PRISMWAM: DECOUPLING STATE REPRESENTATIONS IN WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) typically represent the world through latent embeddings from pretrained video models, which entangle heterogeneous aspects of the world, from *how it looks* (e.g., texture and color) to *how it is physically configured* (e.g., occupancy and pose). This raises an underexplored question: ***what should constitute the state of a WAM?*** We argue that the world state should separate *how the world appears* from *how the world evolves under action*: the former captures *semantic context*, while the latter is governed by *physical state*. To this end, we propose **PrismWAM**, which decouples *physics* from *semantics* while jointly modeling state and action. PrismWAM factorizes an embodied task into a $\text{Semantic Planning} \rightarrow \text{Physical Rollout} \rightarrow \text{Executable Action}$ pipeline: given only the initial RGB observation, semantic planning learns to predict the final goal image, while physical rollout predicts the fine-grained evolution toward this goal in depth space; executable actions are then generated from the depth rollout. Without any embodied action pretraining or additional data augmentation, PrismWAM achieves state-of-the-art manipulation performance in both simulation and the real world. Notably, PrismWAM even surpasses a broad range of models with large-scale embodied action pretraining (e.g., OpenWAM-α). More broadly, we reframe World Action Models from modeling *how the world looks* to *how it responds to action*, paving the way for action-centric world models grounded in occupancy, flow, tactile sensing, proprioception, and beyond.

1 INTRODUCTION

World models have emerged as a promising paradigm for enabling autonomous agents to predict the consequences of their actions (Ha & Schmidhuber, 2018; LeCun, 2022). By learning how the world evolves over time, these models aim to provide a foundation for planning and action (Hafner et al., 2019). Recent World Action Models (WAMs) extend this paradigm by jointly modeling video and action, often relying on pretrained video backbones to capture the rich visual dynamics of the physical world (Ye et al., 2026b; Li et al., 2026b; Ye et al., 2026a). Yet, a key question remains underexplored: ***are video latents truly the most appropriate representations for world states?***

Our key observation is that video latents capture a mixture of factors that are not equally relevant to action generation (NVIDIA et al., 2025a). While appearance and texture provide rich semantic context for understanding the scene, they are often invariant to the actions being modeled. For example, when an agent moves or grasps an object, its geometry and pose may change, while its color, texture, and intrinsic appearance typically remain unchanged. Therefore, video latents which entangles the **physical state** with the **semantic context** can make the representation substantially richer than what is necessary for modeling how the world evolves under action.

This motivates a simple principle: a good state representation in WAMs should distinguish between physical state and semantic context. The physical state refers to action-relevant properties such as geometry, pose, occupancy, contact, and potentially dynamics, whereas semantic context provides the appearance information necessary to interpret that state, such as object identity, category, and texture; see Fig. 1 (a). The distinction is not between information that is useful and useless; rather, it decouples *what actions can change* from *what helps us understand what is being acted upon*.

Therefore, we propose **PrismWAM** to separate semantic context from physical dynamics through a $\text{Semantic Planning} \rightarrow \text{Physical Rollout} \rightarrow \text{Executable Action}$ workflow (see Fig. 1 (b)), rather

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

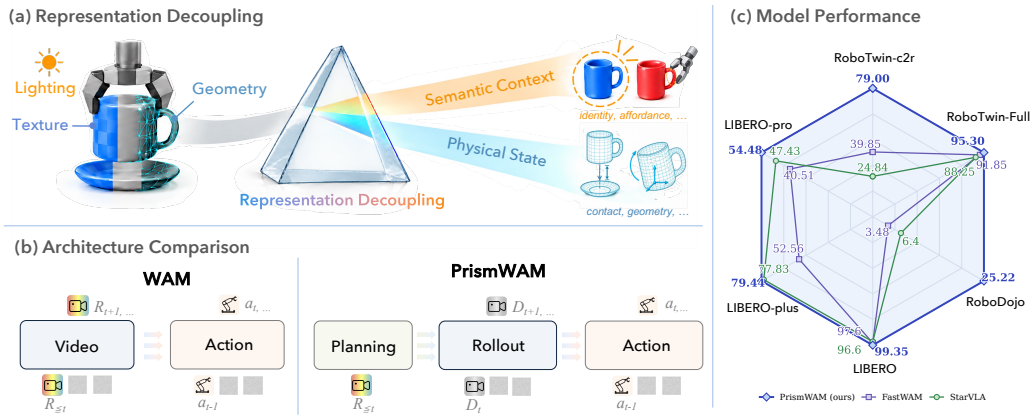


Figure 1: **Overview of PrismWAM.** (a) Our key insight is to decouple physical state from semantic context, yielding state representations that are better aligned with action-induced dynamics. (b) We follow a causal workflow of *what to do* → *how the world responds* → *how to act*, corresponding to *Semantic Planning* → *Physical Rollout* → *Executable Action*. (c) Our best model consistently outperforms state-of-the-art methods by a large margin across six benchmarks.

than treating a single video latent as a monolithic state. Semantic planning infers ***what should happen in the end*** from RGB observations, physical rollout models ***how the world physically evolves*** toward the goal, and executable action determines ***how to make it happen***. We instantiate this principle with a Mixture-of-Transformers (MoT) architecture built upon a pretrained video backbone (Team Wan et al., 2025), as shown in Fig. 2. During training, the semantic planner predicts the target from the initial RGB observation and transfers its task-level knowledge to the physical rollout via attention layers, while the action branch learns to map the predicted physical evolution to continuous actions. We use depth as the physical representation for its accessibility, physical grounding, and spatial alignment with RGB. Notably, the semantic planner serves only as a training-time teacher, with action inference relying solely on the initial RGB observation and learned depth rollout.

Extensive experiments across diverse simulation and real-world benchmarks validate the effectiveness and efficiency of PrismWAM. **(1)** Without action pretraining, PrismWAM substantially outperforms existing VLAs and WAMs under the same setting across all benchmarks, improving the success rate over the strongest baselines by **39.15** points (79.00 vs. 39.85) on RoboTwin-C2R and **18.82** points (25.22 vs. 6.4) on RoboDojo; see Fig. 1(c). **(2)** PrismWAM outperforms existing WAMs regardless of action pretraining, with particularly strong gains in precision manipulation and generalization: on RoboDojo Precision, it surpasses OpenWAM-α by **19.99** points (38.44 vs. 18.45), while improving the success rate on RoboTwin-C2R by **10.00** points (79.00 vs. 69.00), showing a higher ceiling for complex tasks and novel scenes. **(3)** PrismWAM even matches or surpasses large-scale pretrained models on several benchmarks despite being trained with only ∼2,300 hours of depth data. On RoboDojo, it improves the score by **4.99** points over G0.5 (25.22 vs. 20.23) and **5.15** points over Xiaomi-Robotics-1 (25.22 vs. 20.07), showing that scaling action pretraining is not the only path to stronger embodied foundation models; principled architectural design offers a more data-efficient alternative. Our key contributions are listed as follows:

- **A decoupling principle for World Action Models.** We identify the entanglement of semantics and physics in video latents as a fundamental limitation of existing WAMs, forcing action generation to disentangle high-level task planning from fine-grained physical state transitions.
- **An effective and efficient WAM architecture.** We instantiate the principle with a Mixture-of-Transformers (MoT), where an RGB semantic planner guides depth-based physical rollout, and develop a learning pipeline combining dense depth supervision with sparse RGB goal supervision, enabling fast inference without explicit RGB goal prediction.
- **Strong performance and data efficiency.** PrismWAM outperforms existing VLAs and WAMs, with particularly strong gains in precision manipulation and generalization. It also surpasses a number of models with large-scale action pretraining, demonstrating that principled architectural design offer a more data-efficient path toward stronger embodied foundation models.

2 Related Works

Vision-Language-Action Policies. The general and embodied knowledge encoded in pretrained vision-language models (VLMs) (Comanici et al., 2025; Bai et al., 2025; Li et al., 2026a; Wang et al., 2026b; Xiaomi Embodied Intelligence Team, 2025) provides a foundation for vision-language-action (VLA) models that map visual observations and language instructions to robot actions (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Li et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025b; Shukor et al., 2025). Internet-scale image–text pretraining provides semantic priors for complex instruction understanding and long-horizon manipulation (Brohan et al., 2023; Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025b; Shi et al., 2025). However, such pretraining emphasizes recognition and semantic alignment, with limited explicit supervision for continuous state transitions and how actions change the environment.

World Action Models. Advances in video generative models (Team Wan et al., 2025; NVIDIA et al., 2025a; NVIDIA, 2026) have paved the way for world action models (WAMs), a promising paradigm that jointly models future states and robot actions for generalist robotic manipulation. Most jointly model future RGB observations and action sequences (Zhu et al., 2025; Kim et al., 2026). Some methods guide actions based on predicted future RGB observations (Shen et al., 2025; Bi et al., 2025; Li et al., 2026b; Ye et al., 2026b; Xu et al., 2026; Zhang et al., 2026b; Wang et al., 2026a). Others use future prediction as training supervision and generate actions directly from observed context (Yuan et al., 2026; Ye et al., 2026a; Fang et al., 2026; Gu et al., 2026; Zhang et al., 2026a). Both approaches primarily learn state evolution in RGB appearance space, which entangles action-relevant geometry and pose with texture, illumination, and background. This requires modeling appearance details that may be irrelevant to control and inferring physical changes from visual variations that may not reflect motion (Qiu et al., 2026; Zhang et al., 2026a).

Physical Representations for Manipulation. Physical or spatial representations support precise manipulation and spatial generalization (Shridhar et al., 2022; Goyal et al., 2023; Ze et al., 2024; Ke et al., 2024; Qu et al., 2025). Such information can directly guide action generation through depth or point-cloud features extracted from current observations (Yuan et al., 2025; Li et al., 2025a; Ke et al., 2024). Complementary approaches introduce physical or spatial structure through supervision, including representation alignment (Li et al., 2025b; Wu et al., 2026; Shi et al., 2026; Yang et al., 2026) and auxiliary prediction (Guo et al., 2026; Li et al., 2026c; Zhang et al., 2026a; 2025).

Recent world action models take RGB and geometry as joint inputs and predict their future states together (Zhao et al., 2026; Lopetegui Gonzalez et al., 2026). This couples semantic and physical information, retaining the RGB-based modeling issues discussed above. PrismWAM separates their roles along a $\mathrm{Semantic\ Planning} \rightarrow \mathrm{Physical\ Rollout} \rightarrow \mathrm{Executable\ Action}$ chain, distinguishing semantic task understanding from the physical state evolution that guides action execution.

3 Method

PrismWAM follows a $\mathrm{Semantic\ Planning} \rightarrow \mathrm{Physical\ Rollout} \rightarrow \mathrm{Executable\ Action}$ chain for prediction and control. The RGB stream learns semantic planning representations from a sparse future-frame target, while the depth stream predicts a dense physical rollout that guides action generation. We first present the problem formulation (§3.1), then describe PrismWAM's architecture, training, and inference (§3.2).

3.1 Problem Formulation

We formulate a world–action model (WAM) that jointly predicts world-state evolution and the corresponding actions. Let x_0 denote the current world state, $x_{1:F}$ its future world state trajectory, and $a_{1:H}$ the corresponding action sequence. Given task context $c = (\ell, s)$, comprising the language instruction ℓ and proprioceptive state s, the policy can be expressed as

$$\pi_\theta(a_{1:H} \mid x_0, c) = \int \underbrace{p_\theta(a_{1:H} \mid x_{1:F}, x_0, c)}_{\text{action generation}} \underbrace{p_\theta(x_{1:F} \mid x_0, c)}_{\text{world-state evolution}} \, \mathrm{d}x_{1:F}. \tag{1}$$

Here, $p_\theta(x_{1:F} \mid x_0, c)$ models how the current state may evolve over the prediction horizon, while $p_\theta(a_{1:H} \mid x_{1:F}, x_0, c)$ predicts actions conditioned on the current state and a future state trajectory.

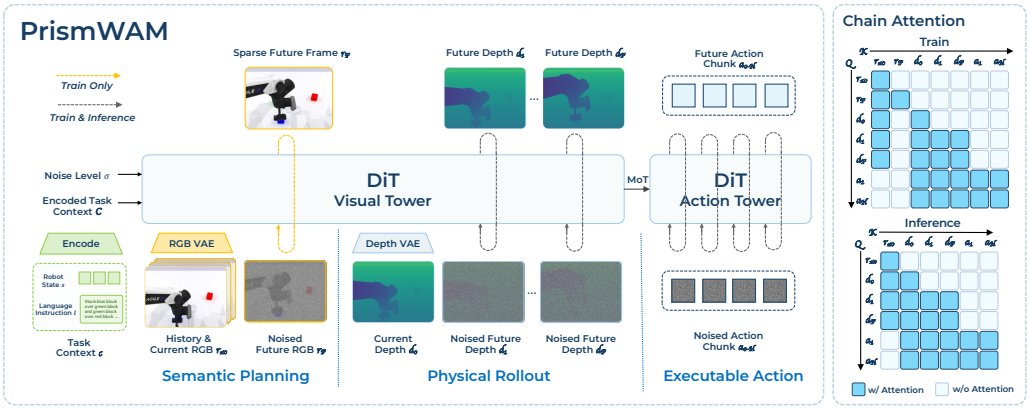


Figure 2: **PrismWAM decouples semantic planning from physical rollout. Left**: the RGB stream learns task semantics through sparse supervision from a single intended-future frame, while the depth stream predicts a dense geometric trajectory jointly with actions. **Right**: chain attention routes observed-RGB features to the depth stream and physical rollout features to the action tower.

Marginalizing over possible future world state trajectories yields the policy's action distribution. The future world state trajectory $x_{1:F}$ and action sequence $a_{1:H}$ cover the same time interval.

Semantic Planning and Physical Rollout. We use historical and current RGB observations $r_{\leq 0}$ for semantic planning and the current depth observation d_0 as the initial physical state. Semantic planning captures task intent and the intended scene configuration, while physical rollout models the intermediate geometric changes needed for action execution. Guided by RGB representations, the depth stream predicts the sequence $d_{1:F}$, and the action tower jointly generates executable actions from the evolving rollout features.

Let z_R denote semantic planning representations extracted from $r_{\leq 0}$ and z_D the physical rollout features produced during depth prediction. The resulting policy is

$$\pi_\theta(a_{1:H} \mid r_{\leq 0}, d_0, c) = \iint \underbrace{p_\theta(a_{1:H} \mid z_D, c)}_{\text{Executable Action}} \underbrace{p_\theta(z_D \mid d_0, z_R, c)}_{\text{Physical Rollout}} \underbrace{p_\theta(z_R \mid r_{\leq 0}, c)}_{\text{Semantic Planning}} \, \mathrm{d}z_R \, \mathrm{d}z_D. \quad (2)$$

3.2 PRISMWAM

Architecture with Sparse–Dense Temporal Supervision. Building on the pretrained video backbone WAN (Team Wan et al., 2025), PrismWAM uses RGB and depth streams with asymmetric temporal supervision, coupled to an action tower through a mixture-of-transformers (MoT) architecture (Fig. 2, left). The RGB stream predicts a single intended-future frame r_F, a sparse target for learning task semantics. The depth stream predicts the denser sequence $d_{1:F}$, providing intermediate spatial states that guide action evolution. This assigns semantic planning to sparse RGB prediction and physical rollout to dense geometric prediction, whose representation is less entangled with texture and illumination.

The model receives RGB history $r_{\leq 0}$, current depth d_0, instruction ℓ, and proprioception s. Camera views are resized into matching RGB and depth mosaics, and RGB values and clipped metric depth are normalized. The frozen RGB and depth VAEs encode observations and future training targets into latent representations, which modality-specific patch projections convert into visual tokens. Frozen T5 embeddings and projected, normalized proprioception form the context C, which all branches access through cross-attention. Appendix A.1.1 specifies input layouts, temporal sampling, and attention masks.

During joint training, the RGB token sequence combines clean history latents with noisy latents of the single future target, while the depth sequence combines clean current-depth latents with noisy future-depth latents. The action branch linearly projects each action vector in the noisy sequence into an action token. Gaussian corruption is applied only to prediction targets; observed RGB and depth remain fixed. Each branch receives its own noise-level embedding.

Chain Attention. A structured attention mask controls information flow between these token groups (Fig. 2, right). Within each visual stream, future tokens attend to observed tokens and one another, while observed tokens cannot read future tokens. Depth tokens additionally read only the observed-RGB tokens. Action tokens attend to one another and to both observed and future depth tokens, with no direct access to RGB. These cross-modal connections are directed: RGB does not read depth or actions, and depth does not read actions. Thus, action prediction uses evolving depth features without accessing the future-RGB target.

Specifically, let $h_m^{(j)}$, $m \in \{R, D, A\}$, denote the RGB, depth, and action features after j layers, with $h_m^{(0)}$ their input embeddings and $h_{R,\le 0}^{(j)}$ the observed-RGB subset. Both visual streams and the action tower receive the encoded task context C through cross-attention. With modality-specific noise levels $\sigma_R, \sigma_D, \sigma_A$, the three feature sequences update synchronously as

$$\begin{aligned} h_R^{(j+1)} &= \mathcal{B}_V^{(j)}\left(h_R^{(j)} \mid C, \sigma_R\right), \\ h_D^{(j+1)} &= \mathcal{B}_V^{(j)}\left(h_D^{(j)} \mid h_{R,\le 0}^{(j)}; C, \sigma_D\right), \\ h_A^{(j+1)} &= \mathcal{B}_A^{(j)}\left(h_A^{(j)} \mid h_D^{(j)}; C, \sigma_D, \sigma_A\right). \end{aligned} \tag{3}$$

Here $\mathcal{B}_V^{(j)}$ is the visual block shared by the RGB and depth streams, and $\mathcal{B}_A^{(j)}$ is the independently parameterized action block. The visual streams retain modality-specific input, conditioning, and output projections. After the final layer, visual heads project tokens back to latent grids to predict RGB and depth flow velocities, while the action head predicts a velocity vector for each step in normalized action space. These outputs are supervised by the joint flow-matching objective below.

Training Details. We initialize the depth VAE from the pretrained Wan VAE and pretrain it on 2,300 hours of depth data (see Appendix A.1.3 for the data mixture, losses, and optimization settings); both VAEs and T5 remain frozen during policy training. Policy training follows two stages. First, we adapt the pretrained Wan backbone to depth space by predicting depth sequences conditioned on current depth and language, using depth-prediction loss alone. We then jointly optimize the RGB and depth streams, action tower, and conditioning projections with sparse RGB goal supervision, dense intermediate depth supervision, and action supervision, connecting semantic planning, physical rollout, and executable action generation.

For joint training, let $R_{\le 0}, R_F$ and $D_0, D_{1:F}$ denote the encoded RGB and depth observations and targets, and $A_{1:H}$ the normalized action chunk; visual subscripts identify the source frames rather than the number of temporally compressed latent frames. For targets $(X_R, X_D, X_A) = (R_F, D_{1:F}, A_{1:H})$, we independently sample Gaussian noise ϵ_m and noise levels $\sigma_m \in [0, 1]$ across modalities:

$$\widetilde{X}_m = (1 - \sigma_m) X_m + \sigma_m \epsilon_m, \qquad v_m = \epsilon_m - X_m, \qquad \epsilon_m \sim \mathcal{N}(0, I). \tag{4}$$

Conditioned on $R_{\le 0}, D_0, C$, the model predicts velocities $\widehat{v}_m$ and minimizes the joint flow-matching objective (Lipman et al., 2023):

$$\mathcal{L} = \mathcal{L}_R + \mathcal{L}_D + \mathcal{L}_A, \qquad \mathcal{L}_m = \mathbb{E}[w(\sigma_m)\,\mathrm{MSE}(\widehat{v}_m, v_m)]\,. \tag{5}$$

Here w weights noise levels, and the expectation is over training examples, noise, and noise levels. MSE is averaged over valid prediction targets, excluding observations and padding. The noise schedule and weighting function are specified in Appendix A.1.1; benchmark-specific budgets, data mixtures, and optimizer settings are collected in table 7.

Inference. At inference (Figure 2, lower right), the RGB stream encodes $r_{\le 0}$ once and caches its features for semantic guidance, without generating future RGB frames. Future depth latents and the normalized action chunk are initialized from Gaussian noise and jointly denoised, with the observed RGB, current depth, and task context held fixed. The resulting depth latents can be decoded into a depth sequence for visualization, but action generation uses intermediate rollout features and does not require decoding depth maps. We denormalize the predicted action chunk, execute a prefix, and replan from the next observation.

4 EXPERIMENT

Without embodied action pretraining, we evaluate PrismWAM across six simulation benchmarks spanning both in-distribution manipulation and out-of-distribution transfer (§4.2). We further examine how our role-decoupled representations support control through robustness analyses, controlled ablations, and discussion (§4.3), and validate the approach in real-world manipulation (§4.4). Additional implementation and evaluation details are provided in Appendices A.1 and A.2.

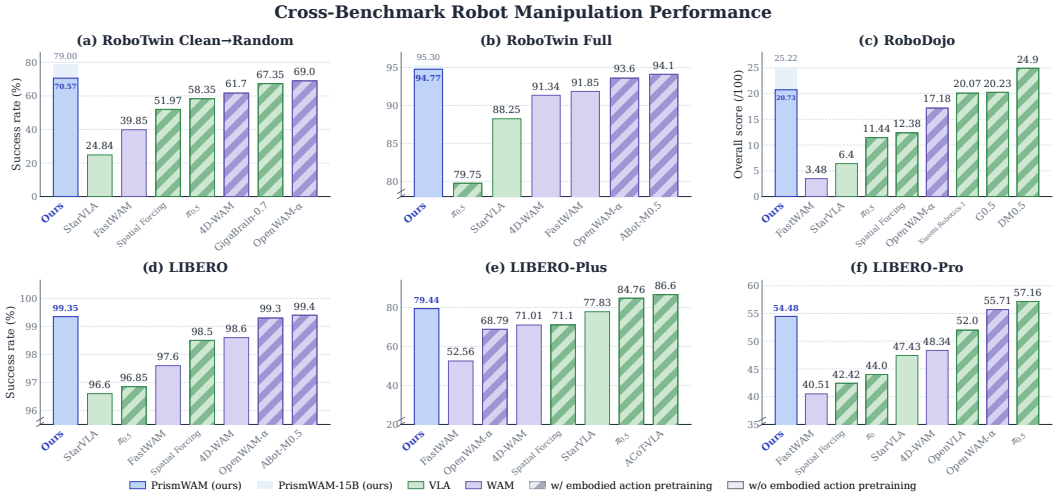


Figure 3: **Cross-benchmark performance.** Across six manipulation benchmarks, PrismWAM consistently outperforms all compared methods without embodied action pretraining and remains competitive with VLAs and WAMs using large-scale embodied action pretraining.

4.1 EXPERIMENT SETUP

Benchmarks. We evaluate in-distribution manipulation and out-of-distribution transfer across six simulation benchmarks: RoboTwin clean-to-random and RoboTwin-Full (Chen et al., 2025), LIBERO (Liu et al., 2023), LIBERO-Plus (Fei et al., 2025), LIBERO-Pro (Zhou et al., 2025), and RoboDojo (Chen et al., 2026b). RoboTwin-Full and LIBERO assess performance under their respective mixed-domain and standard protocols. RoboTwin clean-to-random, LIBERO-Plus, and LIBERO-Pro probe transfer across diverse visual, spatial, semantic, task, and environmental shifts, while RoboDojo evaluates manipulation capabilities using both task scores and success rates.

Baselines. We compare PrismWAM with representative cross-benchmark VLAs and WAMs, including methods trained without embodied action pretraining (starVLA (StarVLA Community, 2026), FastWAM (Yuan et al., 2026)), methods that use embodied action pretraining ($\pi_{0.5}$ (Physical Intelligence et al., 2025), OpenWAM-α (Wang et al., 2026a)), and approaches that employ physics-aware or spatially aware pretraining or fine-tuning (X-WAM (Guo et al., 2026), Spatial-Forcing (Li et al., 2025b), 4D-WAM (Yang et al., 2026)). Hatched bars in fig. 3 denote pretrained models and methods fine-tuned from them. We additionally include top-performing benchmark-specific methods (GigaBrain Team et al., 2026; Chen et al., 2026a; Zhong et al., 2026; Kim et al., 2024; Black et al., 2024; Liu et al., 2026; Xiaomi Robotics Team et al., 2026; Dexmal Team, 2026).

4.2 MAIN RESULTS

Across six benchmarks, PrismWAM *substantially outperforms all compared VLAs and WAMs without embodied action pretraining*. On RoboTwin-C2R, PrismWAM achieves 53.16% success in randomized scenes, compared with starVLA at 3.16% (+50.00 pp) and FastWAM at 1.90% (+51.26 pp). These results demonstrate the strong *zero-shot generalization* capability of PrismWAM without embodied action pretraining. PrismWAM further outperforms the available geometry-aware baselines across RoboTwin-Full and LIBERO, reaching 94.77% on RoboTwin-Full versus 91.34% for

Table 1: **RoboTwin and RoboDojo performance.** RoboTwin: success (%); RoboDojo: Score (/100) / SR (%). Bold: best; underline: second best.

Method	*RoboTwin-C2R*			*RoboTwin-Full*			*RoboDojo*					
	Clean	*Rand.*	*Avg.*	*Clean*	*Rand.*	*Avg.*	*Gen.*	*Precision*	*Long-H.*	*Memory*	*Open*	*Overall*
w/ Embodied Action Pretraining												
$\pi_{0.5}$	70.70	46.00	58.35	82.70	76.80	79.75	13.38/8.17	12.40/5.50	23.54/14.67	5.89/4.67	1.98/1.67	11.44/6.93
OpenWAM-α	89.40	48.70	69.00	93.74	93.46	93.60	20.71/14.83	18.45/9.25	34.93/**25.33**	10.41/**9.11**	1.41/1.08	17.18/11.92
w/o Embodied Action Pretraining												
starVLA	46.52	3.16	24.84	88.18	88.32	88.25	3.94/2.34	9.90/4.33	14.15/6.50	3.34/2.44	0.67/0.58	6.40/3.24
FastWAM	77.80	1.90	39.85	91.90	91.80	91.85	2.34/1.11	1.96/0.00	9.14/5.17	3.55/3.44	0.42/0.42	3.48/2.03
PrismWAM (ours)	87.98	53.16	70.57	94.74	94.80	94.77	22.58/15.94	30.37/21.33	38.97/25.25	**10.45**/9.00	1.30/0.92	20.73/14.48
PrismWAM-15B (ours)	**93.00**	**65.00**	**79.00**	**94.90**	**95.70**	**95.30**	**31.26/23.56**	**38.44/30.25**	**44.05**/25.17	10.01/8.89	**2.35/1.92**	**25.22/17.96**
w/ Explicit Spatial Information												
X-WAM	70.00	25.80	47.90	89.80	90.70	90.25	7.39/3.33	6.72/1.83	17.47/9.08	6.32/4.67	0.57/0.25	7.69/3.83
Spatial-Forcing	77.20	26.74	51.97	–	–	–	14.12/9.34	17.32/10.58	23.26/14.58	5.43/4.11	1.78/1.58	12.38/8.04
4D-WAM	81.50	41.80	61.65	91.92	90.76	91.34	13.35/8.83	18.08/10.00	23.30/14.00	7.38/6.00	1.44/1.00	12.71/7.97

Table 2: **LIBERO performance and transfer.** Success rates (%).

Method	*LIBERO*					*LIBERO-Plus*								*LIBERO-Pro*					
	Spatial	*Object*	*Goal*	*Long*	*Overall*	*Camera*	*Robot*	*Lang.*	*Light*	*Bg.*	*Noise*	*Layout*	*Overall*	*Object*	*Pos.*	*Sem.*	*Task*	*Env.*	*Overall*
w/ Embodied Action Pretraining																			
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.85	71.23	**76.06**	85.82	97.72	96.19	**87.32**	**86.23**	**84.76**	85.40	**28.15**	95.50	**25.30**	51.45	**57.16**
OpenWAM-α	**99.6**	99.6	**99.8**	98.2	99.3	33.21	75.87	87.57	97.02	85.59	39.85	77.38	68.79	84.45	26.10	95.95	25.25	46.80	55.71
w/o Embodied Action Pretraining																			
starVLA	97.8	98.6	96.2	93.8	96.6	64.04	57.16	85.75	93.78	96.00	80.95	77.25	77.83	76.65	2.80	94.85	17.80	45.05	47.43
FastWAM	98.2	**100.0**	97.0	95.2	97.6	16.01	46.26	72.93	81.96	55.48	39.41	66.49	52.56	73.00	0.05	82.90	14.90	31.70	40.51
PrismWAM (ours)	99.4	**100.0**	98.2	**99.8**	**99.35**	60.60	67.61	**92.13**	**99.65**	**99.54**	67.46	81.70	79.44	**88.65**	6.55	**98.55**	23.45	**55.20**	54.48
w/ Explicit Spatial Information																			
X-WAM	96.8	98.6	95.4	95.6	96.60	**72.92**	46.32	88.48	93.61	88.29	62.90	80.79	74.80	83.95	21.20	97.25	17.70	45.55	53.13
Spatial-Forcing	99.4	99.6	98.8	96.0	98.5	56.85	37.94	84.39	95.36	91.82	65.90	79.02	71.10	73.95	2.05	94.45	9.80	31.85	42.42
4D-WAM	98.8	99.5	98.6	97.4	98.6	45.15	64.26	90.63	94.29	57.71	69.08	79.21	71.01	80.75	3.90	95.85	21.10	40.10	48.34

the strongest compared geometry-aware baseline, 4D-WAM. These results suggest that *the key to effectively leveraging physical information is not only to incorporate it, but also to decouple it from semantic representations*. Moreover, PrismWAM *surpasses both pretrained and non-pretrained WAM baselines* in most settings, achieving 70.57% clean/random average success on RoboTwin-C2R compared with 69.00% for pretrained OpenWAM-α. Beyond WAM comparisons, PrismWAM remains competitive with large-scale pretrained policies (table 4). Results on RoboDojo also suggest our *data-efficiency*, achieving strong control without additional embodied backbone pretraining.

4.3 Analysis and Discussion

RQ1: Does physical-state prediction offer more than appearance robustness? Depth encodes distance rather than texture or illumination, making it naturally insensitive to changes in color and lighting. This raises a key question: does physical-state prediction offer more than appearance robustness? Results across multiple benchmarks point to broader benefits for generalization and precise control. PrismWAM achieves 81.70% under layout perturbations on LIBERO-Plus, compared with 77.38% for OpenWAM-α and 66.49% for FastWAM; on LIBERO-Pro, it achieves 88.65% when object attributes such as size, shape, and color change, exceeding OpenWAM-α at 84.45% and FastWAM at 73.00% (table 2). On RoboDojo's challenging Precision tasks, PrismWAM substantially outperforms pretrained OpenWAM-α, achieving 21.33% success versus 9.25% (table 1). These results suggest that physical-state prediction goes beyond appearance robustness: *it supports broader generalization, as illustrated by robustness to changes in scene layout and object shape, and exposes physical structure for precise control*.

RQ2: Can PrismWAM remain reliable without ideal depth observations? Real-world depth sensors may produce measurement noise and missing values, motivating evaluation beyond ideal depth inputs. We therefore construct sensor-like corruptions guided by manufacturer-reported error characteristics of commercial depth sensors, with full settings provided in section A.6. As shown in fig. 4, we train and evaluate PrismWAM on RoboTwin-C2R across three corruption levels constructed from *Metric Jitter*, *Edge Missing*, and *Patch Hole*. On randomized scenes, success consistently decreases as corruption becomes stronger. In the clean setting, light corruption yields a small improvement before performance declines at higher severity. Nevertheless, even at the strongest corruption level, the average performance drops by only approximately 2 percentage points. These results demonstrate that PrismWAM remains robust to diverse forms of depth corruption.

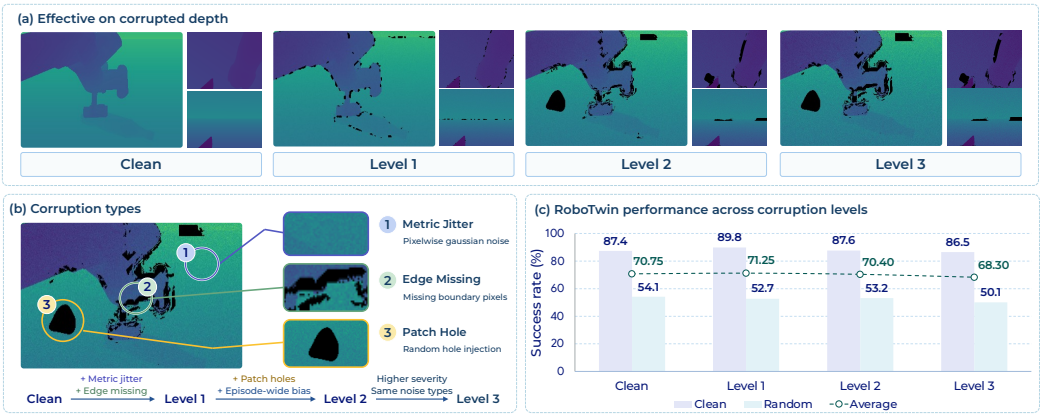


Figure 4: **Robustness to progressive depth corruption. Top:** Depth observations across corruption levels. **Bottom left:** Corruption types and their level-wise composition. **Bottom right:** RoboTwin performance under progressively increasing corruption of the original depth observations.

Table 3: Component ablations.

Variant	Clean	Random	Avg.
PrismWAM (full)	87.40	54.10	70.75
Representation composition			
w/o RGB stream	79.10	40.10	59.60
w/o depth stream	83.00	2.40	42.70
Representation learning			
w/o future-RGB supervision	81.60	51.40	66.50
Single-stage training	89.30	24.70	57.00

Table 4: Embodied pretraining and RoboDojo performance. Hours of additional embodied backbone pretraining data; SR in %.

Method	Hours	Score	SR
OpenWAM-α	$\approx$6,400	17.18	11.92
$\pi_{0.5}$	10,000+	11.44	6.93
Xiaomi-Robotics-1	100,000+	20.07	13.93
PrismWAM	0	20.73	14.48
PrismWAM-15B	0	25.22	17.96

RQ3: Do PrismWAM's design choices work together effectively? We examine the contributions of *representation composition* and *representation learning* through ablations on RoboTwin-C2R (table 3). Removing either the RGB or depth stream reduces both clean-scene success and transfer, supporting the complementary roles of semantic context and physical state. Removing future-RGB supervision or replacing the two-stage recipe with single-stage training also weakens transfer, supporting predictive semantic learning and the staged training strategy. Together, these results support *representation decoupling through complementary streams and targeted training*.

4.4 Real-world Validation

Tasks and evaluation conditions. We compare PrismWAM with OpenWAM (Wang et al., 2026a) on three real-robot tasks: *Pick Up Plate*, *Ranking Block Size*, and *Open Microwave*. These tasks cover bimanual plate handling, single-arm size-dependent ordering, and multistep bimanual retrieval, respectively. In-distribution (ID) trials use the original setups and task-specific initial arrangements. Out-of-distribution (OOD) trials vary illumination, distractors, object appearance or placement, and the tablecloth while preserving each task's goal. Figure 5 and table 5 summarize the settings and results. Task instructions, training recipes, are detailed in Appendix A.3.

PrismWAM achieves pooled success of 88.9% in ID and 68.6% in OOD, compared with 74.1% and 48.6% for OpenWAM, improving success by 14.8 and 20.0 percentage points, respectively. OOD gains hold across all three tasks: 86.7% versus 66.7% on Pick Up Plate, 60.0% versus 30.0% on Ranking Block Size, and 50.0% versus 40.0% on Open Microwave. In ID, PrismWAM improves plate handling and size ordering, while OpenWAM performs better on microwave retrieval (90.0% versus 80.0%). These results extend the scene-change robustness observed in simulation to physical manipulation, supporting the practical value of combining RGB semantic guidance with depth-based physical rollouts.

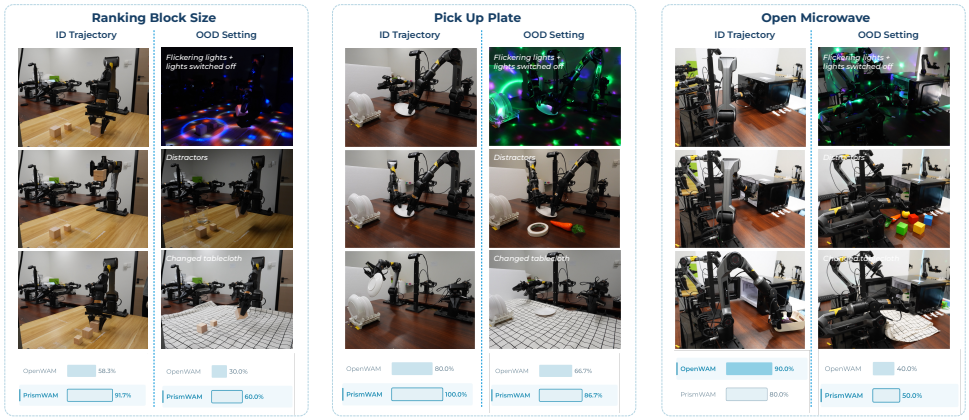


Figure 5: **Real-world evaluation.** ID trajectories (left), representative OOD scenes (right), and OpenWAM versus PrismWAM success rates (bottom) across three tasks.

Table 5: **Real-world success under ID and OOD conditions.** Entries are successes/trials (%). Both models are evaluated on the same three tasks and trial counts. Pooled rates divide total successes by total trials across tasks. Bold marks the higher success rate within each task and condition.

	OpenWAM		**PrismWAM**	
Task	ID	OOD	ID	OOD
Pick Up Plate	4/5 (80.0)	10/15 (66.7)	**5/5 (100.0)**	**13/15 (86.7)**
Ranking Block Size	7/12 (58.3)	3/10 (30.0)	**11/12 (91.7)**	**6/10 (60.0)**
Open Microwave	**9/10 (90.0)**	4/10 (40.0)	8/10 (80.0)	**5/10 (50.0)**
Pooled (3 tasks)	20/27 (74.1)	17/35 (48.6)	**24/27 (88.9)**	**24/35 (68.6)**

5 Conclusion

We introduce **PrismWAM**, a World Action Model built on the principle of decoupling semantic context from physical state. Rather than treating video latents as a monolithic world representation, PrismWAM decomposes world-action modeling into Semantic Planning $\rightarrow$ Physical Rollout $\rightarrow$ Executable Action: RGB future-frame prediction captures task-level intent, depth models physical state transitions, and the resulting rollout is mapped directly to continuous actions without RGB goal prediction at inference. Across diverse simulation and real-world manipulation benchmarks, PrismWAM achieves strong performance and generalization with only 2,300 hours of depth data and without embodied action pretraining, while matching or surpassing large-scale action-pretrained models on challenging settings. These results suggest that scaling action data is not the only path to stronger embodied foundation models; principled state representations and architectural design offer a more data-efficient alternative. More broadly, PrismWAM paves the way for a new generation of WAMs that shifts the focus from modeling *how the world looks* to *how the world responds to action*. While depth is one practical realization of this principle, it can extend to occupancy, flow, tactile sensing, proprioception, and other representations capturing action-induced state transitions.

Limitations and future work. At present, large-scale real-world manipulation data with synchronized depth observations remains relatively limited, which constrains the scale of depth pretraining compared with existing action-pretrained models. We view this primarily as a reflection of the current data landscape: as depth sensing becomes more accessible and embodied datasets incorporate richer physical modalities, such data is likely to become increasingly available. Physical-state pretraining may therefore emerge as a natural scaling direction for World Action Models, with PrismWAM taking an early step toward establishing this new paradigm.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting, language polishing, and structural editing, as well as with literature search and code development. AI-generated suggestions were used only as assistance and were reviewed, revised, and verified by the authors. All technical claims, experimental results, interpretations, and final written content were validated by the authors, who take full responsibility for the work.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model, 2025. URL https://arxiv.org/abs/2512.13030.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Danny Driess, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

Ronghan Chen, Yandan Yang, Zuojin Tang, Dongjie Huo, Tong Lin, Haoning Wu, Haoyun Liu, Yuzhi Chen, Lulu Zheng, Botai Yuan, Tianlun Li, Mingxin Wang, Dekang Qi, Bin Hu, Wei Mei, Yuze Xuan, Haolong Yang, Yanqing Zhu, Mu Xu, Zhiheng Ma, and Xinyuan Chang. ABot-M0.5: Unified mobility-and-manipulation world action model, 2026a. URL https://arxiv.org/abs/2607.00678.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies, 2026b. URL https://arxiv.org/abs/2607.04434.

Tianxing Chen et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025. URL https://arxiv.org/abs/2506.18088.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Dexmal Team. DM0.5: An open-world foundation model for general-purpose embodied intelligence, July 2026. URL https://www.dexmal.com/blog/dm0.5/index_en.html.

Pengcheng Fang, Hongli Chen, and Xiaohao Cai. Privileged foresight distillation: Zero-cost future correction for world action models, 2026. URL https://arxiv.org/abs/2604.25859.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models, 2025. URL https://arxiv.org/abs/2510.13626.

GigaBrain Team et al. GigaBrain-0.7: Scaling embodied foundation models to emergent capabilities with a three-system architecture, 2026. URL https://arxiv.org/abs/2608.15875.

Ankit Goyal et al. RVT: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2306.14896.

Qiwen Gu, Jifan Li, Bingjie Gao, Rui Chen, Jing Tang, Xiangxiang Chu, and Junqiao Zhao. The right future for action: Learning action-relevant predictive states in world action models, 2026. URL https://arxiv.org/abs/2609.23369.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D world action modeling from video priors with asynchronous denoising, 2026. URL https://arxiv.org/abs/2604.26694.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. URL https://arxiv.org/abs/1803.10122.

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

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3D Diffuser Actor: Policy diffusion with 3D scene representations, 2024. URL https://arxiv.org/abs/2402.10885.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning, 2026. URL https://arxiv.org/abs/2601.16163.

Yann LeCun. A path towards autonomous machine intelligence, 2022. URL https://openreview.net/pdf?id=BZ5a1r-kVsf. Position paper, version 0.9.2, June 27, 2022.

Chengmeng Li, Junjie Wen, Yan Peng, Yaxin Peng, Feifei Feng, and Yichen Zhu. PointVLA: Injecting the 3D world into vision-language-action models, 2025a. URL https://arxiv.org/abs/2503.07511.

Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for Vision-Language-Action model, 2025b. URL https://arxiv.org/abs/2510.12276.

Kehan Li, Bohan Hou, Minghao Zhu, et al. RynnBrain 1.1: Towards more capable and generalizable embodied foundation model, 2026a. URL https://arxiv.org/abs/2607.17977.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026b. URL https://arxiv.org/abs/2601.21998.

Qixiu Li et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024. URL https://arxiv.org/abs/2411.19650.

Ying Li, Xiaobao Wei, Jiajun Cao, Hao Wang, Xiaowei Chi, Chengyu Bai, Qianpu Sun, Jiajun Li, Xiaojie Zhang, Peidong Jia, Jian Tang, Sirui Han, and Shanghang Zhang. WAM4D: Fast 4D world action model via spatial register tokens, 2026c. URL https://arxiv.org/abs/2606.14048.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Yicheng Liu et al. G0.5: One autoregressive stream for robot reasoning and action, 2026. URL https://arxiv.org/abs/2608.11739.

Javier Alejandro Lopetegui Gonzalez, Paul Pacaud, and Cordelia Schmid. Spatially aware world action model via geometric latent diffusion, 2026. URL https://arxiv.org/abs/2609.02531.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026. URL https://research.nvidia.com/labs/cosmos-lab/cosmos3/technical-report.pdf.

NVIDIA et al. World simulation with video foundation models for physical AI, 2025a. URL https://arxiv.org/abs/2511.00062.

NVIDIA et al. GR00T N1: An open foundation model for generalist humanoid robots, 2025b. URL https://arxiv.org/abs/2503.14734.

Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Lu Qiu, Yizhuo Li, Yi Chen, Yuying Ge, Yixiao Ge, and Xihui Liu. Making foresight actionable: Repurposing representation alignment in world action models, 2026. URL https://arxiv.org/abs/2606.12217.

Delin Qu et al. SpatialVLA: Exploring spatial representations for visual-language-action model, 2025. URL https://arxiv.org/abs/2501.15830.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=UPHlqbZFZB.

Jiaxin Shi, Xidong Zhang, Fucai Zhu, Zhe Li, Siyu Zhu, and Weihao Yuan. 3DThinkVLA: Endowing vision-language-action models with latent 3D priors via 3D-thinking-guided co-training, 2026. URL https://arxiv.org/abs/2606.04436.

Lucy Shi et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2022. URL https://arxiv.org/abs/2209.05451.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing, 2026. URL https://arxiv.org/abs/2604.05014.

Team Wan et al. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, Yu-Kai Wang, Yuhan Rui, Shenzhe Yao, Zhen Yuan, Zhenhao Shen, Kefei Zhu, Zijie Zhu, Ning Gao, Xiaowei Chi, Guanqi He, Shanghang Zhang, Hao Dong, Lin Shao, and Hang Zhao. OpenWAM: An open, modular exploration towards systematic world-action model pretraining, 2026a. URL https://arxiv.org/abs/2609.07398.

Ziyi Wang, Xumin Yu, Yongming Rao, et al. Hy-Embodied-VLM-1.0: Efficient physical-world agents, 2026b. URL https://arxiv.org/abs/2607.12894.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, Yibin Liu, Shuai Yang, Tianxiang Zhou, Kejia Zhang, Lei Zhou, Cheng Su, Nan Xue, Bin Tan, Han Zhang, Youchao Zhang, Fei Liao, Xing Zhu, Yujun Shen, and Kecheng Zheng. From foundation to application: Improving VLA models in practice, 2026. URL https://arxiv.org/abs/2607.06403.

Xiaomi Embodied Intelligence Team. MiMo-Embodied: X-Embodied foundation model technical report, 2025. URL https://arxiv.org/abs/2511.16518.

Xiaomi Robotics Team et al. Xiaomi-Robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories, 2026. URL https://arxiv.org/abs/2607.15330.

Gangwei Xu, Qihang Zhang, Jiaming Zhou, Xing Zhu, Yujun Shen, Xin Yang, and Yinghao Xu. Next Forcing: Causal world modeling with multi-chunk prediction, 2026. URL https://arxiv.org/abs/2606.11187.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, Qiao Sun, Pengwei Wang, Lingqiao Liu, Yan Wang, Yuxiang Gao, Feras Dayoub, and Haoang Li. 4D-WAM: Infusing spatiotemporal awareness into world action models through trajectory fields, 2026. URL https://arxiv.org/abs/2608.08023.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaWorld-Policy: An efficient action-centered world–action model, 2026a. URL https://arxiv.org/abs/2603.17240.

Seonghyeon Ye et al. World action models are zero-shot policies, 2026b. URL https://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. DepthVLA: Enhancing vision-language-action models with depth-aware spatial reasoning, 2025. URL https://arxiv.org/abs/2510.13375.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL https://arxiv.org/abs/2603.16666.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D Diffusion Policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2403.03954.

Jianjun Zhang, Jian Zhu, Taiyi Su, Chong Ma, Zitai Huang, Yi Xu, and Hanli Wang. Learning 4D geometric priors for inference-efficient world action models, 2026a. URL https://arxiv.org/abs/2607.05468.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, Jiaming Zhou, Yishu Shen, Yudong Jin, Fangyi Xu, Shuailei Ma, Jiaqi Liao, Guanxing Lu, Zifan Shi, Yongkun Wen, Yujie Zhao, Weixuan Tang, Xinyang Wang, Chaojian Li, Jiapeng Zhu, Ka Leong Cheng, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Native video-action pretraining for generalizable robot control, 2026b. URL https://arxiv.org/abs/2607.08639.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge, 2025. URL https://arxiv.org/abs/2507.04447.

Haoyu Zhao, Xingyue Zhao, Siteng Huang, Xin Li, Deli Zhao, and Zhongyu Li. RynnWorld-4D: 4D embodied world models for robotic manipulation, 2026. URL https://arxiv.org/abs/2607.06559.

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. ACoT-VLA: Action chain-of-thought for vision-language-action models, 2026. URL https://arxiv.org/abs/2601.11404.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2025. URL https://arxiv.org/abs/2510.03827.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets, 2025. URL https://arxiv.org/abs/2504.02792.

A Experimental Details

This appendix specifies the model implementation, training recipes, evaluation protocols, component ablations, and depth-corruption experiments. Appendix B provides per-task RoboTwin and RoboDojo results; the controlled component comparisons are analyzed separately in Appendix A.5.

A.1 Implementation and Training

The four main PrismWAM-5B policies follow the same Semantic Planning → Physical Rollout → Executable Action design as §3.2. All use Wan2.2-TI2V-5B initialization; tables 7 and 8 summarize their benchmark-specific recipes and shared settings. Evaluation of PrismWAM-14B is specified in Appendix A.2.1.

A.1.1 Architecture, Initialization, and Optimization

Model dimensions and conditioning. The *RGB stream* learns semantic planning representations, the *depth stream* predicts the physical rollout, and the *action tower* generates executable actions. The two visual streams have separate token sequences and modality-specific input, conditioning, and output projections, but share all 30 visual transformer blocks; the action tower has independent parameters. Table 6 gives their dimensions. Chain attention routes observed-RGB features to the depth stream and depth-rollout features to the action tower. Depth cannot attend to the future-RGB target, and action cannot attend directly to RGB. Within a visual stream, observed tokens cannot read future tokens. Frozen T5 embeddings provide language context with a maximum length of 128 tokens and width 4096; joint training adds projected current proprioception to this context. No additional VLM feature extractor or LoRA adapter is used.

Table 6: PrismWAM model dimensions. The RGB and depth streams share a visual backbone; the action tower has independent parameters. Optimization settings appear in tables 7 and 8.

Setting	RGB / depth streams	Action tower
Transformer layers	30 (shared)	30
Hidden / feed-forward width	3072 / 14336	1024 / 4096
Attention heads / head dimension	24 / 128	24 / 128
Latent channels	48	Benchmark action dimension
Visual latent patch size (T, H, W)	$(1, 2, 2)$	–
Flow-matching loss coefficient	1 per predicted modality	1

Two-stage initialization. The first stage trains a Wan-initialized depth stream for physical-rollout prediction, conditioned on current depth and language, with depth flow-matching loss only. It has no action tower or proprioceptive encoder. Joint training initializes the shared visual blocks and depth-specific modules from this depth predictor, RGB-specific modules from Wan, and the action tower from Wan-derived weights interpolated to width 1024. The action tower does not inherit the depth predictor's weights. All trainable transformer and conditioning parameters use the same learning rate. Each stage starts a fresh optimizer and schedule; the VAEs and text encoder remain frozen throughout policy training, including the corruption experiments.

Flow-matching schedule. For each predicted modality, the training scheduler independently samples $u \sim \mathcal{U}(0,1)$ and sets $\sigma = 5u/(1+4u)$. The continuous timestep is 1000σ; 1000 is the scheduler's reference discretization, not the number of inference steps. The target velocity and Gaussian interpolation are those in eqs. (4) and (5). The implemented loss weight is

$$w(\sigma) = \frac{y(\sigma) - y_{\min}}{\overline{y} - y_{\min} + 10^{-10}}, \qquad y(\sigma) = \exp[-2(\sigma - 1/2)^2], \tag{6}$$

where $y_{\min}$ and $\overline{y}$ are the minimum and mean on the 1000-point shifted grid obtained from $u \in \{1, 0.999, \ldots, 0.001\}$. Losses exclude conditioning frames and padded targets, and the retained modality losses are summed with unit coefficients. Inference uses ten Euler updates on the same shift-5 schedule. Observed RGB features are computed once and cached across these denoising updates; future RGB is not generated for control. This latent diffusion noise is separate from the sensor-depth corruptions in Appendix A.6.

A.1.2 Benchmark Recipes and Input Construction

Table 7 compares benchmark-specific settings; table 8 lists settings shared by all four policies. Main results use the final joint-training checkpoint. The details below specify transformations and boundary handling needed to interpret these tables.

Mosaic geometry and temporal boundaries. For the three-camera layout, the head image occupies 256×320 pixels above two 128×160 wrist cells. LIBERO's two views each occupy 224×224 pixels. These layouts are identical for RGB and depth and during training and evaluation. RGB history is encoded as one causal clip with two latent frames; the future RGB image is encoded separately as one latent frame. History before an episode starts repeats its first observation; episode-tail windows use repeat padding and validity masks.

Action and proprioception representations. RoboTwin reads current proprioception and future absolute targets from the same recorded joint sequence. RoboDojo uses separate measured-state and commanded-action sequences. Both concatenate left then right arm, each with six joints followed by gripper opening; neither subtracts current state from its targets. Action mean/std statistics also normalize proprioception, with a standard-deviation floor of 10^{-6}. RoboTwin-Full fits these statistics to its combined training data.

LIBERO retains the recorded end-effector commands (three translation deltas, three rotation deltas, and gripper). The raw gripper $g \in \{-1, +1\}$ maps to $(1-g)/2$ for training and is inverted for execution. Proprioception comprises end-effector position, axis-angle orientation, and two gripper-joint coordinates. Action and state use separate min–max statistics mapped to $[-1, 1]$, with matching inverse transforms at deployment.

Depth and RGB encoding. Metric depth is clipped to $[0, 1500]$ mm and linearly mapped to $[-1, 1]$ before the depth VAE; LIBERO's equivalent limit is 1.5 m. RGB is scaled to $[-1, 1]$. Views are resized bilinearly and mosaicked in the order above. The RGB VAE is the official Wan2.2 48-channel encoder; depth uses the separately trained 48-channel VAE below and its saved per-channel latent normalization statistics. Both have spatial compression 16 and temporal compression 4. Training can reuse precomputed latents from the matching frozen VAEs; corrupted depth is encoded online.

A.1.3 Depth-VAE Pretraining

We use two Wan2.2-compatible, single-channel depth VAEs with 48 latent channels: a mosaic-pretrained checkpoint for the four simulation policies and a head-view checkpoint for the real-world recipe. Both are initialized from the official RGB VAE. The encoder's RGB boundary channels and decoder's RGB output channels are averaged into their single-depth-channel counterparts; matching internal weights are loaded directly. Spatial and temporal compression factors are 16 and 4, respectively. The autoencoder is frozen during all subsequent policy training, including depth-corruption experiments.

Training corpora and mixture. Both checkpoints use processed metric depth from AgiBotWorld-Beta and simulated depth from RoboTwin. AgiBot depth is stored as uint16 millimeter PNGs inside indexed archives; the compact index permits random frame access without unpacking each

Table 7: Benchmark-specific training and deployment settings. C2R denotes clean-to-random transfer. Stage lengths and warmup count optimizer updates. Inputs describe joint training; depth pretraining uses current depth and language only. H denotes the action horizon and K the executed prefix before replanning. Shared settings appear in table 8.

Setting	**RoboTwin C2R**	**RoboTwin Full**	**LIBERO**	**RoboDojo**
Training data				
Training data	Clean	Clean + randomized	Spatial, Object, Goal, Long	Simulation, ARX X5
Number of tasks	50	50	40	35
Demonstrations per task	50 clean	50 clean + 500 randomized	50	100
Total demonstrations	2,500	27,500	2,000	3,500
Training language	Seen instructions	Seen instructions	Task instructions	Recorded instructions
Training budget				
Updates: depth / joint	15k / 15k	15k / 15k	15k / 15k	15k / 40k
Global batch	256	1024	256	1024
GPUs $\times$ clips/GPU	32×8	32×32	8×32	32×32
Warmup: depth / joint	750 / 750	750 / 750	750 / 750	750 / 2000
Observations and prediction targets				
Camera layout	Three-view	Three-view	Two-view	Three-view
Mosaic size $(H \times W)$	384×320	384×320	224×448	384×320
Depth clip: frames / stride	9 / 4	9 / 4	5 / 4	9 / 4
Future-RGB target	$t + 32$	$t + 32$	$t + 16$	$t + 32$
Action representation	14-D absolute joints	14-D absolute joints	7-D end-effector deltas	14-D absolute joints
Proprioceptive dimension	14	14	8	14
Target action indices	$[t + 1, t + 32]$	$[t + 1, t + 32]$	$[t, t + 15]$	$[t + 1, t + 32]$
Action / state statistics	Shared mean/std	Shared mean/std	Separate min–max	Shared mean/std
Closed-loop inference				
Horizon / replan (H/K)	32 / 24	32 / 24	16 / 16	32 / 24

Three-view mosaics place the head view above the left/right wrist views; two-view mosaics place the external view on the left and the wrist view on the right. RoboTwin-Full samples each task's combined window pool without enforcing equal clean/random probabilities. Depth clips include the current conditioning frame. Strides refer to visual frames; actions retain consecutive dataset steps. LIBERO-Plus and LIBERO-Pro evaluate the standard-LIBERO checkpoint without further training.

episode. Its index-build report records 217 tasks, 161,566 episodes, and 285,156,280 indexed frames. RoboTwin uses all 50 tasks with both clean and randomized demonstrations, totaling 27,500 episodes. These are corpus-inventory counts, not independently sampled clips or additional hours obtained by counting overlapping windows.

Sampling first chooses AgiBot or RoboTwin with probabilities 0.75 and 0.25, then samples an eligible window from that source, with replacement. The source probabilities do not depend on the relative numbers of windows and do not force an exact 3:1 ratio in every minibatch. No InternA1, RoboMIND, or demonstrations from our downstream real-world tasks are included in these two VAE checkpoints. At global batch size 256, 50k updates correspond to 12.8 million clip presentations, with expected source exposures of 9.6 million AgiBot and 3.2 million RoboTwin clips; these are not unique-episode counts.

Input construction and augmentation. Each clip contains nine frames at offsets $[0, 4, \ldots, 32]$ from a valid start. Depth is clipped to 0–1500 mm and mapped linearly to $[-1, 1]$. The mosaic model uses 384×320 inputs: a 256×320 head view above two 128×160 wrist cells. RoboTwin supplies all three depth views. AgiBot supplies head depth only; its two wrist cells are filled with 1500 mm, or $+1$ after normalization. These fill values are included in reconstruction, rather than masked out. The head-view model instead resizes only the external/head image to 256×320, with no wrist cells.

Training chooses random complete temporal clips, samples one rotation uniformly from $[-5°, 5°]$, and applies a horizontal flip with probability 0.5. The spatial parameters are shared across time and, for mosaics, across views; transforms are applied within each view before assembly, without swapping camera slots. No synthetic range noise or edge holes are added during this VAE pretraining.

Optimization and latent normalization. Both runs use 32 GPUs with batch size eight per GPU, bf16, and AdamW with $\beta = (0.9, 0.95)$, weight decay 0.01, and gradient clipping at 1.0. Linear warmup is followed by cosine decay over the horizon in table 9. The objective is

$$\mathcal{L}_{\text{VAE}} = \|\hat{d} - d\|_1 + \|\hat{d} - d\|_2^2 + 10^{-6}\, D_{\text{KL}}(q(z \mid d) \,\|\, \mathcal{N}(0, I)), \tag{7}$$

with mean reductions for the reconstruction terms and stochastic posterior sampling. Reconstruction has no additional valid-depth mask. Per-channel latent means and standard deviations are computed from encoder posterior means and stored with each depth checkpoint for downstream normalization. The mosaic run resumes its original 50k schedule from a 20k checkpoint; its update budget is 50k in total. Depth-VAE pretraining is distinct from the policy's depth-only adaptation stage. In particular, clean-only policy training does not imply that the frozen depth encoder has never seen randomized RoboTwin depth.

Checkpoint evaluation. Standard LIBERO, Plus, and Pro evaluate the same final checkpoint, with no training on Plus or Pro. RoboTwin evaluation uses unseen instructions and the fully randomized scene distribution for Random. The PrismWAM-5B main-table RoboTwin evaluations use 100 episodes per task and condition; the ablation and depth-corruption comparisons use 20 episodes, as detailed in Appendices A.5 and A.6.

A.2 Benchmarks and Evaluation Protocols

RoboTwin: clean-to-random transfer. We evaluate the clean-only 15k+15k PrismWAM-5B checkpoint on 50 RoboTwin tasks (Chen et al., 2025), with 100 episodes per task in each condition. Clean and randomized evaluations yield 4399/5000 (87.98%) and 2658/5000 (53.16%) successes, respectively, averaging 70.57%.

RoboTwin-Full: mixed training. The PrismWAM-5B policy in this setting uses the same task list but includes clean and randomized demonstrations during training. The completed evaluation uses 50 tasks $\times$ 100 episodes per condition, yielding 4737/5000 clean (94.74%) and 4740/5000 randomized (94.80%) successes, averaging 94.77%. This completed sweep is used in both the main comparison and the per-task appendix.

Table 8: Shared settings for the four main PrismWAM policies in table 7. Each stage starts a fresh optimizer and learning-rate schedule.

Setting	Value
Optimization	
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$
Peak / minimum LR	10^{-4} / 10^{-6}
LR schedule	5% linear warmup, then cosine decay; reset each stage
Weight decay / grad. clip	0.01 / 1.0
Precision / accumulation	bf16 / 1
Memory optimization	DeepSpeed ZeRO-2; gradient checkpointing
Save interval / seed	1000 updates / 42
Losses: depth / joint	Depth / RGB + depth + action; unit weight per loss
Frozen modules	RGB VAE, depth VAE, and T5 text encoder
Sampling and visual context	
Sampling	Uniform over tasks; uniform over windows within each task
RGB history	5 frames, stride 4: $t + [-16, -12, -8, -4, 0]$
Depth range	0–1500 mm
Depth corruption	None in main benchmark training
Inference	
Euler steps / flow shift	10 / 5
RGB computation	Observed RGB is cached; no future RGB generation

Table 9: Depth-VAE training settings. Both selected checkpoints are at 50k updates. The real-world head-view checkpoint is taken partway through a 200k schedule; it does not use the mosaic model's 50k decay horizon.

Setting	Simulation VAE	Real-world VAE
Input layout	Three-cell mosaic	Head view
Input size $(H \times W)$	384×320	256×320
Latent shape (C, T, H, W)	$48 \times 3 \times 24 \times 20$	$48 \times 3 \times 16 \times 20$
AgiBot / RoboTwin probability	0.75 / 0.25	0.75 / 0.25
Global batch / accumulation	256 / 1	256 / 1
Selected checkpoint	50k	50k
Configured schedule horizon	50k	200k
Warmup updates	5000	5000
Peak / terminal learning rate	10^{-4} / 10^{-5}	10^{-4} / 10^{-5}

Table 10: PrismWAM on LIBERO Standard and LIBERO-Plus, grouped by base suite. Plus additionally corrupts external-view depth in its Sensor Noise subset; all other categories use the standard Plus protocol. Overall rates pool episodes.

	LIBERO Standard		LIBERO-Plus	
Suite	**Success / Total**	**SR (%)**	**Success / Total**	**SR (%)**
Spatial	497/500	99.40	2075/2402	86.39
Object	500/500	100.00	2127/2518	84.47
Goal	491/500	98.20	1830/2591	70.63
Long	499/500	99.80	1936/2519	76.86
Overall	**1987/2000**	**99.35**	**7968/10030**	**79.44**

LIBERO Standard and LIBERO-Plus. LIBERO (Liu et al., 2023) covers Spatial, Object, Goal, and Long; LIBERO-Plus (Fei et al., 2025) tests the same standard-trained PrismWAM policy with a 16-step action horizon under seven perturbation categories without further training. Standard evaluation contains 2000 episodes (500 per suite) and yields 1987 successes (99.35%). For Plus, the Sensor Noise subset is replaced with 1601 episodes using matched agent-view depth corruption, while the other categories retain the standard Plus protocol. The resulting composite is 7968/10030 (79.44%); the Sensor Noise subset is 1080/1601 (67.46%). Overall rates pool successful and evaluated episodes, rather than averaging rounded category percentages. Table 10 reports the four base suites; table 11 reports the seven perturbation categories. Both Plus breakdowns use the same composite evaluation and sum to 7968 successful episodes. The matched-noise records replace the original Sensor Noise records by scenario identifier, so no scenario is counted twice.

Table 11: PrismWAM on LIBERO-Plus by perturbation category. Sensor Noise uses matched external-view RGB/depth corruption; the other six categories retain their standard protocol. Overall pools all 10030 scenarios, rather than averaging the seven percentages.

Perturbation	Success / Total	SR (%)
Camera viewpoints	969/1599	60.60
Robot initial states	1048/1550	67.61
Language instructions	1416/1537	92.13
Light conditions	1138/1142	99.65
Background textures	1071/1076	99.54
Sensor noise (matched RGB/depth)	1080/1601	67.46
Object layout	1246/1525	81.70
Overall	**7968/10030**	**79.44**

Success remains above 99% under lighting and background changes, while camera viewpoints (60.60%) and robot initial states (67.61%) are more challenging. Matched Sensor Noise yields 67.46%. This image-domain stress test differs from the metric depth-error model in Appendix A.6; its result should not be interpreted as a calibrated sensor-noise tolerance.

LIBERO-Plus matched RGB/depth noise. The native Sensor Noise protocol corrupts external-view RGB. Our extension also corrupts the corresponding depth observation with the *same operator, severity, and random draw*; wrist RGB and depth remain unchanged. Table 12 defines the five operator families. The extension runs on reset observations and subsequent environment-step observations, before policy preprocessing, without retraining the LIBERO policy.

Let I be the external RGB image, $z \in [0, 1]$ the simulator's normalized depth buffer, and $C_{k,s}(\cdot; \omega)$ the released corruption operator with family k, severity s, and random draw ω. We first quantize depth as $u = \lfloor 255z \rfloor$, embed it into a three-channel image $E(u)$, and compute

$$I' = C_{k,s}(I; \omega), \qquad z' = \frac{\mathrm{ch}_0[C_{k,s}(E(u); \omega)]}{255}. \tag{8}$$

For most operators, $E(u) = (u, u, u)$. Motion blur uses $(u, u \oplus 255, u \oplus 85)$, where $\oplus$ is bitwise XOR, to prevent the image library from collapsing equal channels into a different grayscale processing path. The transform acts independently on channels, and only channel zero is retained as depth. The extra channels do not become policy inputs.

To reuse the random draw, the evaluator saves the NumPy RNG state before corrupting RGB, restores that state before corrupting depth, and finally restores the post-RGB state. Depth processing therefore consumes no additional draws from the subsequent environment random sequence. Each scenario fixes the family and severity; stochastic parameters are resampled on each reset/step observation. The policy then performs its usual nonlinear buffer-to-metric conversion,

$$d' = \frac{nf}{f - (f - n)z'}, \qquad d_{\mathrm{VAE}} = 2\frac{\mathrm{clip}(d', 0, 1.5\,\mathrm{m})}{1.5\,\mathrm{m}} - 1, \tag{9}$$

where n and f are the simulator's near and far clipping distances in meters. The metric depth follows the same camera orientation, resizing, and horizontal mosaicking as uncorrupted observations. The corruption is applied to z, not directly to metric d: severity is inherited from the image-corruption operator and does not denote a fixed error in millimeters. The evaluated transformation includes the initial 8-bit quantization.

Table 12: Operators used for matched external-view RGB/depth corruption. The ten severity presets for each family are those of the released LIBERO-Plus implementation. Matching reuses the stochastic parameters, not an equal RGB and metric-depth error.

Operator	Noise IDs	Operation on the depth-buffer image
Motion blur	1–10	Directional blur using the same sampled direction.
Gaussian blur	11–20	Gaussian smoothing with the same severity preset.
Zoom blur	21–30	Blend the original and zoomed images.
Fog	31–40	Apply the same sampled spatial fog field.
Glass blur	41–50	Local pixel displacements and blur with matched random draws.

This is an image-domain stress test, distinct from the metric Gaussian range-noise and edge-hole model in Appendix A.6. Its 1601 completed scenarios replace the original Sensor Noise subset, while the other six categories retain their original results. For the same checkpoint, Sensor Noise changes from 1520/1601 (94.94%) with RGB-only corruption to 1080/1601 (67.46%) with matched depth corruption; the composite total is $8408 - 1520 + 1080 = 7968$ successes over 10030 scenarios.

LIBERO-Pro. We evaluate the same standard-LIBERO-trained PrismWAM checkpoint with a 16-step action horizon, without additional Pro training. LIBERO-Pro (Zhou et al., 2025) separates Object, Position, Semantic, Task, and Environment perturbations across the four base suites. The evaluation covers 200 tasks $\times$ 50 episodes, with 2000 episodes per perturbation type.

Environment-instance differences must be considered when comparing Environment and five-category overall scores, as noted in table 2. Our completed result is 5448/10000 (54.48%): Object 88.65%, Position 6.55%, Semantic 98.55%, Task 23.45%, and Environment 55.20%. X-WAM's completed evaluation yields 5313/10000 (53.13%). Its category- and suite-level breakdown is reported in table 13; consistent with the main comparison, the locally generated and frozen Environment instances make its Environment and Overall results reference comparisons rather than identical-test-set leaderboard scores.

Baseline evaluation settings. The four core Plus baseline evaluations use 10030 native scenarios, MuJoCo 3.3.2, robosuite 1.4.0, and seed 10000, without Plus fine-tuning. Completed counts are $\pi_{0.5}$ 8501/10030, starVLA-PI 7806/10030, OpenWAM-α 6900/10030, and FastWAM 5272/10030. PrismWAM yields 7968/10030 with the modified Sensor Noise protocol described above. All five evaluations are complete; model-specific action horizons and rollout settings are retained.

The six core Pro evaluations each cover 200 tasks with 50 episodes per task. Completed counts are $\pi_{0.5}$ 5716/10000, OpenWAM-α 5571/10000, PrismWAM 5448/10000, X-WAM 5313/10000, starVLA-PI 4743/10000, and FastWAM 4051/10000. The X-WAM run uses its 30k checkpoint

Table 13: X-WAM on LIBERO-Pro. Success rates are shown by perturbation category and base suite for the completed 10000-episode evaluation.

Breakdown	Success / Total	SR (%)
Perturbation category		
Object	1679/2000	83.95
Position	424/2000	21.20
Semantic	1945/2000	97.25
Task	354/2000	17.70
Environment	911/2000	45.55
Base suite		
Spatial	1529/2500	61.16
Object	1425/2500	57.00
Goal	1081/2500	43.24
Long	1278/2500	51.12
Overall	**5313/10000**	**53.13**

with seed 42, action horizon 32, replanning interval 24, and 10 action-denoising steps. Each model retains its evaluation settings; these comparisons do not isolate the effect of modality alone.

RoboDojo. RoboDojo (Chen et al., 2026b) contains 42 canonical simulation tasks across Generalization, Memory, Precision, Long-Horizon, and Open capabilities, as well as 18 real-world tasks. Our simulation sweep covers all 42 canonical tasks as 54 standard/randomized evaluation entries per seed, totaling 2100 episodes per seed. Generalization uses 25 episodes per standard/randomized entry; the other tasks use 50 episodes each. The 15k+40k, global-batch-1024 PrismWAM-5B checkpoint is evaluated with seeds 0, 1, and 2 and simulator-depth inputs. These results exclude RoboDojo's 18 real-world tasks; our separate real-robot evaluation is described in Appendix A.3.

We use the official category-balanced aggregation: Generalization standard/randomized splits are combined within canonical tasks before the five capability categories are averaged equally. The three-seed mean is 20.73/100 progress score and 14.48% success. The separate episode-pooled rate, 938/6300 = 14.89%, is not substituted for this category-balanced macro-average. Per-task and category results are averaged across seeds before reporting.

A.2.1 Evaluation of the 14B Model

Table 14 summarizes the three PrismWAM-14B results shown in fig. 3. RoboTwin C2R evaluates a clean-only policy on Clean and fully randomized scenes; RoboTwin-Full evaluates a policy trained with both clean and randomized demonstrations. Each setting uses 50 tasks with 20 episodes per task and condition, giving 1000 Clean and 1000 Random episodes. Evaluation uses unseen instructions. The reported result is the arithmetic mean of the Clean and Random success rates.

For RoboDojo, each of three evaluation seeds covers the complete simulation suite in 2100 episodes, totaling 6300 episodes. For each seed, we compute the official progress score by combining the standard/randomized Generalization results within canonical tasks and then averaging the five capability categories equally. The reported 25.22/100 is the arithmetic mean of these three seed-level progress scores.

Table 14: PrismWAM-14B results and evaluation protocols for fig. 3. RoboTwin results average Clean and Random success rates; RoboDojo reports the three-seed mean of the official category-balanced progress score.

Benchmark	**Metric**	**Result**	**Evaluation episodes**
RoboTwin C2R	Mean success (%)	79.00	1000 Clean; 1000 Random
RoboTwin-Full	Mean success (%)	95.30	1000 Clean; 1000 Random
RoboDojo	Progress score (/100)	25.22	$3 \times 2100 = 6300$

A.3 Real-World Evaluation Details

Task scope and data preparation. We evaluate three real-world tasks against OpenWAM. Ranking Block Size uses a single arm for size-dependent spatial ordering; Pick Up Plate and Open Microwave use two arms for plate handling and multistep object retrieval. Table 15 gives a representative instruction for each task, including an explicit target object, ordering direction, or final location. Each task has 50 goal-preserving training paraphrases; a stable episode-based hash assigns one to each demonstration. Both policies retain the six-task training mixture; the three tasks reported here form the evaluation subset.

We retain successful demonstrations, remove exactly stationary prefixes and suffixes using the recorded robot state, and exclude incomplete action/video windows. Multitask WAM training samples tasks uniformly, then samples a complete window within the selected task. Actions retain the native 30 Hz sampling; temporal strides below apply to visual frames, not action subsampling.

Table 15: Real-world tasks and representative training instructions. The quoted instructions are drawn from each task's 50 training paraphrases. Ordering and target positions are specified relative to the robot or task object.

Task	Manipulation	Representative instruction
Pick Up Plate	Bimanual plate handling	"Bimanually remove the rack's robot-nearest plate; lay it flat, face-up, between rack and robot."
Ranking Block Size	Single-arm ordering	"Arrange cubes along robot depth: largest farthest, medium centered, smallest nearest."
Open Microwave	Bimanual retrieval	"Open the microwave; put the inside basket to microwave's right; leave door open."

Training recipes. Table 16 compares the PrismWAM and OpenWAM settings. Both use full policy fine-tuning, AdamW with $\beta = (0.9, 0.95)$, gradient clipping at 1.0, no gradient accumulation, seed 42, and checkpoints every 5000 updates. Linear warmup is followed by cosine decay to the listed minimum learning rate. PrismWAM resets the optimizer and schedule between its two stages. Both runs use bf16, DeepSpeed ZeRO-2, and gradient checkpointing, with VAEs and text encoders frozen. PrismWAM uses the head-view depth VAE from Appendix A.1.3.

Views and temporal context. PrismWAM uses one external RGB view and its aligned depth, each resized to 384×320. The policy forms no camera mosaic and uses no wrist images in this recipe. The head-view depth autoencoder is pretrained at 256×320, whereas the policy input is 384×320. Depth is refined with LingBot-Depth v0.5 and causal temporal smoothing, stored in millimeters, and normalized over 0–1500 mm. RGB history uses $t + [-16, -12, -8, -4, 0]$; depth prediction uses $t + [0, 4, \ldots, 32]$; the isolated RGB target is $t + 32$.

OpenWAM places that view in the upper 256×320 region of a 384×320 canvas; the two lower 128×160 cells are black placeholders, not wrist observations. OpenWAM predicts nine-frame RGB sequences at stride four. OpenWAM applies clip-consistent brightness, contrast, and saturation jitter of 0.2 while keeping the placeholder cells black. PrismWAM applies no RGB augmentation and uses 128 text tokens; OpenWAM uses 512. Each retained prediction loss has unit weight.

Joint-space actions for PrismWAM. PrismWAM conditions on measured joint state at t and predicts recorded absolute joint commands at $t+1, \ldots, t+32$. Each arm contributes six joint angles in degrees followed by gripper opening in millimeters. Bimanual vectors concatenate the left then right arm. A single-arm vector occupies the first seven of 14 slots, with padded dimensions masked. Per-task action mean and standard deviation normalize both action targets and proprioception; no state subtraction is applied.

End-effector actions for OpenWAM. OpenWAM uses absolute end-effector targets derived by forward kinematics from future measured states, rather than recorded joint commands. Each arm contributes position, a six-dimensional rotation representation, and gripper opening. Position and

Table 16: Real-world training configurations. PrismWAM and OpenWAM both use multitask training. R, D, and A denote RGB, depth, and action prediction losses. Abs. denotes absolute targets. Camera layouts and action transformations are detailed below.

Setting	**PrismWAM**	**OpenWAM**
Initialization and optimization		
Initialization	Wan2.2-5B	OpenWAM-α
Training scope	Multitask	Multitask
Updates	15k + 15k	30k
Global batch	256	256
Peak learning rate	10^{-4}	10^{-4}
Minimum learning rate	10^{-6}	10^{-6}
Warmup updates	750 per stage	1500
Weight decay	10^{-2}	10^{-2}
Prediction losses	D, then $R{+}D{+}A$	$R + A$
Observations and action targets		
RGB layout	External view	External + black wrist cells
RGB resolution	384×320	384×320
Observed RGB frames	5, stride 4	1
Depth input	External view	None
Action horizon	32	32
Action targets	Abs. joints	Abs. end-effector
Gripper targets	Absolute	Absolute
Normalization	Per-task mean/std	Min–max
RGB augmentation	None	Color jitter

gripper opening are in meters; each arm uses its own base frame. The left and right ten-dimensional vectors occupy slots 0–9 and 34–43 of the 80-dimensional interface, with other slots masked. State and action share training-data min–max statistics; rotation coordinates retain identity normalization. Thus the WAM recipes share task identities and external-camera sources while retaining method-specific action representations and supervision sources.

Physical evaluation and aggregation. Table 5 compares OpenWAM with PrismWAM on Pick Up Plate, Ranking Block Size, and Open Microwave under ID and OOD conditions. Pooled success divides total successes by total trials across all three tasks, rather than averaging task percentages.

Task-specific evaluation conditions. For *Pick Up Plate*, ID uses the original plate-and-rack setup. OOD varies illumination, adds distractors, or changes the rack contents and plate colors; table 17 reports each recorded condition. For *Ranking Block Size*, ID covers all six initial size permutations, and OOD varies illumination, distractors, the tablecloth, and initial placement; table 18 gives the individual counts. The target ordering remains largest farthest from the robot and smallest nearest.

For *Open Microwave*, the task is to open the door, move the basket from inside the microwave to the tabletop on its right, and leave the door open. Each model has ten ID and ten OOD trials: OpenWAM achieves 9/10 and 4/10, and PrismWAM achieves 8/10 and 5/10, respectively. OOD conditions comprise flickering colored lights, low ambient light with flickering colored lights, no ambient light with flickering colored lights, distractors, and a changed tablecloth. Only aggregate OOD counts are reported; the trial allocation and success counts for individual conditions are not specified.

A.4 Comparison and Reporting Conventions

Reporting conventions. An em dash denotes an unavailable result, not zero performance. Bold and underlined entries in the main comparison tables mark the best and second-best reported values, respectively, not statistical significance. Evaluation counts and aggregation rules are specified in the corresponding tables.

Table 17: **Pick Up Plate: condition-level physical evaluation.** Entries are successes/trials. OOD changes preserve the plate-handling goal.

Split	Condition	OpenWAM	PrismWAM
ID	Original setup	4/5	5/5
OOD	Flickering lights	2/2	2/2
	Flickering lights + reduced brightness	1/2	2/2
	Flickering lights + lights switched off	0/2	2/2
	Distractor wooden blocks	2/2	2/2
	Distractor carrot	2/2	2/2
	Empty plate rack	2/2	1/2
	Changed colors of plates in the rack	1/2	2/2
	Changed color of the handled plate	0/1	0/1
	OOD total	10/15	13/15

Table 18: **Ranking Block Size: condition-level physical evaluation.** Entries are successes/trials. Large, medium, and small denote the three cube sizes in the recorded initial ordering.

Split	Condition	OpenWAM	PrismWAM
ID	Large–medium–small	2/2	2/2
	Large–small–medium	1/2	2/2
	Medium–large–small	2/2	2/2
	Medium–small–large	0/2	2/2
	Small–large–medium	1/2	1/2
	Small–medium–large	1/2	2/2
	ID total	7/12	11/12
OOD	Distractors	1/2	2/2
	Changed tablecloth	0/1	0/1
	Flickering lights	1/2	2/2
	Flickering lights + reduced brightness	1/2	0/2
	Flickering lights + lights switched off	0/2	2/2
	Randomized initial positions	0/1	0/1
	OOD total	3/10	6/10

Pretraining groups. The *w/ Embodied Action Pretraining* and *w/o Embodied Action Pretraining* groups refer to additional pretraining with robot-action supervision before target-benchmark training. Generic VLM/video initialization, visual world-model or latent-action pretraining without robot-action supervision, and separate encoder pretraining are excluded from this definition. PrismWAM uses Wan initialization and a separately pretrained depth VAE; absence of embodied action pretraining therefore does not imply training all components from scratch or using no external data.

Method variants and protocol differences. The standard-LIBERO starVLA-OFT result uses Qwen3-VL-4B, whereas Plus/Pro use the Qwen3-VL-PI checkpoint trained on all four LIBERO suites for 100k updates; these are distinct variants. FastWAM uses its official Fast policy checkpoint with two 224×224 camera views for Plus/Pro. OpenWAM uses its official LIBERO checkpoint at training step 10,690. Qwen-RobotManip uses its Context variant on standard LIBERO (99.2) and its base variant on Plus. OpenVLA-OFT+ uses Plus mix-SFT rather than standard-only transfer.

On LIBERO-Pro, Environment results are unavailable for X-VLA, MolmoAct, and NORA, so their totals are not presented as complete five-category aggregates. Environment-instance differences are marked separately in table 2.

The RoboDojo GPT-6-Astra row represents a zero-shot API agent with a non-learned control harness, not a fine-tuned VLA/WAM. Its pretraining-group placement refers only to the absence of additional embodied training in that deployment, not to undisclosed foundation-model training data. These cross-model benchmark comparisons are separate from the controlled RoboTwin ablations below.

A.5 Ablation and Qualitative Evaluation

Experimental design. We study six changes to the semantic, physical, and action pathways of PrismWAM. All models use RoboTwin clean-only demonstrations, the Wan2.2-5B backbone, global batch size 256, and 30k optimizer updates. The reference recipe allocates 15k updates to depth-only adaptation and 15k to joint training. We retain task-balanced sampling over 50 tasks, seen training instructions, three-view 384×320 inputs, 32-step action chunks, current proprioception, and the normalization in Appendix A.1.2. There is no sensor-depth augmentation. Language and proprioception remain available to every action policy, and the applicable VAEs and T5 remain frozen.

Table 19: Component ablations on RoboTwin clean-to-random transfer. Success rates (%) use 50 tasks with 20 episodes per task and condition, totaling 1000 Clean and 1000 Random episodes per model. Mean averages the two rates; Δ_{Random} is the change from PrismWAM in percentage points. Every model uses clean-only training, global batch size 256, and 30k total updates.

Variant	Clean	Random	Mean	Δ_{Random}
PrismWAM	87.40	**54.10**	**70.75**	–
w/o depth stream	83.00	2.40	42.70	−51.70
w/o RGB stream	79.10	40.10	59.60	−14.00
w/o future-RGB supervision	81.60	51.40	66.50	−2.70
Observed RGB → action	89.10	40.30	64.70	−13.80
Future RGB → depth	87.00	50.00	68.50	−4.10
w/o staged depth pretraining	**89.30**	24.70	57.00	−29.40

Objectives and initialization. Table 20 summarizes the objectives. All staged variants reset the optimizer and cosine schedule between stages; each stage uses peak/minimum learning rates $10^{-4}/10^{-6}$ and 750 warmup updates. The RGB-removal, RGB-supervision, and two connectivity ablations reuse the same depth-adapted checkpoint. Action uses the same Wan-derived initialization. Removing a stream removes its private modules at training and inference.

Evaluation and reading the task plots. We evaluate the final checkpoint on 20 episodes per task under Clean and fully randomized scenes, including unseen wall and table textures. All policies use unseen instructions, policy seed 0, instruction-sampling seed 42, environment seeds starting

Table 20: Training objectives and changed pathways. R, D, and A denote future-RGB, future-depth, and action prediction losses, each with coefficient one when present. History always denotes observed RGB; future denotes the diffusing RGB target at $t + 32$.

Variant	First 15k	Last 15k	Changed pathway
PrismWAM	D	$R + D + A$	History → depth → action
w/o depth stream	R	$R + A$	History → action
w/o RGB stream	D	$D + A$	Depth → action
w/o future-RGB supervision	D	$D + A$	RGB input retained; R removed
Observed RGB → action	D	$R + D + A$	Add history → action
Future RGB → depth	D	$R + D + A$	Add future → depth
w/o staged depth pretraining	$R + D + A$, jointly for 30k		No depth-only adaptation

at 100000, ten denoising steps, and replanning every 24 actions. Every task has 20 unique completed task/seed records. Accepted seed sets overlap partially across policies, so the trial counts and configured protocol match without implying fully paired rollouts. One successful episode changes a task's rate by five percentage points; individual examples describe patterns rather than establish statistical significance. This comparison is separate from the 100-episode main benchmark. The following six findings connect aggregate results to task-level behavior, with complete 50-task plots in Appendix A.5.7.

A.5.1 W/O DEPTH STREAM: PHYSICAL PREDICTION AND TRANSFER

This variant removes depth throughout training. It first learns RGB prediction for 15k updates from five observed history frames, with an isolated future target at $t + 32$, then trains RGB and action jointly for 15k updates. Action reads observed RGB history, never the future-RGB target. No depth observations, depth VAE, depth tokens, or depth-specific prediction modules are loaded. Thus the comparison removes the entire depth pathway, including its adaptation stage, rather than just masking depth at inference.

Clean success remains 83.00%, compared with 87.40% for PrismWAM. Random success, however, falls from 54.10% to 2.40%: 49 of 50 tasks decline and 44 have zero successful episodes. Figure 6 shows that this spans different physical skills. Pick Dual Bottles and Place Burger and Fries drop from 95% to 0%; Grab Roller, Put Bottles in Dustbin, and Stack Two Bowls drop from 90% to 0%. The failure therefore extends beyond tasks requiring explicit color interpretation.

Key finding 1: Depth is central to transfer under scene changes. The small Clean gap and large Random gap distinguish fitting familiar scenes from learning a transferable control representation. Predicting physical evolution in depth space reduces the need to represent texture and illumination, allowing observed RGB to provide scene semantics while depth supplies the physical basis for action. This interpretation is consistent with the broad transfer loss when that pathway is removed. Random changes lighting, background, clutter, and table height together; the experiment supports robustness across these changes jointly, rather than attributing the entire effect to illumination alone.

A.5.2 W/O RGB STREAM: SCENE SEMANTICS BEYOND GEOMETRY

We retain the same depth-adapted checkpoint and train depth plus action for 15k updates, removing RGB data, tokens, and private projections. Language and proprioception remain available. This separates visual RGB information from linguistic conditioning and differs from the earlier depth-only system in **??**, which also omitted language.

The clearest Clean losses occur in a predefined nine-task RGB-related group: eight tasks involving color cues and Rotate QR Code, which additionally involves a surface pattern. Success falls from 82.78% to 47.78%, compared with 88.41% to 85.98% on the other 41 tasks. The RGB-related group accounts for 63 of the 83 net lost Clean successes. As fig. 7 shows, Rank Blocks by Color drops from 90% to 15%, Stack Three Blocks from 90% to 20%, and Handover Block and Stack Two Blocks each from 100% to 50%. On the eight color-related tasks alone, the mean falls from 82.50% to 44.38%.

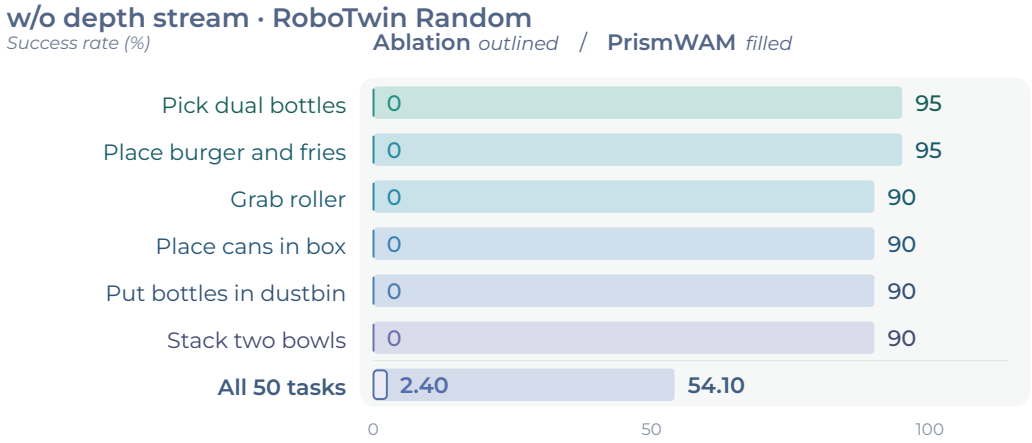


Figure 6: **Removing depth broadly degrades transfer.** Selected Random tasks cover bimanual grasping, multi-object placement, tool handling, transport, and stacking. Outlined bars show the ablation; pale filled bars show PrismWAM. Every task uses 20 episodes. The final row uses all 50 tasks, including those not displayed.

Key finding 2: RGB supplies task-relevant semantics that depth cannot fully recover. Geometric observations can represent shape and arrangement, but they do not preserve the color or surface cues that disambiguate many task goals. The concentrated Clean decline supports conditioning depth prediction on observed RGB rather than asking a depth-only policy to infer those cues from geometry and language. The contribution is broader under distribution shift: on the other 41 Random tasks, success also falls from 59.27% to 46.59%. RGB and depth are therefore complementary, rather than interchangeable modalities.

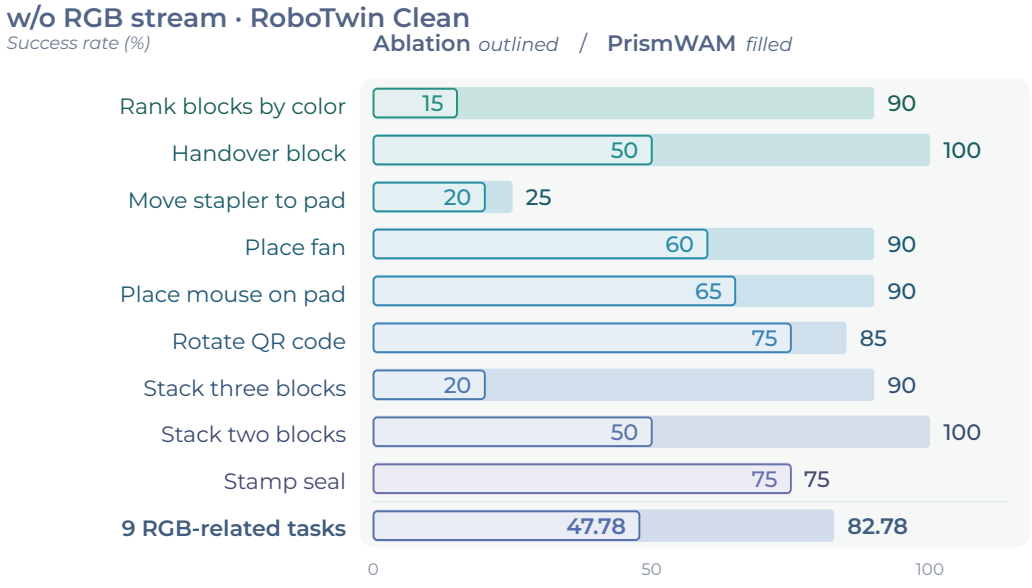


Figure 7: **RGB observations support semantic task execution.** All nine tasks in the predefined RGB-related group are shown, including the unchanged Stamp Seal result. Outlined bars denote removal of the RGB stream and pale filled bars denote PrismWAM; language and proprioception are retained. The mean pools all 180 Clean episodes in this group.

A.5.3 w/o Future-RGB Supervision: Learning Useful Semantic Features

Here RGB remains an input to the observed-RGB-to-depth-to-action pathway. We remove only the future-RGB target, its tokens, and its prediction head, setting $\lambda_R = 0$. The RGB conditioning path and shared visual blocks still receive gradients from depth and action losses. Depth adaptation, action initialization, and the 15k joint-training budget remain unchanged.

Clean success drops from 87.40% to 81.60%, with the RGB-related group falling from 82.78% to 60.00%. This group accounts for 41 of the 58 net lost successes, while the complementary group declines by 2.07 points. Figure 8 highlights Rank Blocks by Color (90% to 40%), Stack Three Blocks (90% to 25%), and Stack Two Blocks (100% to 60%). Under Random, removing the objective reduces the RGB-related group from 30.56% to 19.44%, accounting for 20 of the 27 net lost successes. The other 41 tasks change from 59.27% to 58.41% in aggregate, with both gains and declines.

Key finding 3: Future RGB is useful as a training target. Observing RGB alone does not recover the performance obtained with an explicit RGB prediction objective. The concentration of losses in color ordering and stacking is consistent with this auxiliary task strengthening scene-level semantic features used by the depth predictor. Crucially, this benefit does not require generated RGB to condition execution: PrismWAM learns from the future-RGB target during training while excluding future-RGB tokens from depth and action conditioning. The next two ablations test whether adding those conditioning routes improves control.

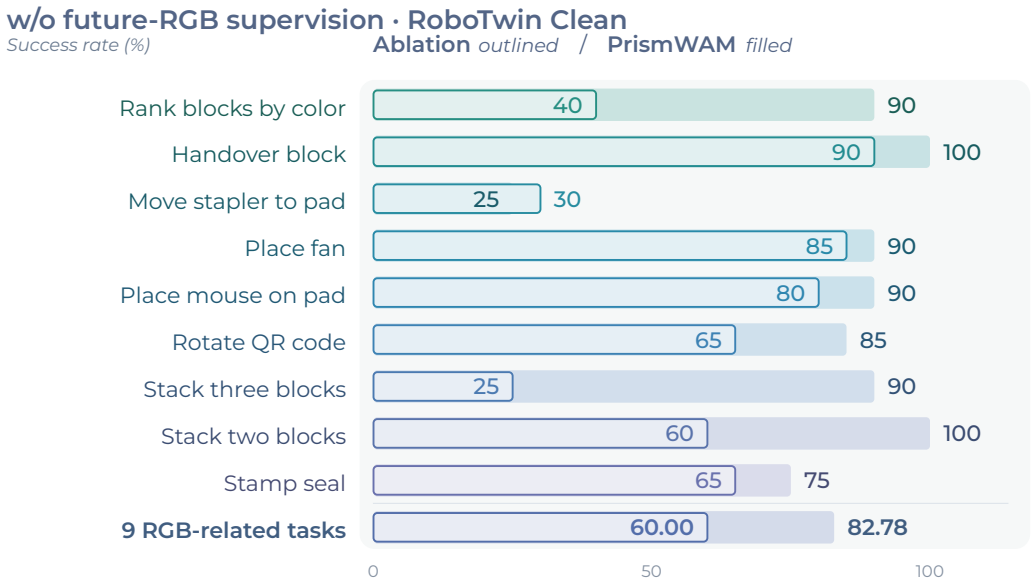


Figure 8: **RGB prediction supervision improves semantic behavior even when RGB observations are retained.** The task set, episode count, and scale match fig. 7. The improvement in Move Stapler to Pad is retained; the largest losses occur in color ordering and stacking. The final row averages the nine displayed tasks.

A.5.4 Observed RGB to Action: Testing a Direct Semantic Connection

This variant adds one directed connection: action can attend to observed RGB history as well as depth. Depth also reads only observed RGB. All three losses, shared visual blocks, initialization, and training budgets are unchanged. Neither depth nor action reads future-RGB tokens, and no action-to-visual edge is added. This is a direct observed-RGB connection, not unrestricted bidirectional attention. Observed features remain cached across denoising steps.

Clean success rises from 87.40% to 89.10%, but Random falls from 54.10% to 40.30%. Across the 50 Random tasks, 39 decline, four tie, and seven improve. The largest losses include Place Phone

on Stand (60% to 5%), Put Bottles in Dustbin (90% to 40%), Place Fan (55% to 10%), and Move Playing Card Away (65% to 25%). The effect is not universal: Turn Switch improves from 35% to 55% (fig. 9).

Key finding 4: More direct visual access does not imply better transfer. The direct RGB path helps fit familiar scenes but weakens aggregate performance under randomization. These results favor a division of roles in which RGB guides the physical rollout and action evolves jointly with that rollout. Routing semantic information through depth constrains action to use a representation of the predicted physical state, whereas a direct appearance-dependent path can bypass this constraint. The observed tradeoff supports the original connectivity; it does not imply that semantic information is unnecessary for control or that every task is harmed by direct access.

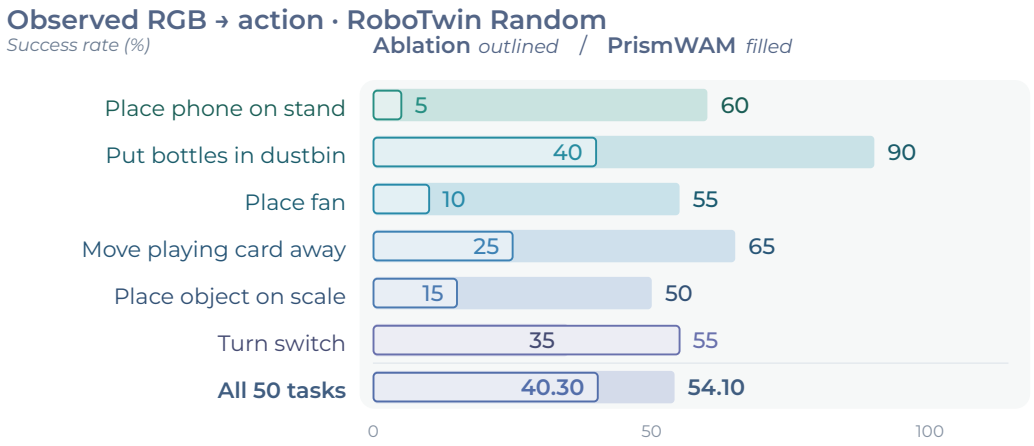


Figure 9: **Direct observed-RGB access by action weakens aggregate Random transfer.** The panel includes the five largest declines and an improved task, Turn Switch. Outlined bars denote the added RGB-to-action edge. The final aggregate covers all 50 tasks, rather than only these selected examples.

A.5.5 Future RGB to Depth: Supervision Versus Control-Time Feedback

We keep five observed RGB frames at stride four, the same depth-adapted checkpoint, Wan-based action initialization, and all three losses. Depth additionally attends to the diffusing future-RGB latent at $t + 32$: it reads the current noisy or partially denoised tokens, not an externally supplied clean future image. Action receives this information only through depth; there is no direct RGB-to-action connection. Inference jointly denoises future RGB, depth, and actions, whereas PrismWAM caches observed RGB and generates no future RGB for control.

This change yields 87.00%/50.00% Clean/Random success, versus 87.40%/54.10% for PrismWAM. Random declines on 28 tasks, ties on eight, and improves on 14. Adjust Bottle drops from 50% to 15%, Stack Three Bowls from 70% to 40%, and Stack Two Blocks from 65% to 40%. Move Stapler to Pad improves from 0% to 25%, illustrating the task-dependent effects (fig. 10).

Key finding 5: Future-RGB supervision is more useful here than future-RGB feedback. Together with the previous supervision ablation, this comparison separates learning to predict appearance from relying on that prediction for execution. Removing RGB supervision hurts semantic tasks; feeding the predicted RGB back into depth provides no aggregate gain and lowers Random success by 4.10 points. A plausible explanation is that appearance-prediction errors under distribution shift contaminate the physical rollout through this extra edge. The qualitative prediction diagnostic in Appendix A.5.8 illustrates such RGB degradation, although success rates alone do not measure error propagation. The results favor retaining RGB as an auxiliary target while conditioning depth on observed RGB history.

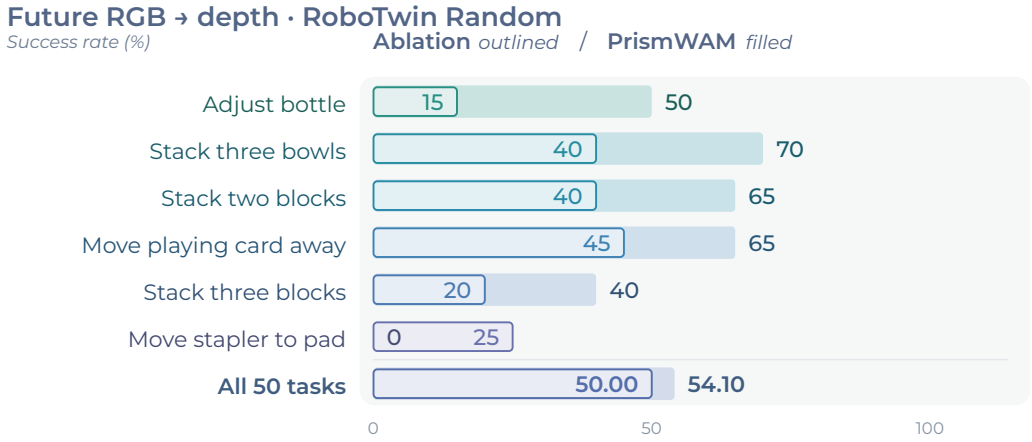


Figure 10: **Feeding diffusing future RGB to depth provides no aggregate transfer gain.** Selected tasks show losses in reorientation, stacking, and object moving, alongside a gain in stapler placement. Action reads future-RGB information only indirectly through depth. The final row covers all 50 Random tasks.

A.5.6 W/O STAGED DEPTH PRETRAINING: ADAPTING A NEW MODALITY

We initialize the three-stream policy from Wan and train jointly for 30k updates, omitting the depth-only adaptation stage. The pretrained depth VAE remains frozen, so this ablates policy adaptation rather than depth-VAE initialization. A single optimizer uses a 30k cosine schedule with 1500 warmup updates and the same peak/minimum learning rates. Total optimizer updates match the staged recipe, but action receives 30k joint updates instead of 15k.

Clean success improves from 87.40% to 89.30%, while Random drops from 54.10% to 24.70%. Random success declines on 46 tasks, ties on three, and improves on only one. Grab Roller falls from 90% to 0%, Lift Pot from 70% to 0%, and Open Laptop and Scan Object each from 60% to 5% (fig. 11). Twice as many action-training updates therefore do not compensate for removing depth adaptation.

Key finding 6: Depth needs dedicated adaptation before joint optimization. The video backbone starts with an RGB prediction prior, while depth introduces a different representation of physical state. A depth-only first stage gives the shared backbone a dedicated opportunity to learn this modality before it must optimize RGB, depth, and action together. The strong Random advantage supports retaining that adaptation stage. Competition from the already familiar RGB objective is a plausible reason why joint training from the start is less effective, but these results do not establish RGB-loss dominance or modality collapse. A mixed RGB–depth first-stage control or modality-specific gradient measurements would be needed to isolate that mechanism.

A.5.7 COMPLETE TASK-WISE RESULTS AND CAPABILITY SUMMARY

Table 21 separates the predefined nine-task RGB-related group from its 41-task complement. This grouping comes from our evaluation setup, rather than an official capability taxonomy or selection by observed performance. It contains Blocks Ranking RGB, Handover Block, Move Stapler to Pad, Place Fan, Place Mouse to Pad, Rotate QR Code, Stack Three Blocks, Stack Two Blocks, and Stamp Seal. Each group pools 180 or 820 episodes per condition. Table 22 consolidates the six findings, relating aggregate changes to representative tasks and their implications. To retain both improvements and regressions beyond the selected plots, figs. 12 and 13 report every task under every ablation.

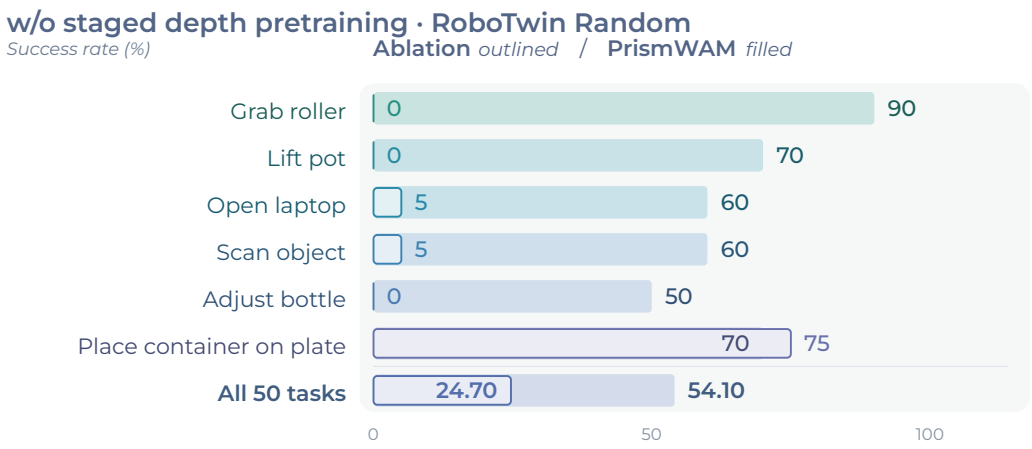


Figure 11: **Dedicated depth adaptation improves transfer despite fewer action-training updates.** Outlined bars denote single-stage 30k joint training. Selected tasks include the only improvement, Place Container on Plate; the final aggregate covers all 50 Random tasks. The depth VAE is pretrained and frozen in both models.

Table 21: Task-group success rates (%). We use a predefined nine-task RGB-related subset; the remaining 41 form the complementary group. Each task contributes 20 episodes per condition.

	Clean		Random	
Variant	RGB-related	Other	RGB-related	Other
PrismWAM	82.78	88.41	30.56	59.27
Without depth stream	67.78	86.34	0.00	2.93
Without RGB stream	47.78	85.98	10.56	46.59
Without future-RGB supervision	60.00	86.34	19.44	58.41
Direct observed-RGB access by action	86.11	89.76	13.33	46.22
Future-RGB access by depth	82.78	87.93	25.56	55.37
Without staged depth pretraining	81.11	91.10	9.44	28.05

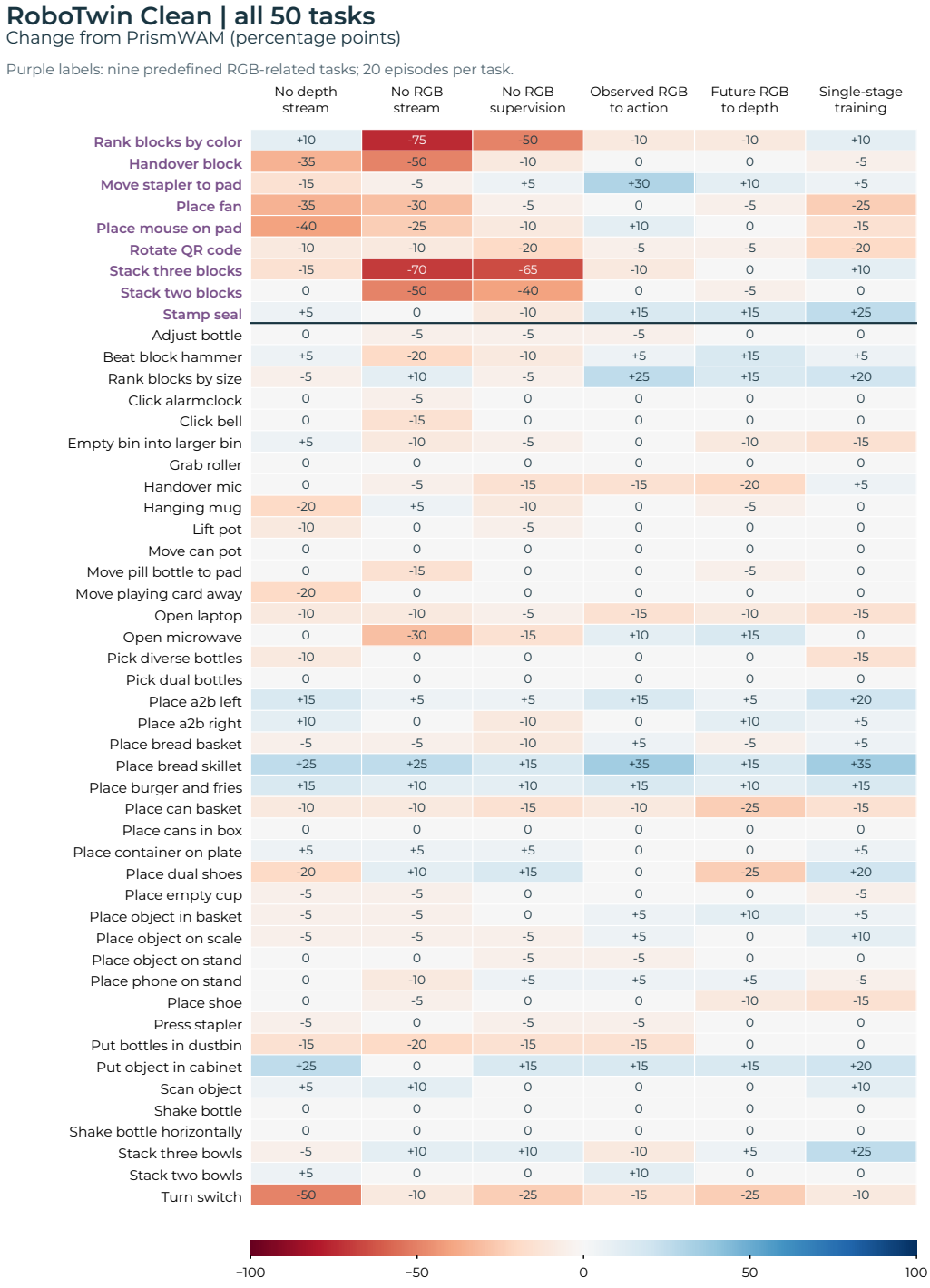


Figure 12: **Clean success changes across all 50 tasks.** Values are ablation minus PrismWAM in percentage points. Each task uses 20 episodes per model; five points correspond to one success. Purple task names mark the predefined RGB-related group. Positive and negative changes are both retained.

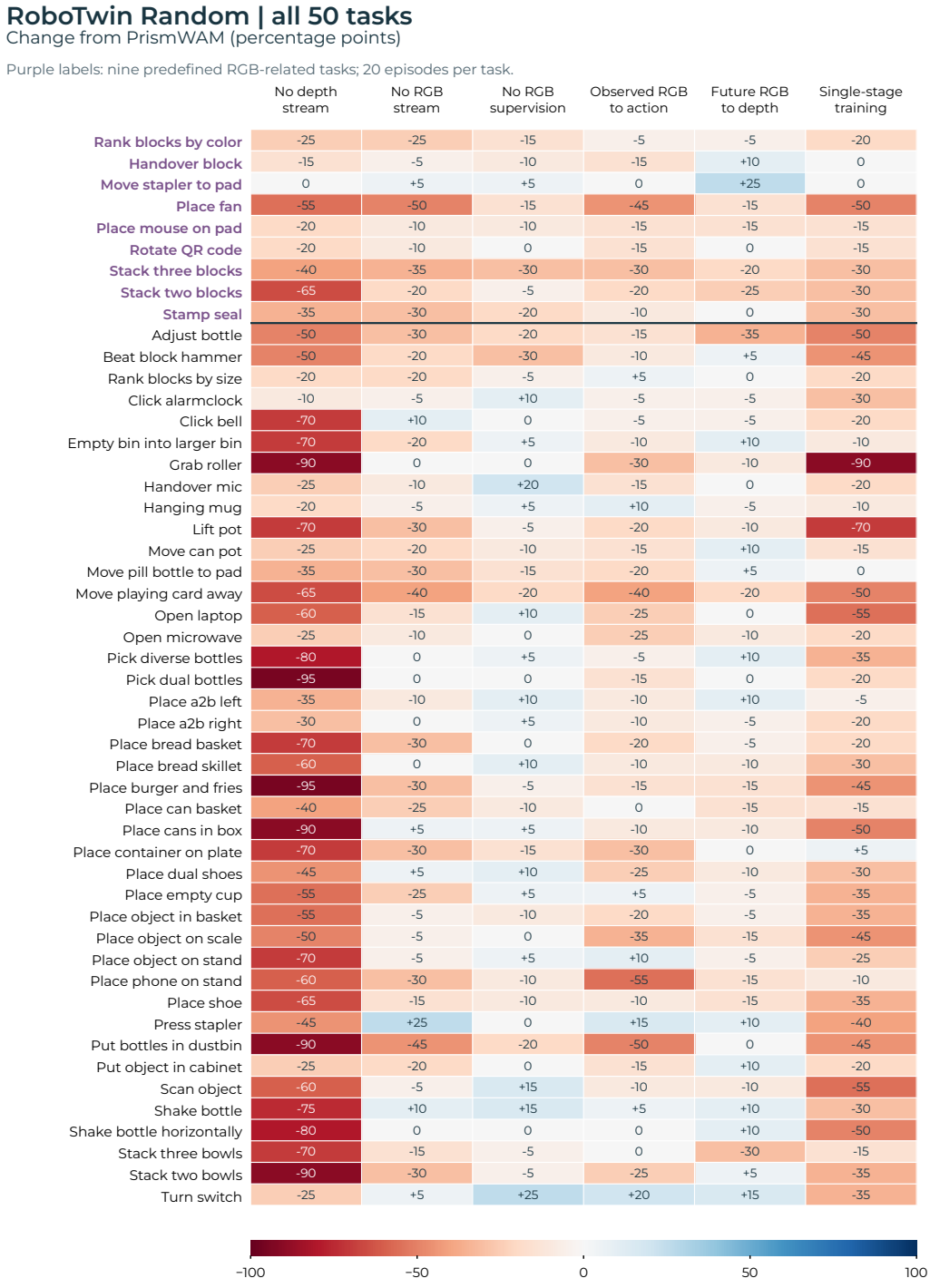


Figure 13: **Random success changes across all 50 tasks.** Layout and scale match fig. 12. Depth removal broadly degrades transfer; removing RGB observations or supervision strongly affects semantic tasks, while both additional RGB-access paths have task-dependent effects and lower aggregate Random success.

Table 22: Summary of the six ablation findings. Clean and Random columns report aggregate changes relative to PrismWAM (87.40% and 54.10%), in percentage points. Task examples show PrismWAM → ablation success rates (%), with 20 episodes per task and condition. Examples illustrate declines; the complete task matrices retain gains as well. Interpretations concern the tested interventions, rather than a measured failure mechanism.

Ablation	ΔClean	ΔRandom	Representative task evidence	Supported finding
w/o depth stream	-4.40	-51.70	*Random:* Pick dual bottles $95 \rightarrow 0$; place burger and fries $95 \rightarrow 0$.	Depth prediction is central to transfer across scene changes.
w/o RGB stream	-8.30	-14.00	*Clean:* Rank blocks by color $90 \rightarrow 15$; stack three blocks $90 \rightarrow 20$.	Observed RGB provides semantic cues unavailable from geometry alone.
w/o future-RGB supervision	-5.80	-2.70	*Clean:* Rank blocks by color $90 \rightarrow 40$; stack three blocks $90 \rightarrow 25$.	Auxiliary RGB prediction benefits color-dependent task execution.
Observed RGB to action	$+1.70$	-13.80	*Random:* Place phone on stand $60 \rightarrow 5$; put bottles in dustbin $90 \rightarrow 40$.	The direct RGB path improves Clean but weakens aggregate transfer.
Future RGB to depth	-0.40	-4.10	*Random:* Adjust bottle $50 \rightarrow 15$; stack three bowls $70 \rightarrow 40$.	Generated-RGB feedback adds no aggregate gain over observed RGB.
w/o staged depth pretraining	$+1.90$	-29.40	*Random:* Grab roller $90 \rightarrow 0$; lift pot $70 \rightarrow 0$.	Dedicated depth adaptation improves transfer despite fewer action updates.

A.5.8 Qualitative Execution and Prediction Diagnostics

10k-step prediction and reconstruction diagnostic. We additionally use an earlier modality pilot trained on ten clean RoboTwin tasks for 10k steps. RGB and depth models share the nine-frame, stride-four prediction format and use modality-specific VAEs. The recorded diagnostic contains Grab Roller episodes 0, 74, and 149 from both clean and fully randomized scenes. Successive prediction windows are conditioned on the previous predicted endpoint, not on future ground-truth frames. The saved PSNR first averages normalized-image PSNR over the 17-frame sequence and then averages the three episode scores. It includes the initial frame, the entire three-camera mosaic, and repeated tail frames when required by the fixed horizon, without masking invalid depth. Depth is normalized using the configured fixed range; pseudocoloring is applied only for display, after metric calculation. The VAE score instead compares ground-truth encode–decode reconstruction with the ground-truth input. These metrics are preliminary prediction diagnostics.

The qualitative examples illustrate possible failure modes but do not identify which prediction errors cause action failures. Isolating that relationship would require matched rollout-error measurements; the component ablations above instead quantify changes in closed-loop task success.

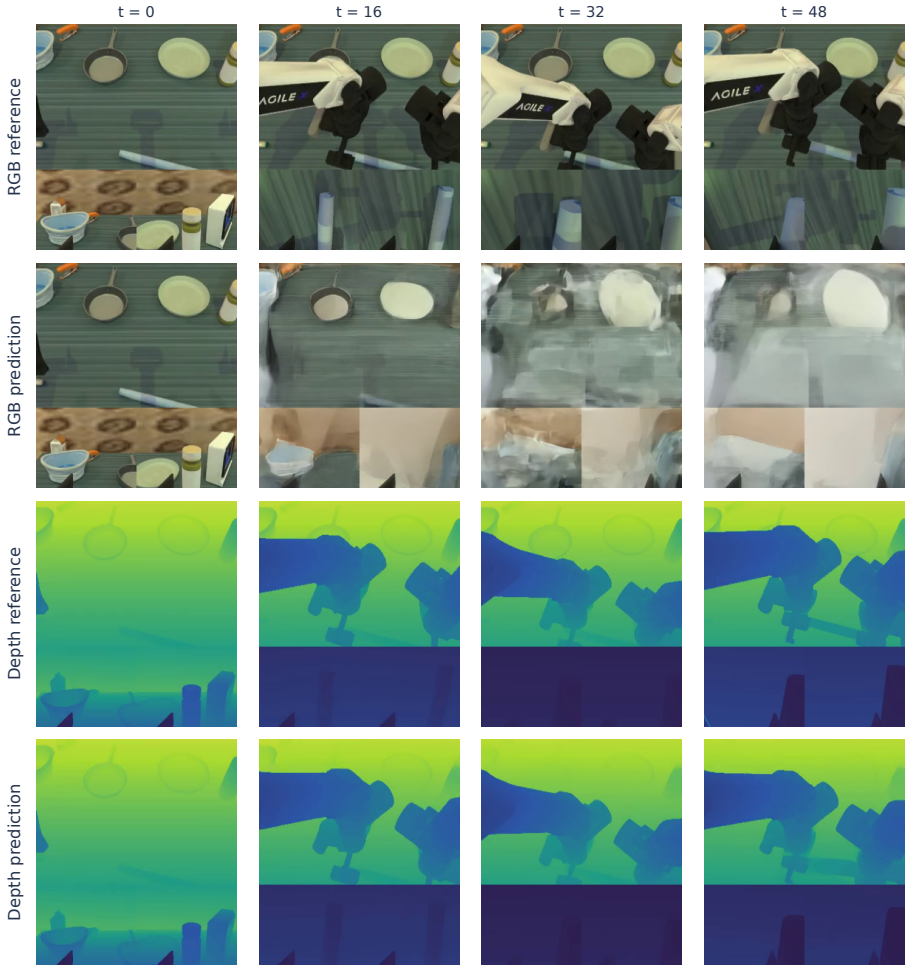


Figure 14: **Autoregressive prediction in the historical 10k-step pilot.** Grab Roller, randomized episode 74, at offsets of 0, 16, 32, and 48 control steps. Each modality has its own reference and prediction rows. The RGB prediction progressively loses scene and robot detail, while depth retains a more coherent spatial structure. Predictions are not exact physical trajectories, and neither row shows closed-loop execution. The displayed offsets precede the padded tail of this episode.

A.6 Depth Corruption Settings and Evaluation

Corruption levels and scope. The RoboTwin study perturbs raw metric depth before clipping, normalization, resizing, mosaicking, and VAE encoding. It leaves RGB observations, language, proprioception, and action targets unchanged. These are controlled measurement-error and missing-depth stress tests, not a calibrated model of a particular sensor. We define three corruption levels in table 23. **Level 1** combines Gaussian range noise and fragmented edge holes. **Levels 2 and 3** increase the metric errors and edge width, and add global shifts, motion-tracked edge holes, and intermittent patch holes. Uncorrupted depth provides the reference condition.

Metric jitter and global shift. For a valid native-resolution depth pixel $d > 0$, measured in millimeters, range noise is additive Gaussian noise $\epsilon_d \sim \mathcal{N}(0, \sigma(d)^2)$ with

$$\sigma(d) = \sigma_{0\mathrm{m}} + (\sigma_{1\mathrm{m}} - \sigma_{0\mathrm{m}}) \left(\frac{d}{1000\,\mathrm{mm}} \right)^2 . \tag{10}$$

The noise is independently drawn across pixels, cameras, and distinct frame indices. Repeated reads of the same frame within a session reuse the same draw. Levels 2 and 3 also sample a signed offset $b \sim \mathcal{U}(-B, B)$ once per training clip or evaluation episode; this offset is shared across all three cameras and held constant over time. The range-noise floor is nonzero even at short range. For example, Level 1 has a standard deviation of 12.72 mm at 300 mm and 20 mm at 1 m. This operation does not Gaussian-blur the input depth image: Gaussian filtering below is applied to random fields used to choose correlated dropout locations.

Table 23: Depth profiles used in fig. 4. Radius is the number of native-resolution binary-dilation iterations, not a diameter or a mosaic-space size. The edge fraction is a selection threshold within the edge band, not a whole-image missing fraction. Patch fraction applies only during an active event.

Setting	$\sigma_{0\mathrm{m}}/\sigma_{1\mathrm{m}}$ (mm)	Shift range (mm)	Edge radius (px)	Base edge fraction	Patch fraction
Clean depth	0/0	0	–	0	0
Level 1	12/20	0	1	25%	0
Level 2	24/40	$[-25, 25]$	5	25%	1.5%
Level 3	36/60	$[-50, 50]$	7	24%	1.5%

Edge detection and Level 1 dropout. An edge is marked on both sides of a horizontal or vertical pair of valid pixels whose depth difference exceeds 30 mm. The binary edge map is dilated with four-neighbor connectivity and intersected with valid depth. Detection uses the original depth, before adding range noise or global shift. Level 1 selects the highest-scoring 25% of pixels in the one-pixel-dilated band. Its score field is obtained by Gaussian-smoothing white noise with standard deviation 2 pixels. Independent field anchors are generated every 12 original frames, linearly interpolated between anchors, and standardized over valid pixels. This produces fragmented but spatially correlated edge holes that change gradually in image coordinates.

Tracked edge dropout and distance weighting. For Levels 2 and 3, a fixed 512×512 random texture combines coarse and fine Gaussian-filtered fields: coarse standard deviation 18/24 pixels, fine standard deviation 1 pixel, and coarse/fine weights 0.92/0.08. Approximate RGB motion correspondences transport texture coordinates between consecutive frames, so nearby points on a persistently visible surface tend to retain similar dropout scores. The implementation uses OpenCV DIS optical flow with its fast preset. A correspondence is accepted only when its forward–backward error is at most $1.5 + 0.05\|f\|$ pixels, warped-depth disagreement is at most $40 + 0.05d$ mm, and grayscale disagreement is at most 40 on the 0–255 scale. Unreliable or newly visible pixels receive fresh texture coordinates. This is approximate motion persistence, not object-instance tracking or guaranteed persistence through occlusion.

Let $q(p)$ be a pixel's ascending score quantile within the current edge band. A pixel is removed when $q(p) \geq 1 - p_{\text{edge}}(d)$, where

$$p_{\text{edge}}(d) = f_{\text{edge}} \left[1.5 - \text{clip}\left(\frac{d - 300}{600}, 0, 1 \right) \right]. \tag{11}$$

Here d is in millimeters and f_{edge} is 0.25/0.24 for Levels 2 and 3. Thus the multiplier is 1.5 at or below 300 mm and 0.5 at or above 900 mm; closer edges are more likely to be missing. These thresholds produce correlated regions, rather than independent Bernoulli dropout at each edge pixel. Level 3 uses a slightly smaller base fraction but a wider edge band and coarser field, alongside larger metric errors.

Intermittent patch holes. Levels 2 and 3 also create at most one connected patch per camera at an event's onset, targeting 1.5% of valid pixels. A Gaussian-filtered random field with standard deviation 6 pixels defines the patch shape. In each 80-frame cycle, its start is sampled uniformly from offsets 16–32 and its duration from 4–8 frames, both inclusive. The mask stays fixed in image coordinates for that event and is absent outside it; it is not continuously moved or regenerated at every frame. It is intersected with current valid depth, retaining only the largest connected component if that intersection splits it. Therefore the visible patch can be smaller than its initial target area. Event timing and shapes are camera-specific.

Composition and invalid values. Level 1 always applies range noise and edge holes. Levels 2 and 3 additionally apply a global shift, motion-tracked edge holes, and the intermittent patch process. Patches appear only during active events. Nonfinite or nonpositive source pixels remain invalid. Additive perturbations are summed on valid pixels, then the union of edge and patch masks is set to zero. Remaining positive depth is floored at 10^{-3} mm before the usual 0–1500 mm clipping and normalization. Zero denotes missing depth rather than a physical near-plane surface; no extra mask channel is provided to the policy. Resizing follows the normal bilinear preprocessing, so native-resolution hole boundaries can become intermediate-valued pixels in the VAE input.

Training and evaluation. Each corruption model uses PrismWAM with 15k updates of depth-prediction pretraining followed by 15k updates of joint training, global batch size 256, and clean-only demonstrations from all 50 RoboTwin tasks. Corruption is applied in both stages to all three cameras' current conditioning depth and future depth targets. The VAEs remain frozen, and corrupted depth is encoded online. A new corruption seed is drawn for each sampled training clip, while the sampled global shift remains constant within the clip. Motion fields are advanced through consecutive original frames, including intervening frames that are not VAE targets, and repeated timestamps reuse their corrupted observation.

Evaluation uses the final joint-training checkpoint and applies its matching corruption level to *observed* depth during closed-loop rollout. It uses the same ten denoising steps, 24-action replanning interval, unseen instructions, and 50 tasks $\times$ 20 episodes per Clean/Random scene condition as the ablation protocol. No ground-truth future frames are available to the corruption tracker during rollout. This study measures performance after training with each corruption level, rather than zero-shot corruption robustness of a single clean-trained policy.

The figure's Clean/Random scene success rates are 87.40%/54.10% with clean depth, 89.80%/52.70% at Level 1, 87.60%/53.20% at Level 2, and 86.50%/50.10% at Level 3. Their means are 70.75%, 71.25%, 70.40%, and 68.30%. Thus Level 3 is 2.45 percentage points below the clean-depth baseline on the mean. Clean/Random describe scene distributions, while clean or corrupted *depth* describes sensor inputs. The clean-depth baseline here uses the 20-episode evaluation, distinct from the 100-episode main benchmark.

Visualization protocol. The observation panels in fig. 4 use zero-based frame 26 of Adjust Bottle clean demonstration episode 0, with seed 42 and the same episode/camera identifiers across profiles. The preview processes preceding frames in order to initialize tracked fields and renders every panel with a fixed 0–1500 mm color scale, with invalid values shown in black. These are corrupted observations before VAE encoding, not predicted futures.

B PER-TASK BENCHMARK RESULTS

The PrismWAM columns in this appendix report the 5B model. The RoboTwin clean-to-random tables report PrismWAM and Depth-only evaluations with 100 episodes per task. The RoboTwin-Full tables report the completed evaluation with 100 episodes per task and condition; RoboDojo reports the mean over seeds 0, 1, and 2. Following the task-wise comparison in Fast-WAM (Yuan et al., 2026), each row represents one task and each column one model. X-WAM (Guo et al., 2026) uses official task-level results for RoboTwin clean-to-random and RoboDojo, and its paper-reported 100-episode results for RoboTwin-Full. Each table reports one evaluation condition or metric. RoboTwin uses clean/randomized success rates; RoboDojo uses progress score and success rate. LIBERO suite-level results and LIBERO-Plus perturbation-category results appear in Appendix A.2 and tables 10 and 11.

An em dash marks an unavailable result, not zero performance. We do not infer task-level scores from aggregate results, or move results between different training protocols.

B.1 ROBOTWIN: CLEAN-ONLY TRAINING

Both evaluation conditions use 50 tasks with 100 episodes per task. Our clean-only checkpoint is evaluated on clean and fully randomized scenes. OpenWAM-α's per-task results are unavailable. The mean row retains the aggregate precision.

Table 24: Per-task RoboTwin success rates (%) on clean scenes after clean-only training. All methods retain the protocol and precision of their cited source.

Task	Prism WAM (Ours)	Depth-only	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	OLA-Sem	Giga Brain 0.7	4D-WAM	X-WAM
Adjust Bottle	98	98	84	99	99	–	72	98	80	92
Beat Block Hammer	81	58	75	82	93	–	65	94	92	92
Blocks Ranking RGB	87	22	39	81	72	–	64	68	88	85
Blocks Ranking Size	56	50	14	52	45	–	36	42	54	33
Click Alarmclock	100	99	60	99	65	–	100	98	100	95
Click Bell	100	100	75	95	31	–	100	98	98	100
Dump Bin Bigbin	84	79	54	94	92	–	96	90	86	88
Grab Roller	100	100	94	100	99	–	100	100	100	95
Handover Block	99	5	6	66	30	–	24	16	79	21
Handover Mic	95	87	66	98	98	–	89	100	85	97
Hanging Mug	38	14	1	14	17	–	16	26	70	17
Lift Pot	99	93	68	91	98	–	100	86	88	77
Move Can Pot	99	99	33	96	60	–	46	86	94	77
Move Pillbottle Pad	100	95	40	92	46	–	78	32	89	78
Move Playingcard Away	99	97	62	94	84	–	90	58	82	83
Move Stapler Pad	33	15	14	33	20	–	58	26	43	26
Open Laptop	91	91	22	84	93	–	90	78	85	72
Open Microwave	57	4	10	17	86	–	76	82	68	9
Pick Diverse Bottles	93	88	43	72	66	–	76	38	77	74
Pick Dual Bottles	100	99	60	87	72	–	92	26	92	99
Place A2B Left	83	72	24	69	71	–	65	68	71	70
Place A2B Right	80	60	25	60	68	–	85	58	62	59
Place Bread Basket	94	87	82	87	63	–	84	60	87	82
Place Bread Skillet	90	78	79	85	76	–	81	80	82	82
Place Burger Fries	98	96	85	97	82	–	94	82	86	93
Place Can Basket	81	70	4	77	56	–	36	20	83	63
Place Cans Plasticbox	100	100	95	98	31	–	97	8	94	93
Place Container Plate	100	99	87	100	92	–	97	94	93	95
Place Dual Shoes	73	71	30	67	70	–	87	68	66	38
Place Empty Cup	99	97	85	90	96	–	98	90	99	93
Place Fan	88	54	30	59	66	–	76	78	48	47
Place Mouse Pad	89	41	11	55	34	–	59	26	54	49
Place Object Basket	86	50	4	82	70	–	54	74	75	52
Place Object Scale	88	66	20	83	78	–	69	64	79	66
Place Object Stand	97	92	66	94	85	–	81	88	87	83
Place Phone Stand	88	76	18	85	65	–	75	36	84	35
Place Shoe	92	96	46	87	82	–	79	88	83	76
Press Stapler	92	90	91	90	79	–	98	90	76	85
Put Bottles Dustbin	90	73	11	55	78	–	44	68	68	71

Continued on next page

Table 24: RoboTwin – Clean (continued)

Task	Prism WAM (Ours)	Depth-only	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	OLA-Sem	Giga Brain 0.7	4D-WAM	X-WAM
Put Object Cabinet	89	76	29	42	39	–	40	78	69	28
Rotate QRcode	83	65	11	81	90	–	46	54	79	47
Scan Object	94	84	63	68	49	–	42	32	73	32
Shake Bottle Horizontally	100	100	98	100	99	–	92	100	98	97
Shake Bottle	100	100	100	100	100	–	89	100	98	96
Stack Blocks Three	90	6	20	89	77	–	74	60	95	72
Stack Blocks Two	98	42	53	99	91	–	91	80	100	97
Stack Bowls Three	84	62	46	80	81	–	88	70	82	69
Stack Bowls Two	94	89	65	94	94	–	94	90	94	93
Stamp Seal	80	32	12	30	53	–	87	52	74	58
Turn Switch	70	52	16	39	52	–	86	42	86	68
Mean	87.98	71.38	46.52	77.80	70.70	–	75.12	66.80	81.50	70.00

Table 25: Per-task RoboTwin success rates (%) on randomized scenes after clean-only training. The model and task order matches the clean evaluation in table 24.

Task	Prism WAM (Ours)	Depth-only	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	OLA-Sem	Giga Brain 0.7	4D-WAM	X-WAM
Adjust Bottle	54	16	0	0	77	–	71	98	36	13
Beat Block Hammer	48	11	2	0	23	–	55	74	28	21
Blocks Ranking RGB	32	0	2	0	44	–	66	76	10	36
Blocks Ranking Size	27	6	0	0	21	–	37	28	8	0
Click Alarmclock	86	56	0	60	50	–	91	96	70	61
Click Bell	78	56	7	7	35	–	98	84	77	52
Dump Bin Bigbin	80	66	4	0	84	–	94	66	57	40
Grab Roller	89	62	44	2	97	–	100	100	68	16
Handover Block	37	0	0	0	12	–	17	32	25	21
Handover Mic	24	4	1	0	6	–	80	100	13	8
Hanging Mug	17	3	0	0	14	–	15	36	10	0
Lift Pot	67	0	0	0	35	–	91	86	9	0
Move Can Pot	19	1	0	0	10	–	45	82	17	34
Move Pillbottle Pad	19	0	0	0	32	–	71	58	22	26
Move Playingcard Away	65	23	0	0	65	–	92	82	35	11
Move Stapler Pad	8	0	0	0	18	–	53	40	12	2
Open Laptop	50	29	2	0	68	–	74	88	82	6
Open Microwave	19	11	1	0	59	–	43	50	12	24
Pick Diverse Bottles	78	48	15	0	42	–	77	50	44	26
Pick Dual Bottles	96	52	20	1	57	–	84	56	67	55
Place A2B Left	39	18	0	0	49	–	64	72	19	18
Place A2B Right	41	8	1	1	39	–	80	68	42	21
Place Bread Basket	77	24	5	0	60	–	80	68	77	40
Place Bread Skillet	66	23	2	0	47	–	67	50	69	37
Place Burger Fries	90	29	11	0	86	–	75	86	72	36
Place Can Basket	31	1	0	0	10	–	34	22	51	15
Place Cans Plasticbox	95	40	2	0	73	–	85	18	48	36
Place Container Plate	68	64	3	0	72	–	92	94	82	52
Place Dual Shoes	43	51	2	0	45	–	76	72	19	7
Place Empty Cup	70	43	0	0	80	–	94	90	70	36
Place Fan	42	3	1	0	26	–	68	74	17	6
Place Mouse Pad	29	1	0	0	31	–	55	44	9	6
Place Object Basket	44	0	1	0	29	–	49	64	40	20
Place Object Scale	37	9	1	0	38	–	63	54	24	14
Place Object Stand	76	39	0	0	47	–	77	76	42	28
Place Phone Stand	47	8	0	0	37	–	63	52	21	4
Place Shoe	55	20	0	0	50	–	78	92	52	31
Press Stapler	50	31	10	3	63	–	95	74	61	20
Put Bottles Dustbin	82	46	0	0	50	–	37	54	19	21
Put Object Cabinet	28	6	0	0	22	–	27	82	47	16
Rotate QRcode	26	4	0	0	21	–	40	58	67	2
Scan Object	72	55	5	0	40	–	32	46	53	14
Shake Bottle Horizontally	80	73	9	9	100	–	70	92	70	66
Shake Bottle	79	70	5	10	100	–	70	98	72	67
Stack Blocks Three	33	0	1	0	25	–	52	78	82	26
Stack Blocks Two	60	14	1	0	43	–	90	92	34	76
Stack Bowls Three	59	34	0	0	52	–	81	64	27	38
Stack Bowls Two	74	62	0	0	70	–	90	74	81	67
Stamp Seal	23	1	0	0	21	–	70	54	10	16
Turn Switch	49	17	0	3	25	–	70	52	11	4
Mean	53.16	24.76	3.16	1.90	46.00	–	67.56	67.90	41.80	25.80

B.2 RoboTwin-Full: Mixed Clean/Randomized Training

These tables use the mixed-training protocol, distinct from the clean-only evaluation above. The evaluation is complete: each condition contains 50 tasks with 100 episodes per task. Clean and randomized success rates are 4737/5000 (94.74%) and 4740/5000 (94.80%), respectively, averaging 94.77%, consistent with the main table. All PrismWAM task evaluations are complete; em dashes denote unavailable baseline entries. Aggregate benchmark results are not used to infer per-task values.

Table 26: Per-task RoboTwin-Full success rates (%) on clean scenes. PrismWAM uses 50 tasks with 100 episodes each. Em dashes denote unavailable external results.

Task	Prism WAM (Ours)	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	ABot M0.5	Qwen Robot Manip	LingBot VA 2.0	X-WAM
Adjust Bottle	100	–	–	–	–	–	–	–	100
Beat Block Hammer	98	–	–	–	–	–	–	–	98
Blocks Ranking RGB	96	–	–	–	–	–	–	–	99
Blocks Ranking Size	80	–	–	–	–	–	–	–	76
Click Alarmclock	100	–	–	–	–	–	–	–	98
Click Bell	100	–	–	–	–	–	–	–	100
Dump Bin Bigbin	93	–	–	–	–	–	–	–	90
Grab Roller	100	–	–	–	–	–	–	–	100
Handover Block	98	–	–	–	–	–	–	–	88
Handover Mic	97	–	–	–	–	–	–	–	88
Hanging Mug	73	–	–	–	–	–	–	–	46
Lift Pot	100	–	–	–	–	–	–	–	100
Move Can Pot	100	–	–	–	–	–	–	–	80
Move Pillbottle Pad	100	–	–	–	–	–	–	–	96
Move Playingcard Away	100	–	–	–	–	–	–	–	100
Move Stapler Pad	93	–	–	–	–	–	–	–	67
Open Laptop	100	–	–	–	–	–	–	–	96
Open Microwave	51	–	–	–	–	–	–	–	89
Pick Diverse Bottles	95	–	–	–	–	–	–	–	91
Pick Dual Bottles	100	–	–	–	–	–	–	–	99
Place A2B Left	97	–	–	–	–	–	–	–	90
Place A2B Right	96	–	–	–	–	–	–	–	92
Place Bread Basket	94	–	–	–	–	–	–	–	90
Place Bread Skillet	91	–	–	–	–	–	–	–	90
Place Burger Fries	99	–	–	–	–	–	–	–	97
Place Can Basket	82	–	–	–	–	–	–	–	84
Place Cans Plasticbox	100	–	–	–	–	–	–	–	99
Place Container Plate	98	–	–	–	–	–	–	–	98
Place Dual Shoes	99	–	–	–	–	–	–	–	83
Place Empty Cup	100	–	–	–	–	–	–	–	98
Place Fan	100	–	–	–	–	–	–	–	84
Place Mouse Pad	99	–	–	–	–	–	–	–	84
Place Object Basket	88	–	–	–	–	–	–	–	85
Place Object Scale	100	–	–	–	–	–	–	–	93
Place Object Stand	100	–	–	–	–	–	–	–	97
Place Phone Stand	92	–	–	–	–	–	–	–	75
Place Shoe	100	–	–	–	–	–	–	–	97
Press Stapler	96	–	–	–	–	–	–	–	94
Put Bottles Dustbin	94	–	–	–	–	–	–	–	85
Put Object Cabinet	90	–	–	–	–	–	–	–	66
Rotate QRcode	93	–	–	–	–	–	–	–	84
Scan Object	100	–	–	–	–	–	–	–	86
Shake Bottle Horizontally	100	–	–	–	–	–	–	–	100
Shake Bottle	100	–	–	–	–	–	–	–	99
Stack Blocks Three	97	–	–	–	–	–	–	–	97
Stack Blocks Two	100	–	–	–	–	–	–	–	100
Stack Bowls Three	84	–	–	–	–	–	–	–	88
Stack Bowls Two	96	–	–	–	–	–	–	–	98
Stamp Seal	100	–	–	–	–	–	–	–	93
Turn Switch	78	–	–	–	–	–	–	–	61
Mean	**94.74**	–	–	–	–	–	–	–	89.80

Table 27: Per-task RoboTwin-Full success rates (%) on randomized scenes. PrismWAM uses 50 tasks with 100 episodes each. Em dashes denote unavailable external results.

Task	Prism WAM (Ours)	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	ABot M0.5	Qwen Robot Manip	LingBot VA 2.0	X-WAM
Adjust Bottle	100	–	–	–	–	–	–	–	99
Beat Block Hammer	98	–	–	–	–	–	–	–	96
Blocks Ranking RGB	91	–	–	–	–	–	–	–	95
Blocks Ranking Size	80	–	–	–	–	–	–	–	82
Click Alarmclock	100	–	–	–	–	–	–	–	99
Click Bell	100	–	–	–	–	–	–	–	100
Dump Bin Bigbin	92	–	–	–	–	–	–	–	96
Grab Roller	100	–	–	–	–	–	–	–	100
Handover Block	96	–	–	–	–	–	–	–	79
Handover Mic	100	–	–	–	–	–	–	–	89
Hanging Mug	75	–	–	–	–	–	–	–	55
Lift Pot	100	–	–	–	–	–	–	–	99
Move Can Pot	99	–	–	–	–	–	–	–	84
Move Pillbottle Pad	100	–	–	–	–	–	–	–	98
Move Playingcard Away	100	–	–	–	–	–	–	–	98
Move Stapler Pad	95	–	–	–	–	–	–	–	70
Open Laptop	100	–	–	–	–	–	–	–	97
Open Microwave	46	–	–	–	–	–	–	–	92
Pick Diverse Bottles	95	–	–	–	–	–	–	–	92
Pick Dual Bottles	100	–	–	–	–	–	–	–	100
Place A2B Left	99	–	–	–	–	–	–	–	87
Place A2B Right	97	–	–	–	–	–	–	–	89
Place Bread Basket	95	–	–	–	–	–	–	–	91
Place Bread Skillet	91	–	–	–	–	–	–	–	96
Place Burger Fries	98	–	–	–	–	–	–	–	99
Place Can Basket	82	–	–	–	–	–	–	–	82
Place Cans Plasticbox	100	–	–	–	–	–	–	–	98
Place Container Plate	98	–	–	–	–	–	–	–	100
Place Dual Shoes	92	–	–	–	–	–	–	–	81
Place Empty Cup	100	–	–	–	–	–	–	–	99
Place Fan	95	–	–	–	–	–	–	–	92
Place Mouse Pad	99	–	–	–	–	–	–	–	86
Place Object Basket	93	–	–	–	–	–	–	–	87
Place Object Scale	98	–	–	–	–	–	–	–	89
Place Object Stand	100	–	–	–	–	–	–	–	96
Place Phone Stand	96	–	–	–	–	–	–	–	80
Place Shoe	100	–	–	–	–	–	–	–	99
Press Stapler	94	–	–	–	–	–	–	–	90
Put Bottles Dustbin	96	–	–	–	–	–	–	–	95
Put Object Cabinet	95	–	–	–	–	–	–	–	76
Rotate QRcode	95	–	–	–	–	–	–	–	83
Scan Object	98	–	–	–	–	–	–	–	79
Shake Bottle Horizontally	100	–	–	–	–	–	–	–	99
Shake Bottle	100	–	–	–	–	–	–	–	99
Stack Blocks Three	98	–	–	–	–	–	–	–	95
Stack Blocks Two	100	–	–	–	–	–	–	–	100
Stack Bowls Three	85	–	–	–	–	–	–	–	82
Stack Bowls Two	95	–	–	–	–	–	–	–	98
Stamp Seal	100	–	–	–	–	–	–	–	95
Turn Switch	84	–	–	–	–	–	–	–	72
Mean	**94.80**	–	–	–	–	–	–	–	90.70

B.3 RoboDojo: Simulation

The current checkpoint is evaluated over all 42 canonical simulation tasks, represented by 54 standard/randomized evaluation entries per seed. Generalization uses 25 episodes per standard/randomized entry; the remaining tasks use 50 episodes each, totaling 2100 episodes per seed and 6300 episodes across seeds 0, 1, and 2. Progress scores (/100) and success rates (%) are reported in separate tables, each with one column per model. Our evaluation averages three seeds and uses simulator depth; protocol differences preclude a matched-input comparison. The compared methods follow table 1.

Randomized Generalization results for GPT-6-Astra are unavailable. The corresponding cells are left unreported; standard-task results are not substituted for randomized-task results. The final row of each table reports the official five-category aggregate: standard/randomized results are first combined inside Generalization, then the five categories are averaged equally. Averaging seeds 0, 1, and 2 yields 20.73 score / 14.48% SR; 938/6300 = 14.89% is the separate episode-pooled success rate.

Table 28: Per-task RoboDojo progress scores (/100). PrismWAM reports the mean over seeds 0, 1, and 2; 4D-WAM reports the completed seed-0 self-run; remaining external values are transcribed from the official task-level leaderboard snapshot.

Task	Prism WAM (Ours)	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	Liber-0 Preview	Liber-0 Lite	GPT-6 Astra	4D-WAM	X-WAM
Generalization – standard										
stack bowls	81.67	15.27	5.87	76.00	89.60	83.93	87.33	66.50	67.60	39.87
push T	0.00	0.00	0.00	0.00	0.00	0.00	0.00	60.00	0.00	0.00
pack objects into box	21.87	8.07	2.87	22.80	30.67	30.60	35.53	11.60	27.40	16.20
fold clothes	65.33	23.47	32.00	48.00	88.27	81.33	74.67	76.40	80.00	25.33
hang mugs	39.40	4.33	2.20	9.53	29.40	33.13	33.47	0.90	9.80	12.73
sweep blocks	2.67	1.33	0.00	0.00	2.67	6.67	2.67	4.00	0.00	0.00
pour liquid into cup	60.00	18.67	0.00	28.00	54.67	57.33	45.33	4.00	0.00	6.67
make toast	23.67	3.67	2.00	9.33	16.00	19.67	27.67	0.50	15.00	7.00
arrange largest number	8.40	1.20	0.80	5.13	7.80	28.47	20.20	68.70	7.60	2.47
sort nesting dolls by size	10.67	1.33	0.00	10.67	9.33	17.33	29.33	18.00	12.00	4.00
store laptop and headphones	28.00	4.80	0.80	19.20	44.27	37.60	44.80	1.60	18.40	8.53
stack blocks	49.27	8.33	5.40	22.53	25.27	75.60	76.93	88.10	51.80	12.13
Generalization – randomized										
stack bowls (Rand.)	32.00	1.20	0.80	14.20	10.27	19.07	12.07	–	3.00	7.07
push T (Rand.)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
pack objects into box (Rand.)	17.60	1.07	0.27	15.40	14.27	16.93	16.20	–	4.20	7.67
fold clothes (Rand.)	12.53	0.53	2.13	16.53	26.67	8.53	8.27	–	7.20	11.73
hang mugs (Rand.)	8.33	0.80	0.00	5.00	3.20	4.33	6.47	–	2.40	1.00
sweep blocks (Rand.)	1.33	0.00	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
pour liquid into cup (Rand.)	17.33	0.00	0.00	1.33	14.67	12.00	13.33	–	0.00	5.33
make toast (Rand.)	16.67	0.00	0.67	2.67	9.00	16.00	19.67	–	5.00	6.67
arrange largest number (Rand.)	1.60	0.13	0.27	0.40	2.07	7.27	5.67	–	0.80	1.00
sort nesting dolls by size (Rand.)	0.00	0.00	0.00	0.00	0.00	0.00	1.33	–	0.00	0.00
store laptop and headphones (Rand.)	39.20	0.27	0.00	13.07	16.80	38.67	42.93	–	6.40	0.80
stack blocks (Rand.)	4.27	0.00	0.00	1.20	2.20	5.20	4.60	–	1.80	1.20
Precision										
fasten screws	36.27	1.60	0.40	15.13	24.67	61.80	60.47	36.40	26.20	13.20
insert tubes	72.40	13.47	3.73	17.87	26.27	82.53	81.07	6.00	38.80	16.80
plug in charger	7.33	0.00	0.00	0.00	4.00	27.33	18.67	0.00	0.00	0.00
pour balls into vase	40.00	5.33	0.00	13.33	17.33	30.67	26.67	4.00	2.00	2.67
play Xylophone	2.00	0.00	0.00	0.00	0.00	0.00	0.00	8.00	0.00	6.67
deposit coin	17.33	5.20	2.80	3.20	9.07	18.67	21.87	16.00	11.20	3.07
insert key	14.20	11.50	4.30	11.90	13.70	0.90	0.70	14.40	9.60	6.10
build tower	53.40	42.07	4.47	37.73	52.53	84.33	83.40	16.40	56.80	5.27
Long-horizon										
put bottles into dustbin	95.10	48.90	57.37	79.93	94.03	97.90	98.00	35.30	83.10	79.90
play tic tac toe	64.03	2.40	0.00	8.23	64.07	95.67	94.53	10.70	10.60	11.07
classify objects	6.97	2.90	3.10	24.67	5.53	19.73	20.67	69.00	4.80	6.13
fill pen holder	55.90	17.57	4.07	23.23	15.37	49.77	50.53	5.00	31.40	7.10
fill egg holder	4.83	1.13	0.73	2.23	5.80	12.57	14.27	5.50	5.00	3.33
organize table	56.83	22.33	7.83	23.33	62.50	47.50	45.00	39.50	25.50	28.83
play stacking toy	0.73	0.00	0.00	0.00	0.13	26.73	23.67	6.60	0.00	0.07

Continued on next page

Table 28: RoboDojo – Progress score (continued)

Task	Prism WAM (Ours)	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	Liber-0 Preview	Liber-0 Lite	GPT-6 Astra	4D-WAM	X-WAM
make kong	27.33	18.00	0.00	26.67	32.00	18.00	17.33	0.00	26.00	3.33
Memory										
cover blocks	25.57	14.73	0.00	19.07	22.23	100.00	100.00	49.30	15.00	13.77
match and pick from conveyor	32.00	4.67	20.67	14.00	36.67	64.00	60.67	62.00	28.00	22.00
swap T	1.33	0.00	0.00	0.00	0.00	53.33	50.67	18.00	0.00	0.00
press by number	1.33	0.00	0.00	0.00	0.00	0.00	0.00	70.00	0.00	0.67
imitate sorting sequence	2.47	0.63	0.63	1.60	2.90	8.63	2.77	58.90	1.30	0.80
swap blocks	0.00	0.00	0.00	0.00	0.67	0.67	0.00	0.00	0.00	0.67
Open										
align blocks	0.00	0.00	0.00	0.00	0.00	0.00	0.00	50.00	0.00	0.00
solve equation	0.00	0.00	0.00	0.00	0.00	0.00	0.00	40.00	0.00	0.00
stack blocks by language	4.27	0.53	0.00	1.73	0.93	15.60	4.40	48.00	2.40	1.33
general pickup	4.00	4.67	3.33	12.00	8.67	34.00	16.67	84.00	8.00	1.33
classify objects by language	2.13	0.20	0.00	0.60	1.33	3.87	3.27	46.00	0.60	1.07
pick from conveyor by image	0.00	0.00	0.00	1.33	0.00	0.00	0.00	4.00	0.00	0.67
store tools in toolbox	0.00	0.00	0.00	0.17	0.33	0.00	0.17	2.50	0.50	0.17
pour by language	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.00
Overall (five categories)	20.73	6.40	3.48	11.41	17.18	30.74	29.24	28.97	12.71	7.69

Table 29: Per-task RoboDojo success rates (%). PrismWAM reports the mean over seeds 0, 1, and 2; 4D-WAM reports the completed seed-0 self-run; remaining external values retain the official leaderboard precision.

Task	Prism WAM (Ours)	starVLA	Fast WAM	$\pi_{0.5}$	Open WAM-α	Liber-0 Preview	Liber-0 Lite	GPT-6 Astra	4D-WAM	X-WAM
Generalization – standard										
stack bowls	78.67	10.67	2.67	72.00	88.00	81.00	85.00	62.00	64.00	30.67
push T	0.00	0.00	0.00	0.00	0.00	0.00	0.00	60.00	0.00	0.00
pack objects into box	2.67	2.67	0.00	4.00	9.33	12.00	19.00	0.00	4.00	0.00
fold clothes	61.33	16.00	22.67	38.67	86.67	79.00	72.00	72.00	76.00	16.00
hang mugs	17.33	0.00	0.00	0.00	10.67	13.00	17.00	0.00	4.00	2.67
sweep blocks	2.67	1.33	0.00	0.00	2.67	7.00	3.00	4.00	0.00	0.00
pour liquid into cup	60.00	18.67	0.00	28.00	54.67	57.00	45.00	4.00	0.00	6.67
make toast	8.00	0.00	0.00	1.33	4.00	8.00	15.00	0.00	0.00	1.33
arrange largest number	2.67	0.00	0.00	1.33	1.33	19.00	12.00	60.00	0.00	0.00
sort nesting dolls by size	10.67	1.33	0.00	10.67	9.33	17.00	29.00	18.00	12.00	4.00
store laptop and headphones	10.67	0.00	0.00	9.33	25.33	20.00	28.00	0.00	4.00	1.33
stack blocks	42.67	5.33	0.00	13.33	14.67	72.00	73.00	86.00	44.00	1.33
Generalization – randomized										
stack bowls (Rand.)	24.00	0.00	0.00	4.00	2.67	11.00	3.00	–	0.00	2.67
push T (Rand.)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
pack objects into box (Rand.)	0.00	0.00	0.00	1.33	0.00	4.00	3.00	–	0.00	0.00
fold clothes (Rand.)	8.00	0.00	1.33	9.33	24.00	8.00	7.00	–	4.00	8.00
hang mugs (Rand.)	1.33	0.00	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
sweep blocks (Rand.)	1.33	0.00	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
pour liquid into cup (Rand.)	17.33	0.00	0.00	1.33	14.67	12.00	13.00	–	0.00	5.33
make toast (Rand.)	4.00	0.00	0.00	0.00	1.33	5.00	9.00	–	0.00	0.00
arrange largest number (Rand.)	0.00	0.00	0.00	0.00	0.00	1.00	1.00	–	0.00	0.00
sort nesting dolls by size (Rand.)	0.00	0.00	0.00	0.00	0.00	0.00	1.00	–	0.00	0.00
store laptop and headphones (Rand.)	26.67	0.00	0.00	1.33	6.67	20.00	23.00	–	0.00	0.00
stack blocks (Rand.)	2.67	0.00	0.00	0.00	0.00	0.00	0.00	–	0.00	0.00
Precision										
fasten screws	12.00	0.00	0.00	2.67	4.00	41.00	37.00	16.00	6.00	0.67
insert tubes	60.67	2.00	0.00	3.33	6.67	73.00	72.00	0.00	12.00	2.67
plug in charger	7.33	0.00	0.00	0.00	4.00	27.00	19.00	0.00	0.00	0.00
pour balls into vase	40.00	5.33	0.00	13.33	17.33	31.00	27.00	4.00	2.00	2.67
play Xylophone	2.00	0.00	0.00	0.00	0.00	0.00	0.00	8.00	0.00	6.67
deposit coin	9.33	1.33	0.00	0.67	3.33	12.00	15.00	2.00	6.00	0.00
insert key	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
build tower	39.33	26.00	0.00	24.00	38.67	81.00	80.00	2.00	54.00	2.00
Long-horizon										
put bottles into dustbin	92.67	28.00	40.00	69.33	91.33	97.00	97.00	10.00	76.00	68.67
play tic tac toe	33.33	0.00	0.00	1.33	58.67	89.00	86.00	0.00	0.00	0.00
classify objects	2.00	0.00	1.33	12.67	1.33	9.00	11.00	56.00	0.00	0.67
fill pen holder	32.67	6.00	0.00	7.33	6.00	25.00	31.00	0.00	10.00	0.00
fill egg holder	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
organize table	14.00	0.00	0.00	0.00	13.33	11.00	7.00	0.00	0.00	0.00
play stacking toy	0.00	0.00	0.00	0.00	0.00	16.00	15.00	0.00	0.00	0.00
make kong	27.33	18.00	0.00	26.67	32.00	18.00	17.00	0.00	26.00	3.33
Memory										
cover blocks	19.33	10.00	0.00	13.33	16.67	100.00	100.00	46.00	8.00	4.67
match and pick from conveyor	32.00	4.67	20.67	14.00	36.67	64.00	61.00	62.00	28.00	22.00
swap T	1.33	0.00	0.00	0.00	0.00	53.00	51.00	18.00	0.00	0.00
press by number	1.33	0.00	0.00	0.00	0.00	0.00	0.00	70.00	0.00	0.67
imitate sorting sequence	0.00	0.00	0.00	0.00	0.67	6.00	1.00	36.00	0.00	0.00
swap blocks	0.00	0.00	0.00	0.00	0.67	1.00	0.00	0.00	0.00	0.67
Open										
align blocks	0.00	0.00	0.00	0.00	0.00	0.00	0.00	50.00	0.00	0.00
solve equation	0.00	0.00	0.00	0.00	0.00	0.00	0.00	40.00	0.00	0.00
stack blocks by language	2.00	0.00	0.00	0.00	0.00	7.00	3.00	40.00	0.00	0.00
general pickup	4.00	4.67	3.33	12.00	8.67	34.00	17.00	84.00	8.00	1.33
classify objects by language	1.33	0.00	0.00	0.00	0.00	1.00	1.00	30.00	0.00	0.00
pick from conveyor by image	0.00	0.00	0.00	1.33	0.00	0.00	0.00	4.00	0.00	0.67
store tools in toolbox	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pour by language	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Overall (five categories)	14.48	3.24	2.03	6.91	11.92	25.52	24.23	22.48	7.97	3.83