi-structured dynamics models. In *Conference on Robot Learning*, pages 2061–2079. PMLR, 2025.

[33] Chenhao Li, Andreas Krause, and Marco Hutter. Offline robotic world model: Learning robotic policies without a physics simulator. *arXiv preprint arXiv:2504.16680*, 2025.

[34] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[35] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16961–16969. IEEE, 2024.

[36] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[37] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & chemical engineering*, 23(4-5):667–682, 1999.

[38] Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, et al. Factory: Fast contact for robotic assembly. *arXiv preprint arXiv:2205.03532*, 2022.

[39] Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, Stephen Spencer, Jessica Yung, Michael Dennis, Sultan Kenjeyev, Shangbang Long, Vlad Mnih, Harris Chan, Maxime Gazeau, Bonnie Li, Fabio Pardo, Luyu Wang, Lei Zhang, Frederic Besse, Tim Harley, Anna Mitenkova, Jane Wang, Jeff Clune, Demis Hassabis, Raia Hadsell, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 2: A large-scale foundation world model. 2024. URL https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model/.

[40] Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, pages 2279–2291. PMLR, 2021.

[41] Jean-Jacques E Slotine and Weiping Li. On the adaptive control of robot manipulators. *The international journal of robotics research*, 6(3):49–59, 1987.

[42] Laura Smith, J Chase Kew, Xue Bin Peng, Sehoon Ha,

Jie Tan, and Sergey Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *2022 international conference on robotics and automation (ICRA)*, pages 1593–1599. IEEE, 2022.

[43] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

[44] Laura Smith, Yunhao Cao, and Sergey Levine. Grow your limits: Continuous improvement with real-world rl for robotic locomotion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10829–10836. IEEE, 2024.

[45] Tyler Westenbroek, Fernando Castaneda, Ayush Agrawal, Shankar Sastry, and Koushil Sreenath. Lyapunov design for robust and efficient robotic reinforcement learning, 2022. URL https://arxiv.org/abs/2208.06721.

[46] Tyler Westenbroek, Jacob Levy, and David Fridovich-Keil. Enabling efficient, reliable real-world reinforcement learning with approximate physics-based models. In *Conference on Robot Learning*, pages 2478–2497. PMLR, 2023.

[47] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016. doi: 10.1109/ICRA.2016.7487277.

[48] Wenli Xiao, Haoru Xue, Tony Tao, Dvij Kalaria, John M Dolan, and Guanya Shi. Anycar to anywhere: Learning universal dynamics model for agile and adaptive mobility. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8819–8825. IEEE, 2025.

[49] Amber Xie, Oleh Rybkin, Dorsa Sadigh, and Chelsea Finn. Latent diffusion planning for imitation learning. *arXiv preprint arXiv:2504.16925*, 2025.

[50] Jie Xu, Eric Heiden, Iretiayo Akinola, Dieter Fox, Miles Macklin, and Yashraj Narang. Neural robot dynamics. *arXiv preprint arXiv:2508.15755*, 2025.

[51] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 1(2):6, 2023.

[52] Patrick Yin, Tyler Westenbroek, Simran Bagaria, Kevin Huang, Ching-an Cheng, Andrey Kobolov, and Abhishek Gupta. Rapidly adapting policies to the real world via simulation-guided fine-tuning. *arXiv preprint arXiv:2502.02705*, 2025.

[53] Patrick Yin, Tyler Westenbroek, Zhengyu Zhang, Ignacio Dagnino, Eeshani Shilamkar, Numfor Mbiziwo-Tiapo, Simran Bagaria, Xinlei Liu, Galen Mullins, Andrey Kolobov, and Abhishek Gupta. Emergent dexterity via diverse resets and large-scale reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nAO9LcV7nE.

[54] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142, 2020.

[55] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34: 28954–28967, 2021.

[56] Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations, 2025. URL https://arxiv.org/abs/2505.10911.

[57] Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. *arXiv preprint arXiv:2412.07762*, 2024.

[58] Chuning Zhu, Max Simchowitz, Siri Gadipudi, and Abhishek Gupta. Repo: Resilient model-based reinforcement learning by regularizing posterior predictability. *Advances in Neural Information Processing Systems*, 36: 32445–32467, 2023.

[59] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

APPENDIX

A. Diverse Data Generation Details

Algorithm 2 details the data generation process. Data generation proceeds with running many environments in parallel. For the j-th environment, we sample a diagonal action-noise covariance Σ_j, where each element on the diagonal is sampled between a minimum and maximum variance: $\sigma_i \sim \mathcal{U}[\sigma_i^{\texttt{min}}, \sigma_i^{\texttt{max}}]$. When each environment is reset, we sample contiguous noise intervals during which Gaussian noise is added to policy actions. In addition, each environment is assigned a randomly selected policy checkpoint from $\{\pi^k\}_{k=1}^K$, or the expert policy π^e. Together, policy mixing and temporally structured action noise produce a dataset that captures both expert-like trajectories and systematic deviations beyond the optimal manifold, which is critical for world-model training. During rollouts, we query the optimal state-based value function V^e at each visited state to generate value targets v_t for distilling an approximate optimal value function (Section IV-A). We also record an expert action flag b_t^e that distinguishes the expert actions from the expert policy π^e versus noised or earlier-checkpoint actions to support behavior cloning (Section IV-A).

Algorithm 2 Diverse Data Generation

Input: Expert policy π^e, policy checkpoints $\{\pi^k\}_{k=1}^K$, optimal value function V^e
Output: Simulation dataset $\mathcal{D}_{\texttt{sim}}$
for environment $j \in \{1 \dots N_E\}$ **do**
    Sample action noise variance: $\Sigma_j = \text{diag}(\sigma)$ with $\sigma_i \sim \mathcal{U}[\sigma_i^{\texttt{min}}, \sigma_i^{\texttt{max}}]; i \in \{1, \dots, \dim(\mathcal{A})\}$
for $t = 0$ to number of steps N_T **do**
    for environment $j \in \{1 \dots N_E\}$ **do**
        if s_t is terminal **then**
            Reset environment
            Sample noise intervals $\mathcal{T}_{\texttt{noise}}$
            Randomly select a policy π^k from $\{\pi^k\}_{k=1}^K$ and assign it to the environment.
        $\varepsilon_t \sim \mathcal{N}(0, \Sigma_j)$ if $t \in \mathcal{T}_{\texttt{noise}}$, else $\varepsilon_t \leftarrow 0$
        $a_t \leftarrow \pi^k(s_t) + \varepsilon_t$
        $b_t^e \leftarrow 1$ if (using expert π^e and $\varepsilon_t = 0$), else $b_t^e \leftarrow 0$
        $v_t \leftarrow V^e(s_t)$
        $(s_{t+1}, o_t, r_t) = \texttt{EnvStep}(s_t, a_t)$
        Add $(o_t, a_t, b_t^e, r_t, v_t)$ to $\mathcal{D}_{\texttt{sim}}$

B. Manipulation Experiment Details

Expert Policy Training. For our manipulation experts we use the expert policies π^e and value function V^e from [53], replicating the training from this work exactly. For the sake of brevity, we refer the reader to [53] for more details, including the exact rewards we use.

Data Generation. To generate the simulation dataset $\mathcal{D}_{\texttt{sim}}$, environments are reset with sub-optimal policies $\{\pi^k\}_{k=1}^K$ with probability 0.5, and Gaussian action perturbations are injected in contiguous intervals sampled from $\mathcal{U}[1, 5]$ steps, interleaved with noise-free intervals sampled from $\mathcal{U}[5, 10]$ steps. We generate 100k trajectories for each tasks, of which approximately 36% optimal actions. We save policies every 100 checkpoints up to checkpoint 1000.

World Model Structure. The encoder for the world model first passes each of the three camera images through a ResNet-18 encoder pretrained on imagenet, producing embeddings of size 3×512, which are stacked and concatenated with the robots 6 joint observations and then passed through an MLP to produce the latent z. Specific parameters are in Table II.

TABLE II: World model architectural parameters for manipulation.

Parameter	Value
Embedding dimension	64
All transformers MLP hidden size	256
Dynamics transformer layers	3
Dynamics transformer heads	4
Reward transformer layers	1
Reward transformer heads	1
Value transformer layers	1
Value transformer heads	1
Base policy transformer layers	4
Base policy transformer heads	8

World Model Pretraining. We pretrain the world model for two epochs over the full simulation dataset $\mathcal{D}_{\texttt{sim}}$. Using a batch size of 256 and approximately 200k gradient updates. Optimization is performed with Adam using an initial learning rate of 2×10^{-4}, which is annealed to 1×10^{-4} via a cosine decay schedule, with a linear warmup over the first 10,000 steps. We apply data augmentation by injecting zero-mean Gaussian noise into the proprioceptive observations, and visual augmentations such as color jitter, gaussian blurring and random cropping.

Hardware Deployment. We use MPPI as implemented in TD-MPC [15] with the hyperparameters listed in Table III.

TABLE III: MPPI parameters for manipulation.

Parameter	Value
Candidate actions batch size	250
Noised base policy actions batch size	100
Solver iterations	3
Initial action standard deviation	1.0
Minimum action standard deviation	0.05
Base policy action standard deviation	0.1
Elites	64
Temperature	0.4
Momentum	0.0
Discount	0.99

C. Quadruped Experiment Details

Expert Policy Training. We train a state-based expert policy π^e and its associated optimal value function V^e using PPO in IsaacLab [36]. Both the policy and value networks are MLPs with three hidden layers of width 512 and operate on privileged simulator state variables listed in Table IV. The expert is trained with a dense state-based reward composed of the terms

summarized in Table V; full implementation details will be released with the public code. To improve robustness and coverage, we apply domain randomization over the parameters in Table IV and randomize terrain conditions across steps, boxes, rough terrain, and slopes, using a curriculum that gradually increases terrain difficulty. Training is performed with 4096 parallel simulation environments over 5000 PPO iterations, with 24 environment steps per iteration, for a total of 490M environment steps. We save policy checkpoints at iterations $\{0, 50, 100, 150, 200, 250, 300, 400, 500, 1000, 2000\}$.

Data Generation. To generate the simulation dataset $\mathcal{D}_{\texttt{sim}}$, environments are reset with sub-optimal policies $\{\pi^k\}_{k=1}^K$ with probability 0.5, and Gaussian action perturbations are injected in contiguous intervals sampled from $\mathcal{U}[1, 50]$ steps, interleaved with noise-free intervals sampled from $\mathcal{U}[25, 500]$ steps. We run 4096 parallel environments for 25000 steps, yielding approximately 100M data points, of which 55.7% correspond to uncorrupted expert actions. Data generation takes approximately 7 hours with a single NVIDIA RTX 4500 Ada GPU.

World Model Structure. Figure 11 illustrates the world model architecture, and Table VII lists the corresponding model parameters used in the quadruped experiments. The observation space for the quadruped is given in Table VI. The history encoder processes a history of proprioceptive observations (all observations except the height map) and actions by first projecting each input, assigning a type embedding to distinguish observations from actions, and interleaving the resulting embeddings to form the history representation h_t. The latent encoder E_θ encodes the local terrain height map using a CNN followed by spatial encoding, flattening, and projection; this representation is concatenated with the projected latest proprioceptive observation and passed through an MLP to produce the latent state embedding. Commands consist of the desired forward, lateral, and yaw velocities of the base $g_t := (v_t^x, v_t^y, \omega_t)$. The future commands $g_{t:t+T-1}$ are concatenated to the inputs of the base policy π_θ, reward model R_θ, and value model V_θ.

TABLE IV: Privileged simulator state space for the quadruped, along with domain randomization ranges, where applicable.

State Variable	Dim.	Domain Rand.
Base linear velocity	3	-
Base angular velocity	3	-
Projected gravity vector	3	-
Commanded base twist	3	-
Joint angles	12	-
Joint speeds	12	-
Previous action	12	-
Cosine / sine of phase	2	-
Height map	$(21, 15)$	-
Foot force wrenches	$(4, 6)$	-
Foot heights	4	-
Base mass	1	$-1.0, +3.0$ kg
Static / dynamic friction	2	$[0.2, 1.2]$
Coefficient of restitution	1	$[0.0, 0.3]$
Joint stiffness	12	$\pm 10\%$
Joint damping	12	$\pm 10\%$
Joint friction	12	$[0.0, 0.05]$

TABLE V: State-based reward terms used for quadruped expert policy training.

Term	Weight
Commanded x, y-velocity tracking reward	1.5
Commanded yaw rate tracking reward	0.75
Desired gait reward	0.05
Desired gait foot height reward	0.2
Base z-velocity penalty	-2.0
Base angular velocity penalty	-0.05
Base orientation penalty	-4.0
Deviation from default hip joint angles penalty	-0.25
Joint torque penalty	-2.0×10^{-4}
Joint acceleration penalty	-2.5×10^{-7}
Action rate penalty	-0.01

TABLE VI: Observation space for the quadruped.

Observation	Dimension
Base linear velocity (local frame)	3
Base angular velocity (local frame)	3
Projected gravity vector	3
Joint angles	12
Joint speeds	12
Cosine and sine of phase	2
Height map	$(21, 15)$

World Model Pretraining. We pretrain the world model for two epochs over the full simulation dataset $\mathcal{D}_{\texttt{sim}}$. Using a batch size of 512, this corresponds to approximately 3.69×10^5 gradient update steps. Optimization is performed with Adam using an initial learning rate of 2×10^{-4}, which is annealed to 1×10^{-4} via a cosine decay schedule, with a linear warmup over the first 10,000 steps. We apply data augmentation by injecting zero-mean Gaussian noise into the input observations (both proprioceptive and height map) during training. Pretraining requires approximately 28 hours on a single NVIDIA RTX 4500 Ada GPU.

Hardware Deployment. We use MPPI as implemented in TD-MPC [15] with the hyperparameters listed in Table VIII. To encourage straight-line locomotion during quadruped experiments, we compute commanded base velocities g_t from the robot's current base pose using a PD controller on position, and provide these commands to the world model.

Detailed Results. Table IX reports detailed real-world quadruped locomotion results on both tasks across commanded

TABLE VII: World model architectural parameters for the quadruped.

Parameter	Value
Embedding dimension	64
Proprioceptive observations MLP hidden dims	128, 128
CNN kernel size	3
CNN strides	2, 2, 2
CNN features	8, 16, 32
All transformers MLP hidden size	256
Dynamics transformer layers	2
Dynamics transformer heads	8
Reward transformer layers	1
Reward transformer heads	1
Value transformer layers	1
Value transformer heads	1
Base policy transformer layers	4
Base policy transformer heads	8

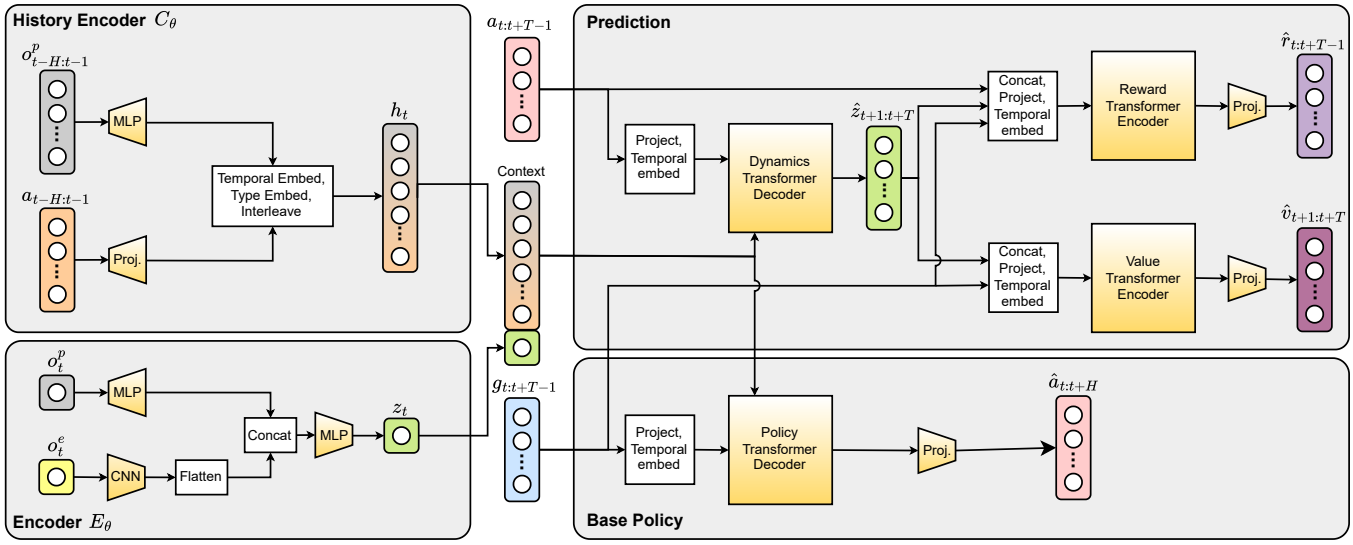

Fig. 11: Detailed world model architecture for the quadruped.

TABLE VIII: MPPI parameters for the quadruped.

Parameter	Value
Candidate actions batch size	450
Noised base policy actions batch size	22
Solver iterations	8
Initial action standard deviation	2.0
Minimum action standard deviation	0.05
Base policy action standard deviation	0.05
Elites	64
Temperature	0.25
Momentum	0.0
Discount	0.99

forward speeds. We report both success rate (successful trials out of five) and average forward progress (mean $\pm$ standard deviation) over all trials at each speed. The *Pretrained model* corresponds to zero-shot deployment of the simulation-pretrained world model without any real-world finetuning. While this model occasionally achieves partial forward progress, it fails to complete the task reliably, highlighting the severity of the sim-to-real dynamics gap. The *Single-step BC policy*, which serves as the initial policy for the RLPD and IQL baselines prior to finetuning, improves performance in some settings but remains inconsistent and rarely achieves full task completion.

After real-world finetuning (32.1 minutes of data for Foam and 35.7 minutes for Slippery Slope), `SimDist` consistently achieves the highest success rates and forward progress across all tested speeds. In contrast, both IQL and RLPD exhibit limited improvement despite access to the same real-world data budget. In particular, RLPD destabilized the robot during adaptation on the Foam task and is therefore not reported for that condition.

D. Ablations Details

Each ablation corresponds to a separately pretrained world model. Across all ablations, the same planning hyperparameters and evaluation environments are used, and each configuration is evaluated on an identical set of randomized environments. Next, we discuss platform specific details.

Manipulation. For both manipulation tasks, we evaluate success rates over the initial condition and domain randomization described in the environments from [53].

Quadruped. The robot is commanded to walk forward at a specified target speed until episode termination. For each evaluated model, we instantiate environments covering all combinations of parameters listed in Table X, resulting in 1080 distinct environments per model. All models are evaluated on the same fixed set of environments to ensure fair comparison. Each environment is run for a single episode, during which state-based reward is accumulated until termination. Episodes terminate either after 1000 simulation steps or upon failure, defined as body contact with the ground or violation of base orientation limits. Results are summarized in Table I, where we report the average accumulated reward per episode across all environments.

TABLE IX: Real-world quadruped results for both tasks. Success is reported as successful trials out of five. Forward progress is reported as mean ± standard deviation (meters) across trials at each commanded speed. The *Pretrained model* corresponds to zero-shot deployment of the simulation-trained world model. The *Single-step BC policy* is the behavior cloning policy used to initialize IQL and RLPD prior to finetuning. `SimDist`, IQL, and RLPD results reflect performance after real-world finetuning using 35.7 minutes (Slippery Slope) and 32.1 minutes (Foam) of data. RLPD results on the Foam task are not reported, as the method destabilized the robot prior to evaluation.

	Speed	Pretrained model		Single-step BC policy		`SimDist` (ours)		IQL		RLPD	
	$\mathrm{m\,s^{-1}}$	Fwd. Prog.	Success	Fwd. Prog.	Success	Fwd. Prog.	Success	Fwd. Prog.	Success	Fwd. Prog.	Success
Slippery Slope	0.1	0.70 ± 0.56	0/5	1.49 ± 0.29	2/5	**1.78 ± 0.08**	**4/5**	0.00 ± 0.00	0/5	0.32 ± 0.01	0/5
	0.3	0.43 ± 0.13	0/5	1.43 ± 0.25	1/5	**1.82 ± 0.00**	**5/5**	0.39 ± 0.56	0/5	0.34 ± 0.05	0/5
	0.5	0.90 ± 0.40	0/5	0.62 ± 0.23	0/5	**1.82 ± 0.00**	**5/5**	0.46 ± 0.42	0/5	0.35 ± 0.01	0/5
Foam	0.2	2.98 ± 0.02	3/5	2.07 ± 1.01	1/5	**3.00 ± 0.00**	**5/5**	0.92 ± 0.48	1/5	–	–
	0.7	2.39 ± 0.71	2/5	1.54 ± 0.81	1/5	**3.00 ± 0.00**	**5/5**	2.25 ± 0.87	2/5	–	–
	1.2	1.70 ± 0.99	0/5	2.45 ± 0.47	2/5	**3.00 ± 0.00**	**5/5**	2.73 ± 0.34	3/5	–	–

TABLE X: Parameter values used to construct quadruped ablation environments in simulation. Each environment is defined by a unique combination of commanded forward speed, ground friction coefficient, terrain type, and terrain difficulty. All combinations are evaluated for each model, yielding 1080 environments per model.

Speed	Friction	Terrain	Terrain Difficulty
0.2	0.2	Boxes	0.2
0.4	0.4	Rough	0.4
0.6	0.6	Stairs Up	0.6
0.8	0.8	Stairs Down	0.8
1.0	1	Slope Up	1
1.2	1.2	Slope Down	