DREAMSTEER: Latent World Models can steer VLA Policies during deployment without any finetuning

Hanchen Cui[1,2,*], **Sergio Arnaud**[1], **Arjun Majumdar**[1], **Daniel Dugas**[1], **Elie Aljalbout**[1], **Karthik Desingh**[2†], **Krishna Murthy Jatavallabhula**[1†], **Franziska Meier**[1†]

[1]Fundamental AI Research (FAIR), Meta [2]University of Minnesota Twin Cities

[*]Work done during an internship at Meta [†]Joint last authors

Abstract: Pretrained vision–language–action (VLA) policies show promising zero-shot generalization, but often fail under deployment-time distribution shift, leading to decreased robustness and inconsistent instruction following. While prior work commonly tackles this by finetuning on in-distribution data, it assumes demonstrations collected on tasks in the target environment. In this work, we propose **DREAMSTEER**, a deployment-time steering framework for pretrained VLAs **without any finetuning or parameter modifications**. The key insight in DREAMSTEER is to leverage a *latent world model* and a *value model* to *steer* pretrained VLA policies. During deployment, DREAMSTEER samples candidate action chunks from a VLA policy and predefined motion primitives, imagines their outcomes using an action-conditioned latent world model, and ranks the imagined trajectories with a language-conditioned value model. Across four real-world manipulation benchmarks with unseen objects, DREAMSTEER improves task success rate from 23.75% to 66.25% and instruction-following accuracy from 38.75% to 56.25% over the base VLA policy. Videos are available on the project website: https://dream-steer.github.io/.

Keywords: Latent World Models, Deployment-time Policy Steering

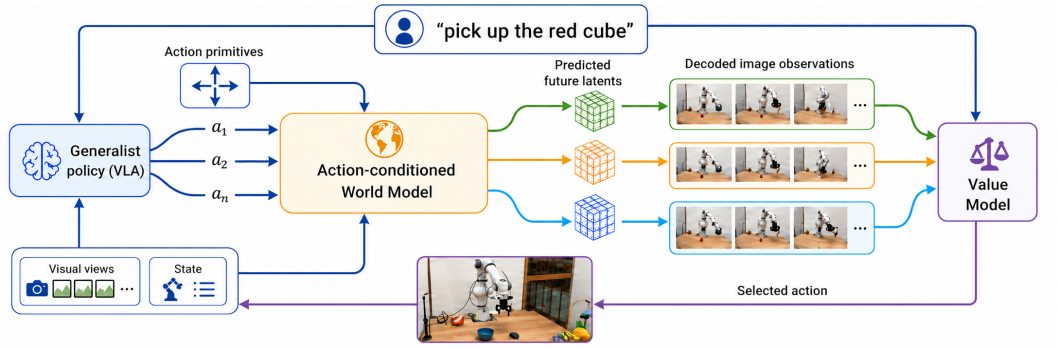

Figure 1: **DREAMSTEER: deployment-time policy steering.** A frozen VLA policy proposes candidate action chunks, which are augmented with a small set of predefined Cartesian action primitives. A latent world model predicts future observations, and a language-conditioned value model ranks the resulting trajectories before execution. **All models are trained frozen, and no target-environment data is used during model training**.

1 Introduction

Recent progress in large-scale multimodal pretraining has enabled VLA policies [1, 2, 3, 4, 5] that integrate perception, language understanding, and control within a unified framework. Trained on large-scale robot datasets, these models show promising transfer to new tasks, objects, and environments. However, pretrained VLAs remain brittle under deployment-time distribution shift, often failing on unseen objects or producing behaviors that violate language instructions. While finetuning can improve performance, it requires additional target-domain data and modifies the pretrained

policy, which may not be desired or feasible. This raises a key question: **how can we improve pretrained VLA policies at deployment time without target-domain data finetuning?**

In the context of large language models, a similar challenge is tackled by *test-time steering* approaches that sample candidate outputs and rerank them using external judges or scoring models to improve reliability without retraining [6, 7]. Pretrained VLAs provide a natural analogue: diffusion-based policies [1, 2] and autoregressive VLAs [3, 4, 5] both produce stochastic action samples that can be steered at inference time. However, unlike text outputs, the quality of an action chunk cannot be evaluated directly before execution in the physical world.

World models [8, 9, 10] provide a natural mechanism for evaluating candidate actions before execution. Conceptually, they enable *thinking before acting*. Given the current observation and a candidate action chunk, an action-conditioned world model can predict the resulting future observations. These imagined rollouts can then be ranked by a language-conditioned value model, enabling look-ahead action evaluation at deployment time.

In this paper, we introduce **DREAMSTEER**, a deployment-time steering framework for pretrained VLA policies using latent world models with zero finetuning. Given a language instruction and current observation, DREAMSTEER samples multiple candidate action chunks from a pretrained VLA policy and augments them with predefined motion primitives. An action-conditioned latent world model predicts imagined future observations for each candidate, and a language-conditioned value model ranks the resulting rollouts. DREAMSTEER then executes the highest-scoring action chunk in the real environment. **All components remain fixed during deployment, and no target-environment data is used for adaptation**. As illustrated in Fig. 2, single-sample policy execution may fail under deployment-time distribution shift, while evaluating multiple imagined futures before execution allows DREAMSTEER to select more reliable actions.

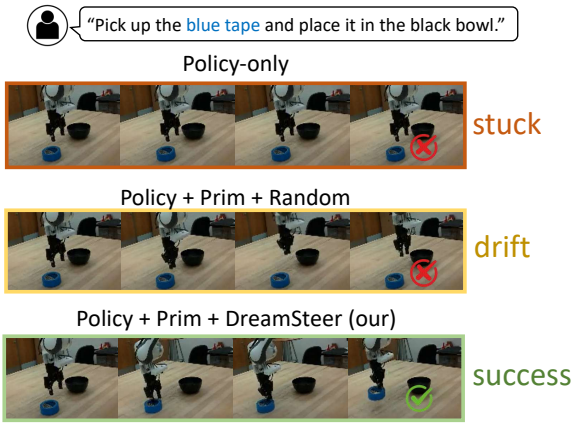

Figure 2: **DREAMSTEER improves deployment-time robustness** of a pretrained, frozen VLA policy.

Our contributions are summarized as follows:

1. We introduce **DREAMSTEER**, a deployment-time steering framework that evaluates candidate VLA action chunks through imagined latent-world-model rollouts before execution.
2. DREAMSTEER composes a pretrained VLA policy, a generalized latent world model, and a language-conditioned value model at deployment time, without finetuning any component on target-environment data.
3. We demonstrate DREAMSTEER improves OOD object manipulation success from 23.75% to 66.25%, and instruction-following accuracy from 38.75% to 56.25% over the π_0 VLA policy.

2 Related Work

Generalist visuomotor policies. Large-scale multimodal pretraining has enabled vision–language–action (VLA) models to emerge as generalist manipulation policies. Recent models such as π_0 [1], GR00T [2], OpenVLA [3], RT-2 [4], and FAST [5], trained on large-scale datasets including DROID [11], exhibit a broad range of manipulation behaviors. Despite their broad capabilities, pretrained VLAs often remain brittle under deployment-time distribution shift [12, 13, 14], producing semantically incorrect behaviors or failing on unseen objects and environments. Instead of

collecting new data and finetuning the policy, DREAMSTEER treats the pretrained VLA as a fixed generative action prior and steers sampled action chunks at deployment time.

Robot World Models. World models predict future states conditioned on observations and actions, enabling agents to evaluate action consequences before execution [8, 9, 10]. In robotics, world models support planning, policy evaluation, and data generation, with approaches differing primarily in prediction space and action conditioning. Pixel-space models such as Cosmos [8] and DreamGen [15] generate visually rich futures but remain computationally expensive for online steering. In contrast, latent models such as DINO-WM [10] and VT-WM [16] directly predict future states in compact latent space conditioned on robot actions, making them better suited for inference-time action evaluation. DREAMSTEER uses an action-conditioned latent world model for efficient rollout and candidate evaluation. Candidate action chunks are rolled out in latent space, then decoded into image observations. This avoids expensive pixel-space video generation while preserving an image-based interface for evaluation.

Policy Steering and Model-Based Policy Refinement. This class of methods improves pretrained policies using action proposals, predictive models, and external evaluators. Most policy steering approaches follow this decomposition: a policy proposes actions, a world model rolls out their consequences, and a value or reward model evaluates the result. Prior methods make different choices within this design space. V-GPS [12] reranks single-step actions by a value model, without explicit rollout prediction. FOREWARN [13], GPC [14], VLA-Reasoner [17], and LaDi-WM [18] rely on task-specific adaptation or policy refinement. In contrast, DREAMSTEER is a generalizable policy steering framework that performs fully deployment-time steering using a pretrained latent world model and a language-conditioned value model, without finetuning any component on target-task data. Table 1 summarizes the main differences.

Table 1: **Comparison to policy steering and model-based refinement methods.** *Training-free composition* means fully plug-and-play, each component does not rely on other components and any target-task data. "–" denotes not applicable.

Method	Action chunks	World model (space | adaptation)	Generalized evaluator	Zero-shot steering	Training-free composition
V-GPS [12]	✗	–	✓	✓	✗
FOREWARN [13]	✓	Latent | task-finetuned	✗	✗	✗
GPC [14]	✓	Pixel | task-finetuned	✗	✗	✗
VLA-Reasoner [17]	✓	Pixel | generalized	✗	✗	✗
LaDi-WM [18]	✓	Latent | task-finetuned	–	✗	✗
DREAMSTEER (ours)	✓	Latent | generalized	✓	✓	✓

3 Methodology

We consider language-instructed robot deployment in an unseen environment. At each timestep t, the agent receives an observation $o_t \in \mathcal{O}$ and acts according to a natural language instruction ℓ. The action space is denoted by $\mathcal{A}$. Explicit reward signals and new demonstrations in target environment are not available. Our goal is to steer a pretrained generative VLA policy at test time, without any modifications to its parameters. The policy $\pi_\theta(a_{t:t+H-1} \mid o_t, \ell)$ defines a stochastic distribution over action chunks $a_{t:t+H-1} \in \mathcal{A}^H$ of horizon H. DREAMSTEER addresses the problem of deciding which sampled action chunk to execute by evaluating candidate futures before acting.

3.1 World Model Training

The world model is a central component of DREAMSTEER. To support zero-shot deployment-time steering, it must be generalizable, efficient, and robust. The world model, shown in Fig. 3, is an action-conditioned latent dynamics model trained on heterogeneous robot and human interaction data. Its design follows three principles: multi-embodiment training for broad generalization, latent-space prediction for efficient rollout, and spatio-temporal factorization for scalable trajectory evaluation. Additional details are provided in the supplementary material Section 7.

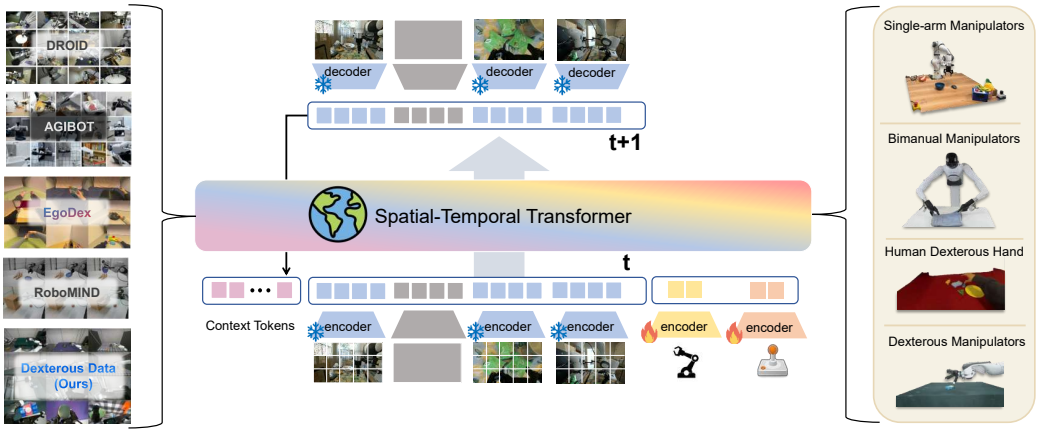

Figure 3: **Heterogeneous action-conditioned world model.** The world model is trained on diverse robot and human interaction datasets spanning multiple embodiments.

Multi-Embodiment Training. To learn a generalizable world model, we train on heterogeneous datasets [11, 19, 20, 21], spanning single-arm manipulators, bimanual robots, dexterous hands, and human demonstrations. Visual observations are mapped into a shared latent space, while embodiment-specific action and state inputs are encoded into latent tokens using learned tokenizers. A shared spatio-temporal transformer is trained across all embodiments. The key insight is that all these embodiments interact with a shared physical world, allowing the backbone to learn transferable object interaction dynamics despite embodiment gaps. Object interaction is both the most important and most difficult component of action-conditioned prediction: robot motion is largely specified by the action input, and static scene information is already present in the observation context, while object interaction dynamics must be learned from diverse interaction data. Missing modalities are masked during training to support partially observed multi-view and multi-embodiment data.

Latent-Space Dynamics. Rather than predicting pixels directly, $\mathcal{W}_\phi$ predicts future states in the latent space of a frozen DINOv2 visual encoder. DINOv2 latent representations capture rich semantic and world knowledge while preserving task-relevant structure for downstream evaluation. Pixel-space video generation is computationally expensive and often requires iterative denoising, making it poorly suited for rollout-based steering over multiple candidates. Latent-space dynamics instead enable efficient autoregressive prediction, while decoded latent rollouts can still be evaluated by an image-based value model. In practice, latent rollout is substantially faster than video diffusion models [22]: on the same NVIDIA RTX 4090 setup, generating three 320×192 frames at horizon $H=10$ takes 23.12 s with a video diffusion model, compared to 0.59 s for our latent world model.

Spatio-Temporal Factorization. To support fast rollout over multiple candidate action chunks, we use a factorized spatio-temporal transformer [23]. The model consists of N spatio-temporal blocks operating on visual latent tokens and action/state tokens. Each block applies self-attention to visual and control tokens together with cross-attention for action-conditioned prediction. As illustrated in Fig. 4, attention is factorized into spatial attention within each timestep and causal temporal attention across timesteps for each patch. This factorization reduces attention complexity in the rollout horizon H from quadratic to linear, enabling efficient evaluation of many candidate action chunks during deployment.

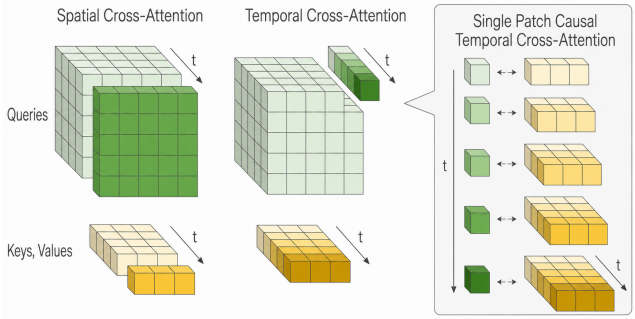

Figure 4: **Spatio-temporal cross attention.** Spatial cross-attention attends within each timestep, while temporal cross-attention applies causal attention across timesteps for each patch.

3.2 DREAMSTEER

DREAMSTEER, shown in Fig. 1, steers a pretrained VLA policy by ranking imagined outcomes of candidate action chunks before execution. At time t, DREAMSTEER constructs a finite candidate set

$$\mathcal{C}_t = \mathcal{C}_t^{\mathrm{VLA}} \cup \mathcal{C}^{\mathrm{prim}},$$

where $\mathcal{C}_t^{\mathrm{VLA}}$ contains stochastic samples from the VLA policy and $\mathcal{C}^{\mathrm{prim}}$ contains predefined Cartesian action primitives. Each candidate action chunk is rolled out through the world model, decoded into image observations, and scored by the value model under instruction ℓ. DreamSteer then executes the candidate with the highest predicted value score:

$$k^\star = \mathrm{argmax}_{k\in\{1,\dots,|\mathcal{C}_t|\}} V_\psi\Big(o_t, \mathcal{W}_\phi(o_t, a_{t:t+H-1}^{(k)}), \ell\Big), \quad a_{t:t+H-1}^\star = a_{t:t+H-1}^{(k^\star)}. \tag{1}$$

Here, π_θ, $\mathcal{W}_\phi$, and V_ψ denote the policy, world model, and value model, respectively, and all remain fixed during deployment.

Stochastic Policy Action Proposals. The policy π_θ is instantiated as a pretrained VLA model. We write the per-step observation as

$$o_t = \{I_t^{\mathrm{wrist}}, I_t^{\mathrm{ext}}, s_t^{\mathrm{robot}}, s_t^{\mathrm{gripper}}\},$$

where I denotes visual observations and s denotes proprioceptive state. Conditioned on o_t and ℓ, the policy samples K_{VLA} temporally extended action chunks:

$$a_{t:t+H-1}^{(k)} \sim \pi_\theta(\cdot \mid o_t, \ell), \qquad k = 1, \dots, K_{\mathrm{VLA}}.$$

These samples expose diverse behaviors already present in the generative action distribution of the pretrained policy.

We augment these policy samples with a small primitive library $\mathcal{C}^{\mathrm{prim}}$. The primitive library consists of fixed short-horizon Cartesian motions, including end-effector translations along left/right, up/down, and forward/backward directions, as well as gripper open and close commands. Each primitive is represented as an action chunk of length H, making it temporally aligned with VLA-generated chunks. The primitives are not intended to solve manipulation tasks on their own. Instead, they provide simple alternatives that improve candidate coverage when the VLA samples alone fail to make progress, for example when the end effector is near the target object but the sampled actions do not move into a grasp-ready pose.

World Model Predictive Visual Dynamics. The world model $\mathcal{W}_\phi$ predicts future observations conditioned on the current observation and a candidate action chunk:

$$\hat{o}_{t+1:t+H}^{(k)} = \mathcal{W}_\phi(o_t, a_{t:t+H-1}^{(k)}).$$

Internally, the model operates in the latent space of a frozen DINOv2 visual encoder [24]. Suppressing modality-specific tokenization for clarity, the rollout can be written as

$$z_t = E(o_t), \quad \hat{z}_{t+1:t+H}^{(k)} = F_\phi\Big(z_t, a_{t:t+H-1}^{(k)}\Big), \quad \hat{o}_{t+1:t+H}^{(k)} = D\Big(\hat{z}_{t+1:t+H}^{(k)}\Big). \tag{2}$$

where E is the frozen observation encoder, F_ϕ is the learned latent dynamics model, and D is the frozen decoder that reconstructs visual observations from predicted latents.

Value Function Scoring and Steering Decision. The value function V_ψ is instantiated as a pretrained Vision-Language-Action-Critic (VLAC) model [25] based on the InternVL2-2B architecture [26]. VLAC provides an instruction-conditioned progress signal over pairs of visual observations. Given an imagined rollout for candidate k, we compute a trajectory-level score by summing pairwise progress estimates along the decoded rollout:

$$S^{(k)} = \sum_{j=1}^{H} \mathrm{VLAC}\Big(\hat{o}_{t+j-1}^{(k)}, \hat{o}_{t+j}^{(k)}, \ell\Big), \qquad \hat{o}_t^{(k)} = o_t. \tag{3}$$

This score estimates how much the candidate action chunk advances the task specified by ℓ. DreamSteer does not require the value model to produce calibrated absolute rewards; it only needs the scores to rank candidates generated at the same timestep. The system then executes the action chunk with the highest score, steering the pretrained VLA policy toward predicted outcomes that better satisfy the language instruction.

4 Experiments

We evaluate DreamSteer on a real-world robot setup to study three questions: whether latent world-model rollouts preserve task-relevant information for value-based ranking, whether DreamSteer improves robustness under deployment-time distribution shift, and how action diversity and value-guided steering contribute to performance. Across all experiments, DreamSteer is **used zero-shot**, i.e. none of the policy, world model, or value model components are finetuned on data from the target evaluation environment.

4.1 Real Robot Setup

Our experiments follow the DROID setup using a 7-DoF Franka Panda manipulator with a Robotiq two-finger gripper. Compared to the original DROID configuration, we replace the exterior cameras with Intel RealSense L515 cameras and mount the robot on a static workbench. **All evaluations are conducted in a different lab environment**, introducing a different robot instance, low-level controller, camera configuration, background, lighting, and object distribution relative to DreamSteer development. These changes occur simultaneously, resulting in substantial distribution shift.

4.2 Evaluating the World Model

We first evaluate whether latent world model rollouts preserve the relative value trends needed for trajectory ranking. DreamSteer does not require calibrated reward prediction or photo-realistic video generation, only imagined rollouts that preserve relative ranking consistency across candidate action chunks. The rollout results are shown in Fig. 5. More visualizations are shown in Fig. 10 in supplementary material. We compare value-model scores on short ground-truth video clips and corresponding imagined rollouts generated from action chunks with horizon $H{=}10$ under the same language instructions. As shown in Fig. 5, the two sets of scores are positively correlated, with Pearson $r = 0.66$ (95% CI $[0.46, 0.79]$, $p < 10^{-6}$) and Spearman $\rho = 0.69$ ($p < 10^{-7}$). This consistency is sufficient for rollout ranking in DreamSteer. We also measure latent prediction error as rollout length increases. As shown in Fig. 6, latent MSE increases gradually over the evaluated horizon, suggesting

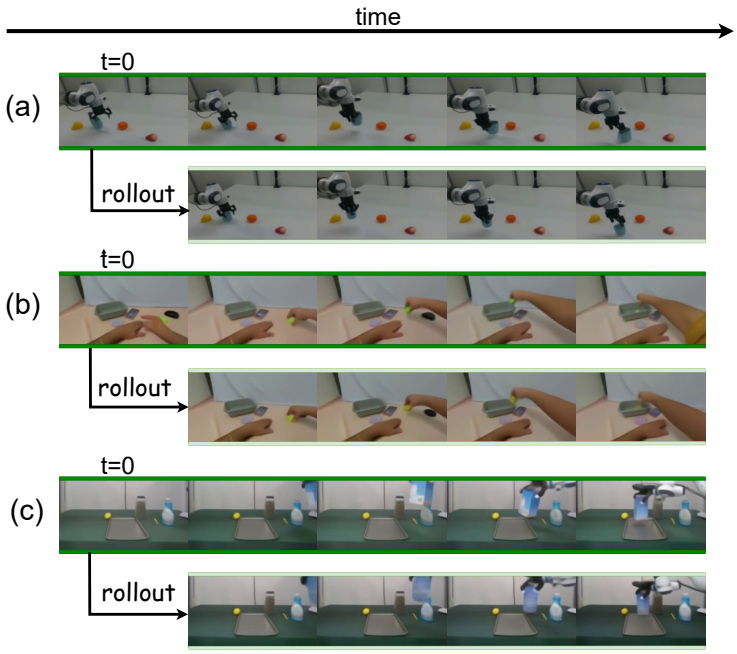

Figure 5: **World model rollouts across multiple embodiments.** Only a single camera view is visualized for clarity.

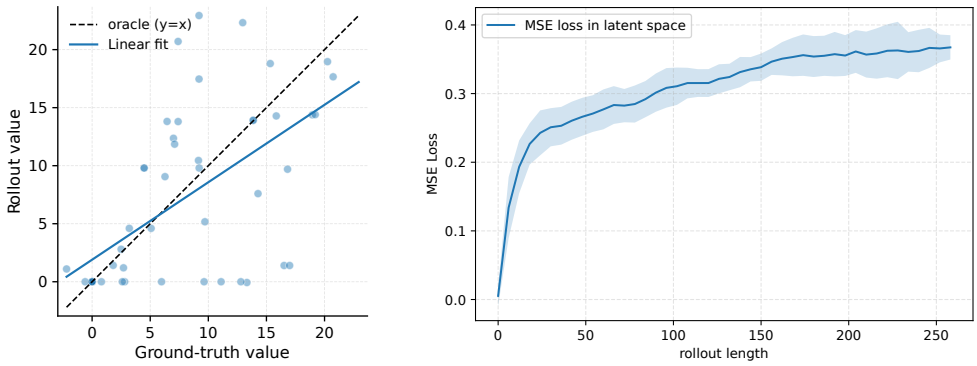

Figure 6: **Evaluative consistency of imagined rollouts.** (a) Value-model scores on ground-truth and imagined trajectories. (b) Latent prediction error versus rollout length; shaded area denotes ± 1 standard deviation.

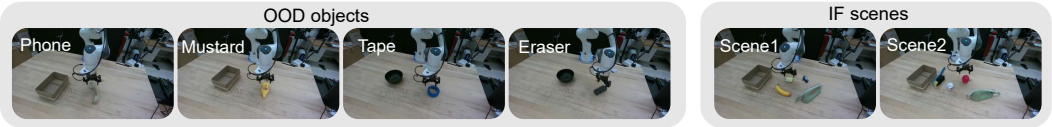

Figure 7: **Real-world evaluation tasks.** Left: OOD object manipulation tasks involving previously unseen objects. Right: instruction-following (IF) scenes containing multiple objects and distractors.

that the model remains sufficiently stable for repeated short-horizon rollout evaluation. All the data for quantitative evaluation are sampled from the unseen RoboArena dataset [27].

4.3 Evaluating DREAMSTEER

We evaluate DREAMSTEER as a complete system with two primary goals: **(1) improving robustness to out-of-distribution (OOD) objects**, and **(2) improving instruction-following (IF) accuracy under language-specified constraints**. The OOD objects and the scenes used to evaluate IF accuracy are shown in Fig. 7.

Across all experiments, we use a π_0 checkpoint pretrained on DROID as the base policy. Unless otherwise specified, we use $K{=}5$ sampled action chunks and horizon $H{=}10$. We use chunk-level steering because imagined observations for $H{=}1$ are often too similar for reliable ranking, while longer action chunks produce more distinguishable future states. Steering is applied once every five control steps for efficiency. Each task is evaluated over 20 trials with randomized object poses and positions. The steering step requires approximately 13 s of computation. This cost scales with the number of candidate action chunks: generating, rolling out, and evaluating a single candidate requires approximately 1 s in our implementation, and we use 13 candidates during deployment. Rollout and evaluation are fully parallelizable and could substantially reduce steering latency, as our current implementation prioritizes feasibility over inference optimization.

OOD Object Generalization. We evaluate DREAMSTEER on OOD object manipulation using a standard pick-and-place task, where the robot must place a specified object into a designated receptacle. The task structure is fixed, but the target objects differ from the policy training distribution in appearance, geometry, and material properties, making them challenging for the pretrained policy. We compare DREAMSTEER against several ablations: π_0 denotes single-sample policy execution; π_0 + DREAMSTEER ranks multiple π_0 action chunks; primitives + DREAMSTEER ranks only predefined Cartesian primitives; and π_0 + primitives + random selects uniformly at random from the same candidate set as the full method.

As shown in Table 2, DREAMSTEER improves OOD manipulation success rate from 23.75% to 66.25%. Ranking multiple π_0 samples improves over single-sample execution, suggesting that useful action chunks exist in the policy distribution but are not reliably selected. Primitives alone achieve zero success, while random selection over the same candidate set also fails, indicating that

both candidate diversity and value-guided ranking are necessary. The full method performs best by combining diverse policy proposals, primitive-based local exploration, and rollout evaluation.

We also study the effect of proposal count K on the whiteboard eraser task. Increasing π_0 samples from $K{=}3$ to $K{=}5$ improves success, while increasing further to $K{=}7$ provides little additional benefit but increases rollout cost. We therefore use $K{=}5$ in all experiments.

Table 2: **OOD object performance.** Success rates over 20 trials per object. The last column reports the aggregate success rate over all 80 trials with 95% Wilson confidence intervals.

Method	Phone	Mustard	Tape	Eraser	Average	95% CI
π_0	4/20	3/20	6/20	6/20	23.75	[15.84, 34.07]
π_0 + DREAMSTEER	7/20	6/20	11/20	10/20	42.50	[32.26, 53.43]
π_0 + primitives + random	0/20	0/20	0/20	0/20	0.00	[0.00, 4.58]
primitives + DREAMSTEER	0/20	0/20	0/20	0/20	0.00	[0.00, 4.58]
π_0 + primitives + DREAMSTEER	**12/20**	**11/20**	**16/20**	**14/20**	**66.25**	**[55.39, 75.65]**

Instruction Following Accuracy. We next evaluate whether DREAMSTEER improves instruction following in complex scenes. We report instruction-following accuracy. A trial is counted as correct if the robot makes consistent contact with, or attempts to grasp, the language-specified object. This metric isolates semantic target selection from low-level execution difficulty, allowing us to evaluate whether test-time steering improves adherence to language constraints. As shown in Table 3, DREAMSTEER improves instruction-following accuracy from 38.75% to 56.25%, a gain of 17.5 percentage points over π_0. These results suggest that imagined rollout scoring helps reject action proposals whose predicted outcomes do not align with the semantic intent of the instruction.

Table 3: **Instruction following performance.** Accuracy over 20 trials per target object. The last column reports the aggregate accuracy over all 80 trials with 95% Wilson confidence intervals.

Method	Sponge	Banana	Pencil	Apple	Average	95% CI
π_0	8/20	9/20	6/20	8/20	38.75	[28.78, 49.73]
π_0 + primitives + DREAMSTEER	**14/20**	**13/20**	**9/20**	**9/20**	**56.25**	**[45.34, 66.57]**

5 Discussion and Limitations

5.1 Why DREAMSTEER Works

Trajectory Ranking Over One-pass Generation. DREAMSTEER improves deployment-time robustness by converting single-sample action generation into a ranking problem over imagined futures. A pretrained VLA policy provides a distribution over plausible action chunks, but under distribution shift a single sample may fail or violate the language instruction. DREAMSTEER instead samples multiple candidate chunks, predicts their outcomes with a world model, ranks the imagined rollouts, and executes the highest-scoring action chunk. The performance improvements suggest that meaningful trajectories often exist in the policy distribution but are not reliably selected during single-sample execution. Selecting among candidate trajectories is easier than generating the correct trajectory in a single pass.

Complementary Generalization Across Components. Another way to understand DREAMSTEER is that the policy, world model, and value model are trained under different data regimes and therefore generalize differently. Generalization often depends on both the scale and diversity of training data. A VLA policy is typically trained on successful robot demonstrations collected from limited embodiments and environments. In contrast, a world model only needs to predict future observations and can leverage broader interaction data, including successful demonstrations, failures, random play, and cross-embodiment trajectories. The value model can generalize more broadly still, since image-based progress estimation can exploit large-scale visual-language pretraining together with diverse action-free human data. DREAMSTEER exploits this complementarity at deployment time: even when an object is out-of-distribution for the policy, it may remain in-distribution for the world model and value model, allowing successful steering.

5.2 Limitations

Failure Mode Analysis. DREAMSTEER can fail for two reasons. First, steering is limited by candidate coverage: when neither the sampled policy actions nor the predefined primitives make meaningful progress toward the instruction, the system cannot recover a successful behavior. Integrating latent-space planning or trajectory optimization could help iteratively refine candidate actions using world-model feedback. Second, steering relies on accurate trajectory ranking. We observe cases where the value model assigns higher scores to suboptimal candidates, often because different action outcomes appear visually similar from the single available camera view. In such cases, the value model cannot reliably distinguish between candidate trajectories, leading to incorrect steering decisions. Future work could improve ranking reliability through multi-view observations.

Latency. World-model rollout and value-model evaluation introduce additional inference overhead. For horizon $H{=}10$, policy inference takes 0.08 s, world-model rollout takes 0.59 s, and value-model evaluation takes 0.37 s on an NVIDIA RTX 4090 GPU. The current implementation prioritizes feasibility over efficiency. Both rollout and scoring could be accelerated further through parallelized rollouts, KV caching, faster attention kernels, and other runtime enhancements.

6 Conclusion

We introduced **DREAMSTEER**, a deployment-time steering framework that improves pretrained VLA policies using latent world-model rollout evaluation, **without training or finetuning any component on target-environment data**. DREAMSTEER combines a pretrained VLA policy, an action-conditioned latent world model, and a language-conditioned value model to rank candidate action chunks before execution. Across real-robot evaluations, DREAMSTEER improves OOD object manipulation success from 23.75% to 66.25% and instruction-following accuracy from 38.75% to 56.25% over the base π_0 policy. Future work could improve both sides of this decomposition: richer candidate proposal mechanisms to increase action coverage, more efficient and robust world model rollout, and multi-view value aggregation.

Acknowledgments

The authors would like to thank Changhyun Choi for providing the robot platform and Mingen Li for assistance with its setup. We also thank Christopher Agia for his helpful discussions and valuable insights. Finally, we thank Adam Imdieke for his hardware support on gripper adapter.

References

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π0: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*.

[2] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[4] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[5] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[6] Z. Kang, X. Zhao, and D. Song. Scalable best-of-n selection for large language models via self-certainty. *arXiv preprint arXiv:2502.18581*, 2025.

[7] R. Pi, H. Bai, Q. Chen, X. S. Wang, J. Shan, X. Liu, and M. Cao. Mr. judge: Multimodal reasoner as a judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20192–20216, 2025.

[8] N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[9] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[10] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning 2024. *URL https://arxiv. org/abs/2411*, 4983.

[11] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[12] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.

[13] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. *arXiv preprint arXiv:2502.01828*, 2025.

[14] H. Qi, H. Yin, Y. Du, and H. Yang. Strengthening generative robot policies through predictive world modeling. *arXiv e-prints*, pages arXiv–2502, 2025.

[15] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjorck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin, et al. Dreamgen: Unlocking generalization in robot learning through neural trajectories. *arXiv e-prints*, pages arXiv–2505, 2025.

[16] C. Higuera, S. Arnaud, B. Boots, M. Mukadam, F. R. Hogan, and F. Meier. Visuo-tactile world models. *arXiv preprint arXiv:2602.06001*, 2026.

[17] W. Guo, G. Lu, H. Deng, Z. Wu, Y. Tang, and Z. Wang. Vla-reasoner: Empowering vision-language-action models with reasoning via online monte carlo tree search. *arXiv preprint arXiv:2509.22643*, 2025.

[18] Y. Huang, J. Zhang, S. Zou, X. Liu, R. Hu, and K. Xu. Ladi-wm: A latent diffusion-based world model for predictive manipulation. *arXiv preprint arXiv:2505.11528*, 2025.

[19] Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Hu, X. Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[20] R. Hoque, P. Huang, D. J. Yoon, M. Sivapurapu, and J. Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

[21] K. Wu, C. Hou, J. Liu, Z. Che, X. Ju, Z. Yang, M. Li, Y. Zhao, Z. Xu, G. Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

[22] Y. Guo, L. X. Shi, J. Chen, and C. Finn. Ctrl-world: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*, 2025.

[23] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, and H. Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

[24] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[25] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu, S. Lin, and J. Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

[26] Z. Chen, W. Wang, Y. Cao, Y. Liu, Z. Gao, E. Cui, J. Zhu, S. Ye, H. Tian, Z. Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

[27] P. Atreya, K. Pertsch, T. Lee, M. J. Kim, A. Jain, A. Kuramshin, C. Eppner, C. Neary, E. Hu, F. Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025.

Supplementary Materials

7 World Model Design and Training

The world model is an action-conditioned latent predictor optimized for efficient and robust rollout, rather than for photorealistic video generation. It is trained on heterogeneous, multi-embodiment data.

7.1 Input Modalities and Latent Encoding

The world model operates on a set of latent representations derived from multiple input modalities, consisting of visual observations, robot state, and action. All inputs are aligned temporally and encoded into a unified latent space before being processed by the spatio-temporal transformer.

Visual observations RGB observations are encoded independently at each timestep using a pre-trained and frozen DINOv2 encoder. Each image is mapped to a grid of latent patch embeddings,

$$x_t \in \mathbb{R}^{H \times W \times D_x},$$

where $H \times W$ denotes the spatial resolution of the latent grid and D_x is the per-patch embedding dimension. After projection, visual latents x_t are denoted as z_t. When an associated decoder is available, it is used only for visualization of predicted futures and not for training.

Action and control signal inputs Robot actions and robot states are represented as per-timestep control tokens. Robot actions are expressed as end-effector delta motions in Cartesian space. These deltas are computed from observed state transitions rather than directly using low-level controller commands, allowing the model to remain agnostic to embodiment-specific control interfaces. Concretely, we treat each action/state component as a separate input key and encode it using a designated tokenizer, implemented as a lightweight embodiment-specific MLP, into latent tokens,

$$c_{t,i} = f_i(a_{t,i}) \in \mathbb{R}^{A_i \times D_c},$$

where $a_{t,i}$ denotes the i-th control component (action and, when available, state) at timestep t, and f_i is the corresponding component-specific tokenizer. A_i denotes the number of tokens produced by component i. The per-component tokens are then concatenated along the token dimension to form the control token sequence

$$c_t = \left[c_{t,1}, \ldots, c_{t,K}\right] \in \mathbb{R}^{A \times D_c}, \quad A = \sum_{i=1}^{K} A_i,$$

with K components provided by the embodiment.

Temporal alignment and batching Visual and control tokens are aligned at the timestep level. Given a sequence of length T, the model receives visual latents

$$x \in \mathbb{R}^{B \times T \times S \times D_x}$$

and corresponding control tokens

$$c \in \mathbb{R}^{B \times T \times A \times D_c},$$

where S is the total number of visual tokens per timestep, aggregated across available views, and A is the number of control tokens per timestep, which may vary across embodiments. These representations are inputs to the spatio-temporal transformer, which predicts future visual latents conditioned on past observations and the provided action/state sequence.

Projection to model dimension Before entering the transformer, both visual and control embeddings are linearly projected to a shared model dimension D. This allows the subsequent attention layers to operate in a unified embedding space while preserving the distinct structural roles of visual grid tokens and control tokens.

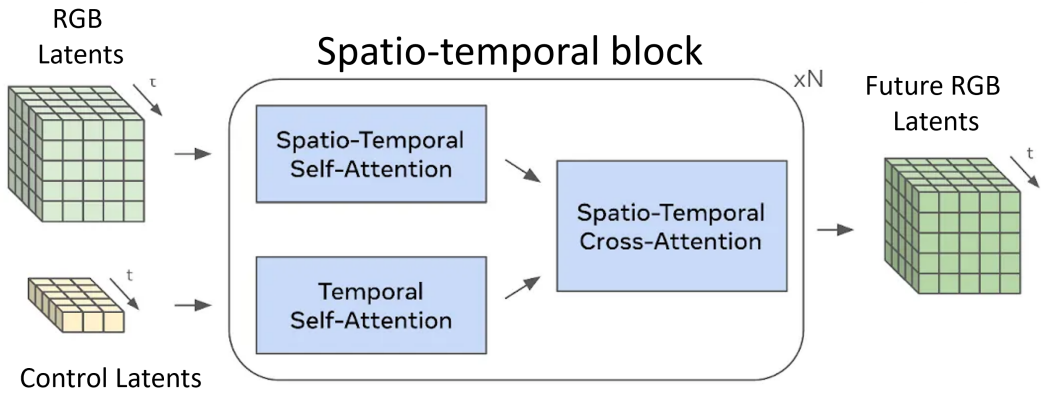

Figure 8: **Spatio-temporal world model architecture.**

7.2 Spatio-Temporal Transformer Architecture

Given the encoded visual and control tokens, the world model predicts future visual latents using a stack of spatio-temporal transformer layers. The transformer operates on two token streams: a grid of visual tokens and a set of per-timestep control tokens, and updates the visual stream through action-conditioned cross-attention.

Spatio-temporal transformer layers The transformer consists of N repeated layers, each composed of three sequential blocks, as shown in Fig. 8: (i) spatio-temporal self-attention over visual tokens, (ii) temporal self-attention over control tokens, and (iii) spatio-temporal cross-attention from visual tokens to control tokens. Spatio-temporal self-attention is factorized into spatial attention applied independently within each timestep and causal temporal attention applied across timesteps at each spatial (or control) location. This factorization avoids full attention over all space-time tokens and improves computational efficiency for long-horizon rollouts. Action conditioning is introduced through cross-attention, where visual tokens act as queries and control tokens serve as keys and values in a time-aligned manner, which is shown in Fig. 4.

7.3 Train with Multi-Embodiment Data

The world model is trained on data collected from multiple robot embodiments with heterogeneous sensory configurations and control interfaces. Differences across embodiments are handled at the input tokenization level, while all subsequent spatio-temporal dynamics parameters are fully shared.

Training datasets The world model is trained on a mixture of multi-embodiment manipulation datasets spanning both robot and human demonstrations. Our training corpus includes: (i) the DROID dataset collected on Franka platforms (76k trajectories, $\sim$350 hours), (ii) Franka subsets from RoboMIND, (iii) the EgoDex dataset consisting of human dexterous hand interactions, (iv) the AgiBot dual-arm manipulation dataset, and (v) an in-house teleoperation dataset containing approximately 1.2k trajectories collected on Franka arms equipped with dexterous hands.

In total, the combined dataset comprises on the order of 10^5 trajectories and several hundred hours of interaction data. The data spans single-arm and dual-arm robots, parallel grippers and dexterous hands, as well as human hand demonstrations, covering diverse viewpoints and control interfaces. This heterogeneity enables the world model to learn embodiment-agnostic interaction dynamics while remaining compatible with varied sensory and action configurations.

Multi-view visual observations Each timestep may include up to four RGB observations, consisting of two external views and two wrist-mounted views. Let

$$\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}, \quad |\mathcal{V}| \leq 4$$

denote the set of available viewpoints at a given timestep. Each view $v \in \mathcal{V}$ is independently encoded by a pretrained visual tokenizer into a grid of latent tokens. Since not all viewpoints are available

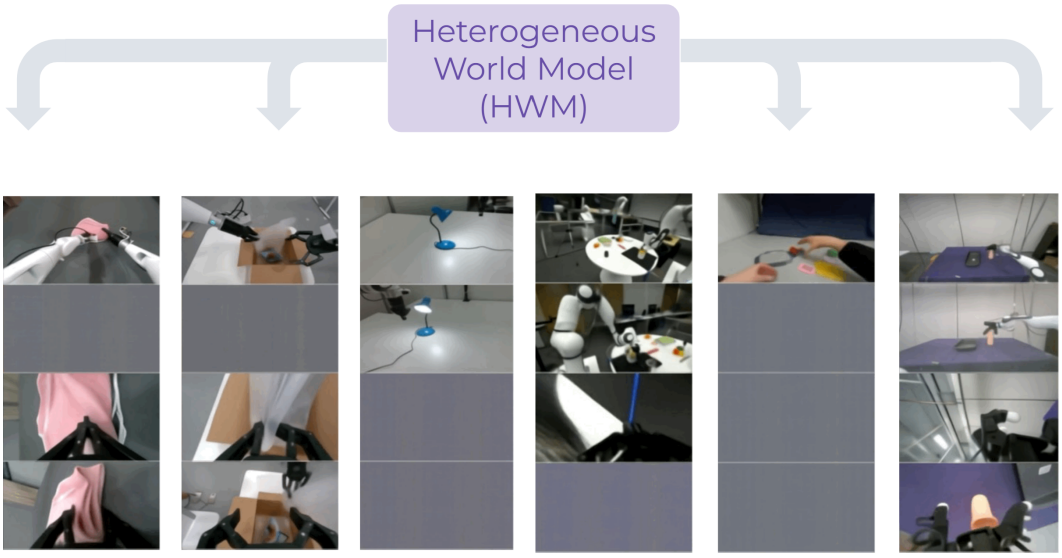

Figure 9: **Mask strategy in multi-view data.**

for every timestep or embodiment, we employ token-level masking to handle missing observations, shown in Fig. 9. For each view v, a binary mask

$$m_t^{(v)} \in \{0, 1\}$$

is expanded to all corresponding latent tokens and applied prior to the dynamics model. Masked tokens are zeroed out and excluded from loss computation, enabling the model to operate on partially observed multi-view inputs without introducing padded or hallucinated observations.

Shared dynamics across embodiments After tokenization and positional embedding, implemented via rotary position embeddings (RoPE), both visual tokens x_t and control tokens $c_t^{(e)}$ are projected to a shared model dimension and processed by a spatio-temporal transformer. All transformer parameters are shared across embodiments; the only embodiment-specific components are the input tokenizers for actions and states. As a result, the model learns a unified latent dynamics representation while remaining compatible with heterogeneous sensory layouts and control interfaces.

7.4 Autoregressive Rollout and Training Objective

Problem formulation Let $\mathbf{o}_{1:T}$ denote a sequence of observations and $\mathbf{a}_{1:T}$ the corresponding action/state tokens. After modality-specific tokenization (Sec. 7.1), observations are represented as latent tokens

$$\mathbf{z}_t \in \mathbb{R}^{S \times D},$$

where S is the number of visual tokens (aggregated across views) and D is the shared model dimension. Actions and proprioceptive states are encoded as control tokens

$$\mathbf{c}_t \in \mathbb{R}^{A \times D}.$$

The world model learns a dynamics function

$$f_\theta : (\mathbf{z}_{1:t}, \mathbf{c}_{1:t}) \rightarrow \hat{\mathbf{z}}_{t+1},$$

implemented by the spatio-temporal transformer described in Sec. 7.2.

Sliding-window autoregressive rollout At both training and inference time, future observations are predicted autoregressively. Given an initial context of length T_c, predictions are generated sequentially:

$$\hat{\mathbf{z}}_{t+1} = f_\theta(\mathbf{z}_{t-T_c+1:t},\ \mathbf{c}_{t-T_c+1:t}),$$

where only the most recent T_c steps are retained to bound memory and computation. Predicted tokens are appended to the context and used to predict subsequent steps:

$$\hat{\mathbf{z}}_{t+k} = f_\theta(\hat{\mathbf{z}}_{t+k-T_c:t+k-1},\ \mathbf{c}_{t+k-T_c:t+k-1}).$$

This sliding-window mechanism enables long-horizon rollout while maintaining fixed computational cost per step.

Teacher-forcing objective For one-step prediction, the model is trained using teacher forcing, where ground-truth observations are provided as input context. The loss is computed as mean squared error (MSE) in latent space:

$$\mathcal{L}_{\text{1-step}} = \frac{1}{T-1}\sum_{t=1}^{T-1} \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2^2 .$$

When modality masks are present (e.g., missing views), the loss is computed only over valid tokens:

$$\mathcal{L}_{\text{1-step}} = \frac{\sum m_{t,s}\, \|\hat{\mathbf{z}}_{t+1,s} - \mathbf{z}_{t+1,s}\|_2^2}{\sum m_{t,s}},$$

where $m_{t,s} \in \{0, 1\}$ denotes token validity.

Open-loop sampling objective To improve long-horizon stability, we additionally train the model using open-loop rollout. Starting from a short ground-truth context (typically one frame), the model generates predictions autoregressively for a rollout horizon H_{rollout}:

$$\hat{\mathbf{z}}_{2:H_{\text{rollout}}+1} = \text{Rollout}_\theta(\mathbf{z}_1,\ \mathbf{c}_{1:H_{\text{rollout}}}) .$$

A final forward pass is then performed using the generated trajectory as context, and predictions are supervised against ground truth:

$$\mathcal{L}_{\text{n-step}} = \frac{1}{H_{\text{rollout}}}\sum_{t=1}^{H_{\text{rollout}}} \|\hat{\mathbf{z}}_{t+1} - \mathbf{z}_{t+1}\|_2^2 .$$

This objective encourages the model to remain stable under its own predictions and mitigates exposure bias.

Combined training objective During training, we alternate between teacher-forcing and sampling losses using a scheduled curriculum. At each optimization step, only one objective is active:

$$\mathcal{L} = \begin{cases} \mathcal{L}_{\text{1-step}}, & \text{teacher-forcing step} \\ \mathcal{L}_{\text{n-step}}, & \text{sampling step.} \end{cases}$$

Both objectives are computed as mean squared error (MSE) in the latent space of the frozen visual encoder:

$$\mathcal{L} = \mathbb{E}\big[\|\hat{\mathbf{z}} - \mathbf{z}\|_2^2\big].$$

No pixel-space reconstruction loss is used. Supervising predictions in latent space significantly reduces computational cost while preserving task-relevant dynamics required for rollout evaluation.

Training details and hyperparameters We train the world model using distributed data-parallel training on 384 NVIDIA H100 GPUs for approximately 2-3 days. Training follows the objectives described in Sec. 7.4, with frozen visual tokenizers and jointly optimized action/state tokenizers. All hyperparameters are listed in Table 4.

Inference At deployment time, the world model operates purely autoregressively. Given past observations and candidate action sequences, future latent trajectories are rolled out and used for downstream evaluation without access to ground-truth future observations.

Hyperparameter	Value
World Model Architecture	
Model dimension D	1536
Transformer layers	8
Attention heads	24
Visual latent dim D_x	1024
Action/state latent dim	16
Context and Rollout	
Training context length T_c	16
Rollout horizon H_{rollout}	4
Optimization	
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	1×10^{-2}
Gradient clipping	1.0
Batch size	384
LR schedule	Cosine decay
Loss	
Latent loss type	MSE (L2)
Teacher forcing	Yes
Sampling loss	Yes
Loss scheduling	Alternating curriculum

Table 4: Training hyperparameters for the spatio-temporal world model.

7.5 World Model Rollout Visualization

We provide qualitative visualizations of autoregressive world model rollouts across multiple embodiments, viewpoints, and manipulation scenarios. In all visualizations, the model predicts future visual latents autoregressively conditioned on past observations and the provided action sequence. The rollout visualizations are shown in Fig. 10.

Unless otherwise specified, rollouts are initialized from a single ground-truth observation frame, after which predictions are generated fully open-loop. We visualize decoded RGB observations obtained from the frozen visual tokenizer decoder for interpretability. Ground-truth and predicted frames are shown side-by-side for comparison.

Beyond low-level physical motion prediction, we examine whether the world model captures higher-level regularities of object functionality and human-designed interactions. As shown in Fig. 11, we consider imagined rollouts in which a robot interacts with a light switch. Conditioned on the toggling action, the model predicts a corresponding change in the environmental state, such as the light turning on or off. This behavior extends beyond immediate contact dynamics and reflects learned associations between object affordances and their functional effects. Such results suggest that the world model internalizes commonsense interaction patterns present in human environments, enabling it to anticipate functional outcomes of actions in addition to physical motion.

8 Value Model and Trajectory Scoring

Model choice To evaluate predicted rollouts during deployment-time steering, we adopt an off-the-shelf vision–language value model, Vision-Language-Action-Critic (VLAC), without any additional finetuning. VLAC is pretrained to estimate task progress between pairs of observations conditioned on a language instruction, producing a scalar progress score that reflects relative advancement toward task completion.

Rollout scoring formulation Given the current observation o_t and a candidate action sequence $a_{t:t+H-1}$, the world model predicts a rollout of future observations

$$\hat{o}_{t+1:t+H}.$$

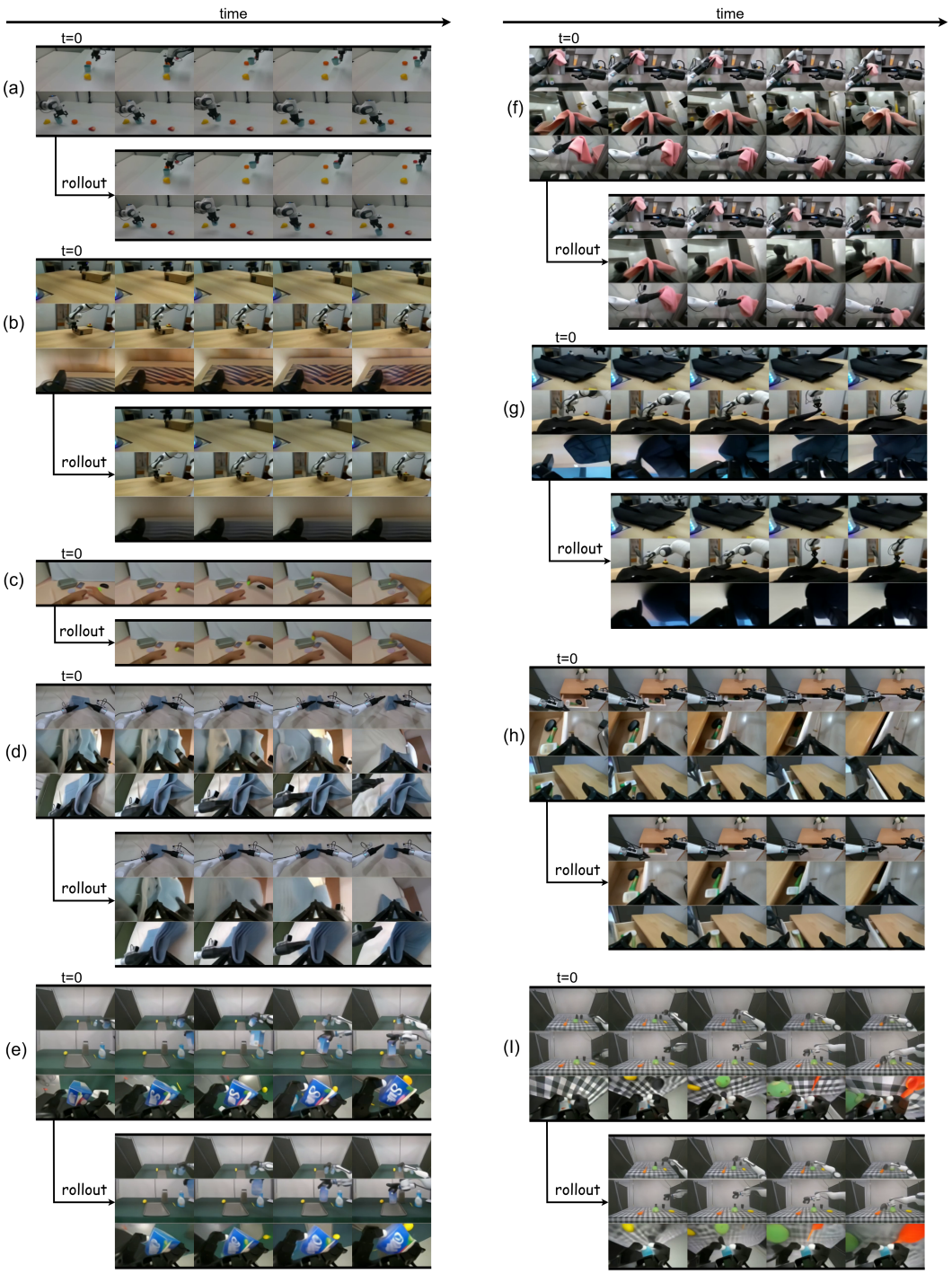

Figure 10: **World model rollout results.**

For a candidate rollout, we compute pairwise progress estimates between consecutive observations:

$$s_{t+j} = \mathrm{VLAC}(\hat{o}_{t+j-1}, \hat{o}_{t+j}, l_{\mathrm{task}}), \qquad \hat{o}_t = o_t,$$

where l_{task} denotes the language instruction.

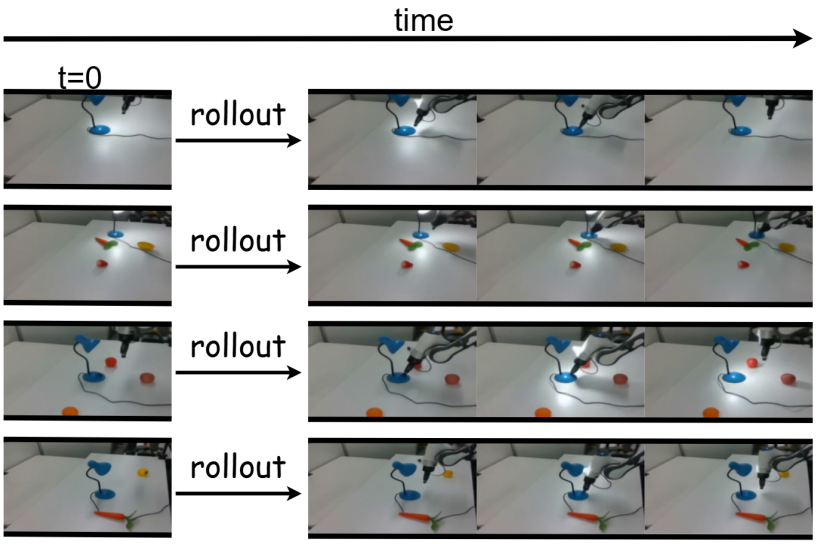

Figure 11: **Functional interaction prediction.** When the robot toggles the light switch in imagination, the world model predicts corresponding changes in scene illumination.

Trajectory aggregation The trajectory-level score is obtained by summing progress estimates along the rollout:

$$S = \sum_{j=1}^{H} \mathrm{VLAC}(\hat{o}_{t+j-1}, \hat{o}_{t+j}, l_{\mathrm{task}}) \,.$$

This score estimates how much the candidate action sequence advances the task specified by the language instruction. During deployment-time steering, multiple candidate action sequences are evaluated and ranked according to their trajectory scores, and the highest-scoring candidate is selected for execution. The value model operates on a single left-view RGB observation.

9 Action Candidate Construction

The pretrained π_0 policy outputs fixed-length action chunks in joint space with horizon $T = 10$:

$$\mathbf{q}_{t:t+T-1}.$$

We convert these joint actions into end-effector Cartesian delta actions using forward kinematics (FK):

$$\Delta \mathbf{p}_t = \mathrm{FK}(\mathbf{q}_{t+1}) - \mathrm{FK}(\mathbf{q}_t),$$

resulting in Cartesian action chunks of the same length $T = 10$.

In addition to policy-generated actions, we construct a set of hand-designed Cartesian action primitives, including directional motions (up, down, forward, backward, left, right) and gripper commands (open, close).

Each primitive defines a fixed Cartesian displacement (or gripper command), which is evenly distributed across the $T = 10$ steps to form a temporally aligned action chunk.

10 Hardware Setup and Evaluation Tasks

A visualization of the robot platform and camera configuration is shown in Fig. 12.

We evaluate real-world performance across two task families designed to assess generalization and language grounding under deployment conditions:

- **Out-of-distribution (OOD) manipulation.** The robot is instructed to manipulate novel objects not seen during training. Language instructions are:

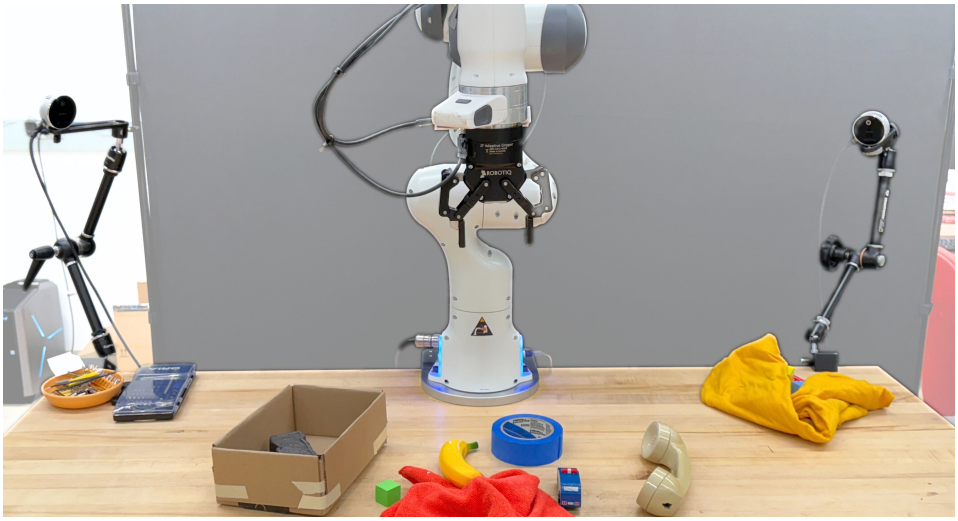

Figure 12: **Robot platform.**

  - "Pick up the {phone} and place it into the brown box."
  - "Pick up the {mustard} and place it into the brown box."
  - "Pick up the {whiteboard eraser} and place it into the black bowl."
  - "Pick up the {blue tape} and place it into the black bowl."

- **Instruction following with distractors.** The robot must identify the correct target object in the presence of visually similar distractors. Instructions and distractors are:
  - "Pick up the {sponge} and place it into the black bowl.". The distractors are banana, police car toy, and pencil case.
  - "Pick up the {banana} and place it into the black bowl." The distractors are sponge, police car toy, and pencil case.
  - "Pick up the {pencil case} and place it into the brown box." The distractors are brush, can, and apple.
  - "Pick up the {apple} and place it into the brown box." The distractors are brush, can, and pencil case.