DREAMSTEER: Latent World Models Can Steer VLA Policies During Deployment

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 generalist policies, such as vision–language–action (VLA) models, promise impressive zero-shot generalization in robot manipulation. However, their real-world performance tapers quickly under distribution shift, leading to decreased robustness and inconsistent instruction following abilities. To address these challenges, we propose DREAMSTEER, a deploy-time steering framework to enhance pretrained VLAs without the need for finetuning on demonstration data collected in the target distribution. The key insight in DREAMSTEER is to leverage a *latent world model* and a *general-purpose value function* to *steer* pretrained VLA policies. During deployment, DREAMSTEER generates diverse action candidates, sourced from the VLA policy and a set of predefined motion primitives, and *imagines* the outcome of each of these action sequences by rolling them out within the latent world model. By evaluating these predicted trajectories with the *value model*, DREAMSTEER identifies and executes the highest-scoring action, resulting in better instruction following, and weeding out task-irrelevant behaviors. Across four real-world manipulation benchmarks of unseen objects, DreamSteer improves task success rates by +42.5 percentage points (from 23.75% to 66.25%) and increases instruction following accuracy by +17.5 percentage points (from 38.75% to 56.25%) compared to the base VLA. These results suggest that latent world models can steer VLA policies during deployment, and present an effective pathway to improve the reliability of generalist robot policies, when finetuning may not be desired or feasible.**

I. INTRODUCTION

Recent progress in large-scale multimodal pretraining has resulted in generalist VLA policies [5, 4, 13, 28, 17] that integrate perception, language understanding, and control within a unified framework. Trained on diverse robotic datasets, these models promise strong zero-shot generalization, transfer to novel environments and tasks without additional fine-tuning. Despite their impressive visuomotor abilities, pretrained VLA policies exhibit two critical limitations under distribution shift: (a) they often fail to generalize robustly to objects unseen (or underrepresented) during training, (b) they struggle to reliably follow high-level language instructions, frequently violating task-specific constraints or semantic intent. A canonical solution to these problems is to fine-tune these models on task-specific data, which can improve performance on target tasks, objects, and environments. However, finetuning is not always desirable, or even feasible. It often causes the model to overfit to aspects of the finetuning distribution, and needs carefully curated mitigation recipes. This raises a central question: **how can we improve performance on unseen objects and instruction-following accuracy without sacrificing generalization?**

Test-time steering has proven effective in large language models [11, 18], where multiple candidate outputs are sampled and ranked at inference time to improve reliability without retraining. Pretrained VLA policies exhibit similar stochasticity, producing diverse action chunks for the same observation and instruction. However, unlike language generation, the quality of an action proposal cannot be evaluated without execution.

World models [1, 2, 27] offer a principled way to bridge this gap by enabling the prediction of action outcomes through imagination rather than execution. By learning the visual dynamics of the environment, world models allow a robot to simulate the future visual state of candidate action sequences conditioned on the current observation. This enables test-time look-ahead reasoning, where candidate actions can be pre-evaluated based on their imagined outcomes using an external, task-dependent scoring or value function before physical execution. Unlike task-specific policies trained only on successful demonstrations, world models can be potentially learned from diverse data that includes both successful, failed trajectories, random play trajectories, and even cross-embodiment data sources, leading to more faithful representations of environment dynamics. Moreover, world models are inherently task-agnostic and reusable across different goals, instructions, and object configurations. As a result, they naturally complement pretrained VLA policies by supporting test-time action evaluation and selection.

In this paper, we present **DreamSteer**, a test-time framework that leverages world models to steer pretrained generalist policies without additional policy training, shown in Fig. 3. Given a language instruction, a pretrained VLA policy first generates multiple candidate action chunks [26]. All candidate sequences, along with a set of predefined action primitives, are then rolled out within a learned world model to predict future observations corresponding to its potential execution. To evaluate these imagined trajectories, we employ a vision–language-model (VLM) [19, 9] as a language-conditioned value function, which scores each rollout based on its alignment with the instruction. The action chunk corresponding to the highest

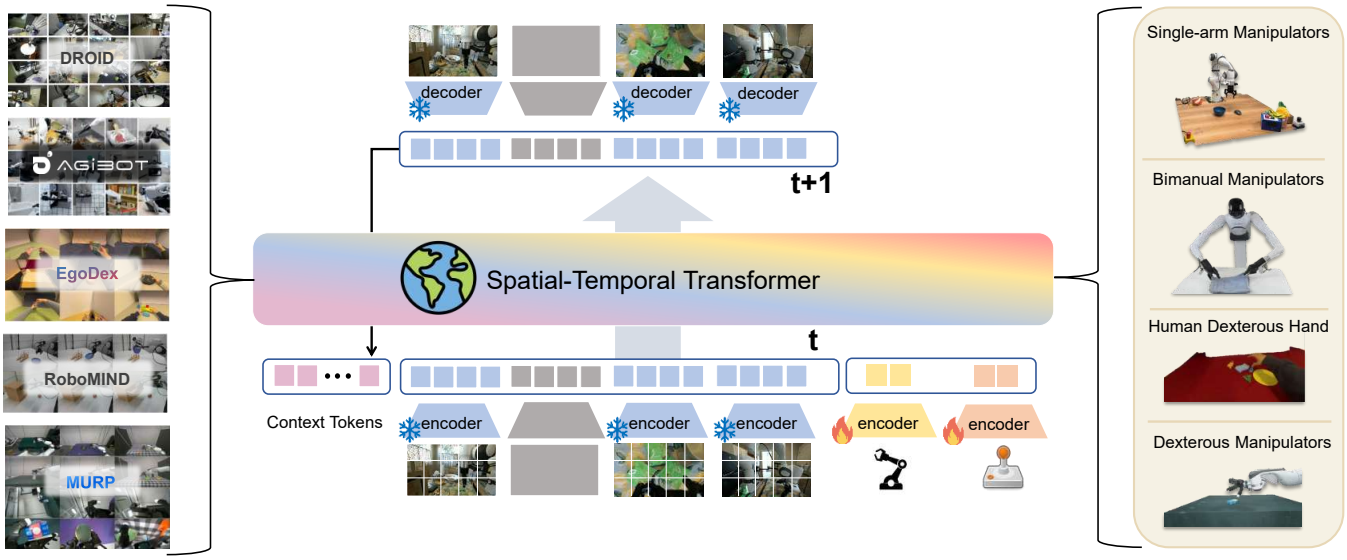

Fig. 1: **Heterogeneous action-conditioned world model.** The world model is trained on diverse datasets [12, 6, 8, 21] spanning multiple embodiments, including single-arm manipulators, bimanual robots, dexterous hands, and human video data. RGB observations are encoded into a shared latent space using a frozen visual encoder, while proprioception and actions from different embodiments are mapped via learned encoders. To provide temporal context, we append context tokens that summarize recent interaction history to the latent representations.

scoring visual rollout is selected and executed in the real environment. By combining the stochastic diversity of pretrained VLA policies with the predictive capabilities of world models, DreamSteer improves test-time decision-making while preserving the generalization strengths of large-scale pretraining.

Our contributions are summarized as follows:

1) We propose **DreamSteer**, a general-purpose test-time steering framework that integrates a pretrained VLA policy, a task-agnostic world model, and a language-conditioned value function, enabling reliable evaluation and selection of action proposals via imagined rollouts.
2) We introduce a cross-embodiment latent world model capable of predicting consistent future states across diverse robotic platforms.
3) We demonstrate that DreamSteer improves generalization to unseen objects, increasing the success rate from 23.75% to 66.25%, and boosts instruction-following accuracy by 17.5% compared to the base VLA, while preserving the broad generalization capabilities of pretrained generalist policies.

II. Related Work

Vision–Language–Action Models Large-scale multimodal pretraining has enabled VLA models to emerge as generalist policies for robotic manipulation. Trained on extensive datasets such as DROID [12], which provide diverse demonstrations across tasks and scenes under a consistent embodiment, recent models including π_0 [5] and GR00T [4] have demonstrated strong zero-shot transfer to novel environments. By integrating visual perception, language conditioning, and action generation within a unified generative framework, these models can produce a wide range of manipulation behaviors without task-specific fine-tuning.

Despite this progress, pretrained VLA models exhibit notable limitations in practice. Their performance often degrades when encountering out-of-distribution objects that differ in appearance, geometry, or material properties from the training data, indicating limited robustness to object-level distribution shifts. In addition, although VLAs generate smooth and plausible behaviors, they frequently fail to reliably align action selection with high-level language instructions, especially in cluttered scenes or under competing visual cues. While fine-tuning on task-specific data can partially mitigate these issues, it typically narrows the policy's generalization by biasing it toward specific objects or tasks. Addressing these limitations without compromising the generality of pretrained VLA models remains an open research problem.

Robotic World Models World models [1, 2, 27] aim to capture environment dynamics by predicting future states, enabling agents to reason about the consequences of actions before execution. Recent advances in large-scale video generation models [14, 20] have shown that predictive models can generalize across diverse domains, providing strong priors for robotic decision-making. In robotics, existing world models can be broadly categorized into three paradigms. Video world models, such as Cosmos [1] and DreamGen [10], predict visually realistic future frames but offer limited controllability, as language conditioning is often coarse. Action-conditioned world models, exemplified by DINO-WM [27], predict future states conditioned on robot actions, providing more accurate and controllable predictions of action outcomes, though their visual fidelity may be limited. A third line of work explores latent action world models, such as LAPA [24], which represent actions in a latent space and enable training on

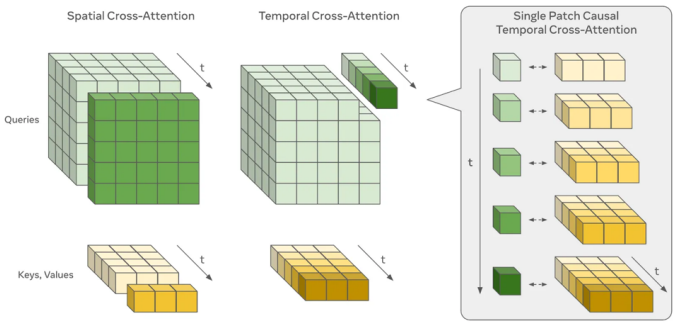

Fig. 2: **Spatio-temporal cross attention.** Each cube represents a D-dimensional vector. Spatial cross-attention (left) performs full cross-attention between the query and the key and values at each timestep independently. Temporal cross-attention (middle) performs causal cross-attention on a per-patch level independently (right).

broader datasets, but face challenges in transferring across embodiments. In this work, we focus on action-conditioned world models, as they most directly capture the consequences of candidate actions, making them well suited for evaluating action proposals during test-time steering.

Policy Steering Pretrained VLA models are generative policies that produce actions stochastically, either through autoregressive sampling or diffusion-based denoising. This stochasticity enables multiple candidate action sequences to be generated for the same instruction, creating an opportunity for policy steering, where an external evaluator selects the most appropriate behavior at test time. V-GPS [15] improves generalist robotic policies by re-ranking sampled actions using a language-conditioned value function trained via offline reinforcement learning. While effective, V-GPS evaluates individual actions rather than temporally extended action chunks, and lacks an explicit model of future state evolution for reasoning over long-horizon consequences. FOREWARN [22] addresses this limitation by incorporating a world model and using a task-specifically fine-tuned VLM to narrate and reason about predicted rollouts. However, this approach relies on language-based narration to detect failures, introducing potential brittleness when reasoning about unseen behaviors.

In contrast, our approach explicitly predicts long-horizon future states, and decouples an action-conditioned world model from a value function to evaluate imagined outcomes of candidate action chunks. By directly scoring predicted rollouts for instruction alignment, our method enables generalized and long-horizon test-time steering without task-specific fine-tuning.

III. Methodology

A. Problem Statement

We consider a partially observed decision-making setting in which an agent interacts with an environment under a natural language instruction l. During the interaction, at each timestep t, the agent receives an observation $o_t \in \mathcal{O}$ and generates an action from the action space $\mathcal{A}$, leading to a new observation under unknown environment dynamics. Unlike standard reinforcement learning, explicit reward signals are unavailable; instead, the goal is to select actions whose resulting outcomes best satisfy the given instruction l. We assume access to a pretrained VLA policy $\pi_\theta(a_{t:t+T} \mid o_t, l)$ that stochastically generates temporally extended action sequences $a_{t:t+T}$. Given a candidate action sequence, an action-conditioned world model $\mathcal{W}_\phi$ predicts the corresponding future observation rollout $\hat{o}_{t:t+T}$. A trajectory-level, language-conditioned value function $V_\psi(o_t, \hat{o}_{t+1:t+T}, l)$ is then used to score imagined trajectories, enabling test-time evaluation and selection of action proposals without policy retraining.

B. Overview

We first describe how our latent world model is trained with heterogeneous data. We then introduce **DreamSteer**, a test-time steering framework that uses the action-conditioned world model, together with a VLM-based value model, to evaluate and select action chunks. By combining the world model's rollouts with a language-conditioned value model evaluation, DreamSteer enables informed action selection at test time without retraining the policy.

C. World Model Training

Our world model design, shown in Fig. 1, follows three principles aimed at enabling efficient and scalable test-time steering. More design and training details are included in the supplementary materials.

Spatio-Temporal Factorization. To support long-horizon prediction under tight latency constraints, we adopt a factorized spatio-temporal transformer [23]. A spatio-temporal transformer consists of a series of N spatio-temporal blocks that process latent visual and action representations. Each block applies spatio-temporal self-attention to image tokens, temporal self-attention to action tokens, and spatio-temporal cross-attention to integrate action information into visual representations. As illustrated in Fig. 2, spatio-temporal attention is factorized into independent spatial attention at each timestep and causal temporal attention applied per patch. By decoupling spatial attention within frames from causal temporal attention across patches, the model reduces computational complexity from quadratic to linear in the prediction horizon T, enabling rapid evaluation of multiple candidate actions during steering.

Latent-Space Dynamics. Rather than predicting pixel-level video, $\mathcal{W}_\phi$ operates entirely in the latent space of a frozen visual encoder. This choice is motivated by classical control principles, which model system dynamics in a state space rather than raw observations. Pixels are high-dimensional, view-dependent, and lack explicit physical structure, whereas latent representations can encode essential properties of the 3D world, such as object identity, geometry, pose and interaction affordances. Additionally, latent dynamics are more computationally efficient than pixel-level video prediction, allowing fast autoregressive rollout, avoiding the computational overhead of iterative denoising and making repeated test-time rollout practical.

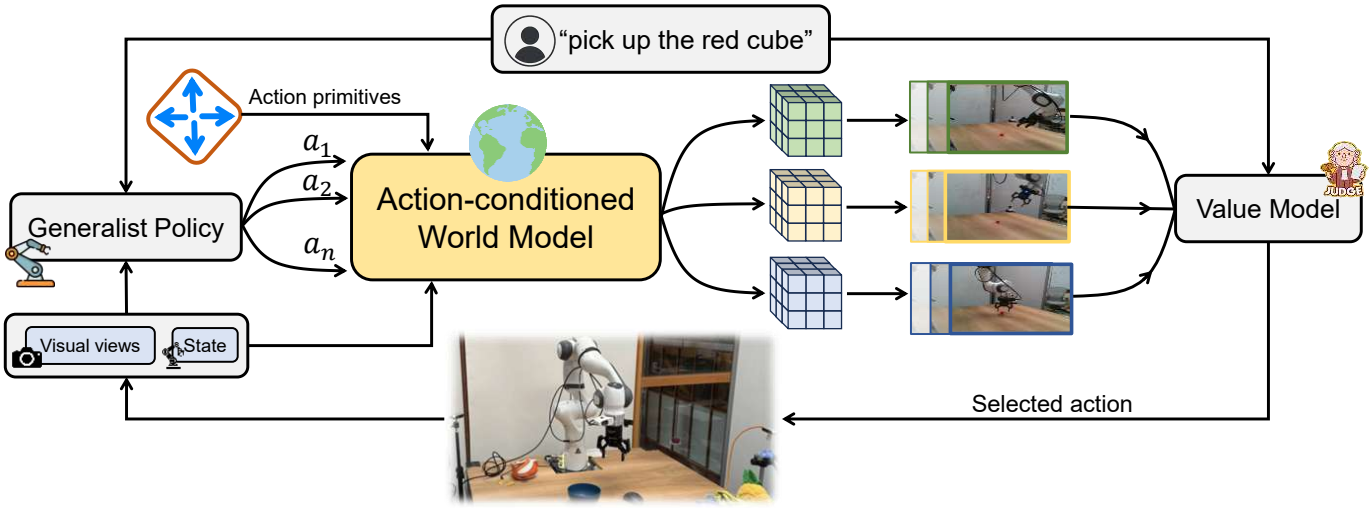

Fig. 3: **DreamSteer: test-time policy steering.** Given a language instruction and current observation, a pretrained generalist policy generates multiple candidate action sequences, which are augmented with a small set of predefined action primitives. An action-conditioned world model predicts the outcomes of each candidate through imagined rollouts, which are evaluated by a language-conditioned value model. The highest-scoring action is then selected and executed in the real environment.

Multi-Embodiment Training. Learning robust object interaction dynamics requires diverse data. By training on data collected from multiple embodiments, the world model is exposed to a wide range of object behaviors, contact patterns, and interaction modes. This enables the model to focus on learning transferable object-centric interaction dynamics rather than embodiment-specific motion patterns. Each embodiment is equipped with its own action encoder that maps raw control signals into a shared latent action space, while all other components of the world model are shared. When training on datasets with missing modalities, unavailable inputs are masked. This design isolates embodiment-specific details to the action encoders while allowing the core model to focus on learning transferable interaction dynamics.

D. DreamSteer

DreamSteer, shown in Fig. 3, is a test-time decision-making framework that improves action selection by evaluating multiple action proposals before execution. At each timestep t, given the current observation o_t and a language instruction l, a mixed candidate set $\mathcal{A}_t = \mathcal{A}_{\text{VLA}} \cup \mathcal{A}_{\text{prim}}$ of action chunks drawn from two sources: (i) $\mathcal{A}_{\text{VLA}}$ samples from the pretrained VLA policy π_θ and (ii) $\mathcal{A}_{\text{prim}}$ represents a primitive action library. Each candidate is rolled out in a learned world model to predict its future visual states, and a language-conditioned value function scores the predicted visual trajectories. The action sequence corresponding to the highest-scoring visual trajectory is then executed in the real environment. Formally,

$$a^{\star}_{t:t+T} = \arg \max_{a_{t:t+T} \in \mathcal{A}_t} V_\psi\big(o_t, \mathcal{W}_\phi(o_t, a_{t:t+T}), l\big), \quad (1)$$

where π_θ is the pretrained VLA policy, $\mathcal{W}_\phi$ the world model, and V_ψ the value function. Algorithm 1 summarizes the full test-time steering procedure.

a) Policy: Stochastic Action Proposals: The policy π_θ is instantiated as a pretrained VLA model that conditions on multimodal observations (I denoting visual observations and s denoting proprioception) and a language instruction l,

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

Conditioned on o_t, the policy π_θ samples K_{VLA} temporally extended action sequences of horizon T.

$$a^{(k)}_{t:t+T} \sim \pi_\theta(\cdot \mid o_t, l), \quad k = 1, \ldots, K_{\text{VLA}},$$

producing diverse candidate behaviors via diffusion or autoregressive sampling. In addition to policy-generated action proposals, we augment the candidate set with K_{prim} predefined action primitives from $\mathcal{A}_{\text{prim}}$. The primitive action library consists of a small set of fixed, short-horizon Cartesian motions, including end-effector translations along the left/right, up/down, and forward/backward directions, as well as gripper open and close actions. Each primitive is a constant Cartesian-space action applied for exactly T steps, ensuring that all primitive candidates are temporally aligned with VLA action chunks. Action primitives serve as a complementary source of candidates when the pretrained VLA policy fails to generate effective actions under distribution shift. These primitives provide simple, physically grounded motions that increase the coverage of feasible behaviors, allowing DreamSteer to recover instruction-aligned actions even when policy samples alone are insufficient.

b) World Model: Predictive Visual Dynamics: The world model $\mathcal{W}_\phi$ predicts future observations conditioned on the current observation and a candidate action sequence, operating in a latent representation space encoded by a frozen DINOv2 [16] visual encoder.

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

Algorithm 1: Test-Time Steering with World Models

Input: Pretrained VLA policy π_θ, world model $\mathcal{W}_\phi$, value function V_ψ.
Input: Per-step observation $o_t = \{I_t^{\text{wrist}}, I_t^{\text{ext}}, s_t^{\text{robot}}, s_t^{\text{gripper}}\}$.
Input: Language instruction l.
Input: VLA proposal count K_{VLA}, primitive set $\mathcal{A}_{\text{prim}}$ with $K_{\text{prim}} = |\mathcal{A}_{\text{prim}}|$, horizon T.
Output: Executed action sequence $a^\star_{t:t+T}$.
while *task not terminated* **do**

```
/* Candidate action set construction */
```

Proposal generation:
for $k \leftarrow 1$ **to** K_{VLA} **do**
  $a^{(k)}_{t:t+T} \sim \pi_\theta(\cdot \mid o_t, l)$
$\mathcal{A}_{\text{VLA}} \leftarrow \{a^{(k)}_{t:t+T}\}_{k=1}^{K_{\text{VLA}}}$;
$\mathcal{A}_t \leftarrow \mathcal{A}_{\text{VLA}} \cup \mathcal{A}_{\text{prim}}$;
$K \leftarrow |\mathcal{A}_t|$;

```
/* Action-conditioned predictive rollout */
```

Predictive rollout: for $k \leftarrow 1$ **to** K **do**
  $\hat{o}^{(k)}_{t+1:t+T} \leftarrow \mathcal{W}_\phi(o_t, a^{(k)}_{t:t+T})$

```
/* Trajectory-level evaluation under instruction l */
```

Evaluation: for $k \leftarrow 1$ **to** K **do**
  $s^{(k)} \leftarrow V_\psi(o_t, \hat{o}^{(k)}_{t+1:t+T}, l)$
Selection: $k^\star \leftarrow \arg\max_{k\in\{1,\dots,K\}} s^{(k)}$;
$a^\star_{t:t+T} \leftarrow a^{(k^\star)}_{t:t+T}$;
Execution: Execute $a^\star_{t:t+T}$ on the robot; observe o_{t+T}; $t \leftarrow t+T$;

It is implemented as a latent dynamics model consisting of an encoder, a dynamics module, and a decoder:

$$\begin{aligned} z_t &= \text{Encoder}(o_t), \\ \hat{z}^{(k)}_{t+1:t+T} &= \text{Dynamics}\big(z_t, a^{(k)}_{t:t+T}\big), \\ \hat{o}^{(k)}_{t+1:t+T} &= \text{Decoder}\big(\hat{z}^{(k)}_{t+1:t+T}\big). \end{aligned} \tag{2}$$

Operating in latent space enables structured and efficient prediction of long-horizon dynamics, while explicit conditioning on actions ensures physically consistent rollouts suitable for evaluating candidate behaviors.

c) Value Function Scoring and Action Selection: The value function V_ψ is instantiated as a pretrained Vision-Language-Action-Critic (VLAC) model [25] based on the InternVL2-2B [7] architecture, which is trained on over 4,000 hours of diverse human and robotic data. To evaluate an imagined rollout $\hat{o}_{t+1:t+T}$, we leverage VLAC's pairwise scoring capability to compute a cumulative trajectory score $S^{(k)} = \sum_{j=1}^{T} \text{VLAC}(\hat{o}_{t+j-1}, \hat{o}_{t+j}, l)$, representing the total estimated advancement toward the goal l. We interpret VLAC's pairwise score as an instruction-conditioned progress signal; summing scores along the rollout provides an approximate trajectory-level progress estimate. DreamSteer then selects and executes the action sequence $\mathbf{a}^{(k)}_{t:t+T}$ that maximizes this score, effectively steering the VLA policy toward semantically aligned behaviors.

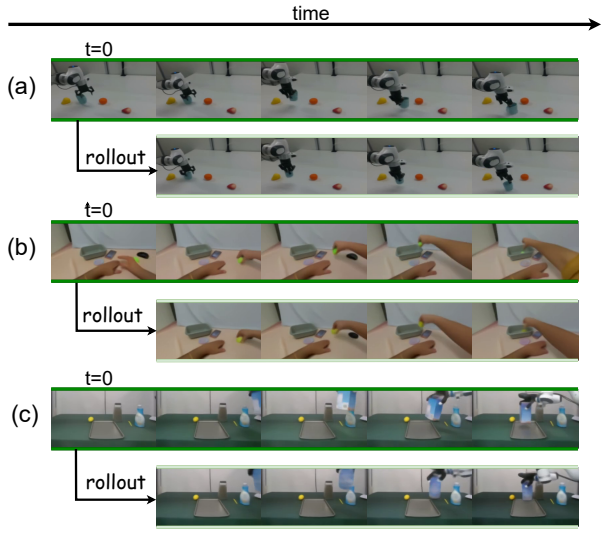

Fig. 4: **World model rollouts across multiple embodiments.** For clarity, we only visualize a single camera view.

IV. Experiments

A. Real Robot Setup

Our real-world experiments follow the DROID setup. We employ a 7-DoF Franka Panda manipulator equipped with a Robotiq two-finger gripper. Compared to the original DROID configuration, we make two minor modifications: the external third-person cameras are replaced with Intel RealSense L515 cameras, and the robot arm is mounted in a static configuration.

B. Evaluating the World Model

Before evaluating DreamSteer as a complete system, we first assess the learned world model in isolation. Our evaluation is designed to address the following key questions:

- **Accuracy & Generalization:** Does the world model accurately reflect real-world dynamics across multiple embodiments and generalize to unseen scenes and objects?
- **Evaluation Consistency and Long-Horizon Reliability:** Are imagined rollouts evaluated by the value model consistent with real video, and do they remain stable over long horizons?

1) Cross-Embodiment Generalization and Physical Consistency: We evaluate whether a single world model can generalize across embodiments, scenes, and object distributions. Trained on heterogeneous data spanning multiple robot platforms and human demonstrations, the model is evaluated on unseen real-world trajectories without embodiment-specific fine-tuning. As shown in Fig. 4, the predicted rollouts remain

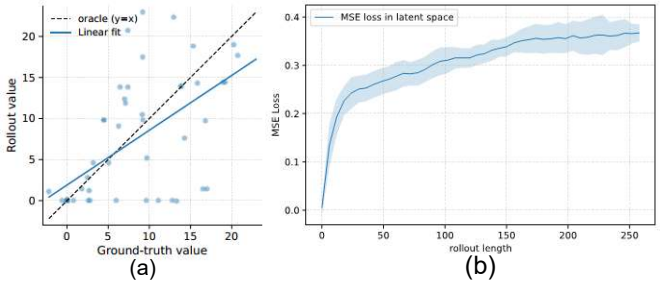

Fig. 5: **Evaluative consistency of imagined rollouts in the world model.** (a) Value model scores for ground-truth video segments and corresponding imagined rollouts. (b) Latent prediction error versus rollout length; shaded area denotes ± 1 standard deviation.

temporally coherent and physically consistent across embodiments and unseen object configurations. In particular, the model preserves object identity and relative motion across time, capturing object-centric interaction dynamics that generalize across embodiments and unseen object configurations, indicating that it captures shared object-centric interaction dynamics rather than embodiment-specific motion patterns or spurious visual correlations.

2) Evaluation Consistency: To assess whether imagined rollouts are suitable for value-based evaluation, we compare value model scores for short ground-truth video clips and their corresponding imagined rollouts under the same language instructions. As shown in Fig. 5 (a), the scores exhibit a clear correlation, indicating that latent rollouts preserve task-relevant semantic information. Each clip spans 10 frames, matching the per step rollout horizon used in DreamSteer. To examine how the rollout quality degrades over longer horizons, we analyze latent prediction error with rollout length. With a context length of one frame, the latent MSE increases gradually and approximately linearly, shown in Fig. 5 (b), suggesting stable behavior over extended rollouts. All evaluation data are randomly sampled from the unseen RoboArena dataset [3].

C. 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 to evaluate IF accuracy are shown in Fig. 6. Across all experiments, DreamSteer steers a fixed pretrained π_0 policy at test time by sampling K action chunks and action primitives, rolling them out in the action-conditioned world model, and selecting the highest-scoring action proposal according to a language-conditioned value function. We use a π_0 checkpoint pretrained on the DROID dataset as the base policy. Each task is evaluated over 20 trials with randomized object positions and poses. The policy produces action chunks of horizon T=10. For efficiency, steering is applied once every five control steps in our experiments.

1) OOD Object Generalization: All OOD evaluations are conducted on a standard pick-and-place manipulation task,

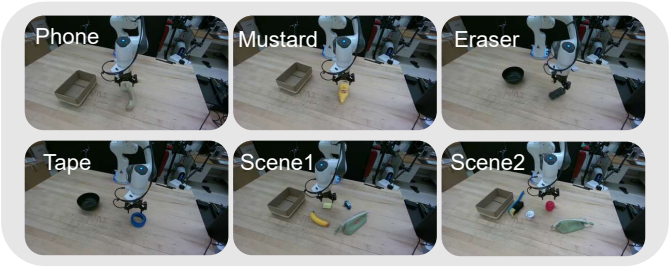

Fig. 6: **Real world tasks of OOD objects and scenes.**

where the robot is instructed to pick up a specified object and place it into a designated receptacle. OOD objects are selected based on preliminary evaluations where the base pretrained VLA policy achieves consistently low success rates, often failing to achieve stable grasps due to differences in object geometry, material properties, or surface appearance. These objects therefore represent realistic object-level distribution shifts that are challenging for the policy but remain physically feasible for the robot.

We first test whether DreamSteer improves performance on object-level distribution shifts. We construct OOD evaluation suites where the task structure remains the same but the target objects differ from training in instance, appearance, geometry, or material properties. We report task success rate as the primary metric. We compare DreamSteer with action primitives against several baselines, including (i) the base VLA policy with single-sample execution, (ii) a VLA-only DreamSteer variant that applies value-based selection over K action candidates sampled from the policy, (iii) a primitive-only steering baseline (prim+DreamSteer), and (iv) a random-selection baseline that selects among combined VLA and primitive candidates (VLA+prim+random). This set of comparisons isolates the respective contributions of value-based selection and incorporating multiple action sources. Here, π_0 denotes the pretrained VLA policy. "prim" denotes the inclusion of predefined action primitives in the candidate set, while "random" selects a candidate uniformly at random without value-based ranking.

TABLE I: Out-of-distribution objects performance

Policy	Phone	Mustard	Tape	Eraser	Average
π_0(k=1)	4/20	3/20	6/20	6/20	23.75%
π_0(k=5)+DreamSteer	7/20	6/20	11/20	10/20	42.5%
π_0(k=5+prim)+random	0/20	0/20	0/20	0/20	0%
prim+DreamSteer	0/20	0/20	0/20	0/20	0%
π_0(k=5+prim)+DreamSteer	**12/20**	**11/20**	**16/20**	**14/20**	**66.25%**

The failure mode of the primitive-only baseline further highlight the complementary role of action primitives in DreamSteer. The predefined primitives consist of simple, coarse Cartesian motions and lack the expressiveness required to accomplish fine-grained manipulation tasks on their own, resulting in 0 success when used without policy-generated actions. In contrast, the pretrained VLA policy can produce

rich, precise motion sequences necessary for task completion, but often fails to generate effective grasping actions on OOD objects, leading to behaviors where the end-effector becomes stuck above the target object. In such cases, action primitives provide a useful mechanism to escape these failure states by introducing structured exploratory motions. However, selecting among VLA and primitive actions at random disrupts coherent task execution and frequently steers the policy away from promising trajectories, underscoring the importance of value-based ranking in DreamSteer.

We analyze the effect of the proposal count K on both performance and efficiency of DreamSteer using a whiteboard eraser picking task. We vary the number of VLA-generated action proposals while keeping all other components fixed. We find that increasing K from 3 to 5 consistently improves task success, indicating that a moderate level of proposal diversity helps mitigate failures caused by insufficient action coverage. However, further increasing K to 7 does not yield additional gains, while incurring higher inference cost due to additional world model rollouts and value evaluations. Based on this trade-off, we set $K = 5$ in all experiments.

2) Instruction Following Accuracy: We next evaluate whether DreamSteer improves instruction-following performance in scenarios where success depends on satisfying explicit language constraints rather than producing plausible manipulation motions. We consider instruction-following tasks that require target disambiguation in multi-object scenes, where the robot must correctly identify and act on the object specified by the instruction in the presence of visually similar distractors. We report instruction-following (IF) accuracy, which measures whether the executed behavior aligns with the language-specified target. A trial is considered successful if the robot makes consistent contact with, or attempts to grasp, the object specified by the instruction, regardless of whether a stable grasp is ultimately achieved. This metric isolates semantic correctness from execution difficulty and allows us to evaluate how effectively test-time steering improves adherence to language constraints. We compare DreamSteer with both VLA and predefined action sources against the underlying pretrained VLA policy.

TABLE II: Instruction following performance

Policy	Sponge	Banana	Pencil case	Apple	Average
π_0(k=1)	8/20	9/20	6/20	8/20	38.75%
$\pi_0(k{=}5{+}\text{prim})$ +DreamSteer	**14/20**	**13/20**	**9/20**	**9/20**	**56.25%**

These results show that DreamSteer improves instruction adherence by filtering action proposals whose predicted outcomes conflict with the semantic intent of the instruction, particularly in scenes with visually similar distractors, where the base policy is prone to semantic ambiguity.

3) Behavior Stability and Execution Efficiency: Beyond task success, we observe that DreamSteer improves the stability and efficiency of the executed behaviors. The base pretrained VLA policy frequently produces erratic or oscillatory motions, resulting in large corrective movements and unstable trajectories during manipulation. By evaluating and filtering candidate actions through imagined rollouts, DreamSteer effectively suppresses such undesirable behaviors and selects smoother, more consistent action sequences.

As a result, DreamSteer also reduces the number of execution steps required to complete a task, requiring roughly half the number of control steps compared to the base VLA policy on OOD objects. This improvement arises because poorly chosen actions often require multiple corrective actions to recover, or lead to irreversible failure states. In contrast, DreamSteer proactively avoids these failure-inducing actions at test time, leading to more stable execution and fewer steps task completion.

D. Failure Analysis

Despite the overall performance gains reported in Section IV-C, DreamSteer exhibits several failure modes that reflect the limitations of its predictive and evaluative components, rather than the test-time steering formulation itself.

a) World Model Hallucination: In some cases, the action-conditioned world model predicts successful object grasping in imagined rollouts even when the object cannot be grasped in the real world ("magic grasp"). We attribute this behavior to training bias toward successful manipulation trajectories, which can lead the model to overestimate grasp feasibility under uncertainty. When the candidate actions yield this over-optimistic rollouts, the value model may fail to reject physically infeasible actions. A practical mitigation is to expand world-model training beyond success-biased demonstrations by incorporating failure trajectories and random-play interaction data. These data expose the model to non-graspable contacts, slips, and unsuccessful attempts, reducing the success bias that leads to unrealistically optimistic rollouts. Qualitative examples of these failure cases are provided in the supplementary material.

b) Value Model Misranking: A second failure mode arises from imperfect trajectory ranking by the value model. While generally effective, the value prediction exhibits sensitivity to viewpoint and visual presentation. The value model operates on observations from a single fixed camera view, which can make certain action outcomes visually ambiguous or partially occluded. As a result, the highest-scoring trajectory does not always correspond to the most effective real-world execution. A potential improvement is to provide the value model with observations from multiple viewpoints and aggregate the resulting scores, for example by averaging, to reduce sensitivity to viewpoint-specific ambiguities.

V. Discussion and Limitations

A. Why DreamSteer Works

The empirical gains observed in our experiments can be directly attributed to the complementary generalization properties and decision structure introduced by DreamSteer. Across both OOD object evaluations and instruction-following tasks,

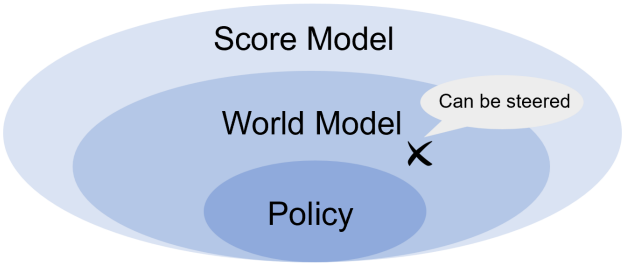

Fig. 7: **Training data and generalization among policy, world model, and score model.**

DreamSteer consistently outperforms the base VLA policy, particularly in regimes where the policy alone exhibits brittle behavior. In general, the generalization ability of learned models is closely tied to the diversity and scale of their training data. This pattern aligns with the fact that pretrained VLA policies are trained on success-only trajectories collected under a single embodiment, which limits their exposure to diverse object instances, failure modes, and interaction variations. In contrast, the world model is trained on heterogeneous, and cross-embodiment data, while the scoring model benefits from larger internet-scale action-free video data. As a result, many evaluation scenarios that are out-of-distribution for the policy, but remain within the distribution of the world model and the scoring function, enabling DreamSteer to recover effective behaviors through test-time selection, as illustrated in Fig. 7.

An important observation underlying DreamSteer is that the pretrained VLA policy succeeds at a task at least once, indicating that effective action trajectories already exist within its action distribution. However, due to the stochastic nature of action generation, these successful trajectories may not be reliably sampled during deployment. DreamSteer addresses this gap by explicitly sampling and evaluating multiple action candidates, enabling the system to recover high-quality actions that are already supported by the policy but would otherwise be missed.

Finally, the improvements in both OOD objects and instruction-following accuracy further illustrate the benefits of reformulating action generation as a discrimination problem. In multi-object scenes with distractors, the base policy often produces plausible but semantically incorrect behaviors, such as manipulating the wrong object. DreamSteer mitigates this failure mode by sampling multiple candidate action sequences and selecting those whose predicted outcomes best align with the language instruction. Our results show that even when the policy fails to consistently generate the correct action, stochastic action primitives sampling frequently produces candidates that partially satisfy the instruction. Discriminating among these candidates using imagined rollouts is easier than requiring the policy to generate a perfectly aligned action sequence in one pass.

B. Limitations

While DreamSteer improves action selection at test time, it does not address all aspects of robotic decision-making. We discuss two inherent limitations of the current framework.

1) Action Candidate Coverage: DreamSteer selects actions from a finite candidate set composed of stochastic policy samples and predefined action primitives, but it does not expand the underlying action space of the pretrained policy. When neither source produces a trajectory that meaningfully progresses toward satisfying the instruction, steering cannot recover a successful behavior. This limitation underscores that DreamSteer improves action *selection* rather than action *generation*, and its effectiveness depends on the diversity and expressiveness of the candidate set.

2) Latency: Policy inference is fast, while world model rollouts and value model evaluation introduce additional but moderate computational overhead. For an action chunk of horizon T=10, averaged over 10 runs, policy inference takes 0.08 s, world model rollout takes 0.59 s, and value model evaluation takes 0.37 s. Policy and value model inference are performed on an NVIDIA RTX 4090 GPU, while the world model runs on an NVIDIA A6000 GPU. We note that the current implementation prioritizes clarity and modularity over aggressive optimization, and both the world model and value model admit substantial opportunities for further optimization, such as more efficient rollout implementations or latent-space trajectory scoring.

VI. Conclusion

In this paper, we introduced **DreamSteer**, a test-time steering framework that enhances the robustness of OOD objects and instruction-following ability of pretrained VLA policies without additional policy training. By combining stochastic action generation with action-conditioned world models and language-conditioned evaluation, DreamSteer enables informed action selection through imagined rollouts, without modifying or retraining the underlying policy. Extensive experiments demonstrate that this approach improves generalization to unseen objects and adherence to language-specified goals, while preserving the broad capabilities acquired through large-scale pretraining.

Beyond empirical gains, DreamSteer exemplifies a general paradigm for improving decision-making in generalist robotic systems by decoupling action generation and action evaluation. Pretrained VLA policies provide broad visuomotor competence by generating diverse candidate actions, while world models enable explicit reasoning about the consequences of these actions at test time. This separation allows an external evaluator to select actions that better align with task instructions, compensating for the limitations of pretrained policies without additional training.

One direction for future work is to further tighten the integration between world models and action evaluation, for example by learning scoring functions directly in latent space. Such approaches could enable faster trajectory evaluation and make test-time steering more practical for real-time robotic control. Another promising direction is to enrich the candidate action set by incorporating diverse action sources to improve action coverage.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 1, 2

[2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 1, 2

[3] Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, Cyrus Neary, Edward Hu, Fabio Ramos, et al. Roboarena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 2025. 6

[4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. 1, 2

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian 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*. 1, 2

[6] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025. 2

[7] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024. 5

[8] Ryan Hoque, Peide Huang, David J Yoon, Mouli Sivapurapu, and Jian Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025. 2

[9] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 1

[10] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through neural trajectories. *arXiv e-prints*, pages arXiv–2505, 2025. 2

[11] Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty. *arXiv preprint arXiv:2502.18581*, 2025. 1

[12] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. 2

[13] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 1

[14] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024. 2

[15] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024. 3

[16] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 4

[17] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 1

[18] Renjie Pi, Haoping Bai, Qibin Chen, Xiaoming Simon Wang, Jiulong Shan, Xiaojiang Liu, and Meng 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. 1

[19] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

[20] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025. 2

[21] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024. 2

[22] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop

policy steering via latent alignment. *arXiv preprint arXiv:2502.01828*, 2025. 3

[23] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020. 3

[24] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024. 2

[25] Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025. 5

[26] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 1

[27] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning 2024. *URL https://arxiv.org/abs/2411*, 4983. 1, 2

[28] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan 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. 1

SUPPLEMENTARY MATERIALS

VII. 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.

A. 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.

a) Visual observations: RGB observations are encoded independently at each timestep using a pretrained 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.

b) 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.

c) 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

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

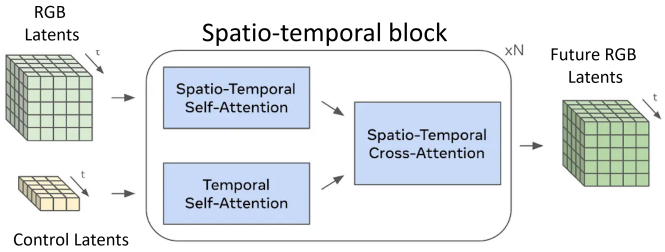

Fig. 8: **Spatio-temporal world model architecture.**

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.

d) 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.

B. 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.

a) 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) spatio-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. 2.

C. 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.

a) Training datasets: The world model is trained on a mixture of multi-embodiment manipulation datasets spanning both robot and human demonstrations. Our training corpus

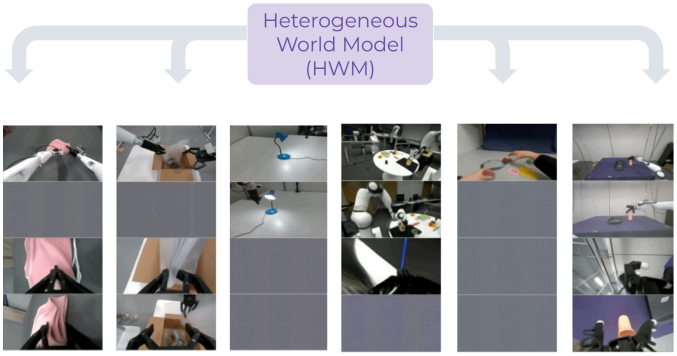

Fig. 9: **Mask strategy in multi-view data.**

includes: (i) the DROID dataset collected on Franka platforms (76k trajectories, ∼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.

b) 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 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.

c) 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.

D. Autoregressive Rollout and Training Objective

a) 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. VII-A), 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. VII-B.

b) 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.

c) 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.

d) 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 horizon H:

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

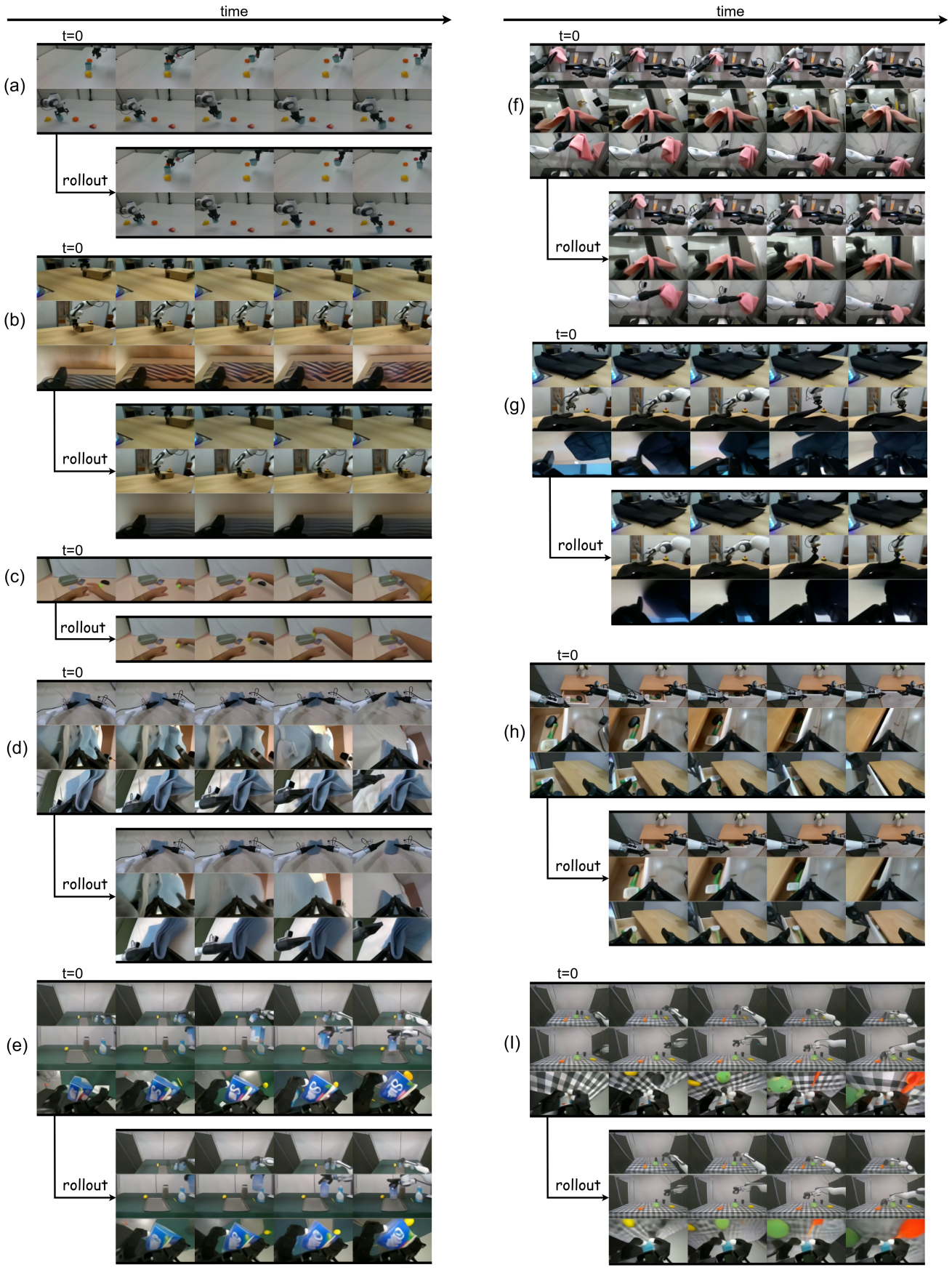

Fig. 10: **World model rollout results.**

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

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

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

e) 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.

f) 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. VII-D, with frozen visual tokenizers and jointly optimized action/state tokenizers. All hyperparameters are listed in Table III.

g) 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.

E. 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, following 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.

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
Sampling horizon H	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 III: Training hyperparameters for the spatio-temporal world model.

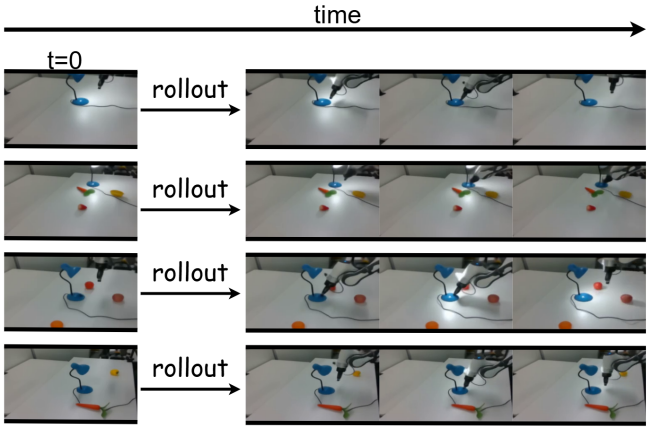

Fig. 11: **Functional interaction prediction.** When the robot toggles the light switch in imagination, the world model predicts corresponding changes in scene illumination.

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.

VIII. Value Model and Trajectory Scoring

a) Model choice: To evaluate predicted rollouts during test-time steering, we adopt an off-the-shelf vision–language value model, Vision-Language-Action-Critic (VLAC), without additional finetuning. VLAC is pretrained to estimate task progress by comparing pairs of observations conditioned on a language instruction, producing a scalar progress delta that

reflects relative advancement toward task completion.

b) 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

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

We evaluate task progress along the rollout using pairwise comparisons with the starting observation:

$$s_{t+k} = \mathrm{VLAC}(\hat{o}_t, \hat{o}_{t+k}, l_{\text{task}}),$$

where l_{task} denotes the language instruction and s_{t+k} represents the predicted progress at horizon step k. In practice, we set $k = 1$ in our experiments.

c) Trajectory aggregation: A trajectory-level score is computed by aggregating progress estimates across the rollout horizon:

$$S = \sum_{k=1}^{H} s_{t+k}.$$

This scalar score reflects the predicted task advancement induced by the candidate action sequence. During test-time steering, multiple candidate sequences are evaluated, and the sequence with the highest trajectory score is selected for execution. The value model operates on a single left-view RGB observation.

IX. 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.

X. 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:
  - "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.

Fig. 12: **Robot platform.**

XI. Failure Modes

A. World Model Hallucination

Magic grasp hallucination is a common failure mode occurs during contact-rich manipulation, where the world model predicts successful grasps despite failure in the real world. Figure 13 shows two examples in which the robot fails to grasp the object during execution, while the world model rollout incorrectly depicts the object as being grasped and lifted.

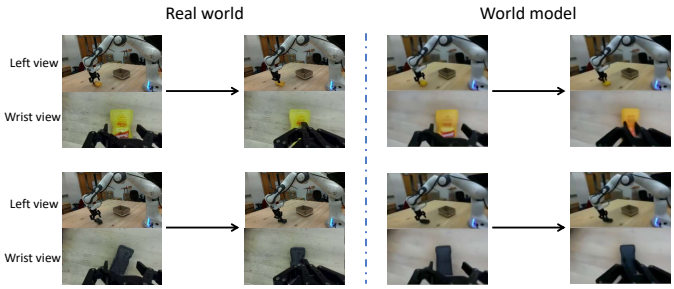

Fig. 13: **World model hallucination examples.** Left: real-world execution where the object is not grasped. Right: world model rollout predicting a successful grasp.

B. *Value Model Misranking*

Fig. 14: **Value model misranking examples.**

We observe failure cases where the value model assigns higher scores to suboptimal action candidates, resulting in incorrect steering decisions during deployment. Figure 14 illustrates two representative real-world examples.

In the phone case, the selected action corresponds to a downward motion that brings the gripper closer to the object. However, this motion does not place the gripper in a grasp-ready pose. In the mustard case, the robot already reaches a grasp-ready pose, yet the value model favors a forward motion, causing the gripper to move away from the target object.

We attribute these misranking failures primarily to limited visual observability in the value model input. The value model operates on a single left-view RGB observation, under which certain action outcomes appear visually ambiguous, leading to incorrect value ordering.