COGWAM: ALIGNING SEMANTIC COGNITION WITH WORLD ACTION MODELING VIA EVENT-DRIVEN INTERFACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot policies increasingly incorporate semantic reasoning and future-world prediction, yet combining these capabilities does not guarantee that local predictions and actions remain aligned with task progress. We introduce CogWAM, a cognition-guided world-action model that establishes an explicit semantic interface between task reasoning and world-action learning through a persistent Semantic State, which stores completed task events and the active subtask. CogWAM updates this state only when observations indicate semantic transitions, allowing task-level context to persist across multiple action chunks. To bridge semantic context with physical prediction and control, CogWAM employs progress-conditioned WORLD and ACTION queries that selectively extract task-relevant information for future-world prediction and action generation. During training, the Semantic State provides shared task-progress context for both branches, while inference removes the future-prediction branch and directly generates actions from observations and the maintained state. We further introduce semantic training strategies to improve transition learning and closed-loop conditioning. Without additional robot-action pretraining, CogWAM achieves 15.56 / 11.70% Score/SR on RoboDojo and state-of-the-art performance on BiCoord, while real-world experiments demonstrate closed-loop dual-arm manipulation with 16.4× fewer Semantic State regenerations than step-wise updating.

1 INTRODUCTION

The promise of physical intelligence lies in embodied agents that can transform abstract intentions into purposeful changes in the physical world (Ahn et al., 2022; Brooks, 1991; Driess et al., 2023; Black et al., 2025). Doing so requires more than predicting isolated actions. As interaction unfolds, a robot needs task-level cognition to relate what has already been accomplished to what should happen next, while grounding its evolving intent in physical interaction (Huang et al., 2022; Shi et al., 2025). Robot policies should therefore connect task semantics, execution progress, and physical dynamics within a unified closed loop.

Current robot foundation models have advanced complementary parts of this loop. Vision-language-action models (VLAs) leverage large-scale vision-language pretraining to achieve strong semantic understanding and multi-task generalization (Kim et al., 2025; Black et al., 2024). Recent approaches further introduce high-level reasoning, temporal memory, and subtask prediction to better track execution progress (Shi et al., 2025; 2026; Yang et al., 2025; Wang et al., 2026; Su et al., 2026). In parallel, world-action models (WAMs) augment action learning with predictions of future world representations, providing supervision for spatial change and interaction dynamics (Zhu et al., 2025; Ye et al., 2026b; Lyu et al., 2026; Chen et al., 2026a; Zhao et al., 2026; Li et al., 2026c; Zhang et al., 2026b). Together, these directions address what should be accomplished next and how physical interaction may evolve. However, incorporating both capabilities does not by itself specify how task progress should be represented, maintained, and shared between them. A locally plausible scene prediction or action may still correspond to a subtask that has already been completed or has not yet been reached. This motivates a shared representation of task progress that both branches can condition on and that is revised in response to observed progress.

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.

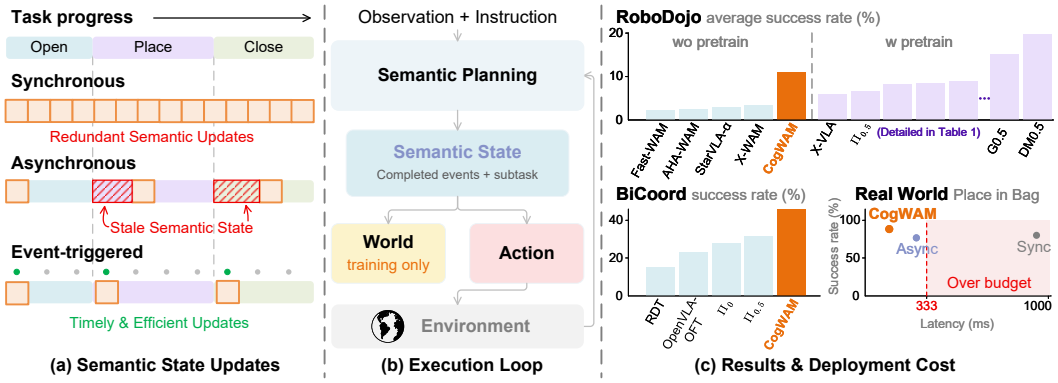


Figure 1: **Overview of CogWAM.** (a) Event-triggered updates maintain a persistent Semantic State while avoiding redundant updates and stale states. (b) The Semantic State couples semantic planning with world prediction and action generation through a closed-loop execution loop. (c) Benchmark results and real-world success–latency trade-off.

To address this alignment problem, we introduce CogWAM (Fig. 1), which explicitly represents task progress as a persistent cognitive state rather than repeatedly inferring it from individual observations Yang et al. (2026). Existing hierarchical and memory-based policies demonstrate the importance of maintaining temporal context for long-horizon execution (Shi et al., 2025; 2026; Torne et al., 2026; Yang et al., 2025), yet they primarily focus on semantic reasoning or memory accumulation without explicitly coupling task progress with predictive world modeling and action generation. CogWAM addresses this gap through a persistent Semantic State that records completed task events and the active subtask. During execution, the state is updated only when observations indicate meaningful task transitions, allowing semantic context to persist across multiple action chunks. This event-driven formulation decouples the semantic timescale from the control timescale: actions require frequent replanning for reactive control (Zhao et al., 2023), whereas task semantics evolve only at a small number of stage transitions (Torne et al., 2026; Sutton et al., 1999).

Beyond maintaining task progress, an effective interface must also determine how semantic context is transferred to different physical objectives. Future-world prediction and action generation optimize different objectives, requiring representations that capture scene evolution and executable control. CogWAM therefore introduces progress-conditioned WORLD and ACTION queries, which are conditioned on the same Semantic State while selectively extracting branch-specific information for future multi-view latent prediction and continuous action generation. This allows task-level semantics to be shared while preserving objective-specific representations Li et al. (2023).

This state-conditioned formulation introduces two training challenges. First, semantic transitions are sparse relative to state persistence, causing imbalance between UPDATE and KEEP decisions. Second, training with annotated states creates exposure bias because closed-loop execution relies on predicted states Bengio et al. (2015). We address these issues with boundary-aware semantic supervision and progressive semantic conditioning. Our main contributions are summarized as follows:

1. We present CogWAM, a semantics-aware world-action architecture that couples a persistent Semantic State with progress-conditioned WORLD/ACTION queries, enabling task-progress-aware future prediction and control.
2. We develop event-triggered semantic planning, which decouples semantic evolution from control frequency, together with training strategies that address sparse transitions and closed-loop semantic conditioning.
3. We demonstrate that explicit semantic interface design consistently improves robot policies across simulation, real-world, and different world-action architectures.

2 Related Work

Vision-Language-Action Models and Long-Horizon Reasoning. Vision-Language-Action (VLA) models leverage pretrained vision-language representations for language-conditioned robotic control (Brohan et al., 2023; Kim et al., 2024; 2025; Zheng et al., 2026). Recent work increasingly augments such policies with higher-level reasoning and temporal context. $\pi_{0.5}$ incorporates high-

level semantic prediction to support open-world and long-horizon manipulation (Black et al., 2025), while Hi Robot explicitly decomposes high-level semantic reasoning and low-level visuomotor execution (Shi et al., 2025). MemoryVLA introduces perceptual–cognitive memory for temporally dependent control (Shi et al., 2026), and Mem maintains multi-scale embodied memory across an episode (Torne et al., 2026). SeqVLA explicitly predicts subtask completion to control transitions during sequential manipulation (Yang et al., 2025). These works demonstrate the importance of reasoning beyond the current observation and global instruction. CogWAM similarly models execution progress, but differs by explicitly transferring maintained task progress into world-action learning through a structured semantic interface.

World-Action Models and Predictive Robot Policies. WAMs augment action learning with predictive models of future scene evolution Du et al. (2023). Unified World Models jointly model future observations and actions (Zhu et al., 2025), while DreamZero scales this paradigm and demonstrates strong zero-shot physical generalization (Ye et al., 2026b). Recent studies increasingly question whether dense future video generation is necessary. Fast-WAM shows that world-model co-training can provide substantial benefits even without explicit future generation at inference time (Yuan et al., 2026). In parallel, recent approaches move toward compact, action-relevant future representations: LaWAM predicts latent visual subgoals that condition robot actions (Chen et al., 2026a), whereas SG-WAM models future dynamics directly in a geometry-aware policy representation space (Zhao et al., 2026). CogWAM follows this trend toward control-relevant future representations, while conditioning future prediction on an explicit execution-aware Semantic State.

Task Progress across Semantic and Control Timescales. A related challenge is the mismatch between task-level semantics and the finer temporal scale of physical control. Hierarchical VLAs address this issue through intermediate semantic commands (Shi et al., 2025; Black et al., 2025), while completion-aware approaches track task progress across action chunks (Yang et al., 2025). WALL-WM further organizes world-action learning around semantically coherent events to address the granularity mismatch among language, visual dynamics, and fixed-length actions (Li et al., 2026d). CogWAM instead decouples Semantic State transitions from fixed-horizon action generation: the state persists across multiple replanning cycles and is updated only when observed progress indicates a semantic transition. This formulation is related to temporal abstraction in reinforcement learning (Sutton et al., 1999), but operates on a language-level task state whose updates are event-triggered by execution progress (Tabuada, 2007).

3 METHOD

3.1 PROBLEM FORMULATION

We consider language-conditioned robot manipulation from multi-view observations. At timestep t, let $\mathcal{O}_t = \{\boldsymbol{I}_t^{(v)}\}_{v=1}^{V}$ denote the current observation, ℓ the task instruction, and $\boldsymbol{a}_t$ an action chunk of horizon H. A standard visuomotor policy models $p_\theta(\boldsymbol{a}_t \mid \mathcal{O}_t, \ell)$.

For long-horizon manipulation, however, the current observation may not reveal which task events have already been completed or which subtask is currently active. CogWAM therefore maintains a persistent Semantic State $S_{t^-} = (m_{t^-}, s_{t^-})$, where m_{t^-} records completed task events and s_{t^-} denotes the active subtask. Given $\mathcal{O}_t$, ℓ, and S_{t^-}, the model first determines the Semantic State S_t consistent with the current task progress.

During training, CogWAM additionally predicts a future multi-view latent representation $\mathcal{Z}_{t+H} = \{\boldsymbol{z}_{t+H}^{(v)}\}_{v=1}^{V}$. We factorize the conditional distribution as

$$p_\theta(S_t, \mathcal{Z}_{t+H}, \boldsymbol{a}_t \mid \mathcal{O}_t, \ell, S_{t^-}) = p_\theta(S_t \mid \mathcal{O}_t, \ell, S_{t^-})\, p_\theta(\mathcal{Z}_{t+H}, \boldsymbol{a}_t \mid \mathcal{O}_t, \ell, S_t)\,. \tag{1}$$

The first factor models task-progress transitions, while the second models progress-conditioned world prediction and action generation. Importantly, action generation does not depend on the predicted future representation; future-world prediction is used only as a training-time objective. At inference time, the world branch is removed, and CogWAM directly predicts $p_\theta(\boldsymbol{a}_t \mid \mathcal{O}_t, \ell, S_t)$ from the current observation and maintained Semantic State. The overall architecture is shown in Fig. 2.

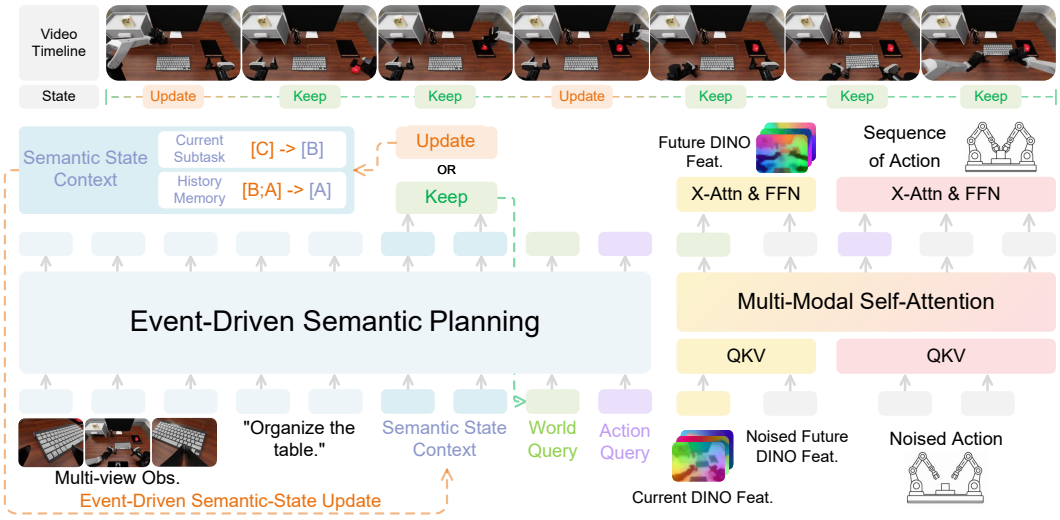


Figure 2: **Architecture of CogWAM.** An event-triggered KEEP/UPDATE mechanism maintains a persistent Semantic State during closed-loop execution. WORLD and ACTION queries condition parallel branches for future multi-view DINO feature prediction and continuous action generation.

3.2 Event-Triggered Semantic State

CogWAM maintains a persistent Semantic State throughout task execution and updates it only when visual evidence indicates a change in task progress. As a language-level summary of execution progress, the Semantic State preserves task history that may no longer be recoverable from the current observation. Instead of regenerating semantic context at every step Yang et al. (2026), CogWAM separates transition detection from state generation.

Semantic-State Transition. Let $d_t \in \{\texttt{KEEP}, \texttt{UPDATE}\}$ denote whether the stored Semantic State remains valid at timestep t. Given the current observation $\mathcal{O}_t$, task instruction ℓ, and previous state S_{t^-}, the VLM Li et al. (2026b) predicts:

$$d_t \sim p_\theta(d_t \mid \mathcal{O}_t, \ell, S_{t^-}) . \tag{2}$$

When $d_t = \texttt{UPDATE}$, the model generates a memory increment Δm_t describing the newly completed task event and the updated active subtask s_t:

$$(\Delta m_t, s_t) \sim p_\theta(\Delta m_t, s_t \mid \mathcal{O}_t, \ell, S_{t^-}, d_t = \texttt{UPDATE}) . \tag{3}$$

The resulting Semantic State is updated as:

$$S_t = \begin{cases} S_{t^-}, & d_t = \texttt{KEEP}, \\ (m_{t^-} \oplus \Delta m_t, s_t), & d_t = \texttt{UPDATE}, \end{cases} \tag{4}$$

where $\oplus$ appends the newly completed event to the stored memory. Thus, d_t determines when the Semantic State changes, while $(\Delta m_t, s_t)$ determines how it changes. The state can persist across multiple action chunks until observed progress supports a transition, decoupling semantic evolution from fixed control horizons. At inference, d_t is predicted in a single forward pass. A KEEP decision avoids autoregressive semantic generation, whereas an UPDATE decision additionally generates $(\Delta m_t, s_t)$. Details of semantic-state supervision are provided in Appendix C.

3.3 Progress-Conditioned World–Action Interface

Maintaining task progress alone does not determine how semantic context should influence different physical prediction objectives. Future-world prediction and action generation capture complementary aspects of robot behavior: the former models scene evolution, while the latter generates executable control trajectories. CogWAM therefore introduces progress-conditioned WORLD and ACTION queries to transfer the same task-progress context to both branches while preserving objective-specific representations.

Given the maintained Semantic State S_t, CogWAM appends two sets of learnable query tokens after the current observation, task instruction, and semantic context in the causal sequence. The resulting hidden representations $\boldsymbol{H}_t^w$ and $\boldsymbol{H}_t^a$ are generated by the WORLD and ACTION queries:

$$\boldsymbol{H}_t^w, \boldsymbol{H}_t^a = f_\theta(\mathcal{O}_t, \ell, S_t, Q^w, Q^a). \tag{5}$$

The causal ordering allows both query sets to attend to the current observation, instruction, and Semantic State, while their outputs are routed to different downstream objectives. Specifically, $\boldsymbol{H}_t^w$ conditions future multi-view latent prediction, whereas $\boldsymbol{H}_t^a$ conditions continuous action generation. Their branch-specific supervision encourages the WORLD and ACTION queries to extract task-relevant information for their respective prediction targets, enabling shared task-level semantics with specialized physical representations. This design is supported by the gradient analysis in Appendix A.2.1, which shows that World and Action objectives share perceptual grounding but require increasingly distinct representations at the query readout stage.

3.4 Latent World–Action Modeling

Given the progress-conditioned representations $\boldsymbol{H}_t^w$ and $\boldsymbol{H}_t^a$, CogWAM jointly learns future-world prediction and continuous action generation over the action horizon.

Multi-View Future Latents. A frozen DINO encoder E_{DINO} maps each current image $\boldsymbol{I}_t^{(v)}$ and its future counterpart $\boldsymbol{I}_{t+H}^{(v)}$ to latent features $\boldsymbol{z}_t^{(v)}$ and $\boldsymbol{z}_{t+H}^{(v)}$. Across V views, we denote the current and future representations as $\mathcal{Z}_t = \{\boldsymbol{z}_t^{(v)}\}_{v=1}^V$ and $\mathcal{Z}_{t+H} = \{\boldsymbol{z}_{t+H}^{(v)}\}_{v=1}^V$. Predicting future DINO features avoids appearance-level reconstruction while preserving object structure and spatial information relevant to physical state changes Nilaksh et al. (2026); Nie et al. (2026). Multi-view targets further capture complementary global and local interaction cues Torne et al. (2026).

World and Action Branches. CogWAM instantiates the world and action branches as two streams in a Mixture-of-Transformers (MoT) architecture (Liang et al., 2024). Both streams share $\mathcal{Z}_t$ as the current visual anchor and interact through multimodal self-attention, while retaining modality-specific cross-attention and feed-forward blocks. The World stream predicts $\mathcal{Z}_{t+H}$ conditioned on $\boldsymbol{H}_t^w$, whereas the Action stream generates $\boldsymbol{a}_t$ conditioned on $\boldsymbol{H}_t^a$. During joint training, a structured attention mask prevents Action tokens from attending to future-latent tokens, avoiding target leakage and ensuring that action generation uses only information available at inference Yuan et al. (2026). Future-latent prediction therefore serves as a training-time co-objective rather than an intermediate plan for action generation. At inference, the World stream is removed, leaving a direct action path conditioned on $\mathcal{Z}_t$ and $\boldsymbol{H}_t^a$.

3.5 Training Strategy

Joint Training Objective. CogWAM jointly optimizes action generation, future-latent prediction, and Semantic State modeling. For the action branch, Gaussian noise $\boldsymbol{\epsilon}^a \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is interpolated with the target action chunk $\boldsymbol{a}_t$ at flow time $\tau \sim \mathcal{U}(0, 1)$. The world branch analogously interpolates $\boldsymbol{\epsilon}^w \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ with the future representation $\mathcal{Z}_{t+H}$:

$$\boldsymbol{a}_{t,\tau} = (1-\tau)\boldsymbol{a}_t + \tau\boldsymbol{\epsilon}^a, \qquad \mathcal{Z}_{t+H,\tau} = (1-\tau)\mathcal{Z}_{t+H} + \tau\boldsymbol{\epsilon}^w. \tag{6}$$

Conditioned on the current visual representation and the corresponding progress-conditioned queries, the two branches predict their velocity fields:

$$\mathcal{L}_{\text{action}} = \mathbb{E}\left[\left\|\boldsymbol{v}_\theta^a\left(\boldsymbol{a}_{t,\tau}, \tau; \mathcal{Z}_t, \boldsymbol{H}_t^a\right) - (\boldsymbol{\epsilon}^a - \boldsymbol{a}_t)\right\|_2^2\right], \tag{7}$$

$$\mathcal{L}_{\text{world}} = \mathbb{E}\left[\left\|\boldsymbol{v}_\theta^w\left(\mathcal{Z}_{t+H,\tau}, \tau; \mathcal{Z}_t, \boldsymbol{H}_t^w\right) - (\boldsymbol{\epsilon}^w - \mathcal{Z}_{t+H})\right\|_2^2\right]. \tag{8}$$

Semantic State prediction uses a decision loss $\mathcal{L}_{\text{dec}}$ for `KEEP`/`UPDATE` classification and an autoregressive loss $\mathcal{L}_{\text{gen}}$ for memory increment and updated subtask. Given annotated decision $d_t^\star$,

$$\mathcal{L}_{\text{sem}} = \mathcal{L}_{\text{dec}} + \mathbf{1}[d_t^\star = \texttt{UPDATE}]\,\mathcal{L}_{\text{gen}}. \tag{9}$$

The complete training objective is

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda_{\text{world}}\mathcal{L}_{\text{world}} + \lambda_{\text{sem}}\mathcal{L}_{\text{sem}}, \tag{10}$$

where λ_{world} and λ_{sem} control the contributions of future-latent and semantic supervision.

Boundary-Aware Semantic Sampling. Semantic transitions are sparse, so uniform sampling strongly favors `KEEP`. We therefore oversample `UPDATE` events together with hard-`KEEP` examples near annotated transition boundaries. These neighboring observations are visually similar to transition states but do not yet justify advancing the subtask, encouraging the model to localize semantic transitions more precisely. This rebalancing is applied only to semantic supervision; world and action training follow the original demonstration distribution.

Progressive Semantic Conditioning. Training the world and action branches only with annotated Semantic States creates a train–inference mismatch, since deployment conditions on model-predicted states. We therefore use the predicted state $\widehat{S}_t$ as semantic context with probability ρ_k and the annotated state $S_t^\star$ otherwise, where ρ_k increases with training progress k. Semantic prediction remains supervised by the annotated target; only the context used to construct $\boldsymbol{H}_t^w$ and $\boldsymbol{H}_t^a$ is progressively replaced. This exposes both branches to the semantic context encountered during closed-loop inference.

4 Experiments and Analysis

We evaluate CogWAM in both simulation and the real world. Following a question-driven evaluation protocol, our experiments are designed to answer the following four questions:

Q1: How does CogWAM perform on multi-task robotic manipulation benchmarks?
Q2: Does the Semantic State improve long-horizon manipulation?
Q3: How should the Semantic State be updated during closed-loop execution?
Q4: How well does CogWAM transfer to real-world robotic manipulation?

4.1 Experimental Setup

Benchmarks. We conduct simulation experiments on RoboDojo Chen et al. (2026b) and BiCoord Peng et al. (2026), both under the multi-task setting. RoboDojo evaluates manipulation capabilities across Generalization, Precision, Long-Horizon, Memory, and Open categories. BiCoord contains 18 long-horizon bimanual tasks with stage-wise annotations and an average of 4.27 stages per task, providing a complementary test bed for multi-stage bimanual coordination.

Metrics. For RoboDojo, we report Score and Success Rate (SR) for each capability dimension and average them across the five dimensions. Following the official protocol, each task is evaluated over 50 episodes; Generalization uses 25 standard and 25 randomized episodes. For BiCoord, we report task Success Rate (SR) and Stage-wise Success Rate (SSR) over 100 rollouts per task. For the update-strategy analysis, we additionally report semantic-model calls, inference latency, and transition delay relative to annotated stage boundaries, measured in the real-robot deployment where the control frequency imposes a fixed replanning budget.

4.2 Simulation Experiments

Q1: Multi-task benchmark performance. We compare CogWAM with representative VLA and WAM baselines on RoboDojo and BiCoord under the multi-task setting.

RoboDojo. As shown in Tab. 1, CogWAM achieves an average Score/SR of **15.56/11.70%** without prior embodied robot-data pre-training, substantially outperforming existing methods under the same setting. Relative to StarVLA-α Ye et al. (2026a), it improves the average Score by **9.16** and SR by **8.46** percentage points, with particularly strong gains in Precision and Long-Horizon manipulation. CogWAM also surpasses several methods that rely on prior embodied robot-data pre-training.

BiCoord. Tab. 2 further evaluates long-horizon bimanual manipulation. Averaged over all 18 tasks, CogWAM achieves an SSR/SR of **58.0/43.8%**, outperforming the strongest evaluated baseline, $\pi_{0.5}$ Black et al. (2025), by **7.2/8.2** percentage points. Its strongest gains occur on sustained multi-stage tasks such as Clean Table, Cook, and Put Objects Cabinet, while performance is less uniformly improved on tasks dominated by low-level bimanual coordination. Together, the two

Table 1: **Multi-Task Evaluation on RoboDojo.** Each cell reports Score / Success Rate (in %). Methods are grouped by whether prior embodied robot-data pre-training is used before RoboDojo-specific training. **Bold** denotes the best result within each group.

Method	Generalization	Precision	Long-Horizon	Memory	Open	Average
w/ Prior Embodied Robot-Data Pre-training						
X-VLA Zheng et al. (2026)	10.47 / 6.78	18.32 / 12.00	16.53 / 9.75	4.76 / 3.56	0.55 / 0.50	10.13 / 6.52
InternVLA-A1.5 Ma et al. (2026)	10.35 / 6.83	15.23 / 10.17	23.80 / 13.75	4.93 / 3.56	1.43 / 1.42	11.15 / 7.14
$\pi_{0.5}$ Black et al. (2025)	13.38 / 8.17	12.40 / 5.50	23.54 / 14.67	5.89 / 4.67	1.98 / 1.67	11.44 / 6.93
Spatial Forcing Li et al. (2026a)	14.12 / 9.34	17.32 / 10.58	23.26 / 14.58	5.43 / 4.11	1.78 / 1.58	12.38 / 8.04
Hy-Embodied-0.5-VLA Zhang et al. (2026a)	11.78 / 8.39	13.81 / 8.00	25.74 / 14.92	13.37 / 12.11	0.65 / 0.58	13.07 / 8.80
Xiaomi-Robotics-1 Team et al. (2026)	**23.54 / 17.00**	26.69 / 18.83	38.39 / 23.67	7.81 / 6.56	**3.94 / 3.58**	20.07 / 13.93
GalaxeaVLA (G0.5) Liu et al. (2026)	18.46 / 12.83	**28.25 / 20.42**	**44.12 / 32.25**	8.61 / 7.33	1.73 / 1.58	20.23 / 14.88
DM0.5 Dexmal (2026)	15.77 / 10.95	24.82 / 16.75	33.70 / 19.50	**47.74 / 47.44**	2.43 / 2.08	**24.90 / 19.34**
w/o Prior Embodied Robot-Data Pre-training						
Fast-WAM Yuan et al. (2026)	2.33 / 1.11	1.96 / 0.00	9.14 / 5.17	3.55 / 3.44	0.42 / 0.42	3.48 / 2.03
AHA-WAM Cai et al. (2026)	5.79 / 3.27	5.86 / 2.42	8.61 / 2.67	2.97 / 2.78	0.88 / 0.83	4.82 / 2.39
StarVLA-α Ye et al. (2026a)	3.94 / 2.33	9.90 / 4.33	14.15 / 6.50	3.34 / 2.44	0.67 / 0.58	6.40 / 3.24
X-WAM Guo et al. (2026)	7.39 / 3.33	6.72 / 1.83	17.47 / 9.08	6.32 / 4.67	0.57 / 0.25	7.69 / 3.83
Fast-WAM + CogWAM Interface	12.28 / 9.00	19.61 / 13.50	23.69 / 14.75	**9.17 / 8.00**	1.93 / 1.75	13.33 / 9.40
CogWAM (Ours)	**15.53 / 12.17**	**24.45 / 19.00**	**28.14 / 19.00**	7.65 / 6.33	**2.05 / 2.00**	**15.56 / 11.70**

Table 2: **Multi-Task Evaluation on BiCoord.** Each cell reports Stage-wise Success Rate / Success Rate (in %). We show the six tasks with the longest average expert trajectories, while **Avg.** is computed over all 18 tasks. Complete per-task results are provided in Appendix Tab. 11.

Method	Clean Table	Cook	Exchange Mics	Exchange Pots	Match Blocks With Signs	Put Objects Cabinet	...	Average (18 Tasks)
RDT Liu et al. (2025)	17.5 / 0.0	31.0 / 9.0	35.0 / 23.0	**96.0 / 92.0**	7.0 / 1.0	49.0 / 31.0	...	34.8 / 16.9
OpenVLA-OFT Kim et al. (2025)	25.0 / 2.0	25.0 / 10.0	**67.5 / 66.0**	56.0 / 53.0	8.7 / 5.0	49.5 / 26.0	...	36.5 / 23.1
π_0 Black et al. (2024)	46.2 / 6.0	29.5 / 14.0	59.5 / 52.0	60.5 / 52.0	**18.7 / 8.0**	24.0 / 6.0	...	43.2 / 27.2
$\pi_{0.5}$ Black et al. (2025)	61.5 / 24.0	48.0 / 31.0	62.0 / 55.0	68.0 / 61.0	15.0 / 6.0	58.5 / 42.0	...	50.8 / 35.6
CogWAM (Ours)	**80.0 / 50.0**	**77.5 / 52.0**	56.5 / 49.0	76.5 / 70.0	17.3 / 7.0	**84.0 / 74.0**	...	**58.0 / 43.8**

benchmarks show that CogWAM substantially strengthens multi-task and long-horizon execution without relying on prior robot-data pre-training.

Interface and Representation Analysis. We further analyze two design choices underlying CogWAM: whether the proposed semantic interface transfers across world-action architectures, and how the predictive representation affects performance. (1) Cross-architecture transfer. Applying the same semantic interface to Fast-WAM Yuan et al. (2026) improves its average Score/SR from **3.48/2.03%** to **13.33/9.40%**, while retaining the original Fast-WAM world-action formulation. This substantial gain shows that the benefit of the semantic interface is not specific to CogWAM's underlying WAM architecture. (2) Predictive representation and model efficiency. We compare predictive formulations while keeping the semantic interface unchanged. Fast-WAM repurposes a pretrained Wan2.2-TI2V-5B video backbone Wan (2025), whereas CogWAM predicts future DINOv3 features with a 0.44B world stream trained from scratch. Despite the reduced world-modeling capacity, CogWAM improves average Score/SR from **13.33/9.40%** to **15.56/11.70%**, with gains on Generalization, Precision, and Long-Horizon, while Wan2.2 remains stronger on Memory. This suggests that control-relevant visual representations can provide effective predictive supervision without large generative backbones. Since representation, capacity, and initialization differ jointly, we interpret this comparison as evidence of formulation efficiency rather than isolated representation quality.

Q2: Effect of the Semantic State. We isolate the contribution of the Semantic State by comparing CogWAM with a world-action MoT counterpart that shares the same backbone and training setting but conditions on only the current observation and global task instruction, without any persistent Semantic State. As illustrated in Fig. 3, this baseline is more prone to subtask drift, repetition of completed stages, and premature transitions. In contrast, CogWAM preserves the active subtask across multiple action chunks and advances it only when execution progress changes. Fig. 4 further quantifies these gains on the Long-Horizon and Memory categories of RoboDojo and the stage-wise progression metrics of BiCoord.

Q3: Semantic State update strategies. We compare three update strategies: *Synchronous*, *Asynchronous*, and our *Event-triggered* update. Synchronous updating regenerates the Semantic State at every replanning step and therefore incurs substantial computation. Asynchronous updating reduces this overhead by running the semantic planner at a fixed lower frequency, but its updates can be misaligned with actual subtask transitions. CogWAM instead performs a lightweight `KEEP`/`UPDATE`



Figure 3: **Qualitative Analysis of Semantic State Progression.** Simulation rollouts illustrate how persistent Semantic States track task progress across action chunks, reducing subtask drift, repetition, and premature transitions. The real-world rollout visualizes the predicted `KEEP`/`UPDATE` decisions together with the corresponding Semantic States throughout execution.

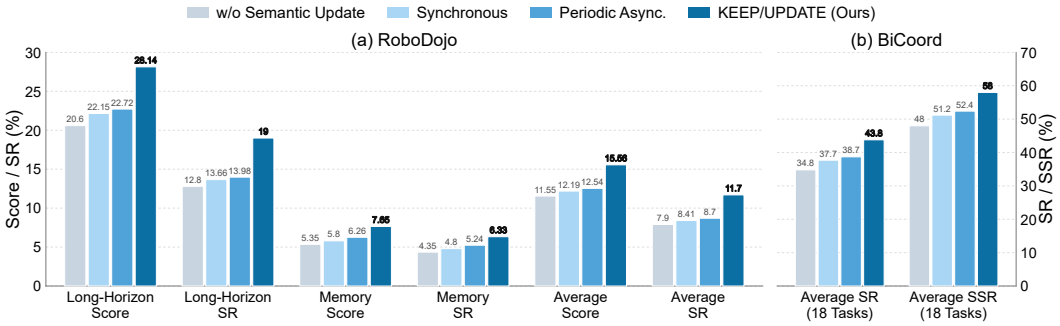


Figure 4: **Effect of Semantic State and Update Strategies.** Task performance on (a) RoboDojo (Score / SR) and (b) BiCoord (SR / SSR), shown with benchmark-specific vertical scales. *w/o Semantic State* conditions the policy only on the current observation and global task instruction, while the remaining variants differ in how the Semantic State is updated.

decision and regenerates the Semantic State only when a semantic transition is detected. As shown in Fig. 4, the event-triggered strategy performs best on both benchmarks; notably, synchronous updating does not surpass asynchronous updating despite regenerating far more often, indicating that update timing matters more than update frequency. We further compare semantic-model calls, inference latency, and transition delay on the real robot (Sec. 4.3).

4.3 Real-World Experiments

Q4: Real-world evaluation. We evaluate CogWAM in the real world along two complementary tracks: task performance on six manipulation tasks and robustness under four controlled distribution shifts. Fig. 5 illustrates the robotic setup and representative task examples. Details of data collection, optimization, compute, and deployment hyperparameters are provided in Appendix B.

Evaluation Protocol. We evaluate CogWAM on six multi-stage tabletop tasks: *Place Objects*, *Organize Utensils*, *Put in Drawer*, *Fill Pen Holder*, *Place in Bag*, and *Block Sorting*, covering repeated manipulation, category-level organization, articulated and deformable object interaction, bimanual coordination, and precision placement. For the basic setting, each task is evaluated over 20 closed-loop trials with predefined initial configurations, yielding 120 trials in total. We further assess robustness under four controlled distribution shifts, each evaluated independently across all six tasks. For each shift dimension, we perform 5 closed-loop trials per task, yielding 30 trials per dimension.

Table 3: **Real-world evaluation.** Success counts across basic manipulation tasks and distribution shifts.

Basic Task	Success	Generalization	Success
Place Objects	20 / 20	Spatial Location	21 / 30
Organize Utensils	17 / 20	Object Appearance	20 / 30
Put in Drawer	18 / 20	Distractor	17 / 30
Fill Pen Holder	18 / 20	Novel Objects	14 / 30
Place in Bag	16 / 20	–	
Block Sorting	5 / 20	–	
Total	**94 / 120**	**Total**	**72 / 120**

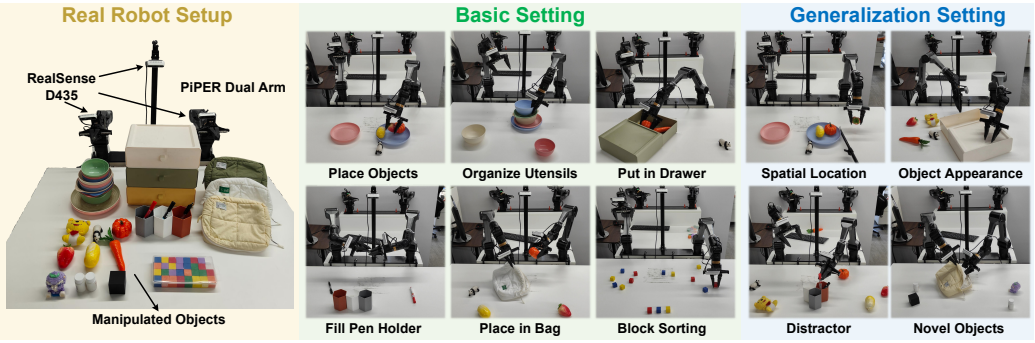


Figure 5: **Real-world setup and task examples.** CogWAM is deployed on a dual-arm AgileX PIPER platform with two 6-DoF manipulators and parallel grippers, using synchronized RGB observations from one fixed head-mounted and two wrist-mounted Intel RealSense D435 cameras.

Table 4: **Deployment cost of update strategies.** *Total* denotes semantic-model calls per rollout, while *Regen.* counts calls that regenerate the Semantic State; *Dec. tok.* denotes tokens decoded per replanning step. Transition delay is the lag between a subtask boundary and the first replanning step at which the strategy can observe it. Latency is measured on an NVIDIA RTX 5090.

	Semantic-model calls		Inference latency		Transition
Update strategy	Total	Regen. ↓	Mean (ms) ↓	Dec. tok. ↓	**delay (ms)** ↓
Synchronous	65.4	65.4	927.8	45	**152**
Asynchronous (1 Hz)	22.1	22.1	309.3	45	502
Event-triggered (ours)	65.4	**4.0**	**146.7**	**0**	**152**

Tab. 3 summarizes the successful trials across both settings. We additionally visualize a representative *Place in Bag* rollout, showing the predicted `KEEP`/`UPDATE` decisions and corresponding subtask memory in Fig. 3.

Real-time efficiency of semantic updating. The real robot runs a 30 Hz control loop and issues one replanning request every 10 control steps, leaving a 333 ms budget per replanning step. On *Place in Bag*, five held-out rollouts average 21.6 s, 65.4 replanning steps, and 4.00 semantic transitions per episode. As shown in Tab. 4, synchronous updating achieves a 152 ms transition delay but incurs 927.8 ms latency, exceeding the budget by $2.8\times$. Asynchronous updating reduces latency to 309.3 ms by updating every 30 control steps, but increases transition delay to 502 ms. In contrast, CogWAM evaluates the `KEEP`/`UPDATE` decision at every replanning step, as frequently as synchronous updating, but requires $16.4\times$ fewer Semantic State regenerations. The `KEEP`/`UPDATE` decision is predicted from the semantic module's final hidden state without autoregressive decoding, enabling frequent transition checks at low cost. This yields the same 152 ms transition delay as synchronous updating, while reducing inference latency from 927.8 ms to 146.7 ms.

5 Conclusion and Discussion

CogWAM shows that explicitly aligning semantic task progress with world-action learning improves execution and deployment efficiency. Across simulation and real-world dual-arm manipulation, it consistently improves task completion without prior embodied robot-data pre-training. Importantly, transferring the same semantic interface to Fast-WAM also yields substantial gains, suggesting that the interface is not tied to a particular WAM architecture and can serve as a reusable bridge between VLM-style semantic reasoning and WAM-style predictive learning. Our analysis further shows that both update timing and predictive representation matter. Event-triggered updating remains aligned with task transitions while requiring far fewer Semantic State regenerations than synchronous updating. Meanwhile, CogWAM's compact DINO-based predictive formulation provides additional gains over the Fast-WAM-based variant, indicating that semantic alignment and predictive representation are complementary. A remaining limitation is that the Semantic State is append-only and cannot revise committed semantic errors. Future work could explore uncertainty-aware state revision and broader transfer across heterogeneous VLM and WAM backbones.

AI USE STATEMENT

In this work, we used GPT-5.6 Sol as a fixed VLM annotator to generate offline semantic supervision for robot demonstrations, as described in Appendix C. Specifically, the model was used to identify completion boundaries of predefined semantic stages from sampled episode observations. The generated annotations were manually inspected and fixed before being used for policy training, and the model was not involved in policy optimization or inference.

We did not use generative AI systems to produce experimental results, modify evaluation outcomes, or make scientific claims. The authors are responsible for all final content, including the methodology, experiments, analyses, and conclusions presented in this paper.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of the proposed CogWAM framework, including the model architecture and World–Action interface design (Appendix A.1, Appendix A.2), training configurations, optimization settings, datasets, and computational resources (Appendix B, Appendix B.2), as well as the semantic supervision generation procedure (Appendix C). Complete quantitative evaluations and qualitative closed-loop rollouts are provided in Appendix D to support reproducible evaluation and future research.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691, 2022.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. NeurIPS, 28, 2015.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. arXiv preprint arXiv:2504.16054, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818, 2023.

Rodney A Brooks. Intelligence without representation. Artificial intelligence, 47(1-3):139–159, 1991.

Jisong Cai, Long Ling, Shiwei Chu, Zhongshan Liu, Jiayue Kang, Zhixuan Liang, Wenjie Xu, Yinan Mao, Weinan Zhang, Xiaokang Yang, et al. Aha-wam: Asynchronous horizon-adaptive world-action modeling with observation-guided context routing. arXiv preprint arXiv:2606.09811, 2026.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. Lawam: Latent world action models for efficient dynamics-aware robot policies, 2026a.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. arXiv preprint arXiv:2607.04434, 2026b.

Dexmal. Opendm: An open-world foundation model for general-purpose embodied intelligence. `https://github.com/dexmal/opendm`, 2026. DM0.5.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. arXiv preprint arXiv:2303.03378, 2023.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. NeurIPS, 36: 9156–9172, 2023.

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. arXiv preprint arXiv:2604.26694, 2026.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608, 2022.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. arXiv preprint arXiv:2502.19645, 2025.

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. In ICLR, volume 2026, pp. 132324–132345, 2026a.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In ICML, pp. 19730–19742. PmLR, 2023.

Kehan Li, Bohan Hou, Minghao Zhu, Tianyi Zhang, Zesen Cheng, Zhikai Wang, Sicong Leng, Xin Li, Xiao Lin, Biying Yao, et al. Rynnbrain 1.1: Towards more capable and generalizable embodied foundation model. arXiv preprint arXiv:2607.17977, 2026b.

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. RSS, 2026c.

Shalfun Li, Victor Yao, Charles Yang, Truth Qu, Regis Cheng, Ryan Yu, Howard Lu, Newton Von, Vincent Chen, Yohann Tang, et al. Wall-wm: Carving world action modeling at the event joints. arXiv preprint arXiv:2606.01955, 2026d.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. arXiv preprint arXiv:2411.04996, 2024.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In ICLR, volume 2025, pp. 29982–30009, 2025.

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, et al. G0.5: One autoregressive stream for robot reasoning and action. arXiv preprint arXiv:2608.11739, 2026.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. RSS, 2026.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. arXiv preprint arXiv:2607.04988, 2026.

Dujun Nie, Fengjiao Chen, Qi Lv, Jun Kuang, Xiaoyu Li, Xuezhi Cao, and Xunliang Cai. Lary: A latent action representation yielding benchmark for generalizable vision-to-action alignment, 2026. URL https://arxiv.org/abs/2604.11689.

Nilaksh, Saurav Jha, Artem Zholus, and Sarath Chandar. Reconstruction or semantics? what makes a latent space useful for robotic world models. arXiv preprint arXiv:2605.06388, 2026. URL https://arxiv.org/abs/2605.06388.

Xingyu Peng, Chen Gao, Liankai Jin, Annan Li, and Si Liu. Bicoord: A bimanual manipulation benchmark towards long-horizon spatial-temporal coordination. arXiv preprint arXiv:2604.05831, 2026.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In ICLR, volume 2026, pp. 18567–18602, 2026.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. arXiv preprint arXiv:2502.19417, 2025.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. arXiv preprint arXiv:2508.10104, 2025.

Haisheng Su, Zongdai Liu, Xin Jin, Haoxuan Dou, Chengming Hu, Baorun Li, Zhanwang Liu, Ruiyan Xu, Jianjie Fang, Xin Zhang, Zhenjie Yang, Xue Yang, Chen Gao, Junchi Yan, Yong Li, and Wei Wu. Worldscape policy 2.0: Empowering steerable world action modeling with reasoning-augmented memory. arXiv preprint arXiv:2607.18840, 2026.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 112(1-2):181–211, 1999.

Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. IEEE Transactions on Automatic control, 52(9):1680–1685, 2007.

Generalist Team. Gen-0: Embodied foundation models that scale with physical interaction. Generalist AI Blog, 2025. https://generalistai.com/blog/gen-0.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, et al. Xiaomi-robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. arXiv preprint arXiv:2607.15330, 2026.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. arXiv preprint arXiv:2603.03596, 2026.

Team Wan. Wan: Open and advanced large-scale video generative models. arXiv preprint arXiv:2503.20314, 2025.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Jiabing Yang, Zhaowen Li, Yan Huang, Liang Wang, and Peng Su. Dim-wam: World-action modeling with diverse historical event memory. arXiv preprint arXiv:2606.27677, 2026.

Ran Yang, Zijian An, Lifeng Zhou, and Yiming Feng. Seqvla: Sequential task execution for long-horizon manipulation with completion-aware vision-language-action model. arXiv preprint arXiv:2509.14138, 2025.

Yi Yang, Zhihong Liu, Siqi Kou, Yiyang Chen, Yanzhe Hu, Jianbo Zhou, Boyuan Zhao, Zhijie Wei, Xiao Xia, Xueqi Li, et al. World-language-action model for unified world modeling, language reasoning, and action synthesis. CoRL, 2026.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla-α: Reducing complexity in vision-language-action systems. In ECCV, 2026a.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. arXiv preprint arXiv:2602.15922, 2026b.

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.

He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, Dingyan Zhong, Yubo Dong, Yihao Wu, Yongming Rao, Dongsheng Zhang, et al. Hy-embodied-0.5-vla: From vision-language-action models to a real-world robot learning stack. arXiv preprint arXiv:2606.14409, 2026a.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native video-action pretraining for generalizable robot control. arXiv preprint arXiv:2607.08639, 2026b.

Ruiteng Zhao, Zhengshen Zhang, Yue Su, Wenshuo Wang, Jiahui Li, Zhiyuan Yang, Francis EH Tay, Marcelo H Ang Jr, and Haiyue Zhu. Sg-wam: Self-guided world modeling in geometry-aware policy space. arXiv preprint arXiv:2608.01397, 2026.

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

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In ICLR, volume 2026, pp. 60580–60606, 2026.

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. In RSS, 2025.

A Implementation Details

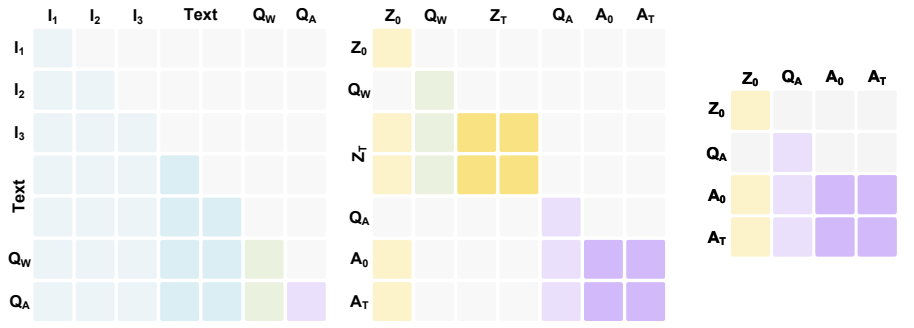
A.1 Model Architecture

CogWAM couples a vision-language backbone (Li et al., 2026b), a Mixture-of-Transformers (MoT) architecture (Liang et al., 2024) whose world and action streams are conditioned separately, and a frozen visual teacher (Siméoni et al., 2025). Tab. 5 summarizes the three components.

Table 5: **Model architecture.** The backbone row reports its 24-layer language model; its vision tower is also 24 layers. The two MoT streams traverse the same 30 layers but keep separate widths, attention projections, and feed-forward blocks; attention head dimension is 128 for both. At inference, the world-prediction stream and future-target encoding are removed, while the frozen DINO encoder is retained to extract current visual features.

	VLM backbone	**World–Action MoT**		**Visual teacher**
	RynnBrain1.1-2B	World stream	Action stream	DINOv3 ViT-B/16
Parameters	2.72 B	442 M	1.01 B	85.7 M
Trained	yes	yes	yes	frozen
Layers	24	30		12
Attention heads	8	24		12
Hidden dim	2048	512	1024	768
Feed-forward dim	6144	2048	4096	3072
Learned planner queries: 16 `WORLD` + 25 `ACTION` at dim 2048 (0.084 M)				
Trainable total: 4.19 B incl. projection layers • Full model incl. frozen teacher: 4.27 B				

The attention masks used in CogWAM are illustrated in Fig. 6. For the VLM backbone (Fig. 6(a)), image tokens follow bidirectional attention, while text tokens and learned WORLD/ACTION queries adopt causal attention. During World–Action MoT training (Fig. 6(b)), future-latent tokens attend to the WORLD conditioning representations and current visual features, while action tokens are masked from future-latent tokens to avoid information leakage. At inference, the world stream is removed, leaving the action stream conditioned on current visual features and ACTION representations.



(a) VLM Attention Mask (b) WAM Training Mask (c) WAM Inference Mask

Figure 6: **Illustration of attention masks in CogWAM.**

A.2 Analysis of VLM-World-Action Interface

CogWAM bridges high-level semantic reasoning and low-level physical control through a structured World–Action interface, adding a predictive World branch alongside the Action branch that conventional policies optimize alone (Black et al., 2025). This section asks why the two branches need separate queries (§A.2.1), what future-world prediction contributes beyond direct action supervision (§A.2.2), and whether the interface transfers to a different World–Action backbone (§A.2.3).

A.2.1 Semantic Interface between VLM and World-Action Learning

Future-world prediction and action generation share the same instruction and observation, but supervise the model through different output modalities: one learns a representation of scene evolution, the other a representation of executable control. Two design questions follow. **How deep into the**

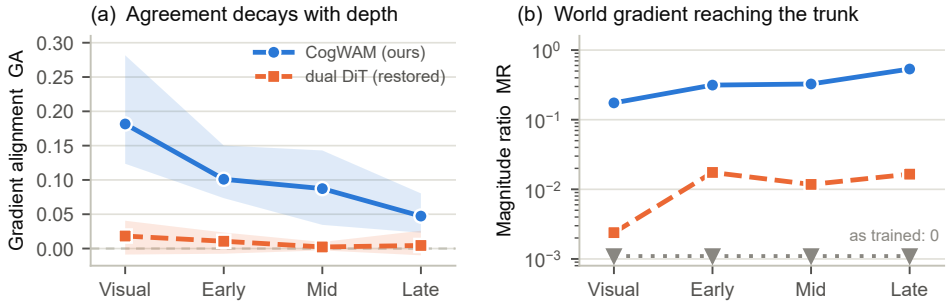


Figure 7: **World and Action gradients inside the shared VLM, by depth.** (a) Gradient alignment, with 95% bootstrap intervals over 120 batches. CogWAM is significantly positive at every depth and decays inward from the visual tower; the dual-DiT baseline with its gradient path restored is indistinguishable from zero throughout. (b) World-gradient magnitude relative to the action gradient. As trained, the dual-DiT baseline detaches the world branch, so no world gradient reaches the trunk and GA is undefined.

shared VLM should the objectives remain coupled, and through which interface should each read out the representation it needs? We answer both by measuring, rather than assuming, how the two objectives interact inside the trunk.

Treating the gradient direction as the update an objective requests from a parameter block, we report

$$\mathrm{GA}(\theta) = \frac{\nabla_\theta \mathcal{L}_{\text{world}} \cdot \nabla_\theta \mathcal{L}_{\text{action}}}{\|\nabla_\theta \mathcal{L}_{\text{world}}\|_2 \, \|\nabla_\theta \mathcal{L}_{\text{action}}\|_2}, \qquad \mathrm{MR}(\theta) = \frac{\|\nabla_\theta \mathcal{L}_{\text{world}}\|_2}{\|\nabla_\theta \mathcal{L}_{\text{action}}\|_2}, \tag{11}$$

over depth-ordered blocks θ of the shared VLM trunk. GA asks whether the objectives request the same update direction, MR whether the world objective carries enough magnitude to matter. We compute both measures from the unweighted world and action losses in Eq. 11. At a fixed checkpoint, multiplying either gradient by a positive scalar does not change GA. Changing the loss weights during training may, however, change the learned parameters and gradient directions, so this scale invariance alone does not make GA directly comparable across differently trained checkpoints. To attribute any difference to the interface alone, we compare the two architectures on a matched configuration that varies only in how the world and action branches attach to the VLM: both use a RynnBrain1.1 backbone Li et al. (2026b), action horizon 16, the same RoboDojo recipe and the same frozen DINOv3-B teacher. This controlled pair is therefore not the final model of Tab. 5, which uses the RynnBrain1.1 backbone at horizon 25; we trade the exact deployment configuration for a clean attribution. All measurements use 120 training batches at 20k steps, report medians with 95% bootstrap intervals, and split the 24-layer language-model stack into equal thirds (layers 0–7, 8–15, 16–23).

Shared grounding, specialized abstraction. Sec. 3.3 motivates separate WORLD and ACTION queries as a way to share task-level semantics while preserving objective-specific representations; the gradients make that trade-off measurable. In CogWAM the objectives are positively but weakly aligned throughout the trunk, and the alignment decreases with depth (Fig. 7a): every interval excludes zero, and the medians fall monotonically from $0.182\,[0.124, 0.282]$ in the visual tower through 0.101 and 0.087 to $0.047\,[0.023, 0.080]$ in the last language-model third (0.081 over the trunk as a whole). Because the blocks are measured on the same batches, a paired test applies; the visual tower exceeds every language-model block (Wilcoxon $p < 10^{-3}$), and LM-early exceeds LM-late; the median of the batch-wise paired differences is $0.023\,[0.011, 0.046]$, although adjacent blocks are not always separable. This is an expectation over batches rather than a per-batch property: 66% to 76% of batches are positive, so there is no systematic opposition between the objectives, but individual batches do disagree. Both objectives must recover the same objects, geometry and scene state, which makes the perceptual trunk genuinely shared; by the semantic depth at which the queries read out, the representation each requires is largely its own. Notably the deeper blocks receive proportionally more world gradient (Fig. 7b, MR rising from 0.17 to 0.53) while agreeing with the action objective less, so the divergence is one of direction rather than of the world signal fading out. A single query would have to summarise two only marginally related requirements within one fixed bottleneck, whereas separate WORLD and ACTION queries let each objective retain the full subspace for roughly 3×10^4 additional parameters.

External heads in the evaluated dual-DiT configuration. The dual-DiT baseline attaches independent world and action heads to a standard VLM. As trained, its recipe detaches the world branch from the backbone, so $\|\nabla_\theta \mathcal{L}_{\text{world}}\|$ is identically zero on every VLM block and GA is undefined: the world modality shapes only its own head. Since that detachment is a choice in the recipe rather than a consequence of using external heads, we also measure the counterfactual in which the path is restored. With the gradient path restored, the evaluated dual-DiT baseline has a whole-trunk GA of $0.012\,[-0.005, 0.022]$, with every per-block interval covering zero. Its MR is 0.02, compared with 0.37 for CogWAM, indicating a substantially smaller world-gradient contribution to the shared trunk. In this evaluated configuration, the external world head contributes little to shared representation learning even when its gradient path is restored.

Query ordering. Sec. 3.4 relies on a structured attention mask to keep ACTION tokens from attending to future-latent tokens; Fig. 6 shows the forward information path. The zero world-loss gradient on ACTION query parameters is consistent with this query ordering, while the attention mask directly prevents action tokens from accessing future-latent inputs. The WORLD queries receive gradients from both objectives ($\text{MR} = 2.63$), with positive alignment between them ($\text{GA} = 0.152$).

Key Findings: Designing a VLM–World-Action Interface

An effective interface should therefore not hand one unified semantic representation to every downstream objective, but preserve shared context at the depth where the objectives agree and let each specialise where that agreement runs out.

A.2.2 Decomposing CogWAM

We next decompose CogWAM into the two additions that distinguish it from action-only control: predictive world supervision and semantic conditioning. Tab. 6 follows the progression *Action MoT* $\rightarrow$ *MoT* $\rightarrow$ *CogWAM*. Action MoT retains the World–Action architecture but is trained only with the action objective. MoT additionally learns to predict future observations in the frozen DINO feature space. CogWAM further introduces the semantic interface, which conditions the policy on VLM-derived task progress and maintains an event-triggered Semantic State containing the active subtask and accumulated task history.

Table 6: **Stepwise decomposition of CogWAM on RoboDojo.** Each cell reports Score / Success Rate (in %). Action MoT $\rightarrow$ MoT adds DINO-space future-world prediction; MoT $\rightarrow$ CogWAM further adds the CogWAM Interface.

	Components			RoboDojo Evaluation					
Method	Pred.	VLM	State	Gen.	Prec.	Long-H.	Mem.	Open	Avg.
Action MoT	–	–	–	8.00 / 5.20	18.20 / 12.60	21.62 / 12.00	3.50 / 2.00	**2.10** / 1.90	10.68 / 6.74
MoT	✓	–	–	11.73 / 8.83	21.24 / 14.00	21.88 / 13.64	5.72 / 4.67	1.05 / 1.00	12.32 / 8.43
CogWAM	✓	✓	✓	**15.53 / 12.17**	**24.45 / 19.00**	**28.14 / 19.00**	**7.65 / 6.33**	2.05 / **2.00**	**15.56 / 11.70**

Pred.: DINO-space future-world prediction; *VLM*: VLM-derived task-progress conditioning; *State*: event-triggered Semantic State.

Future prediction primarily improves perceptual grounding. Moving from Action MoT to MoT raises Generalization by $+3.73$ and Precision by $+3.04$ Score, while Long-Horizon changes by only $+0.26$. This pattern is consistent with future-world prediction acting mainly as a representation-learning signal: predicting how the scene will evolve encourages the shared representation to preserve objects, geometry, and interaction-relevant visual structure that are useful for selecting the next action. It does not, however, explicitly encode where the robot is within a multi-stage task, and therefore provides little direct mechanism for resolving progress ambiguity over long horizons.

The semantic interface reduces action uncertainty by conditioning on task progress. Moving from MoT to CogWAM produces the largest gain on Long-Horizon ($+6.26$ Score), but also improves Generalization ($+3.80$), Precision ($+3.21$), Memory ($+1.93$), and Open ($+1.00$). These broader gains are expected because semantic conditioning is useful beyond tasks that are nominally long-horizon. From a conditional-prediction perspective, an observation and instruction alone may admit several plausible action modes when the same visual configuration can occur at different stages of a

Table 7: **Transferability of the CogWAM Interface on RoboDojo.** Each cell reports Score / Success Rate (in %). Applying the same semantic interface to Fast-WAM substantially improves performance without changing its underlying predictive formulation.

Method	Gen.	Prec.	Long-H.	Mem.	Open	Avg.
Fast-WAM Yuan et al. (2026)	2.33 / 1.11	1.96 / 0.00	9.14 / 5.17	3.55 / 3.44	0.42 / 0.42	3.48 / 2.03
Fast-WAM + CogWAM Interface	12.28 / 9.00	19.61 / 13.50	23.69 / 14.75	**9.17 / 8.00**	1.93 / 1.75	13.33 / 9.40
CogWAM	**15.53 / 12.17**	**24.45 / 19.00**	**28.14 / 19.00**	7.65 / 6.33	**2.05 / 2.00**	**15.56 / 11.70**

task. Introducing the active subtask and task history supplies an additional condition z_t, replacing the harder prediction problem $p(a_t \mid o_t, \ell)$ with $p(a_t \mid o_t, \ell, z_t)$. Equivalently, the interface is intended to reduce the conditional uncertainty of the action distribution, $H(A_t \mid O_t, \ell, z_t) \leq H(A_t \mid O_t, \ell)$, whenever task progress disambiguates otherwise plausible behaviors.

This reduction is most consequential in long-horizon tasks, where perceptually similar states can recur after different histories and therefore require different subsequent actions. The accumulated Semantic State explicitly records which task events have already occurred, reducing this form of history-dependent state aliasing. The same conditioning can nevertheless benefit shorter-horizon Generalization and Precision tasks: specifying the current semantic stage narrows the set of admissible behaviors and gives the action model a more stable task-relevant context, rather than requiring it to infer both task phase and motor command from the observation at every replanning step.

Importantly, the MoT-to-CogWAM comparison adds the complete semantic interface—VLM-derived task-progress conditioning together with the event-triggered Semantic State. We therefore attribute the improvement in Tab. 6 to the semantic interface as a whole, rather than claiming that this ablation isolates either component individually. The disproportionately large Long-Horizon gain is consistent with the additional value of explicit task history, but separating current-subtask conditioning from accumulated semantic memory would require a further component-level ablation.

Unlike generative world models that optimize future reconstruction, CogWAM predicts future representations in a frozen DINO feature space (Siméoni et al., 2025). Future prediction is used only as a training signal, so the World branch need not synthesize pixels at inference time. The compact target instead encourages the shared representation to capture task-relevant scene evolution while keeping the predictive objective lightweight.

A.2.3 Transferability of the CogWAM Interface

The decomposition above evaluates the CogWAM Interface within our DINO-based World–Action formulation. We next ask whether its benefit depends on this particular predictive backbone. To test this, we attach the same interface to Fast-WAM (Yuan et al., 2026), while retaining Fast-WAM's original world-action formulation, Wan2.2-TI2V-5B predictive backbone (Wan, 2025), and action decoder.

Adding the CogWAM Interface raises Fast-WAM's average Score/SR from $3.48/2.03$ to $13.33/9.40$. The improvement is broad rather than confined to one capability: Generalization increases by $+9.95$ Score, Precision by $+17.65$, Long-Horizon by $+14.55$, Memory by $+5.62$, and Open by $+1.51$. This transfer is important because the underlying predictive model is unchanged; the gain therefore does not rely on CogWAM's DINO-space future prediction.

Together with Tab. 6, the result separates two roles. Predictive world modeling shapes the representation of how the scene may evolve, while the CogWAM Interface conditions action generation on task progress and execution history. In conditional-prediction terms, the former enriches the predictive representation of the observation, whereas the latter supplies an additional task-state variable that narrows the set of plausible continuations. This explains why the interface improves not only Long-Horizon tasks, where history-dependent ambiguity is strongest, but also Generalization and Precision tasks, where explicit task-phase conditioning can reduce ambiguity in the next-action distribution.

The remaining gap between Fast-WAM + CogWAM Interface and CogWAM should not be attributed to any single predictive component, since the two systems retain different world-action formulations. The transfer experiment supports the narrower conclusion that the CogWAM Interface is portable: its benefit persists when attached to a substantially different predictive backbone.

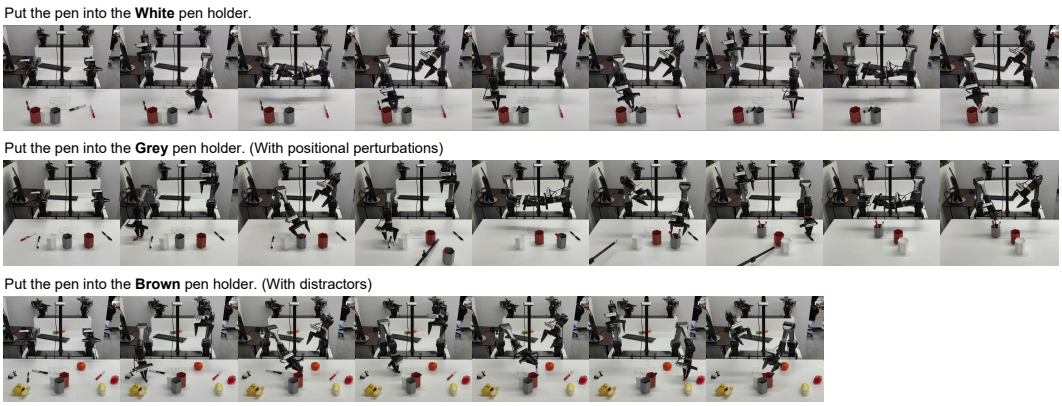


Figure 8: **Language grounding under visual perturbations.** CogWAM follows instructions specifying different target pen holders: white, grey, and brown. The grey-holder setting introduces spatial perturbations of the target objects, while the brown-holder setting introduces additional distractor objects. Despite these variations, CogWAM consistently grounds the instructed target and completes the corresponding manipulation behavior.

A.2.4 Language Grounding under Visual Perturbations

While the previous analysis studies how semantic information is transferred from the VLM to the World-Action model, we further investigate whether the CogWAM Interface enables fine-grained language grounding between instructions and visual entities.

Fig. 8 qualitatively evaluates language grounding on the *Fill Pen Holder* task. We vary the target specified by the instruction while keeping the underlying manipulation objective unchanged. The target object is changed among different pen holders with distinct visual appearances, including white, grey, and brown holders. Furthermore, we introduce two types of visual perturbations: spatial rearrangement of the target objects and additional distractor objects in the scene.

CogWAM successfully identifies the instructed target across these variations, demonstrating robust grounding between linguistic descriptions and visual observations. This behavior highlights a capability provided by the VLM Interface beyond task-progress modeling: instead of relying solely on a compact language embedding associated with demonstrations, the policy receives a semantically grounded representation that explicitly links instructions with the current visual context.

Key Findings: VLM Enables Robust Language Grounding

Beyond task-progress modeling, the VLM interface grounds language instructions to visual entities, allowing CogWAM to robustly follow fine-grained references under visual variations.

B Training Details

B.1 RoboDojo Training Recipe

Objectives. Tab. 8 lists the full training recipe used for the RoboDojo benchmark. The pretrained backbone is fine-tuned an order of magnitude more slowly than the randomly initialized queries and action model. The semantic loss is the sum of two next-token cross-entropies over disjoint label spans, one covering the decision token and one the `Memory Add` and `Current Subtask` continuation; they are combined without a relative coefficient, since their effective balance comes from the 2-of-6 composition of the semantic batch.

Table 8: **Training hyperparameters.** The semantic stream is a second, independent dataloader: every step trains both a physical batch and a balanced semantic batch.

Configuration	Value
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$
Weight decay	10^{-8}
Learning rate (VLM backbone)	1×10^{-5}
Learning rate (planner queries)	1×10^{-4}
Learning rate (action model)	1×10^{-4}
LR schedule	cosine, 2,000 warmup steps, floor 5×10^{-7}
Gradient clipping	1.0 (global norm)
Training steps	50,000
Batch size (physical)	12 / device, 768 global
Batch size (semantic)	6 / device (2 `UPDATE` / 2 hard `KEEP` / 2 random `KEEP`)
Compute	8×8 NVIDIA H20
Precision	bfloat16, DeepSpeed ZeRO-2, gradient checkpointing
Random seed	42
Loss	$\mathcal{L}_{\text{action}} + \lambda_{\text{world}}\mathcal{L}_{\text{world}} + \lambda_{\text{sem}}\mathcal{L}_{\text{sem}}$
λ_{world}, λ_{sem}	1.0, 0.005
Flow-matching schedule	shifted uniform, $\sigma = su/(1+(s-1)u)$, $s = 5.0$
Train timesteps	1000
Regression target	$\epsilon - x_0$
Inference solver	Euler, 20 steps, no classifier-free guidance
Action horizon H	25
Future-world stride	16 frames
Replanning interval	10 frames
Semantic offset	-10 frames
Scheduled-sampling ρ_k	0 until 30k, $\rightarrow$ 0.2 at 40k, $\rightarrow$ 0.5 at 50k
Semantic decoding	greedy, $\leq$ 96 new tokens

Table 9: **Training data.** Per-task counts after the 5% episode-level validation split.

Task	Train ep.	Train frames	Val ep.	Val frames
Place Objects	66	83,314	1	1,244
Organize Utensils	87	112,181	5	6,725
Put in Drawer	437	729,929	23	37,273
Fill Pen Holder	1,477	220,962	78	10,950
Place in Bag	954	392,681	49	19,131
Block Sorting	1,210	410,721	64	21,353
Total	**4,231**	**1,949,788**	**220**	**96,676**

B.2 Real-World Training Data

Sampling. Episode counts and frame counts rank inversely (Tab. 9): *Fill Pen Holder* is 34.9% of episodes but 11.3% of frames, *Put in Drawer* 10.3% and 37.4%. Sampling over episodes would therefore undersample the long-horizon tasks, so the physical stream samples over frames. Validation is held out at the episode level, so no validation frame is ever seen in training.

B.3 Real-World Training Progress

Fig. 9 tracks the action objective over the 50k-step run. Action MSE scores the predicted chunk against the logged one; reverse KL (Team, 2025) asks whether the samples the policy actually draws land near that chunk rather than averaging across modes into an action that matches none of them.

Convergence. Both held-out measures improve steeply through the first 20k steps and then flatten: action MSE falls 34% by 20k and 32% overall, reverse KL 47% and 50%. After 20k the curves separate. Action MSE oscillates within $\pm 3\%$ of its final value and reaches its minimum at 37.5k rather than at the end. The remaining fluctuations are small, with no sustained improvement in action MSE. Reverse KL keeps drifting down, indicating that late training sharpens the sampled

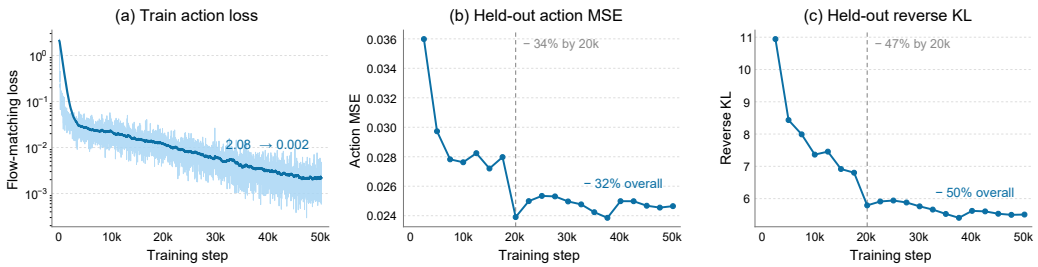


Figure 9: **Action learning over training.** Panel (a) shows per-step training loss (light) with an exponential moving average (dark) on a log scale. Panels (b) and (c) show held-out action MSE and reverse KL at every validation checkpoint.

distribution around the demonstrated action more than it moves the mean prediction. We therefore report 50k as converged rather than as the best of a still-improving run.

Relation to closed-loop behaviour. Offline action metrics track closed-loop quality across checkpoints, but not across tasks. In informal on-robot checks the 20k and 40k checkpoints behaved comparably in grasp accuracy and generalization while 10k was visibly underfit, consistent with the elbow in Fig. 9; we did not run a controlled comparison across checkpoints. Across tasks the picture differs: at 50k, closed-loop success ranges from $20/20$ on *Place Objects* to $5/20$ on *Block Sorting*, a spread the aggregate curves cannot explain, since action MSE compares one predicted chunk to one logged chunk and cannot see whether an error is absorbed at the next replanning step or compounds into a failed grasp. This is also why we track reverse KL: a policy that averages across two viable behaviours can score well on MSE while every sample it draws falls between them, and it is the drawn sample that the robot executes.

C Semantic Supervision

This section details the offline procedure that produces the `KEEP`/`UPDATE` decision, the memory increment Δm_t, and the active subtask s_t.

Offline subtask annotation. We generate subtask supervision once offline using GPT-5.6 Sol as a fixed VLM annotator. For each episode, we provide sparsely sampled video frames, the task instruction, and a predefined ordered checklist of semantic stages, and ask the annotator to localize the completion boundary of each stage. The predicted boundaries are mapped back to the original video timeline and converted into right-open subtask spans, yielding the frame-level active subtask s_t. For tasks with an episode-dependent number of stages, such as *Arrange Largest Number*, the stage count is first inferred from early observations before constructing the checklist. These annotations are fixed before training; GPT-5.6 Sol is used only for offline annotation and is not involved in policy training or inference.

Some RoboDojo tasks require episode-specific checklist construction. For *Arrange Largest Number*, the number of digit-placement stages varies between four and five; we infer this count from up to three early observations using independent VLM queries followed by majority voting, and instantiate the corresponding checklist before boundary annotation. For tasks with a fixed semantic structure, such as *Spell RoboDojo*, the checklist is fixed in advance. Thus, the VLM annotator localizes transitions between predefined semantic stages rather than freely generating the task decomposition.

Canonicalization. The offline annotation procedure above produces right-open segment ends paired with subtask text for each episode. Before constructing semantic supervision, we canonicalize these segments to remove annotation gaps. An unlabelled interval is not retained as a separate segment, because entering and leaving such a gap would otherwise create two artificial semantic transitions. Leading and internal gaps are therefore absorbed into the following labelled segment, while trailing gaps are absorbed into the final labelled segment, ensuring that each semantic stage boundary induces exactly one transition.

Algorithm 1 Offline subtask annotation

Require: Episode $V = \{x_t\}_{t=1}^{T}$, instruction ℓ, task c
1: Retrieve the ordered stage checklist $\mathcal{S}_c$
2: **if** c has an episode-dependent stage count **then**
3: Infer the stage count from early observations and instantiate $\mathcal{S}_c$
4: **end if**
5: Sample representative observations $\mathcal{O}$ from V
6: Predict ordered stage boundaries $\mathbf{b}$ from $(\ell, S_c, \mathcal{O})$
7: Map $\mathbf{b}$ to the original timeline and construct right-open spans
8: **return** Frame-level active-subtask labels $\{s_t^\star\}_{t=1}^{T}$

Decision labels. Let (s_t, m_t) denote the active subtask and the cumulative memory of completed subtasks at frame t. With a semantic offset matching the replanning interval of 10 frames,

$$d_t = \begin{cases} \texttt{UPDATE}, & \text{if } t < 10, \text{ or } \mathrm{N1}(s_t) \neq \mathrm{N1}(s_{t-10}), \text{ or } \mathrm{N1}(m_t) \neq \mathrm{N1}(m_{t-10}), \\ \texttt{KEEP}, & \text{otherwise,} \end{cases} \tag{12}$$

where $\mathrm{N1}(\cdot)$ collapses whitespace, casefolds, and strips terminal punctuation. Frames with $t < 10$ have no valid cached state and are forced to `UPDATE`; we exclude them when scoring, since counting them would inflate recall for free.

The memory increment is computed positionally: we take the longest common prefix of the two item lists and emit the remaining suffix of m_t. Cumulative memory only grows at the tail, and a set-keyed difference erases a genuine second completion whenever two subtasks share a sentence. On this corpus the set-keyed variant produced an empty increment for 46% and 57% of `UPDATE` frames on two tasks, which trains the model to emit `UPDATE` and then write nothing.

Boundary-aware sampling. Only frames on the replanning phase are eligible. They are partitioned into `UPDATE` frames, hard `KEEP` frames one replanning interval before or after an `UPDATE`, and the remaining random `KEEP` frames. Hard `KEEP` frames are visually close to a transition but do not yet support advancing the subtask, so they are where a spurious `UPDATE` is most likely. Each semantic batch draws 2 frames from each pool. The training split holds 197,617 eligible frames, 9.19% of them `UPDATE`; the sampler replays the small pools and subsamples the large one so the model sees a 1:1:1 ratio.

D Additional Results and Visualizations

D.1 Complete RoboDojo Results

Tab. 10 reports task-wise Score / Success Rate for all 42 RoboDojo simulation tasks, complementing the capability-level averages in Tab. 1. For the 12 Generalization tasks, the standard and randomized evaluation settings are reported separately.

D.2 Complete BiCoord Results

Tab. 11 reports all 18 BiCoord tasks, complementing Tab. 2 in the main text, which shows only the six with the longest expert trajectories.

D.3 Qualitative Rollouts

Figs. 10–12 show closed-loop rollouts on the real robot, RoboDojo, and BiCoord. Each strip pairs the video timeline with the `KEEP`/`UPDATE` decision at every replanning step, the active subtask, and the accumulated memory. The pattern the strips make visible is the one Tab. 4 quantifies: `UPDATE` fires a handful of times per episode, at subtask boundaries, and the state is simply carried in between.

Table 10: **Complete task-wise results on RoboDojo.** Each cell reports Score / Success Rate (in %). For Generalization tasks, *Std.* and *Rand.* denote the standard and randomized evaluation settings, respectively. Methods are grouped according to whether prior embodied robot-data pre-training is used before RoboDojo-specific training. Within the group without robot pre-training, **bold** and underlined numbers denote the best and second-best results for each metric, respectively. Methods with robot pre-training are reported for reference and are excluded from the ranking. Fast-WAM+Interface and **CogWAM (Ours)** are our own evaluation logs.

	Without Robot Pre-training				With Robot Pre-training	
Task	Fast-WAM	Fast-WAM+Interface	StarVLA-α	**CogWAM (Ours)**	$\pi_{0.5}$	GalaxeaVLA (G0.5)
Generalization						
stack_bowls (Std.)	5.87 / 2.67	**72.20 / 68.00**	15.27 / 10.67	**72.20 / 68.00**	76.00 / 72.00	64.07 / 58.67
stack_bowls (Rand.)	0.80 / 0.00	3.00 / 0.00	1.20 / 0.00	**4.80** / 0.00	14.20 / 4.00	20.33 / 13.33
push_T (Std.)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
push_T (Rand.)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
pack_objects_into_box (Std.)	2.87 / 0.00	14.40 / 0.00	8.07 / 2.67	**14.60 / 4.00**	22.80 / 4.00	23.80 / 5.33
pack_objects_into_box (Rand.)	0.27 / 0.00	0.40 / 0.00	1.07 / 0.00	**7.60** / 0.00	15.40 / 1.33	12.67 / 1.33
fold_clothes (Std.)	32.00 / 22.67	53.60 / 52.00	23.47 / 16.00	**63.20 / 60.00**	48.00 / 38.67	53.87 / 48.00
fold_clothes (Rand.)	**2.13 / 1.33**	0.00 / 0.00	0.53 / 0.00	0.00 / 0.00	16.53 / 9.33	18.67 / 14.67
hang_mugs (Std.)	2.20 / 0.00	11.00 / **4.00**	4.33 / 0.00	**15.20 / 4.00**	9.53 / 0.00	12.47 / 4.00
hang_mugs (Rand.)	0.00 / 0.00	**1.20** / 0.00	0.80 / 0.00	**1.20** / 0.00	5.00 / 0.00	4.80 / 0.00
sweep_blocks (Std.)	0.00 / 0.00	**4.00 / 4.00**	1.33 / 1.33	0.00 / 0.00	0.00 / 0.00	2.67 / 2.67
sweep_blocks (Rand.)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	1.33 / 1.33
pour_liquid_into_cup (Std.)	0.00 / 0.00	20.00 / 20.00	18.67 / 18.67	**48.00 / 48.00**	28.00 / 28.00	54.67 / 54.67
pour_liquid_into_cup (Rand.)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	1.33 / 1.33	16.00 / 16.00
make_toast (Std.)	2.00 / 0.00	**15.00 / 4.00**	3.67 / 0.00	**15.00 / 4.00**	9.33 / 1.33	13.33 / 5.33
make_toast (Rand.)	0.67 / 0.00	1.00 / 0.00	0.00 / 0.00	**3.00** / 0.00	2.67 / 0.00	10.67 / 1.33
arrange_largest_number (Std.)	0.80 / 0.00	**16.80 / 12.00**	1.20 / 0.00	11.60 / 4.00	5.13 / 1.33	7.67 / 1.33
arrange_largest_number (Rand.)	**0.27** / 0.00	0.20 / 0.00	0.13 / 0.00	0.20 / 0.00	0.40 / 0.00	1.73 / 0.00
sort_nesting_dolls_by_size (Std.)	0.00 / 0.00	**28.00 / 28.00**	1.33 / 1.33	24.00 / 24.00	10.67 / 10.67	14.67 / 14.67
sort_nesting_dolls_by_size (Rand.)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
store_laptop_and_headphones (Std.)	0.80 / 0.00	26.40 / 8.00	4.80 / 0.00	**44.80 / 40.00**	19.20 / 9.33	35.73 / 20.00
store_laptop_and_headphones (Rand.)	0.00 / 0.00	5.60 / 0.00	0.27 / 0.00	**6.40** / 0.00	13.07 / 1.33	20.80 / 2.67
stack_blocks (Std.)	5.40 / 0.00	22.00 / 16.00	8.33 / 5.33	**40.20 / 36.00**	22.53 / 13.33	49.67 / 42.67
stack_blocks (Rand.)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	**0.60** / 0.00	1.20 / 0.00	3.40 / 0.00
Precision						
fasten_screws	0.40 / 0.00	**10.00** / 0.00	1.60 / 0.00	7.20 / 0.00	15.13 / 2.67	30.00 / 8.67
insert_tubes	3.73 / 0.00	**67.20** / 54.00	13.47 / 2.00	66.80 / **56.00**	17.87 / 3.33	58.53 / 42.67
plug_in_charger	0.00 / 0.00	2.00 / 2.00	0.00 / 0.00	**6.00 / 6.00**	0.00 / 0.00	0.67 / 0.67
pour_balls_into_vase	0.00 / 0.00	16.00 / 16.00	5.33 / 5.33	**24.00 / 24.00**	13.33 / 13.33	28.00 / 28.00
play_Xylophone	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
deposit_coin	2.80 / 0.00	**13.60 / 6.00**	5.20 / 1.33	8.80 / **6.00**	3.20 / 0.67	10.93 / 4.67
insert_key	4.30 / 0.00	13.50 / 0.00	11.50 / 0.00	**14.40** / 0.00	11.90 / 0.00	14.90 / 0.00
build_tower	4.47 / 0.00	34.60 / 30.00	42.07 / 26.00	**68.40 / 60.00**	37.73 / 24.00	82.93 / 78.67
Long-Horizon						
put_bottles_into_dustbin	57.37 / 40.00	**95.20 / 92.00**	48.90 / 28.00	87.70 / 82.00	79.93 / 69.33	96.30 / 94.00
play_tic_tac_toe	0.00 / 0.00	**5.80** / 0.00	2.40 / 0.00	5.20 / **2.00**	8.23 / 1.33	65.23 / 40.00
classify_objects	3.10 / 1.33	9.20 / **2.00**	2.90 / 0.00	**9.30 / 2.00**	24.67 / 12.67	10.33 / 4.00
fill_pen_holder	4.07 / 0.00	40.60 / 20.00	17.57 / 6.00	**44.20 / 24.00**	23.23 / 7.33	41.27 / 18.67
fill_egg_holder	0.73 / 0.00	2.20 / 0.00	1.13 / 0.00	**3.20** / 0.00	2.23 / 0.00	3.03 / 0.00
organize_table	7.83 / 0.00	**36.50 / 4.00**	22.33 / 0.00	35.50 / 2.00	23.33 / 0.00	46.33 / 11.33
play_stacking_toy	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.47 / 0.00
make_kong	0.00 / 0.00	0.00 / 0.00	18.00 / 18.00	**40.00 / 40.00**	26.67 / 26.67	90.00 / 90.00
Memory						
cover_blocks	0.00 / 0.00	16.50 / 10.00	14.73 / 10.00	**18.90 / 12.00**	19.07 / 13.33	20.67 / 14.67
match_and_pick_from_conveyor	20.67 / 20.67	**36.00 / 36.00**	4.67 / 4.67	26.00 / 26.00	14.00 / 14.00	29.33 / 29.33
swap_T	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.67 / 0.67	0.00 / 0.00
press_by_number	0.00 / 0.00	**2.00 / 2.00**	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
imitate_sorting_sequence	0.63 / 0.00	0.50 / 0.00	0.63 / 0.00	**1.00** / 0.00	1.60 / 0.00	1.67 / 0.00
swap_blocks	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
Open						
align_blocks	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
solve_equation	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
stack_blocks_by_language	0.00 / 0.00	0.00 / 0.00	0.53 / 0.00	**2.00 / 2.00**	1.73 / 0.00	0.13 / 0.00
general_pickup	3.33 / 3.33	**14.00 / 14.00**	4.67 / 4.67	**14.00 / 14.00**	12.00 / 12.00	12.67 / 12.67
classify_objects_by_language	0.00 / 0.00	**1.40** / 0.00	0.20 / 0.00	0.40 / 0.00	0.60 / 0.00	1.07 / 0.00
pick_from_conveyor_by_image	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	1.33 / 1.33	0.00 / 0.00
store_tools_in_toolbox	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.17 / 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	0.00 / 0.00
Average	3.48 / 2.03	13.33 / 9.40	6.40 / 3.24	**15.56 / 11.70**	11.44 / 6.93	20.23 / 14.88

Table 11: **Multi-Task Evaluation on BiCoord.** Each cell reports Stage-wise Success Rate (SSR) / Success Rate (SR) in %. **Avg.** is averaged over all 18 tasks. **Bold** denotes the best result.

Method / Task	Average	Balance Roller	Build Bridge	Build Tower With Blocks	Clean Table	Collect Pens	Cook	Divide Block Tower	Exchange Mics
RDT Liu et al. (2025)	34.8 / 16.9	43.0 / 15.0	50.2 / 47.0	1.0 / 0.0	17.5 / 0.0	68.8 / 15.0	31.0 / 9.0	12.8 / 2.0	35.0 / 23.0
OpenVLA-OFT Kim et al. (2025)	36.5 / 23.1	74.5 / 49.0	2.0 / 2.0	0.0 / 0.0	25.0 / 2.0	46.2 / 5.0	25.0 / 10.0	11.1 / 0.0	**67.5 / 66.0**
π_0 Black et al. (2024)	43.2 / 27.2	81.5 / 68.0	47.0 / 44.0	5.0 / 2.0	46.2 / 6.0	81.8 / 47.0	29.5 / 14.0	10.8 / 0.0	59.5 / 52.0
$\pi_{0.5}$ Black et al. (2025)	50.8 / 35.6	89.8 / 82.0	54.3 / 51.0	21.6 / 11.0	61.5 / 24.0	84.8 / 57.0	48.0 / 31.0	14.8 / **1.0**	62.0 / 55.0
CogWAM (Ours)	**58.0 / 43.8**	**96.5 / 95.0**	**60.2 / 58.0**	**35.0 / 19.0**	**80.0 / 50.0**	**87.2 / 66.0**	**77.5 / 52.0**	**18.0 / 1.0**	56.5 / 49.0

Exchange Pots	Extract Bottom Block To Top	Fetch Block With Roller	Handover Block With Bowls	Jigsaw	Match Blocks With Signs	Place Plate And Cup	Put Objects Cabinet	Stack Bowls	Sweep Block
96.0 / 92.0	25.5 / 17.0	49.5 / 0.0	**20.0** / 0.0	11.5 / 0.0	7.0 / 1.0	70.0 / 40.0	49.0 / 31.0	28.3 / 3.0	10.0 / 10.0
56.0 / 53.0	55.5 / 53.0	70.5 / 44.0	3.0 / 0.0	25.2 / 0.0	8.7 / 5.0	79.2 / 55.0	49.5 / 26.0	14.0 / 1.0	44.0 / 44.0
60.5 / 52.0	43.5 / 40.0	69.5 / 43.0	9.0 / **4.0**	36.5 / 6.0	**18.7 / 8.0**	66.2 / 25.0	24.0 / 6.0	28.0 / 12.0	**61.0 / 61.0**
68.0 / 61.0	71.9 / 68.0	71.4 / 45.0	5.7 / 2.0	44.5 / 8.0	15.0 / 6.0	73.8 / 45.0	58.5 / 42.0	34.8 / 18.0	34.0 / 34.0
76.5 / 70.0	**95.0 / 95.0**	**73.0 / 46.0**	3.0 / 0.0	**51.0 / 9.0**	17.3 / 7.0	**80.2 / 63.0**	**84.0 / 74.0**	**40.3 / 23.0**	12.0 / 12.0

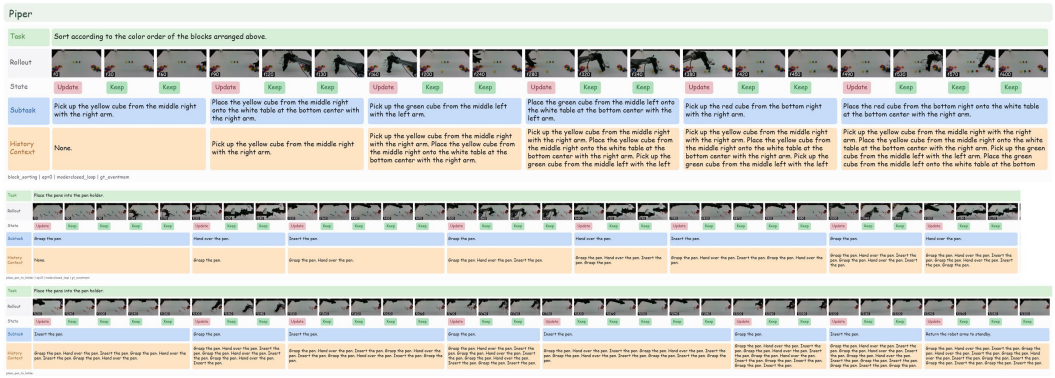


Figure 10: **Closed-loop rollouts on the real robot.** *Block Sorting* and *Fill Pen Holder*. The latter repeats one grasp–hand-over–insert cycle many times, so the memory is the only signal distinguishing the third repetition from the first.

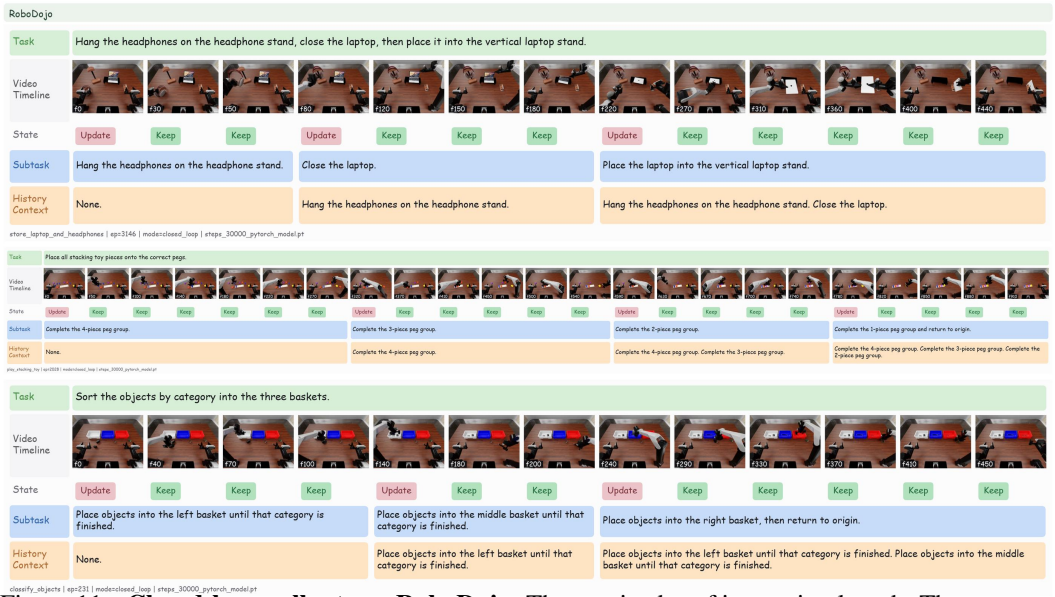


Figure 11: **Closed-loop rollouts on RoboDojo.** Three episodes of increasing length. The memory row grows one entry per `UPDATE`, so the subtask the policy is conditioned on is always paired with an explicit record of what preceded it.

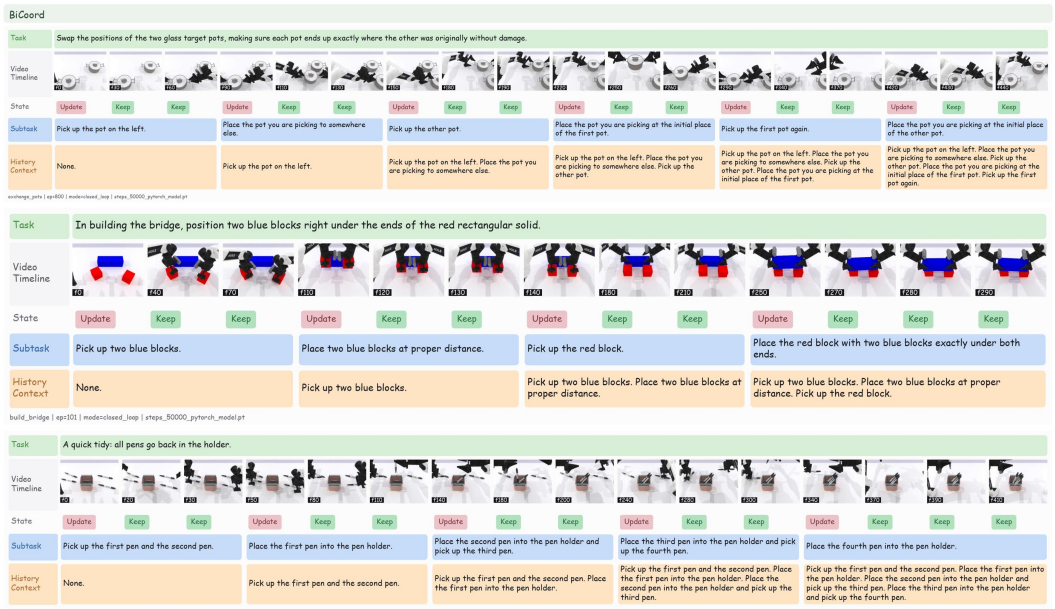


Figure 12: **Closed-loop rollouts on BiCoord.** Bimanual tasks whose stages differ in which arm leads. Subtask text carries the arm assignment, so a transition changes both what is being done and which arm does it.