Robotic World Model: A Neural Network Simulator for Robust Policy Optimization in Robotics

Chenhao Li
ETH Zurich, Switzerland
chenhli@ethz.ch

Andreas Krause
ETH Zurich, Switzerland
krausea@ethz.ch

Marco Hutter
ETH Zurich, Switzerland
mahutter@ethz.ch

https://sites.google.com/view/roboticworldmodel

Abstract

Learning robust and generalizable world models is crucial for enabling efficient and scalable robotic control in real-world environments. In this work, we introduce a novel framework for learning world models that accurately capture complex, partially observable, and stochastic dynamics. The proposed method employs a dual-autoregressive mechanism and self-supervised training to achieve reliable long-horizon predictions without relying on domain-specific inductive biases, ensuring adaptability across diverse robotic tasks. We further propose a policy optimization framework that leverages world models for efficient training in imagined environments and seamless deployment in real-world systems. This work advances model-based reinforcement learning by addressing the challenges of long-horizon prediction, error accumulation, and sim-to-real transfer. By providing a scalable and robust framework, the introduced methods pave the way for adaptive and efficient robotic systems in real-world applications.

1 Introduction

Robotic systems have achieved remarkable advancements in recent years, driven by progress in reinforcement learning (RL) [1, 2] and control theory [3, 4]. A prevalent limitation in many approaches is the lack of adaptation and learning once the policy is deployed on the real system [5, 6, 7, 8]. This results in underutilization of the valuable data generated during real-world interactions. Robotic systems operating in dynamic and uncertain environments require the ability to continually adapt their behavior to new conditions [9]. The inability to exploit real-world experience for further learning restricts the system's robustness and limits its ability to handle evolving scenarios effectively. Truly intelligent robotic systems should operate efficiently and reliably using limited data, adapting to real-world conditions in a scalable manner [10, 11]. While model-free RL algorithms such as Proximal Policy Optimization (PPO) [2] and Soft Actor-Critic (SAC) [1] have demonstrated impressive results in simulation, their high interaction requirements make them impractical for real-world robotics. Sample-efficient methods are therefore essential for leveraging the information in real-world data without extensive environment interactions [12, 13].

A promising solution is the use of predictive models of the environment, commonly referred to as world models [14, 15]. World models simulate environment dynamics to enable planning and policy optimization, often referred to as *learning in imagination* [16]. These models have shown success across diverse robotic domains, including manipulation [17, 18], navigation [11], and locomotion [10]. However, developing reliable and generalizable world models poses unique challenges due to the complexity of real-world dynamics, including nonlinearities, stochasticity, and partial observability [19, 20]. Existing approaches often incorporate domain-specific inductive biases, such as structured state representations or hand-designed network architectures [21, 22, 23], to improve model fidelity. While effective, these methods are limited in their scalability and adaptability to novel environments or tasks. In contrast, a general framework for learning world models without

Preprint. Under review.

Figure 1: Autoregressive imagination, ground-truth simulation, and real-world deployment of RWM. For each environment, the top row showcases the RWM autoregressively predicting future trajectories in imagination. The second row visualizes the ground truth evolution in simulation. Specifically for the ANYmal D quadruped and Unitree G1 humanoid, the framework achieves robust policy optimization through MBPO-PPO, enabling zero-shot deployment on hardware.

domain-specific assumptions has the potential to enhance generalization and applicability across a wide range of robotic systems and scenarios.

In this work, we present a novel approach for learning world models that emphasizes robustness and accuracy over long-horizon predictions. Our method is designed to operate without handcrafted representations or specialized architectural biases, enabling broad applicability to diverse robotic tasks. To evaluate the utility of these learned models, we further propose a policy optimization method using PPO and demonstrate successful deployment in both simulated and real-world environments. To the best of our knowledge, this is the first framework to reliably train policies on a learned neural network simulator without any domain-specific knowledge and deploy them on physical hardware with minimal performance loss.

Our contributions are summarized as follows: **(i)** We introduce a novel network architecture and training framework that enables the learning of reliable world models capable of long autoregressive rollouts, a critical property for downstream planning and control. **(ii)** We provide a comprehensive evaluation suite spanning diverse robotic tasks to benchmark our method. Comparative experiments with existing world model frameworks demonstrate the effectiveness of our approach. **(iii)** We propose an efficient policy optimization framework that leverages the learned world models for continuous control and generalizes effectively to real-world scenarios with hardware experiments, including both quadruped and humanoid systems.

By addressing the challenges associated with learning world models, this work contributes toward bridging the gap between data-driven modeling and real-world deployment. The proposed framework enhances the scalability, adaptability, and robustness of robotic systems, paving the way for broader adoption of model-based reinforcement learning in real-world applications. Supplementary videos for this work are available on `https://sites.google.com/view/roboticworldmodel`.

2 Related work

2.1 World Models for Robotics

World models have emerged as a cornerstone in robotics for capturing system dynamics and enabling efficient planning and control through simulated trajectories. A prominent application of world models is in robotic control, where dynamics models are used to describe real-world dynamics for policy optimization [24]. Extensions to vision-based tasks have been realized through visual foresight techniques [18, 25, 17], which learn visual dynamics for planning in high-dimensional sensory spaces. Similar ideas are applied to train RL agents in such world models aiming to fully replicate real environment interactions [14, 26]. These approaches underline the versatility of world models in tasks requiring rich perceptual inputs.

To improve the generalization of black-box neural network-based world models beyond the training distribution, many works incorporate known physics principles or state structures into model design, addressing potential limitations in control performance. Examples include foot-placement dynamics [21], object invariance [22], granular media interactions [27], frequency domain parameterization [23], rigid body dynamics [20], and semi-structured Lagrangian dynamics models [28]. While these methods demonstrate impressive results, they often require strong domain knowledge and carefully crafted inductive biases, which can restrict their scalability and adaptability to diverse robotic applications. Latent-space dynamics models offer an alternative by abstracting the state space into compact representations, enabling efficient long-horizon planning. Deep Planning Network (PlaNet) [15] and its successor Dreamer [29, 11, 30] exemplify this trend, achieving state-of-the-art performance in continuous control and visual navigation tasks. These frameworks have been extended to real-world robotics [19, 31], demonstrating their potential in both simulation and hardware deployment.

2.2 Model-Based Reinforcement Learning

Model-Based Reinforcement Learning (MBRL) has emerged as a powerful approach to address the limitations of model-free reinforcement learning, particularly in scenarios where sample efficiency and safety are critical. Unlike model-free methods, which learn policies directly from interactions with the environment, MBRL leverages a learned model of the environment to simulate interactions, enabling more efficient and safer policy learning. One of the pioneering methods in MBRL is Probabilistic Ensembles with Trajectory Sampling (PETS), which uses an ensemble of probabilistic neural networks to model the environment dynamics [12]. Building on the idea of latent-space modeling, PlaNet leverages a latent dynamics model to plan directly in a learned latent space [15]. Dreamer extends the concept by incorporating an actor-critic framework into the latent dynamics model, enabling the simultaneous learning of both the dynamics model and the policy [29, 11, 30]. Variations on the architectural design also see success in improving generation capabilities of such latent dynamics models with autoregressive transformer [32] and the stochastic nature of variational autoencoders [33]. Recent advancements in this area include TD-MPC and TD-MPC2, which integrate model-based learning with MPC to achieve high-performance control in dynamic environments [34, 35, 36].

Recognizing the strengths of both model-based and model-free methods, several hybrid approaches have been developed to combine the sample efficiency of MBRL with the robustness of model-free reinforcement learning. One notable example is Model-Based Policy Optimization (MBPO), which uses a model-based approach for planning and policy optimization but refines the policy using model-free updates [13]. It emphasizes selectively relying on the learned model when its predictions are accurate, thus mitigating the negative effects of model inaccuracies. Building on similar principles, Model-based Offline Policy Optimization (MOPO) extends the framework to the offline setting, where learning is conducted entirely from previously collected data without further environment interaction [37]. In contrast to using zeroth-order model-free reinforcement learning for policy optimization, first-order gradient-based optimization is used to improve policy

learning [38, 39]. This allows for more efficient and precise policy updates, particularly in complex, high-dimensional environments, where accurate gradient information is crucial for performance. Our framework extends MBPO by integrating it with PPO over extensive autoregressive rollouts, making it particularly effective for complex robotic control tasks.

3 Approach

3.1 Reinforcement Learning and World Models

We formulate the problem by modeling the environment as a Partially Observable Markov Decision Process (POMDP) [40], defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces, respectively. The transition kernel $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ captures the environment dynamics $p\left(s_{t+1} \mid s_t, a_t\right)$, while the reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ maps transitions to scalar rewards. Observations $o_t \in \mathcal{O}$ are emitted according to probabilities $p\left(o_t \mid s_t\right)$, governed by the observation kernel $O : \mathcal{S} \to \mathcal{O}$. The agent seeks to learn a policy $\pi_\theta : \mathcal{O} \to \mathcal{A}$ that maximizes the expected discounted return $\mathbb{E}_{\pi_\theta}\left[\sum_{t \geq 0} \gamma^t r_t\right]$, where r_t is the reward at time t and $\gamma \in [0, 1]$ is the discount factor.

World models [14] approximate the environment dynamics and facilitate policy optimization by enabling simulated environment interactions in *imagination* [16]. Training typically involves three iterative steps: (1) collect data from real environment interactions; (2) train the world model using the collected data; and (3) optimize the policy within the simulated environment produced by the world model.

Despite the success of existing frameworks in achieving tasks in simplified settings, their application to complex low-level robotic control remains a significant challenge. To address this gap, we propose Robotic World Model (RWM), a novel framework for learning robust world models in partially observable and dynamically complex environments. RWM builds on the core concept of world models but introduces architectural and training innovations that enable reliable long-horizon predictions, even in stochastic and partially observable settings. By incorporating historical context and autoregressive training, RWM addresses challenges such as error accumulation and partially observable and discontinuous dynamics, which are critical in real-world robotics applications.

3.2 Self-supervised Autoregressive Training

To address the inherent complexity of partially observable environments, we propose a self-supervised autoregressive training framework as the backbone of RWM. This framework trains the world model p_ϕ to predict future observations by leveraging both historical observation-action sequences and its own predictions, ensuring robustness over extended rollouts.

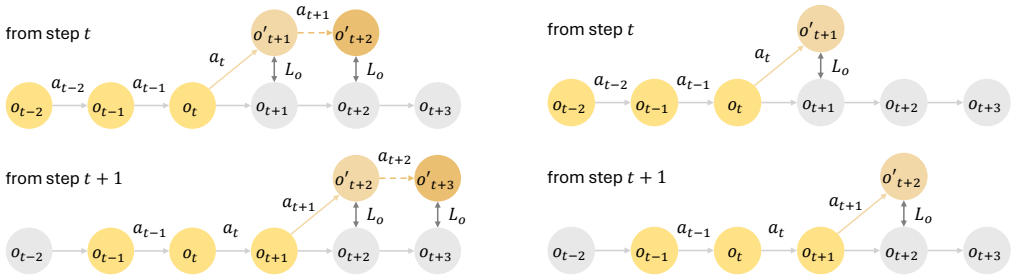
The input to the world model consists of a sequence of observation-action pairs spanning M historical steps. At each time step t, the model predicts the distribution of the next observation $p\left(o_{t+1} \mid o_{t-M+1:t}, a_{t-M+1:t}\right)$. Predictions are generated autoregressively: at each step, the predicted observation o'_{t+1} is appended to the history and combined with the next action a_{t+1} to serve as input for subsequent predictions. This process is repeated over a prediction horizon of N steps, producing a sequence of future predictions. The predicted observation k steps ahead can thus be written as

$$o'_{t+k} \sim p_\phi\left(\cdot \mid o_{t-M+k:t}, o'_{t+1:t+k-1}, a_{t-M+k:t+k-1}\right). \tag{1}$$

A similar process is also applied to predict privileged information c, such as contacts, providing an additional learning objective that implicitly embeds critical information for accurate long-term predictions. Such a training scheme introduces the model to the distribution it will encounter at test time, reducing the mismatch between training and inference distributions. Overall, the model is optimized by minimizing the multi-step prediction error:

$$\mathcal{L} = \frac{1}{N} \sum_{k=1}^{N} \alpha^k \left[L_o\left(o'_{t+k}, o_{t+k}\right) + L_c\left(c'_{t+k}, c_{t+k}\right)\right], \tag{2}$$

where L_o and L_c quantify the discrepancy between predicted and true observations and privileged information, and α denotes a decay factor. This autoregressive training objective encourages the hidden states to encode representations that support accurate and reliable long-horizon predictions.



(a) Autoregressive training. (b) Teacher-forcing training.

Figure 2: Comparison of training paradigms for world models with an example of a history horizon $H = 3$. (a) Autoregressive training operates with an example of a forecast horizon $N = 2$, leveraging historical data and its own predictions for long-horizon robustness. The dashed arrows denote the sequential autoregressive prediction steps. (b) Teacher-forcing training can be viewed as a special case of autoregressive training with a forecast horizon $N = 1$, using ground truth observations for next-step predictions to optimize parallelization but limiting robustness to error accumulation.

Training data is constructed by sliding a window of size $M + N$ over collected trajectories, providing sufficient historical context for prediction targets. To improve gradient propagation through autoregressive predictions, we apply reparameterization tricks to enable effective end-to-end optimization. By incorporating historical observations, RWM captures unobservable dynamics, addressing the challenges of partially observable and potentially discontinuous environments. The autoregressive training mitigates error accumulation, a common issue in long-horizon predictions, and eliminates the need for handcrafted representations or domain-specific inductive biases, enhancing generalization across diverse tasks. This process is illustrated in Fig. 2a, in contrast to the teacher-forcing pipeline in Fig. 2b, which is commonly adopted to train many popular architectures [29, 41]. Specifically, teacher-forcing can be viewed as a special case of autoregressive training with forecast horizon $N = 1$, which boosts training with higher parallelization.

While the proposed autoregressive training framework can be applied to any network architecture, RWM utilizes a GRU-based architecture for its ability to maintain long-term historical context while operating on low-dimensional inputs. The network predicts the mean and standard deviation of a Gaussian distribution describing the next observation. Our framework introduces a *dual-autoregressive mechanism*: **(i)** *Inner autoregression* updates GRU hidden states autoregressively after each historical step within the context horizon M. **(ii)** *Outer autoregression* feeds predicted observations from the forecast horizon N back into the network. This architecture, visualized in Fig. S6, ensures robustness to long-term dependencies and transitions, making RWM suitable for complex robotics applications.

3.3 Policy Optimization on Learned World Models

Policy optimization in RWM is conducted using the learned world model, following a framework inspired by Model-Based Policy Optimization (MBPO) [13] and the Dyna algorithm [42]. During imagination, the actions are generated recursively by the policy π_θ conditioned on the observations predicted by the world model p_ϕ, which is further conditioned on the previous predictions. The actions at time $t + k$ can thus be written as

$$a'_{t+k} \sim \pi_\theta \left(\cdot \mid o'_{t+k} \right), \tag{3}$$

where o'_{t+k} is drawn autoregressively according to Eq. 1. Rewards are computed from imagined observations and privileged information. The approach combines model-based imagination with model-free RL to achieve efficient and robust policy optimization, as outlined in Algorithm 1.

The replay buffer $\mathcal{D}$ aggregates real environment interactions collected by a single agent. The world model p_ϕ is trained on this data following the autoregressive scheme described in Sec. 3.2. Imagination agents are initialized from samples in $\mathcal{D}$ and simulate trajectories using the world model for T steps, enabling policy updates through a reinforcement learning algorithm. The training diagram is visualized in Fig. S7.

Algorithm 1 Policy optimization with RWM

Initialize policy π_θ, world model p_ϕ, and replay buffer $\mathcal{D}$
for learning iterations $= 1, 2, \ldots$ **do**
  Collect observation-action pairs in $\mathcal{D}$ by interacting with the environment using π_θ
  Update p_ϕ with autoregressive training using data sampled from $\mathcal{D}$ according to Eq. 2
  Initialize imagination agents with observations sampled from $\mathcal{D}$
  Roll out imagination trajectories using π_θ and p_ϕ for T steps according to Eq. 3
  Update π_θ using PPO or another reinforcement learning algorithm
end for

While PPO is known for its strong performance in robotic tasks, training it on learned world models poses unique challenges. Model inaccuracies can be exploited during policy learning, leading to discrepancies between the imagined and true dynamics. This issue is exacerbated by the extended autoregressive rollouts required for PPO, which compound prediction errors. We denote this policy optimization method by MBPO-PPO. Despite these challenges, RWM demonstrates its robustness by successfully optimizing policies over a hundred autoregressive steps with MBPO-PPO, far exceeding the capabilities of existing frameworks such as MBPO [13], Dreamer [29, 11, 30], or TD-MPC [34, 36]. This result underscores the accuracy and stability of the proposed training method and its ability to synthesize policies deployable on hardware.

4 Experiments

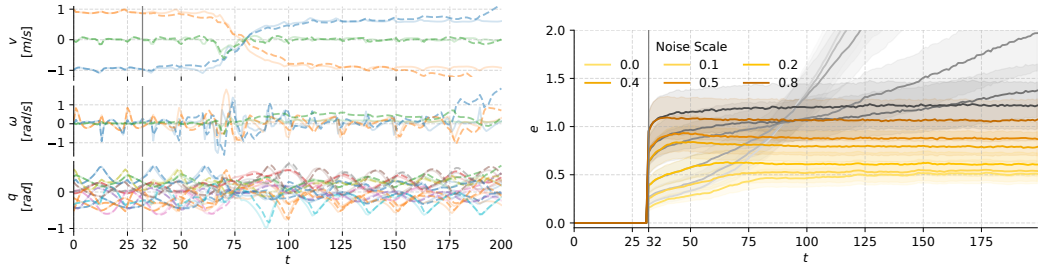
We validate RWM through a comprehensive set of experiments across diverse robotic systems, environments, and network architectures. The experiments are designed to assess the accuracy and robustness of RWM, evaluate its architectural and training design choices, and demonstrate its effectiveness across diverse robotic tasks in Isaac Lab [43] and in real-world deployment combined with MBPO-PPO. We start the analysis by looking into the autoregressive prediction accuracy and robustness of the world model learned with simulation data induced by a velocity tracking policy. The observation and action spaces of the world model are detailed in Table S2 and Table S4. We then compare various network architectures and the error induced across diverse robotic environments and tasks to demonstrate the generality of RWM. And finally, we learn a policy in RWM with the proposed MBPO-PPO and demonstrate the applicability and robustness of the method on ANYmal D [44] and Unitree G1 hardware.

4.1 Autoregressive Trajectory Prediction

The capability of a world model to maintain high fidelity during autoregressive rollouts is critical for effective planning and policy optimization. To evaluate this aspect, we analyze the autoregressive prediction performance of RWM using trajectories collected from ANYmal D hardware. The control frequency of the robot is at $50\,Hz$. The model is trained with history horizon $M = 32$ and forecast horizon $N = 8$. Further details on the network architecture and training parameters are summarized in Sec. A.2.1 and Sec. A.3.1, respectively. The autoregressive trajectory predictions by RWM are visualized in Fig. 3a.

The results demonstrate that RWM exhibits a remarkable alignment between predicted and ground truth trajectories across all observed variables. This consistency persists over extended rollouts, showcasing the model's ability to mitigate compounding errors—a critical challenge in long-horizon predictions. This performance is attributed to the dual-autoregressive mechanism introduced in Sec. 3.2, which stabilizes predictions despite the short forecast horizon employed during training. A comparison of state evolution between the RWM prediction and the ground truth simulation is illustrated in Fig. 1 (bottom). The visualization highlights the ability of RWM to maintain consistency in trajectory predictions over long horizons, even beyond the training forecast horizon. This robustness is pivotal for stable policy learning and deployment, as discussed further in Sec. 4.4.

It is notable that the choice of history horizon M and forecast horizon N plays a critical role in the training and performance of RWM. Our ablation study in Sec. A.4.1 reveals that, while extending both M and N improves accuracy, practical considerations of computational cost necessitate careful tuning of these hyperparameters to achieve optimal performance.



(a) Autoregressive trajectory prediction by RWM.

(b) Prediction error under Gaussian noise.

Figure 3: (Left) Solid lines represent ground truth trajectories, while dashed lines denote predicted state evolution. Predictions commence at $t = 32$ using historical observations, with future observations predicted autoregressively by feeding prior predictions back into the model. (Right) Yellow curves denote RWM at varying noise levels, demonstrating consistent robustness and lower error accumulation across forecast steps. Grey curves represent the MLP baseline, which exhibits significantly higher error accumulation and reduced robustness to noise.

4.2 Robustness under Noise

A critical challenge in training world models is their ability to generalize under noisy conditions, particularly when predictions rely on autoregressive rollouts. Even small deviations from the training distribution can cascade into untrained regions, causing the model to hallucinate future trajectories. To assess the robustness of RWM, we analyze its performance under Gaussian noise perturbations applied to both observations and actions. We compare the results with an MLP-based baseline also trained autoregressively with the same history and forecast horizon, as shown in Fig. 3b, where yellow curves denote the relative prediction error e for RWM, and grey curves represent the MLP baseline.

The results indicate a clear advantage of RWM over the MLP baseline across all noise levels. As forecast steps increase, the relative prediction error of the MLP model grows significantly, diverging more rapidly than RWM. In contrast, RWM demonstrates superior stability, maintaining lower prediction errors even under high noise levels. This robustness can be attributed to the dual-autoregressive mechanism introduced in Sec. 3.2, which ensures stability in long-horizon predictions. This design minimizes the accumulation of errors by continually refining the state representation toward long-term predictions, even in the presence of noisy inputs.

4.3 Generality across Robotic Environments

To assess the generality and robustness of RWM across a diverse range of robotic environments, we compare its performance with several baseline methods, including MLP, recurrent state-space model (RSSM) [15, 29, 11, 30], and transformer-based architectures [41, 45]. These baselines represent widely adopted approaches in dynamics modeling and policy optimization. All models are given the same context during training and evaluation. Their training parameters are detailed in Sec. A.2.2. The relative autoregressive prediction errors e for these models are shown in Fig. 4. The tasks span manipulation scenarios as well as quadruped and humanoid locomotion tasks, allowing for a comprehensive evaluation of the models. In addition, we highlight the importance of the autoregressive training introduced in Sec. 3.2 by including both RWM trained with teacher-forcing (RWM-TF) and autoregressive training (RWM-AR), demonstrating the significant performance gains achieved by the latter.

The results highlight the superiority of RWM trained with autoregressive training (RWM-AR), which consistently achieves the lowest prediction errors across all environments. The performance gap between RWM-AR and the baselines is especially pronounced in complex and dynamic tasks, such as velocity tracking for legged robots, where accurate long-horizon predictions are critical for effective control. The comparison also reveals that RWM-AR significantly outperforms its teacher-forcing counterpart (RWM-TF), underscoring the importance of autoregressive training in mitigating compounding prediction errors over long rollouts. We additionally visualize the imagination rolled out by RWM-AR compared with the ground truth simulation in Fig. 1 and Fig. S9.

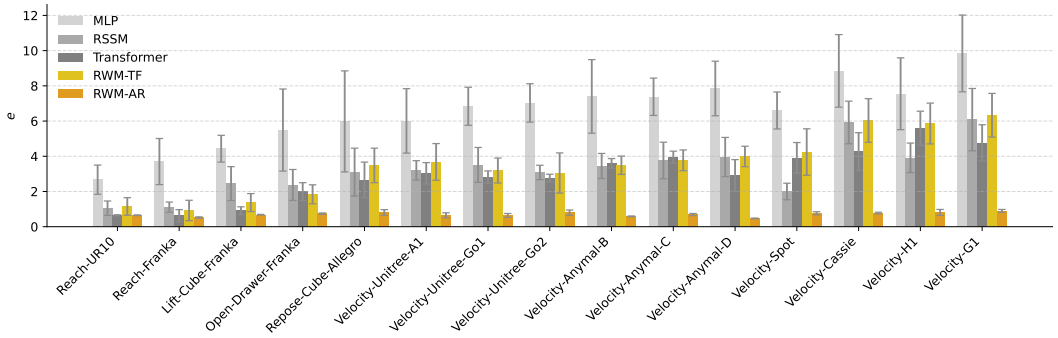


Figure 4: Autoregressive trajectory prediction errors across diverse robotic environments and network architectures. RWM trained with autoregressive training (RWM-AR) consistently outperforms baseline methods, including MLP, recurrent state-space model (RSSM), and transformer-based architectures. RWM-AR demonstrates superior generalization and robustness across tasks, from manipulation to locomotion. Autoregressive training (RWM-AR) reduces compounding errors over long rollouts, significantly improving performance compared to teacher-forcing training (RWM-TF).

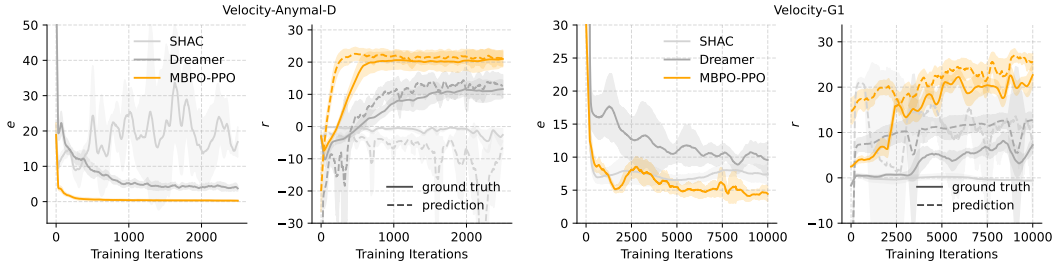


Figure 5: Model error and policy mean reward for the ANYmal D (left) and Unitree G1 (right) velocity tracking task with MBPO-PPO. The policy is trained using estimated rewards computed from predicted observations by RWM. Ground truth rewards, visualized with solid lines, are reported by the simulator for *evaluation* purposes only.

Note that the baselines are trained using teacher forcing as they are traditionally implemented. However, the proposed autoregressive training framework is architecture-agnostic and can also be applied to baseline models. When trained with autoregressive training, RSSM achieves a performance comparable to the proposed GRU-based architecture. Nevertheless, we opt for the GRU-based model due to its simplicity and computational efficiency. On the other hand, training transformer architectures with autoregressive training does not scale effectively, as the multi-step gradient propagation in autoregressive forecasting leads to GPU memory constraints, limiting their practicality for this approach. These results demonstrate that RWM, when combined with autoregressive training, achieves robust and generalizable performance across diverse robotic tasks.

4.4 Policy Learning and Hardware Transfer

Using MBPO-PPO, we train a goal-conditioned velocity tracking policy for ANYmal D and Unitree G1 leveraging RWM. The policy's observation and action spaces are detailed in Sec. A.1.1, and its architecture is described in Sec. A.2.3. Reward formulations are provided in Sec. A.1.2, while training parameters are summarized in Sec. A.3.2. We compare MBPO-PPO with two baselines: Short-Horizon Actor-Critic (SHAC) [38] and DreamerV3 [30]. SHAC employs a first-order gradient-based method that propagates gradients through the world model to optimize the policy. Dreamer integrates a latent-space dynamics model with an actor-critic framework, emphasizing sample efficiency and robustness in continuous control tasks.

Figure 5 illustrates the model error e during policy optimization. While MBPO-PPO demonstrates a significant reduction in model error over training, SHAC struggles with high and fluctuating model error throughout the process. Its reliance on first-order gradients for optimization is not

well-suited for discontinuous dynamics, such as those encountered in legged locomotion, where system behavior changes drastically due to varying contact patterns. The resulting inaccurate gradients lead to suboptimal policy updates, producing chaotic robot behaviors during training. These chaotic behaviors, in turn, generate low-quality training data for updating RWM, exacerbating model inaccuracies. Although Dreamer effectively leverages its latent-space dynamics model for policy optimization, its reliance on shorter planning horizons during training limits its ability to handle long-horizon dependencies, particularly in stochastic environments. As a result, Dreamer encounters moderate compounding errors during policy learning, which hinder its convergence to optimal behaviors.

On the right plot of rewards r, predicted rewards (dashed) from MBPO-PPO initially overshoot the ground truth (solid) due to the policy exploiting small inaccuracies in the model's optimistic estimates. As training progresses, predictions align more closely with ground truth, remaining accurate enough to guide effective learning. In contrast, SHAC fails to converge, producing unstable behaviors that degrade both policy and model quality. Dreamer demonstrates partial convergence, achieving higher rewards compared to SHAC but significantly lagging behind MBPO-PPO.

To evaluate the robustness of the learned policies, we deploy them on ANYmal D and Unitree G1 hardware in a zero-shot transfer setup. SHAC and Dreamer fail to produce a deployable policy due to its collapse during training. However, as shown in Fig. 1, the policy learned using MBPO-PPO demonstrates reliable and robust performance in tracking goal-conditioned velocity commands and maintaining stability under external disturbances, such as unexpected impacts and terrain conditions. The success of MBPO-PPO in hardware deployment is a direct result of the high-quality trajectory predictions generated by RWM, which enable accurate and effective policy optimization. Videos showcasing the robustness of the policies on hardware, including their responses to external disturbances, are available on our webpage. These results underline the effectiveness of RWM and MBPO-PPO in enabling robust and scalable policy deployment for real-world robotic systems.

5 Limitations

The policy learned with RWM and MBPO-PPO surpasses existing MBRL methods in both robustness and generalization. However, it still falls short of the performance achieved by well-tuned model-free RL methods trained on high-fidelity simulators. Model-free RL, being a more mature and extensively optimized paradigm, excels in settings where unlimited interaction with near-perfect simulators is possible. In contrast, the strengths of MBRL are more pronounced in scenarios where accurate or efficient simulation is infeasible, making it an indispensable tool for enabling intelligent agents to eventually learn and adapt in complex, real-world environments. To clarify the computational and performance aspects, we provide a comparison against a PPO-based method with a high-fidelity simulator in Table 1.

Table 1: Comparison with model-free method

Method	RWM pretraining	MBPO-PPO	PPO
state transitions	6M	–	250M
total training time	50 min	5 min	10 min
step inference time	–	1 ms	1 ms
real tracking reward	–	0.90 ± 0.04	0.90 ± 0.03

In this work, the world model is pre-trained using simulation data prior to policy optimization, reducing instability during training (see Sec. A.4.3). However, training from scratch remains challenging as policies can exploit model inaccuracies during exploration, leading to inefficiency and instability. In addition, the need for additional interaction with the environment to fine-tune the world model highlights areas for further refinement. Nevertheless, enabling safe and effective online learning directly on hardware remains challenging (see Sec. A.4.4). Current training in simulation avoids potential hardware damage, but incorporating safety constraints and robust uncertainty estimates will be critical for deploying RWM and MBPO-PPO in real-world, lifelong learning scenarios. These limitations underscore the trade-offs inherent in MBRL frameworks, balancing data efficiency, safety, and performance while addressing the complexities of real-world robotic systems.

6 Conclusion

In this work, we present RWM, a robust and scalable framework for learning world models tailored to complex robotic tasks. Leveraging a dual-autoregressive mechanism, RWM effectively addresses key challenges such as compounding errors, partial observability, and stochastic dynamics. By incorporating historical context and self-supervised training over long prediction horizons, RWM achieves superior accuracy and robustness without relying on domain-specific inductive biases, enabling generalization across diverse tasks. Through extensive experiments, we demonstrate that RWM consistently outperforms state-of-the-art approaches like RSSM and transformer-based architectures in autoregressive prediction accuracy across diverse robotic environments. Building on RWM, we propose MBPO-PPO, a policy optimization framework that leverages long world model rollout fidelity. Policies trained using MBPO-PPO demonstrate superior performance in simulation and transfer seamlessly to hardware, as evidenced by zero-shot deployment on the ANYmal D and Unitree G1 robots. This work advances the field of model-based reinforcement learning by providing a generalizable, efficient, and scalable framework for learning and deploying world models. The results highlight RWM 's potential to enable adaptive, robust, and high-performing robotic systems, setting a foundation for broader adoption of model-based approaches in real-world applications.

Acknowledgments and Disclosure of Funding

This research was supported by the ETH AI Center.

References

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

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

[3] Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE international conference on robotic computing (IRC)*, pages 590–595. IEEE, 2019.

[4] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[5] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[6] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[7] Chenhao Li, Marin Vlastelica, Sebastian Blaes, Jonas Frey, Felix Grimminger, and Georg Martius. Learning agile skills via adversarial imitation of rough partial demonstrations. In *Conference on Robot Learning*, pages 342–352. PMLR, 2023.

[8] Chenhao Li, Sebastian Blaes, Pavel Kolev, Marin Vlastelica, Jonas Frey, and Georg Martius. Versatile skill control via self-supervised adversarial imitation of unlabeled mixed motions. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 2944–2950. IEEE, 2023.

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

[10] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[11] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[12] 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.

[13] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[14] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[15] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[16] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[17] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[18] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

[19] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[20] Yunlong Song, Sangbae Kim, and Davide Scaramuzza. Learning quadruped locomotion using differentiable simulation. *arXiv preprint arXiv:2403.14864*, 2024.

[21] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

[22] Cansu Sancaktar, Sebastian Blaes, and Georg Martius. Curious exploration via structured world models yields zero-shot object manipulation. *Advances in Neural Information Processing Systems*, 35:24170–24183, 2022.

[23] Chenhao Li, Elijah Stanger-Jones, Steve Heim, and Sangbae Kim. Fld: Fourier latent dynamics for structured motion representation and learning. *arXiv preprint arXiv:2402.13820*, 2024.

[24] 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.

[25] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[26] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *arXiv preprint arXiv:2405.12399*, 2024.

[27] Suyoung Choi, Gwanghyeon Ji, Jeongsoo Park, Hyeongjun Kim, Juhyeok Mun, Jeong Hyun Lee, and Jemin Hwangbo. Learning quadrupedal locomotion on deformable terrain. *Science Robotics*, 8(74):eade2256, 2023.

[28] Jacob Levy, Tyler Westenbroek, and David Fridovich-Keil. Learning to walk from three minutes of real-world data with semi-structured dynamics models. *arXiv preprint arXiv:2410.09163*, 2024.

[29] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[30] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[31] Thomas Bi and Raffaello D'Andrea. Sample-efficient learning to solve a real-world labyrinth game using data-augmented model-based reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7455–7460. IEEE, 2024.

[32] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[33] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

[35] Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world. *arXiv preprint arXiv:2310.16029*, 2023.

[36] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[37] 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.

[38] Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. *arXiv preprint arXiv:2204.07137*, 2022.

[39] Ignat Georgiev, Krishnan Srinivasan, Jie Xu, Eric Heiden, and Animesh Garg. Adaptive horizon actor-critic for policy learning in contact-rich differentiable simulation. *arXiv preprint arXiv:2405.17784*, 2024.

[40] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[41] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[42] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[43] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[44] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 38–44. IEEE, 2016.

[45] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

A Technical Appendices and Supplementary Material

A.1 Task Representation

A.1.1 Observation and action spaces

The observation space for the ANYmal D and Unitree G1 world model is composed of base linear and angular velocities v, ω in the robot frame, measurement of the gravity vector in the robot frame g, joint positions q, velocities $\dot{q}$ and torques τ as in Table S2.

Table S2: World model observation space

Entry	Symbol	Dimensions	Entry	Symbol	Dimensions
ANYmal D			*Unitree G1*		
base linear velocity	v	0:3	base linear velocity	v	0:3
base angular velocity	ω	3:6	base angular velocity	ω	3:6
projected gravity	g	6:9	projected gravity	g	6:9
joint positions	q	9:21	joint positions	q	9:38
joint velocities	$\dot{q}$	21:33	joint velocities	$\dot{q}$	38:67
joint torques	τ	33:45	joint torques	τ	67:96

The privileged information is used to provide an additional learning objective that implicitly embeds critical information for accurate long-term predictions. The space is composed of knee and foot contacts as in Table S3.

Table S3: World model privileged information space

Entry	Symbol	Dimensions	Entry	Symbol	Dimensions
ANYmal D			*Unitree G1*		
knee contact	–	0:4	body contact	–	0:26
foot contact	–	4:8	foot height	–	26:28
			foot velocity	–	28:30

The action space is composed of joint position targets as in Table S4.

Table S4: Action space

Entry	Symbol	Dimensions	Entry	Symbol	Dimensions
ANYmal D			*Unitree G1*		
joint position targets	q^*	0:12	joint position targets	q^*	0:29

The observation space for the ANYmal velocity tracking policy is composed of base linear and angular velocities v, ω in the robot frame, measurement of the gravity vector in the robot frame g, velocity command c, joint positions q and velocities $\dot{q}$ as in Table S5.

A.1.2 Reward functions

The total reward is sum of the following terms with weights detailed in Table S6.

Linear velocity tracking x, y

$$r_{v_{xy}} = w_{v_{xy}} e^{-\|c_{xy} - v_{xy}\|_2^2 / \sigma_{v_{xy}}^2},$$

where $\sigma_{v_{xy}} = 0.25$ denotes a temperature factor, c_{xy} and v_{xy} denote the commanded and current base linear velocity.

Angular velocity tracking

Table S5: Policy observation space

Entry	Symbol	Dimensions	Entry	Symbol	Dimensions
ANYmal D			*Unitree G1*		
base linear velocity	v	0:3	base linear velocity	v	0:3
base angular velocity	ω	3:6	base angular velocity	ω	3:6
projected gravity	g	6:9	projected gravity	g	6:9
velocity command	c	9:12	velocity command	c	9:12
joint positions	q	12:24	joint positions	q	12:41
joint velocities	$\dot{q}$	24:36	joint velocities	$\dot{q}$	41:70

Table S6: Reward weights

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
ANYmal D				*Unitree G1*			
$w_{v_{xy}}$	1.0	w_{ω_z}	0.5	$w_{v_{xy}}$	1.0	w_{ω_z}	0.5
w_{v_z}	-2.0	$w_{\omega_{xy}}$	-0.05	w_{v_z}	-2.0	$w_{\omega_{xy}}$	-0.05
w_{q_τ}	$-2.5e^{-5}$	$w_{\ddot{q}}$	$-2.5e^{-7}$	w_{q_τ}	$-2.5e^{-5}$	$w_{\ddot{q}}$	$-2.5e^{-7}$
$w_{\dot{a}}$	-0.01	w_{f_a}	0.5	$w_{\dot{a}}$	-0.05	w_{f_a}	0.0
w_c	-1.0	w_g	-5.0	w_c	-1.0	w_g	-5.0
w_{f_c}	0.0	w_{q_d}	0.0	w_{f_c}	1.0	w_{q_d}	-1.0

$$r_{\omega_z} = w_{\omega_z} e^{-\|c_z - \omega_z\|_2^2 / \sigma_{\omega_z}^2},$$

where $\sigma_{\omega_z} = 0.25$ denotes a temperature factor, c_z and ω_z denote the commanded and current base angular velocity.

Linear velocity z

$$r_{v_z} = w_{v_z} \|v_z\|_2^2,$$

where v_z denotes the base vertical velocity.

Angular velocity x, y

$$r_{\omega_{xy}} = w_{\omega_{xy}} \|\omega_{xy}\|_2^2,$$

where ω_{xy} denotes the current base roll and pitch velocity.

Joint torque

$$r_{q_\tau} = w_{q_\tau} \|\tau\|_2^2,$$

where τ denotes the joint torques.

Joint acceleration

$$r_{\ddot{q}} = w_{\ddot{q}} \|\ddot{q}\|_2^2,$$

where $\ddot{q}$ denotes the joint acceleration.

Action rate

$$r_{\dot{a}} = w_{\dot{a}} \|a' - a\|_2^2,$$

where a' and a denote the previous and current actions.

Feet air time

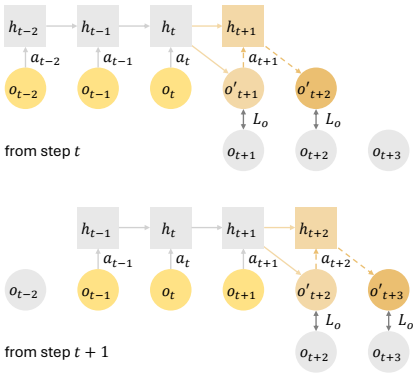


Figure S6: Dual-autoregressive mechanism employed in RWM. Inner autoregression updates GRU hidden states after each historical step within the context horizon, while outer autoregression feeds predicted observations from the forecast horizon back into the network. The dashed arrows denote the sequential autoregressive prediction steps, highlighting robustness to long-term dependencies and transitions.

$$r_{f_a} = w_{f_a} t_{f_a},$$

where t_{f_a} denotes the sum of the time for which the feet are in the air.

Undesired contacts

$$r_c = w_c c_u,$$

where c_u denotes the counts of the undesired contacts.

Flat orientation

$$r_g = w_g g_{xy}^2,$$

where g_{xy} denotes the xy-components of the projected gravity.

Foot clearance

$$r_{f_c} = w_{f_c} h_{f_c},$$

where h_{f_c} denotes the clearance height of the swing feet.

Joint deviation

$$r_{q_d} = w_{q_d} \left\| q - q_0 \right\|_1,$$

where q_0 denotes the default joint position.

A.2 Network Architecture

A.2.1 RWM

The robotic world model consists of a GRU base and MLP heads predicting the mean and standard deviation of the next observation and privileged information such as contacts, as detailed in Table S7. The training scheme is visualized in Fig. S6.

A.2.2 Baselines

The network architectures of the baselines are detailed in Table S8.

Table S7: RWM architecture

Component	Type	Hidden Shape	Activation
base	GRU	256, 256	–
heads	MLP	128	ReLU

Table S8: Baseline architecture

Network	Parameter	Value
MLP	hidden shape	256, 256
	activation	ReLU
RSSM	type	GRU
	hidden size	256
	layers	2
	latent dimension	64
	prior type	categorical
	categories	32
Transformer	type	decoder
	dimension	64
	heads	8
	layers	2
	context length	32
	positional encoding	sinusoidal

A.2.3 MBPO-PPO

The network architectures of the policy and the value function used in MBPO-PPO are detailed in Table S9. The training scheme is visualized in Fig. S7.

A.3 Training Parameters

The learning networks and algorithm are implemented in PyTorch 2.4.0 with CUDA 12.6 and trained on an NVIDIA RTX 4090 GPU.

A.3.1 RWM

The training information of RWM is summarized in Table S10.

A.3.2 MBPO-PPO

The training information of MBPO-PPO is summarized in Table S11.

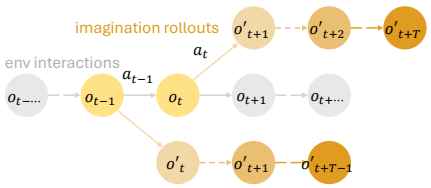


Figure S7: Model-Based Policy Optimization with learned world models. The framework combines real environment interactions with simulated rollouts for efficient policy optimization. Observation and action pairs from the environment are stored in a replay buffer and used to train the autoregressive world model. Imagination rollouts using the learned model predict future states over a horizon of T, providing trajectories for policy updates through reinforcement learning algorithms.

Table S9: Policy and value function architecture

Network	Type	Hidden Shape	Activation
policy	MLP	128, 128, 128	ELU
value function	MLP	128, 128, 128	ELU

Table S10: RWM training parameters

Parameter	Symbol	Value
step time seconds	Δt	0.02
max iterations	–	2500
learning rate	–	$1e^{-4}$
weight decay	–	$1e^{-5}$
batch size	–	1024
history horizon	M	32
forecast horizon	N	8
forecast decay	α	1.0
approximate training hours	–	1
number of seeds	–	5

A.4 Additional Experiments and Discussions

A.4.1 Dual-autoregressive Mechanism

The heatmap on the left in Fig. S8 shows the relative autoregressive prediction error e under different combinations of M and N. Models trained with a longer history horizon M consistently exhibit lower prediction errors, demonstrating the importance of providing sufficient historical context to capture the underlying dynamics. However, the influence of M plateaus beyond a certain point, indicating diminishing returns for very large history horizons. Forecast horizon N, on the other hand, plays a decisive role in improving long-term prediction accuracy. Increasing N during training leads to better performance in autoregressive rollouts, as it encourages the model to learn representations robust to compounding errors over extended prediction horizons. This improvement comes at the cost of increased training time, as shown in the heatmap on the right. Larger N values require sequential computation during training due to the autoregressive nature of the process, significantly lengthening the training duration.

Interestingly, when the forecast horizon $N = 1$ (teacher-forcing), training can be highly parallelized, resulting in minimal training time. However, this setting leads to poor autoregressive performance, as

Figure S8: Ablation study on the history horizon M and forecast horizon N in RWM. The heatmap on the left shows the relative autoregressive prediction error, with darker colors indicating higher errors. Models trained with larger history horizons M exhibit lower errors, although the improvements plateau beyond a certain point. Forecast horizon N has a significant impact, with longer horizons leading to better long-term prediction accuracy due to exposure to extended rollouts during training. The heatmap on the right illustrates training time, with darker colors representing longer durations. Increasing N significantly raises training time due to sequential computation, while shorter horizons (e.g., $N = 1$, teacher-forcing) enable faster training but result in poor prediction accuracy.

Table S11: MBPO-PPO training parameters

Parameter	Symbol	Value
imagination environments	–	4096
imagination steps per iteration	–	100
step time seconds	Δt	0.02
buffer size	$\lvert\mathcal{D}\rvert$	1000
max iterations	–	2500
learning rate	–	0.001
weight decay	–	0.0
learning epochs	–	5
mini-batches	–	4
KL divergence target	–	0.01
discount factor	γ	0.99
clip range	ϵ	0.2
entropy coefficient	–	0.005
number of seeds	–	5

the model lacks exposure to long-horizon prediction during training and fails to effectively handle compounding errors. From the results, an optimal trade-off emerges: moderate values of M and N balance prediction accuracy and training efficiency. For instance, a history horizon of $M = 32$ and forecast horizon of $N = 8$ achieve strong autoregressive performance with manageable training time. These settings ensure sufficient historical context while training the model for robust long-term predictions. Overall, the results highlight the critical interplay between history and forecast horizons in autoregressive training. While extending both M and N improves accuracy, practical considerations of computational cost necessitate careful tuning of these hyperparameters to achieve optimal performance.

A.4.2 Visualization of Imagination Rollouts

The imagination rollouts across various robotic environments compared with the ground-truth simulation is visualized in Fig. S9.

A.4.3 Collision Handling and Model Pretraining

In both phases of the pretraining and online fine-tuning of RWM, we terminate rollouts and reset the environment when ground contact by the base is detected, signaling a failure. We explicitly train RWM to predict such terminations in its privileged information prediction head. This enables the world model to learn transitions leading to unsafe situations. During policy optimization, MBPO-PPO treats these termination predictions as episode-ending events in imagination rollouts, affecting PPO's return computation and state values.

RWM is pretrained with simulation data induced by policies trained for similar tasks under varied dynamics. The policy is learned from scratch purely in imagination, with RWM fine-tuned using a *single*-environment online dataset. Pretraining is essential for two key reasons. First, the online dataset is extremely limited, as it is generated by only a *single* environment, akin to real-world constraints. Training the world model entirely from scratch on such data would lead to severe overfitting and long training times. Second, an immature policy would frequently cause the robot to fall, generating transitions with limited value. In cases of significant failure or domain shift, training the world model solely on these data would result in chaotic imagined rollouts, which in turn would produce poor policy updates. Pretraining stabilizes training and serves as a robust initialization for online fine-tuning, particularly in environments with challenging dynamics.

Importantly, RWM pretraining does not require data from optimal policies. Figure 3 demonstrate that RWM remains robust to domain shifts and injected noise. As an alternative, we warm up the model using data from a suboptimal policy, which significantly stabilizes training. Notably, this pretraining is only necessary for locomotion tasks due to the discontinuous dynamics and environment terminations. Our manipulation experiments do not require such pretraining.

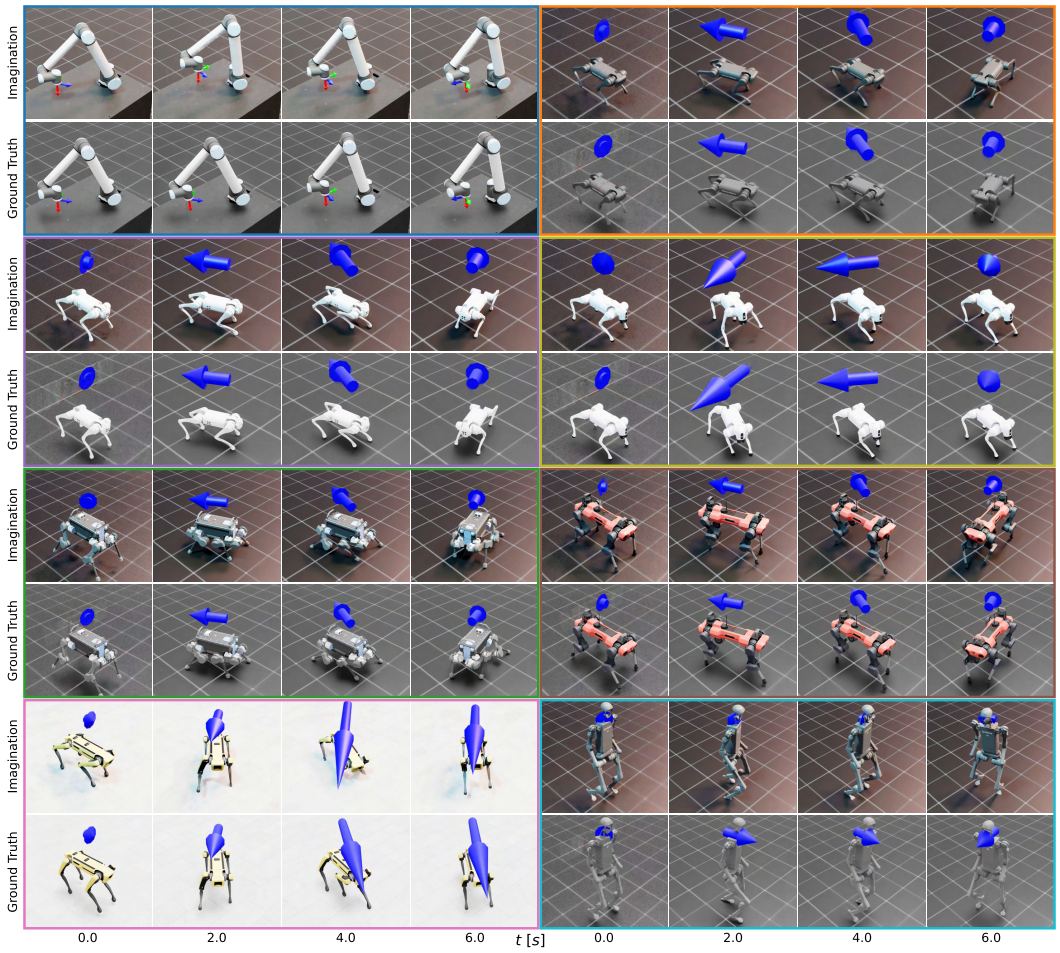


Figure S9: Autoregressive imagination of RWM and ground-truth simulation across diverse robotic systems. For each environment, the top row showcases the RWM autoregressively predicting future trajectories in imagination. The second row visualizes the ground truth evolution in simulation. The visualized coordinate and arrow markers denote the predicted and measured end-effector pose and base velocity, respectively.

A.4.4 Challenges in Real-World Online Learning

We acknowledge that the advantages of our approach would be further demonstrated by performing the policy training phase directly on real hardware. While this is a key long-term objective, several challenges currently prevent real-world deployment.

During online learning, the policy often exploits minor world model errors, leading to overly optimistic behaviors that result in collisions. In simulation, these failures serve as corrective signals, but in real hardware, they pose a risk to the robot. Our experiments show that such failures occur more than 20 times on average during online learning, which would be detrimental to real-world systems. Even if hardware collisions were acceptable, fully automating online learning would require a recovery policy capable of resetting the robot to an initial state—a particularly challenging requirement for large platforms like ANYmal D or Unitree G1. Additionally, privileged information used to fine-tune RWM (e.g., contact forces) must be either measured or estimated using onboard sensors, which may not always be available. To mitigate error exploitation, uncertainty-aware world models could be explored, but integrating such models into RWM would require additional architectural modifications. Due to these challenges, we approximate real-world constraints by using only a *single* simulation environment with domain shifts from pretraining environments. This setup reduces engineering effort while proving the feasibility of our approach. Our ongoing work specifically addresses these issues .

A.5 Ethics and Societal Impacts

This work does not involve human subjects or sensitive data. All experiments are conducted in simulation or on dedicated robotic hardware operated by the authors, with no use of third-party datasets. The research complies with the Code of Ethics of the venue. The proposed framework provides a robust and scalable method for learning world models tailored to complex robotic tasks. This can benefit domains such as healthcare, disaster response, and logistics, and reduce environmental and hardware costs associated with physical experimentation. Potential risks include misuse of the method in surveillance or autonomous enforcement systems, and the acceleration of automation in labor-sensitive sectors. While such uses are not intended or explored in this work, the authors acknowledge the dual-use potential of generalizable control methods. To mitigate safety risks, policy training occurs entirely in simulation, and deployment is limited to policies validated under domain shifts. Failure events are explicitly modeled and used to terminate unsafe rollouts. Online learning on hardware is deferred due to safety concerns and the absence of reliable recovery strategies. Future work will explore uncertainty-aware models and safer online adaptation.