INTENTWAM: TASK INTENT AS CONTEXT FOR CROSS-TASK GENERALIZATION IN WORLD–ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning from demonstrations offers a promising route to unseen robot tasks, but transferring demonstrated behavior across embodiments requires connecting task intent to the current execution scene. We propose IntentWAM, a world–action modeling framework with a unified conditioning interface that combines an Object-Centric Relational Task Specification (ORTS) with target-scene role bindings. ORTS encodes task roles, endpoint scene graphs, and object dynamics from both robot and human demonstrations. Combined with target-scene role bindings, ORTS provides intent tokens that guide visual-future prediction and subsequent action generation. Policy post-training associates demonstrations and robot executions by task identity, without requiring trajectory-specific demonstration construction or temporal correspondence between them; deployment uses external demonstrations as context without parameter updates. On seven held-out RoboTwin tasks, IntentWAM achieves $47.97 - 48.26\%$ mean success across robot and generated human demonstrations with a fixed UR5-WSG execution policy. With filmed human demonstrations from each evaluated layout, real-world evaluation after task-specific adaptation yields 47.50% success on held-out layouts, versus 22.50% for the strongest evaluated baseline.

1 INTRODUCTION

In-context learning (ICL) promises adaptation to unseen robot tasks through demonstrations rather than parameter updates (Fu et al., 2025; Sridhar et al., 2025; Patel et al., 2026). Demonstrations convey object interactions and motion beyond a task instruction, but their embodiments and scenes can differ from those encountered during execution. Existing approaches use different training associations: Vid2Robot pairs existing trajectories by task language (Jain et al., 2024); Human2Robot uses synchronized human–robot recordings (Xie et al., 2026); and Zero-WAM pairs generated human videos with their originating robot trajectories (Zhou et al., 2026). Task-level association thus already enables reuse of existing trajectories. Our focus is how to represent demonstrated task intent and connect it to the current execution scene across demonstration embodiments.

For world–action models (WAMs), demonstration context must guide both visual-future prediction and action generation (Li et al., 2026; Ye et al., 2026; Yuan et al., 2026). Across embodiments and scene layouts, the transferable content is the intended change in the task-relevant objects, rather than the demonstrator's exact movement. Exploiting this distinction requires identifying object roles, specifying changes in their states and relations, and grounding those roles in the target scene. Object-centric motion representations (Xu et al., 2025) and ORION's object graphs and target-scene trajectory transfer (Zhu et al., 2026) provide foundations for this approach. Building on these ideas, we ask: how can object-centric task intent be grounded in the execution scene to guide world–action modeling across demonstration embodiments?

We introduce **IntentWAM**, a world–action modeling framework that conditions visual-future prediction and action generation on a shared object-centric representation of demonstration intent. This representation, the **Object-Centric Relational Task Specification (ORTS)**, combines task roles, endpoint scene graphs, and object dynamics while retaining other visible objects as scene context (Figure 1). For example, cup placement requires identifying the manipulated cup and target con-

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.

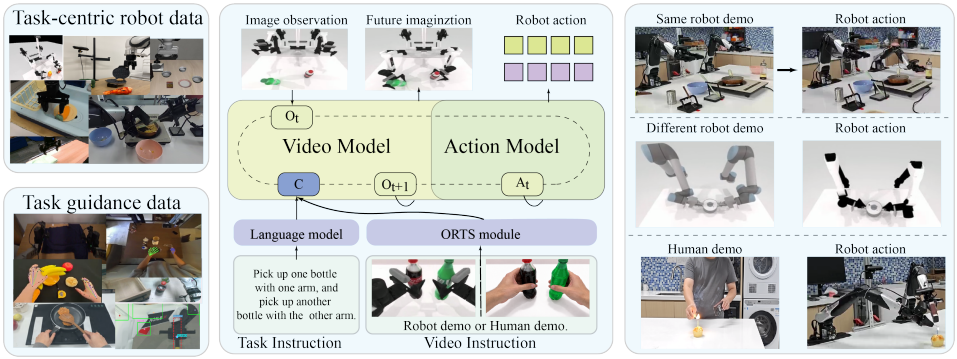

Figure 1: **Overview of IntentWAM.** Robot or human demonstrations supply ORTS specifications. Target-scene role grounding produces task-intent tokens that, together with language and robot history, guide visual-future prediction and action generation. The execution policy remains fixed across demonstration sources.

tainer and transferring the demonstrated change in their configuration. IntentWAM binds these roles to objects in the robot's initial scene and encodes the grounded specification into task-intent tokens. Together with language and robot history, these tokens guide visual-future prediction and subsequent action generation.

We train this interface using task-level demonstration–execution associations. Source demonstrations supply ORTS specifications, while robot executions of the same task supply visual and action supervision. Unlike Zero-WAM's published human-video construction, which pairs generated videos with their originating robot trajectories, our policy post-training does not require a demonstration constructed for its associated execution or temporal correspondence between them. ORTS construction combines pretrained perception with instruction-guided role reasoning. We adapt Zero-WAM's in-context future chunk prediction (IFP) (Zhou et al., 2026) to supply longer-horizon visual supervision. At deployment, an external demonstration supplies context without parameter updates.

On seven RoboTwin tasks held out from policy post-training (Chen et al., 2025b), IntentWAM achieves 47.97–48.26% mean success across robot and generated human demonstrations with a fixed UR5-WSG execution policy. Improvements over Zero-WAM are largest with robot demonstrations, even after its source-specific fine-tuning; IntentWAM retains 45.10–47.10% success under scene variations. In real-world candle lighting, filmed human demonstrations from each evaluated layout guide execution after task-specific adaptation, yielding 47.50% success on held-out layouts compared with 22.50% for the strongest evaluated baseline. Ablations support the contributions of motion summaries, target grounding, and IFP.

Our central contribution is a unified conditioning interface that combines ORTS task roles, endpoint graphs, and object dynamics with target-scene role bindings to guide visual-future prediction and action generation. We establish its effectiveness under task-level demonstration–execution associations through evaluations across demonstration embodiments, held-out tasks, and scene variations, complemented by real-world layout generalization after adaptation and component ablations.

2 Related Work

World–action models. World–action models use predicted visual futures to guide robot actions. Existing approaches extract actions from generated video plans (Du et al., 2023) or jointly model visual futures and actions (Shen et al., 2025; Cen et al., 2025), with multi-chunk supervision extending prediction to later time steps (Xu et al., 2026). Zero-WAM conditions world–action modeling on human videos; its published data construction pairs generated demonstrations with their originating robot trajectories (Zhou et al., 2026). IntentWAM builds on its backbone and auxiliary future supervision, introducing a structured interface that combines demonstrated task intent with target-scene role bindings. Policy post-training associates demonstrations and executions by task identity without requiring this trajectory-specific pairing.

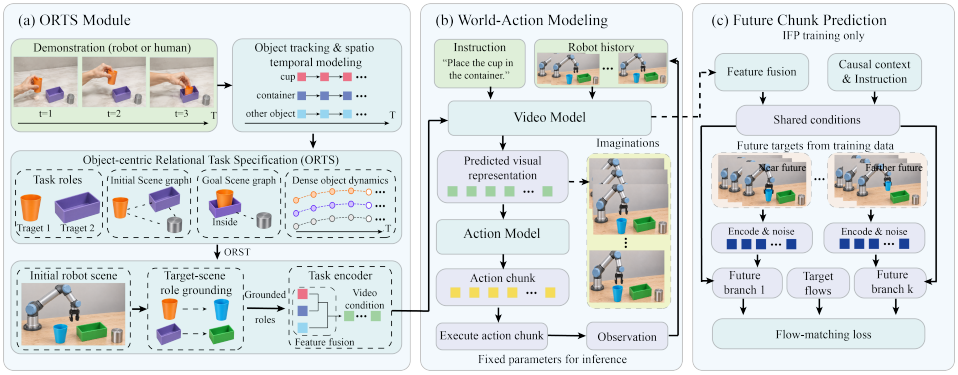

Figure 2: **Architecture of IntentWAM. (a)** ORTS extraction, target-role grounding, and intent-token encoding. **(b)** Intent tokens, language, and robot history guide visual prediction; action decoding uses the predicted representation together with robot history and language, without directly attending to intent tokens. Language also guides role inference in (a); these connections and the history and language inputs to the action model are omitted for clarity. **(c)** Training-only IFP supervises later visual chunks through fused video features and causal context. The dashed feature-fusion connection and all of (c) are omitted at inference. Scene thumbnails, task relations, and imagined futures are schematic; the implemented predicate set is specified in Appendix A.2.

Demonstration-conditioned and in-context policies. Demonstration-conditioned policies specify behavior through examples, differing in their prompt representations and training associations. Video-conditioned policies use demonstrations to specify tasks (Jang et al., 2022; Chen et al., 2025a). Vid2Robot pairs existing robot trajectories by task language, reuses robot trajectories for hindsight human demonstrations, and fuses demonstration and observation features through cross-attention (Jain et al., 2024). Human2Robot uses synchronized human–robot videos to learn robot video prediction and action generation (Xie et al., 2026). Other approaches exploit retargeted human play or unpaired video corpora (Shah et al., 2026; Xu et al., 2023; Kedia et al., 2025). Our focus is explicit object-level task description and target-role binding as a shared condition for world–action modeling.

Object-centric representations and demonstration transfer. Structured task representations and object motion provide interfaces for transferring demonstrated behavior. Program- and graph-based approaches structure task execution (Xu et al., 2018; Huang et al., 2019) or enrich object relations with physical attributes (Wang et al., 2025). Motion-based approaches encode keypoints and action trajectories (Di Palo & Johns, 2024), transform or predict object trajectories (Heppert et al., 2024; Wen et al., 2024), or use object-centric motion fields (Xu et al., 2025; Yin et al., 2025). ORION extracts object graphs, localizes task objects in the execution scene, warps demonstrated object trajectories, and optimizes robot actions (Zhu et al., 2026). IntentWAM builds on the idea of object-centric transfer but uses ORTS and target-role bindings as learned model conditions: intent tokens guide visual-future prediction, whose features then condition action generation.

3 Methodology

3.1 Problem Setup and Overview

Given a task condition $c = (c_{\text{demo}}, c_{\text{lang}})$ comprising a robot or human demonstration and a language instruction, we formulate in-context world–action modeling as

$$p_\theta(V^r_{i+1}, A^r_{i+1} \mid h^r_i, c) = p^v_\theta(V^r_{i+1} \mid h^r_i, c)\, p^a_\theta(A^r_{i+1} \mid V^r_{i+1}, h^r_i, c). \tag{1}$$

Here, V^r_{i+1} and A^r_{i+1} are the next visual latent and action chunks, and $h^r_i = (V^r_{0:i}, A^r_{1:i})$ contains observed video latents and executed actions. V^r_0 encodes the initial observation I^r_0. At deployment, the task is specified through c while model parameters θ remain fixed.

The central challenge is to extract transferable task intent across embodiments and scenes, then ground it in the robot's execution context. IntentWAM represents this intent through the Object-

Centric Relational Task Specification (ORTS), comprising task roles, endpoint scene graphs, and object dynamics (Figure 2). It binds the roles to target objects and encodes the specification and bindings into intent tokens $\mathbf{Z}$ to guide visual prediction and action generation. Policy post-training associates demonstrations and executions by task identity, without requiring trajectory-specific demonstration construction or temporal correspondence between them.

3.2 Object-Centric Task Specification

For the T_d-frame demonstration c_{demo}, we define the Object-Centric Relational Task Specification (ORTS) as

$$\mathcal{I}^d = \left(\mathcal{R}^d, \mathcal{G}_0^d, \mathcal{G}_{T_d-1}^d, \mathcal{M}_{0:T_d-1}^d\right). \tag{2}$$

Its components describe participating roles, the demonstrated initial and goal scenes, and object dynamics. They provide a common object-level interface for robot and human demonstrations without including source joint trajectories or control commands.

Roles and scene graphs. Role assignments $\mathcal{R}^d = \{(q, j_q^d) : q \in \mathcal{Q}^d\}$ associate each participating role q with a source object j_q^d, where $\mathcal{Q}^d$ indexes roles such as manipulated object, target, and tool. Each endpoint scene graph $\mathcal{G}_t^d = (\mathcal{V}_t^d, \mathcal{X}_t^d, \mathcal{E}_t^d)$ records visible objects, their states, and inter-object relations. Roles highlight task participants while the graphs retain supports, obstacles, and other visible objects as context. The cup-placement schematic in Figure 2(a) illustrates distinct cup and container roles and the demonstrated goal configuration. The goal graph records the successful terminal scene; incidental relations involving context objects are not mandatory execution goals. Figure 3 illustrates the same interface for hammering across two robot embodiments.

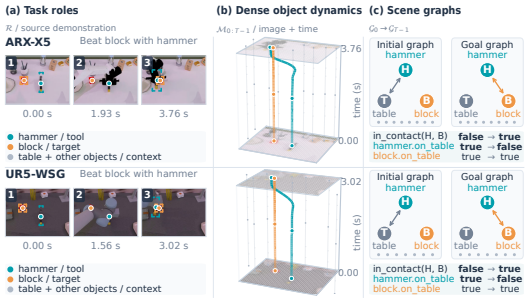

Figure 3: **Annotation-derived ORTS examples.** The same hammering task across two embodiments. H/B/T: hammer/block/table; gray nodes: scene context. (b) Height indicates time; (c) arrows mark contacts involving task objects.

Object dynamics. Endpoint scene graphs specify the observed change but omit the intervening motion. The dynamics component $\mathcal{M}_{0:T_d-1}^d$ retains timestamped image positions, signed velocities, and validity masks for tracked objects. Persistent object identities connect these trajectories to task roles and graph nodes. Static objects remain in the representation, and missing observations are distinguished from valid zero motion. Together, roles identify *which objects participate*, endpoint graphs describe *what changes*, and dynamics record *how objects move*. These image-based measurements remain scene-dependent; target-role grounding connects them to the robot's setting.

Constructing ORTS. The pipeline F_{ORTS} combines pretrained Grounding DINO (Liu et al., 2025) detection with SAM 2 (Ravi et al., 2025) segmentation and tracking. DeepSeek (`deepseek-flash`) reasons over the instruction and detected object semantics to infer task roles, which semantic matching associates with object tracks. Geometric rules construct endpoint graphs, while timestamped tracks supply dynamics and masks. Appendices A.1 and A.2 give the field definitions and extraction details.

3.3 Grounding and Encoding Task Intent

The lower part of Figure 2(a) connects the source specification to the initial robot observation I_0^r. Grounding identifies the target objects that fulfill the demonstrated roles; encoding combines these correspondences with the source task description.

Target-scene role grounding. Grounding matches instruction-conditioned source-role descriptions to objects detected in the initial target observation:

$$\Gamma_0^r = B(\widehat{\mathcal{I}}^d, I_0^r). \tag{3}$$

For each participating role, Γ_0^r records the associated target object and its image geometry. In Figure 2(a), the demonstrated cup and container roles map to the blue cup and green container in the robot scene. Their target geometry locates the intended interaction in that scene, rather than treating source image coordinates as target coordinates. Only task roles require correspondence; context objects remain available without a complete source-to-target scene matching. Grounding uses the initial observation, with no access to the robot's future execution or terminal scene.

Structured intent encoding. The task encoder converts ORTS and target-role bindings into graph, motion, and role tokens. For brevity, we omit hats on the components of the extracted ORTS. Relation-aware attention updates object features in each endpoint graph, and masked attention pooling yields z_0 and z_g for the initial and goal graphs, respectively. The motion encoder embeds per-object position means and standard deviations, mean signed velocities, visibility fractions, and temporal spans, then pools them into z_m. This representation captures motion statistics rather than the complete temporal ordering. For each role q, an MLP combines its source initial-graph node feature n_q^d with the target grounding features g_q^r, producing a role token z_q. The encoder E_ψ forms the intent-token sequence $\mathbf{Z} = E_\psi(\widehat{\mathcal{I}}^d, \Gamma_0^r)$ as

$$\begin{cases} z_q = \mathrm{MLP}_\psi([g_q^r; n_q^d]), \\ \mathbf{Z} = \Pi_\psi\big(\mathrm{Stack}(z_m, z_0, z_g, \{z_q\}_{q\in\mathcal{Q}^d}) + E_{\mathrm{type}}\big)\,. \end{cases} \tag{4}$$

Here, $[\,;\,]$ concatenates features, and Stack assembles the tokens into a sequence. E_{type} identifies token types, and Π_ψ normalizes and projects the sequence to the video model's feature dimension. Motion statistics use valid measurements with image-normalized positions and velocities and timestamps in seconds. Thus, graph and motion tokens describe the demonstrated change, while role tokens attach its participants to the robot's scene. The same interface accepts either robot or human demonstrations.

3.4 Intent-Conditioned World–Action Modeling

We implement the task condition c using intent tokens $\mathbf{Z}$ and language c_{lang} (Figure 2(b)). We build on LingBot-VA's world–action formulation (Li et al., 2026) and the Zero-WAM backbone implementation (Zhou et al., 2026). Intent tokens enter the video transformer as a separate context sequence, bypassing image patch embedding and prediction output heads. At inference, the model samples a visual latent chunk and then an action chunk:

$$\begin{cases} \widehat{V}_{i+1}^r \sim p_\theta^v(\,\cdot \mid h_i^r, c), \\ \widehat{A}_{i+1}^r \sim p_\theta^a(\,\cdot \mid \widehat{V}_{i+1}^r, h_i^r, c). \end{cases} \tag{5}$$

Hats denote model predictions; unaccented V_{i+1}^r and A_{i+1}^r denote execution data in the training objectives. The video branch receives intent tokens $\mathbf{Z}$, language c_{lang}, and robot history h_i^r. The action branch attends to robot-history features, language, and video features computed from the next visual chunk, without directly attending to $\mathbf{Z}$. Demonstration intent therefore influences action generation through the conditioned video features. During training, these video features are computed from execution chunks under teacher forcing; at inference, they are computed from predicted chunks.

3.5 Task-Level Post-Training

Task-level association. For each robot execution ρ^r, we sample a robot or human demonstration according to its task identity:

$$\begin{cases} \kappa = \mathrm{task}(\rho^r), \\ c_{\mathrm{demo}} \sim P_{\mathrm{demo}}(\,\cdot \mid \kappa). \end{cases} \tag{6}$$

Here, $P_{\mathrm{demo}}(\,\cdot \mid \kappa)$ samples from demonstrations of task κ. The policy is conditioned on source ORTS, initial-scene role bindings, and the instruction; windows of ρ^r provide h_i^r and future video–action supervision. During policy post-training, source ORTS and target-role bindings are automatically predicted using the same fixed pipeline as at evaluation; target grounding uses only the initial

observation I_0^r. This association requires shared task identity, without requiring the demonstration to be constructed for the sampled execution or temporally aligned with it.

Auxiliary future supervision. To supervise task evolution beyond the next chunk, we adapt Zero-WAM's in-context future chunk prediction (IFP) (Zhou et al., 2026), drawing on multi-chunk prediction (Xu et al., 2026). We fuse intermediate features of the main video stream into Φ^{i+1} and use K parallel branches to predict flow-matching fields for later visual chunks (Figure 2(c)):

$$\begin{cases} j_k = i + 2 + (k-1)s, \\ \widehat{u}_k = G_{\eta_k}\left(V^r_{j_k,\sigma_k}, \sigma_k; \Phi^{i+1}, \mathcal{K}_i\right). \end{cases} \tag{7}$$

For $k = 1, \ldots, K$, $s \geq 1$ is the chunk stride, $V^r_{j_k,\sigma_k}$ is the future target at noise level σ_k, and G_{η_k} predicts its flow field $\widehat{u}_k$. $\mathcal{K}_i$ contains the permitted causal backbone context and language. The branches access demonstration intent through the main stream's features, without directly attending to $\mathbf{Z}$. The auxiliary branches are used only during training and are omitted at deployment.

Policy optimization. We combine next-chunk video and action supervision with the auxiliary future loss:

$$\mathcal{L}_{\text{policy}} = \lambda_v \mathcal{L}_v + \lambda_a \mathcal{L}_a + \lambda_{\text{IFP}} \mathcal{L}_{\text{IFP}}, \tag{8}$$

Here, $\mathcal{L}_v$ and $\mathcal{L}_a$ supervise the next video and action chunks, $\mathcal{L}_{\text{IFP}}$ supervises the later chunks in Equation 7, and the λ coefficients are nonnegative weights. We update the task encoder and action branch while freezing video weights; gradients through the video backbone remain enabled to train the encoder. The pretrained perception and language models and matching rules remain fixed. Appendices A.3 and A.4 detail the losses and masks.

4 Experiments

Our experiments address three questions. **(Q1)** How well does IntentWAM generalize zero-shot to unseen tasks in simulation using demonstrations from different embodiments, and how robust is it to scene variations? **(Q2)** How well does IntentWAM generalize to held-out real-world layouts after task adaptation? **(Q3)** How do individual components contribute to execution, and does changing the demonstrated goal affect execution under fixed language and initial scenes?

4.1 Simulation Evaluation

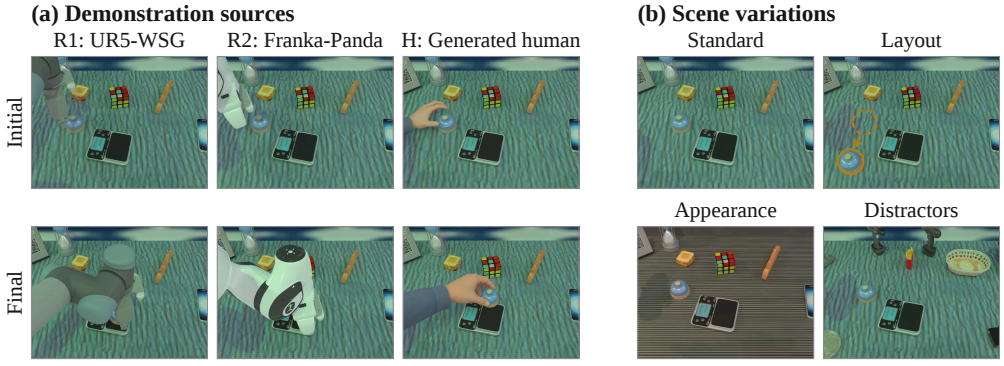

Figure 4: **Simulation evaluation conditions.** (a) R1, R2, and H demonstrations, with initial frames above and final frames below. (b) Standard scenes and layout, appearance, and distractor variations.

Benchmark and evaluation settings. We evaluate Q1 on RoboTwin 2.0 (Chen et al., 2025b), using 43 tasks for post-training and seven held-out tasks, with the pretrained perception and language models fixed. Zero-shot evaluation permits an independent demonstration of the test task as context, with all parameters fixed during execution. UR5-WSG executes every test task, conditioned on same-embodiment robot (R1), cross-embodiment robot (R2), or generated human (H) videos (Figure 4(a)). R2 source embodiments belong to the training pool; the held-out dimension is task

identity. We evaluate both standard scenes and controlled variations in target-object layout, appearance, and distractors (Figure 4(b)). Across scene conditions, demonstrations and policy parameters remain fixed, and the intended task outcome is unchanged. Appendix C.4 details the perturbations and their relationship to training-time randomization. Appendix C specifies the task split, training embodiments, and demonstration selection.

Table 1: **Held-out task success on RoboTwin 2.0 (%).** R1/R2 denote same-/cross-embodiment robot demonstrations; H denotes generated human videos. Average weights the listed task scores equally and is rounded to two decimal places. All methods use task language; † marks language-enabled Zero-WAM (Appendix C.2).

Task	Language only		R1		R2		H	
	WAN-Action	LingBot-VA	Zero-WAM†	Intent-WAM	Zero-WAM†	Intent-WAM	Zero-WAM†	Intent-WAM
Place object on scale	3.00	6.10	18.70	24.20	12.40	25.30	25.20	23.40
Stamp seal	7.30	3.60	31.40	46.90	26.40	46.90	45.60	47.90
Open microwave	2.20	29.30	36.80	51.20	28.40	52.20	58.80	53.80
Move stapler to pad	10.60	23.30	53.70	77.20	47.10	76.40	70.20	74.20
Place bread into basket	15.20	17.30	21.10	37.40	15.30	35.50	35.00	36.30
Place empty cup	38.30	42.30	61.50	88.80	53.50	87.30	83.10	86.10
Stack three blocks	0.00	0.00	3.20	12.10	3.20	13.00	10.20	14.10
Average ↑	10.94	17.41	32.34	48.26	26.61	48.09	46.87	47.97

Baselines and evaluation. WAN-Action (Wan et al., 2025) and LingBot-VA (Li et al., 2026) provide language-only references; Zero-WAM (Zhou et al., 2026) conditions on source videos, while IntentWAM extracts ORTS and grounds its roles in the target scene. Zero-WAM uses separate R1/R2 checkpoints fine-tuned with robot demonstrations on the 43 training tasks and the official checkpoint for H; IntentWAM uses one checkpoint across sources. All methods receive official task instructions; Zero-WAM and IntentWAM share test demonstrations and target initializations. We use official success checkers, a 1,200-action-step limit, and ten evaluation seeds with 100 rollouts per task and condition per seed, separately for each eligible R2 source embodiment. Each task–source pair has one fixed demonstration, so these repetitions measure execution variability conditional on that demonstration. Results average R2 embodiments within each task and then weight the seven tasks equally. Appendices C.3 and C.6 give model configurations and statistical reporting details.

Zero-shot task generalization (Q1). With demonstrations as context, IntentWAM achieves 47.97–48.26% mean success across sources, compared with 17.41% for the strongest evaluated language-only baseline (Table 1). Compared with Zero-WAM after fine-tuning on robot demonstrations, IntentWAM improves success from 32.34% to 48.26% under R1 and from 26.61% to 48.09% under R2, outperforming it on all seven tasks in both conditions. Under H, it achieves comparable success to the official checkpoint (47.97% versus 46.87%). One IntentWAM checkpoint thus supports all three demonstration sources, with the largest gains under robot demonstrations.

Table 2: **Scene robustness on RoboTwin 2.0 (%).** Seven-task mean success with fixed demonstrations and policies. Columns and Standard follow Table 1; Mean drop is computed from the displayed scene means (pp). Reporting follows Section 4.1.

Scene condition	Language only		R1		R2		H	
	WAN-Action	LingBot-VA	Zero-WAM†	Intent-WAM	Zero-WAM†	Intent-WAM	Zero-WAM†	Intent-WAM
Standard ↑	10.94	17.41	32.34	48.26	26.61	48.09	46.87	47.97
Layout ↑	8.90	15.10	28.60	46.30	24.80	46.50	44.20	45.70
Appearance ↑	9.80	17.80	32.00	46.00	24.40	47.00	45.50	47.10
Distractors ↑	7.40	16.40	26.40	47.00	22.10	45.10	44.20	46.70
Mean drop (pp) ↓	2.24	0.98	3.34	1.83	2.84	1.89	2.24	1.47

This pattern is consistent with IntentWAM's object-centric conditioning mechanism. ORTS represents the demonstrated object roles, state changes, and motion, while target-scene grounding

connects this specification to the robot's execution context. The resulting intent tokens guide visual prediction and action generation through a shared interface across demonstration embodiments. The system remains effective under task-level demonstration–execution associations. IntentWAM's post-training permits reuse of same-task demonstrations without constructing each for its associated execution. The comparison evaluates complete systems with different training histories; it does not isolate the interface's contribution (Appendix C.3).

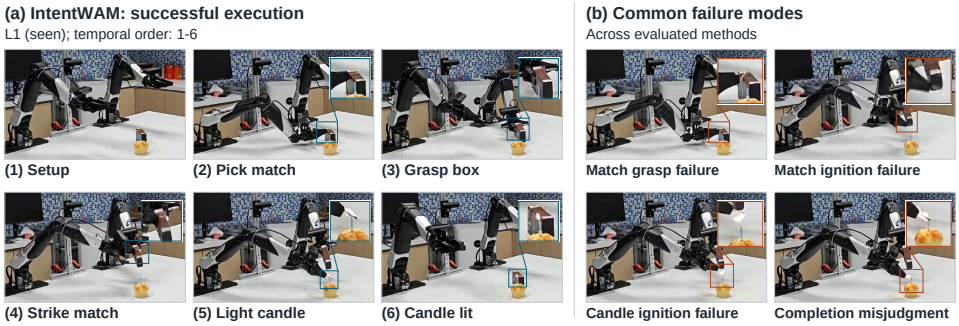

Figure 5: **Real-world execution examples.** (a) Successful execution at seen L_1, in temporal order. (b) Qualitative execution issues across methods.

Robustness to scene variations (Q1). IntentWAM maintains 45.10–47.10% mean success under scene perturbations, compared with 15.10–17.80% for LingBot-VA, the strongest language-only baseline (Table 2). Although LingBot-VA has the smallest mean drop (0.98 percentage points), its standard-scene success is only 17.41%; a small decrease from a low starting success rate provides limited evidence of robustness. We therefore consider absolute success together with performance degradation. Among the demonstration-conditioned methods, IntentWAM exceeds Zero-WAM in all nine source–perturbation combinations. Its mean drops under R1, R2, and H are 1.83, 1.89, and 1.47 points, respectively, compared with 3.34, 2.84, and 2.24 for Zero-WAM, combining higher perturbed-scene success with smaller degradation.

These results are consistent with IntentWAM's extraction of task intent from demonstrations and its grounding in the target scene. ORTS specifies object roles and intended state changes, while grounding binds these roles to current objects and their geometry. This separation is designed to retain task-relevant guidance across layout, appearance, and distractor variations.

4.2 Real-World Evaluation

Layout generalization after adaptation. We evaluate bimanual candle lighting on Nero after adapting at one layout, L_1. All four methods—π_0 (Black et al., 2025), LingBot-VA (Li et al., 2026), RICL (Sridhar et al., 2025), and IntentWAM—receive 30,000 updates on the same 400 robot trajectories from four auxiliary tasks and 20 candle-lighting trajectories at L_1. Policies then remain fixed for 10 trials at each of five layouts: seen L_1 and four layouts L_2–L_5 held out from target-task adaptation. A human evaluator scores success as candle ignition within a three-minute limit.

Table 3: **Real-world layout generalization.** Success rates (%).

Layout	π_0	LingBot-VA	RICL	IntentWAM
Seen L_1	20.00	40.00	30.00	**70.00**
Held-out L_2–L_5	0.00	22.50	15.00	**47.50**

We record two human demonstration videos of candle lighting at each layout, L_1–L_5. Only the two L_1 videos are used as demonstration conditions during task adaptation; videos from L_2–L_5 are excluded from training. At evaluation, IntentWAM extracts ORTS from one video randomly selected from the two corresponding to the evaluated layout, with all model parameters fixed. RICL retrieves four context groups from the same 20 L_1 robot trajectories, while π_0 and LingBot-VA receive no test demonstrations. This evaluates transfer beyond the adaptation layout with layout-specific human guidance; the robot training budget is shared, but test inputs and model configurations differ. Appendix B details the platform, layout split, and method configurations.

Results and remaining challenges (Q2). IntentWAM achieves 70.00% success at L_1 and 47.50% on held-out layouts, exceeding all evaluated baselines (Table 3). On held-out layouts, it exceeds the strongest instruction-only baseline, LingBot-VA, by 25.00 percentage points and the demonstration-conditioned RICL by 32.50 points.

Nevertheless, success declines from 70.00% at L_1 to 47.50% on held-out layouts, indicating that execution reliability remains a substantial challenge. The qualitative examples across methods in Figure 5(b) illustrate grasping, ignition, and completion-recognition issues, highlighting difficulties in precise manipulation and execution-state monitoring. With the same frozen policy, reusing L_1 human demonstrations yields 44.69% success across L_2–L_5, versus 47.50% with current-layout demonstrations in a separate contemporaneous comparison (320 trials per condition; Appendix B.8). This supports cross-layout reuse of existing demonstrations, with a 2.81-point lower mean.

4.3 Component and Behavioral Analysis

Component-ablation protocol. We retrain each variant with one component removed under shared initialization, data, and optimization (Appendix C.7). Evaluation uses UR5-WSG, the seven held-out tasks, standard scenes, and fixed H demonstrations.

Component contributions (Q3). All three components contribute, with Full achieving 47.97% mean success (Table 4). Object dynamics adds motion guidance beyond endpoint states: removing it reduces success to 23.95% (24.02 points) despite retaining both endpoint graphs. Removing target grounding causes the largest drop, to 18.33%, although source ORTS and robot observations remain available. This supports explicitly connecting demonstrated roles to execution-scene objects. Auxiliary future prediction raises success from 32.14% to 47.97% through training-time supervision, with its branches removed at deployment.

Table 4: **RoboTwin component ablations.**

Variant	Full	w/o OD	w/o TG	w/o IFP
Success (%) ↑	47.97	23.95	18.33	32.14
Drop (pp)	—	24.02	29.64	15.83

Table 5: **Demonstration-goal switching (%).**

Method	Goal ↑	Tower ↑
Zero-WAM	3.67	14.33
IntentWAM	**16.33**	**20.50**

Switching demonstrated goals (Q3). We switch R1 demonstrations among six valid stacking orders while holding language and each initial execution scene fixed, with both policies unchanged (Appendix C.8). This intervention tests whether changing the demonstrated goal affects execution. Goal requires a completed tower in the demonstrated order; Tower accepts any valid order. IntentWAM achieves 16.33% Goal success versus 3.67% for Zero-WAM, and 20.50% Tower versus 14.33% (Table 5). These results support the role of demonstrated goals in guiding execution, while overall stacking reliability remains limited.

5 Conclusion

We presented IntentWAM, which combines object-centric task descriptions and target-scene role bindings into a unified interface for world–action modeling across demonstration embodiments. Policy post-training uses task-level associations without requiring trajectory-specific demonstration construction or temporal correspondence between demonstrations and executions. Experiments support zero-shot task generalization and scene robustness in simulation, alongside human-guided layout generalization after real-world task adaptation. These results support grounded task intent as a shared condition for visual-future prediction and action generation from diverse demonstrations.

Limitations and Future Work. Evaluation covers one simulation execution embodiment and one adapted real-world task; execution reliability remains limited and depends on visual perception and initial-scene role grounding. Future work will prioritize generalization to complex backgrounds and cluttered scenes with similar objects, alongside online grounding updates and broader real-world evaluation.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript revision, literature searches and reference verification, implementation assistance, and feedback on experimental methodology and interpretation of reported results. We also used generative video tools to create human demonstration videos, including robot-to-human video conversion. ORTS specifications extracted from training-task videos served as conditioning inputs for training the ORTS module's intent-token encoder and the action branch during policy post-training. Generated videos of held-out tasks were used only as inference-time demonstrations. The pretrained perception and language models in the ORTS extraction pipeline remained fixed. We used DeepSeek (`deepseek-flash`) to infer task-role descriptions from task instructions and detected object semantics. We also used DeepSeek to spot-check reference annotations in the target-object identification evaluation. We did not use generative AI tools to formulate mathematical claims or proofs. The authors take responsibility for the final content of this work, including its text, claims, references, and artifacts.

REPRODUCIBILITY STATEMENT

Appendix A details ORTS construction, target-scene grounding, intent encoding, and training objectives. Appendix C specifies simulation datasets, task splits, baseline configurations, scene variations, and evaluation protocols. Appendix B describes the real-world platform, adaptation data, demonstration inputs, and layout-generalization evaluation. We will release the code upon acceptance.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010. URL `https://www.roboticsproceedings.org/rss21/p010.html`.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025. doi: 10.48550/arXiv.2506.21539. URL `https://arxiv.org/abs/2506.21539`.

Guangyan Chen, Meiling Wang, Qi Shao, Zichen Zhou, Weixin Mao, Te Cui, Minzhao Zhu, Yinan Deng, Luojie Yang, Zhanqi Zhang, et al. See once, then act: Vision-language-action model with task learning from one-shot video demonstrations. *arXiv preprint arXiv:2512.07582*, 2025a.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b.

Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.096. URL `https://www.roboticsproceedings.org/rss20/p096.html`.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 9156–9172, 2023. doi: 10.52202/075280-0403. URL `https://proceedings.neurips.cc/paper_files/paper/2023/hash/1d5b9233ad716a43be5c0d3023cb82d0-Abstract-Conference.html`.

Max Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, Will Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. Icrt: In-context imitation learning via next-token prediction. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5937–5944. IEEE, 2025.

Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Ditto: Demonstration imitation by trajectory transformation. In *2024 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 7565–7572. IEEE, 2024.

De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8557–8566. IEEE, 2019.

Vidhi Jain, Maria Attarian, Nikhil J Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, Igor Gilitschenski, Yonatan Bisk, and Debidatta Dwibedi. Vid2Robot: End-to-end video-conditioned policy learning with cross-attention transformers. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.052. URL https://www.roboticsproceedings.org/rss20/p052.html.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *conference on Robot Learning*, pp. 991–1002. PMLR, 2022.

Kushal Kedia, Prithwish Dan, Angela Chao, Maximus A Pace, and Sanjiban Choudhury. One-shot imitation under mismatched execution. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15649–15656. IEEE, 2025.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Computer Vision – ECCV 2024*, volume 15105 of *Lecture Notes in Computer Science*, pp. 38–55, Cham, 2025. Springer Nature Switzerland. doi: 10.1007/978-3-031-72970-6_3. URL https://doi.org/10.1007/978-3-031-72970-6_3.

Austin Patel, Ben Pekarek, Joel Enrique Castro Hernandez, and Shuran Song. Behavior prompting policy: Demonstrations as prompts for manipulation. *arXiv preprint arXiv:2606.30457*, 2026.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45c1f6a8cbf2da59ebf2c802b4f742cd-Abstract-Conference.html.

Rutav Shah, Shuijing Liu, Qi Wang, Zhenyu Jiang, Sateesh Kumar, Mingyo Seo, Roberto Martín-Martín, and Yuke Zhu. MimicDroid: In-context learning for humanoid robot manipulation from human play videos. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2026. URL https://ut-austin-rpl.github.io/MimicDroid/.

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*, volume 38, pp. 95597–95621, 2025. doi: 10.52202/085713-3197. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/89a3b655a8b68ae1c76b768152c9c19d-Abstract-Conference.html.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. RICL: Adding in-context adaptability to pre-trained vision-language-action models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 5022–5038. PMLR, 2025. URL https://proceedings.mlr.press/v305/sridhar25a.html.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Jianren Wang, Kangni Liu, Dingkun Guo, Zhou Xian, and Christopher G. Atkeson. One-shot video imitation via parameterized symbolic abstraction graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. URL https://www.jianrenw.com/PSAG/.

Chuan Wen, Xingyu Lin, John Ian Reyes So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.092. URL https://www.roboticsproceedings.org/rss20/p092.html.

Sicheng Xie, Haidong Cao, Zejia Weng, Zhen Xing, Haoran Chen, Shiwei Shen, Jiaqi Leng, Zuxuan Wu, and Yu-Gang Jiang. Human2Robot: Learning robot actions from paired human-robot videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(13):11078–11086, 2026. doi: 10.1609/aaai.v40i13.38086. URL https://ojs.aaai.org/index.php/AAAI/article/view/38086.

Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3795–3802. IEEE, 2018.

Gangwei Xu, Qihang Zhang, Jiaming Zhou, Xing Zhu, Yujun Shen, Xin Yang, and Yinghao Xu. Next forcing: Causal world modeling with multi-chunk prediction. *arXiv preprint arXiv:2606.11187*, 2026.

Mengda Xu, Zhenjia Xu, Cheng Chi, Manuela Veloso, and Shuran Song. Xskill: Cross embodiment skill discovery. In *Conference on robot learning*, pp. 3536–3555. PMLR, 2023.

Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2475–2499. PMLR, 2025. URL https://proceedings.mlr.press/v270/xu25a.html.

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*, 2026.

Zhao-Heng Yin, Sherry Yang, and Pieter Abbeel. Object-centric 3D motion field for robot learning from human videos. In *Advances in Neural Information Processing Systems*, volume 38, pp. 55923–55943, 2025. doi: 10.52202/085713-1872. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/50f91c82158baed92e9b7e5491a6d997-Abstract-Conference.html.

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.

Jiaming Zhou, Qihang Zhang, Gangwei Xu, Cunxin Fan, Yujie Zhao, Ruilin Wang, Yiming Luo, Shuai Yang, Xing Zhu, Yujun Shen, et al. Zero-wam: In-context world-action modeling from human videos for open-ended task generalization. *arXiv preprint arXiv:2608.26103*, 2026.

Yifeng Zhu, Arisrei Lim, Peter Stone, and Yuke Zhu. Vision-based manipulation from single human video with open-world object graphs. *Autonomous Robots*, 50(2):27, 2026.

A METHOD DETAILS

This appendix expands the ORTS fields, extraction and encoding pipeline, and training objectives summarized in Figure 2 and Section 3. Throughout, $c = (c_{\text{demo}}, c_{\text{lang}})$ denotes the external task condition, $\mathbf{Z}$ the encoded ORTS intent tokens, and h_i^r the observed video–action history defined in Section 3.1.

A.1 ORTS GRAPHS AND OBJECT DYNAMICS

Task roles and endpoint graphs. Let $\mathcal{V}_t^d$ contain all objects in the source object inventory that satisfy its visibility criterion at frame t. At each endpoint, we define

$$\mathcal{G}_t^d = \left(\mathcal{V}_t^d, \mathcal{X}_t^d, \mathcal{E}_t^d\right), \qquad t \in \{0, T_d - 1\}, \tag{9}$$

where $\mathcal{X}_t^d$ records object states and $\mathcal{E}_t^d$ records inter-object relations. The role assignments $\mathcal{R}^d = \{(q, j_q^d) : q \in \mathcal{Q}^d\}$ map participating roles to source object identities, as in Section 3.2. These annotations highlight participating entities without filtering the graphs: supports, obstacles, and other visible objects remain as context. Object identities link roles, graphs, and dynamics within the demonstration; the two endpoint graphs may contain different object sets. All source demonstrations end in successful task completion, so $\mathcal{G}_{T_d-1}^d$ records the demonstrated goal configuration. This graph provides evidence of the intended outcome; incidental relations involving scene-context objects are not treated as mandatory execution goals.

Dense object dynamics. To describe motion beyond the endpoints, we retain per-frame observations for the union $\mathcal{V}_\cup^d = \bigcup_{t=0}^{T_d-1} \mathcal{V}_t^d$:

$$\mathcal{M}_{0:T_d-1}^d = \left\{ \left(\tau_t^d, \mathbf{u}_{jt}^d, \dot{\mathbf{u}}_{jt}^d, \mathbf{m}_{jt}^d\right) \right\}_{\substack{j \in \mathcal{V}_\cup^d \\ 0 \le t < T_d}}, \tag{10}$$

where τ_t^d is the source timestamp, $\mathbf{u}_{jt}^d$ and $\dot{\mathbf{u}}_{jt}^d$ are object j's image position and signed velocity in the source camera, and $\mathbf{m}_{jt}^d$ contains separate visibility, position-validity, and velocity-validity masks. Static objects and objects entering or leaving view are retained; missing measurements remain distinct from valid zero motion. ORTS excludes demonstrator joint trajectories and control commands but retains scene-dependent object motion, whose use in the robot's environment requires subsequent grounding and encoding.

A.2 EXTRACTION, ROLE GROUNDING, AND TOKEN ENCODING

Demonstration-to-ORTS extraction. ORTS construction combines pretrained visual perception, instruction-conditioned role reasoning, and semantic and geometric rules. Grounding DINO (Liu et al., 2025) supplies object-category labels, detection scores, and bounding boxes at selected demonstration frames. These boxes initialize SAM 2 (Ravi et al., 2025), which segments the objects and propagates their masks through the video to maintain track identities. The implementation uses Grounding DINO Tiny and SAM 2.1 Hiera Base Plus; the reference configuration detects at up to eight keyframes with box and text thresholds of 0.30 and 0.25. Detection uses a fixed vocabulary of object-category prompts. Mask geometry supplies image centers and boxes, while visibility and geometry validity are recorded separately. We use the DeepSeek model `deepseek-flash` to infer role descriptions from the task instruction and detected object semantics; these descriptions are subsequently associated with tracked objects. The complete pipeline therefore depends on both components of the task condition:

$$\widehat{\mathcal{I}}^d = F_{\text{ORTS}}(c_{\text{demo}}, c_{\text{lang}}), \tag{11}$$

where the hat distinguishes video-derived estimates from annotation-derived specifications. The pretrained perception and language models are fixed; this pipeline is separate from the trainable intent-token encoder.

Role semantics and matching. The role vocabulary comprises manipulated object, target, tool, and support, while their assignment depends on the instruction. DeepSeek combines the instruction with detected object semantics to infer which objects serve these roles and produces corresponding semantic descriptions. For example, a cup can be the manipulated object in a cup-moving task or the target container in an object-placement task. The instruction supplies the task semantics, while the demonstration supplies the observed object configurations and motion. The subsequent matching rule lowercases each inferred object description and the detector label and accepts a match when either string contains the other. A unique matching object supplies the role binding; no match is recorded as missing, and multiple matches are recorded as ambiguous. Neither missing nor ambiguous matches are assigned an object identity. Language-model reasoning supplies instruction-dependent role semantics; visual perception and matching associate them with object instances and geometry.

Endpoint predicates and motion. The initial and goal graphs use the first and last demonstration frames. The implemented relation vocabulary is `left_of`, `right_of`, `above`, `below`, `near`, and `in_contact`. Image-center offsets determine directional relations and Euclidean image distance determines proximity; the reference pixel thresholds are 6 and 60, respectively. Contact is a visual estimate based on bounding-box proximity with a 2-pixel tolerance, rather than a measurement of physical contact. The current state rule estimates `on_table` from a largely stationary support region and the object's box bottom relative to its lowest observed image position. Relations with unavailable geometric evidence remain unknown; the rule set does not include a separate containment predicate. Object velocities are finite differences of tracked centers using source-video timestamps, with two-sided differences where both temporal neighbors are available and one-sided differences at the boundaries. Missing positions or invalid time intervals produce invalid velocities, preserving the distinction from valid zero motion.

Target-scene role grounding. Task roles in the extracted specification $\widehat{\mathcal{I}}^d$ identify participants in the source demonstration. The same pretrained perception models provide candidate object labels and image geometry in the robot's initial observation. Semantic matching associates each source role with a target candidate:

$$\Gamma_0^r = B\left(\widehat{\mathcal{I}}^d, I_0^r\right), \tag{12}$$

where I_0^r is the initial observation and Γ_0^r denotes the *grounded roles*: each source task role is associated with a target-scene object and its image geometry. For example, the manipulated-object and target roles in a cup-placement demonstration are associated with the cup and container in the robot's scene. The matching rule uses the instruction-conditioned semantic description associated with each source role and the labels detected in I_0^r, with the same unique-match requirement as source-role assignment. Source and target object identities are local to their respective scenes and are not compared directly. For each role, Γ_0^r stores the target object identity, bounding box, image center, and binding and geometry-validity indicators. Missing or ambiguous bindings remain unavailable; objects absent from the initial observation are not localized using later target frames. Only participating roles require correspondence; other objects remain scene context, and no target terminal scene is used. This stage binds the inferred role descriptions through semantic matching; the downstream role-token encoder learns how to use the supplied correspondences. As shown in Figure 2(a), Γ_0^r is the output of role grounding and is passed directly to the task encoder.

Intent-token encoding. The task encoder combines the predicted source specification $\widehat{\mathcal{I}}^d$ with the predicted target-role bindings Γ_0^r. The specification describes the demonstrated states, relations, and motion, while the grounded roles identify the participating objects in the robot's scene. Equation 4 encodes these complementary inputs as $\mathbf{Z} \in \mathbb{R}^{L_Z \times d_v}$, where $L_Z = 3 + |\mathcal{Q}^d|$ before batch padding accounts for two graph tokens, one motion token, and one token per role, and d_v is the video-stream feature dimension. Before encoding, image positions and velocities are normalized by image dimensions while preserving source timing in seconds and the associated masks. Relation-aware attention encodes each endpoint graph, followed by masked attention pooling into a graph token. The current dynamic encoder embeds per-object position means and standard deviations, mean velocities, visibility fractions, and temporal spans, then pools across objects into a motion-summary token. For each role, the encoder fuses its source initial-graph node feature with its target grounding features. Absent source nodes and unavailable role bindings are masked while retaining their token slots. The

resulting graph, motion, and role tokens receive component-type embeddings and are projected to the video-stream dimension. The video branch receives h_i^r, $\mathbf{Z}$, and c_{lang} to implement the conditional distribution in Equation 5. The action branch receives robot-domain visual and action features and c_{lang}, without directly attending to c_{demo} or $\mathbf{Z}$. Its visual features can carry demonstration information, so the overall action distribution remains conditioned on c.

Conditioning inputs and policy supervision. For both robot and human demonstrations, policy post-training uses source ORTS automatically predicted from the demonstration video and instruction by the fixed perception–reasoning pipeline described above. Target-role bindings are automatically predicted from the initial execution image using the same perception and semantic-matching procedure as at evaluation. These predictions serve as policy conditioning inputs during both training and evaluation; simulator-derived ORTS or target bindings are not substituted for them. The intent encoder is optimized through the video, action, and auxiliary future losses in Equation 8; no separate object-label prediction loss is added to this policy objective. The pretrained perception and language models remain fixed during policy post-training.

Target-object identification accuracy. The ORTS data collection comprises 20,000 RoboTwin trajectories covering 43 tasks, 731 simulated object instances, and 147 object categories, together with 10,000 web images covering more than 200 task objects. These corpus sizes are distinct from the numbers of scored evaluation samples. We assess instruction-conditioned target-object identification on 2,000 RoboTwin samples selected by task and 1,000 randomly sampled web-image samples. Each evaluation sample consists of one task instruction and one corresponding image. Both evaluation sets were held out from ORTS module development and hyperparameter tuning. Reference object identities are obtained from simulator object IDs for RoboTwin and existing annotations for web images. We used DeepSeek to spot-check the reference annotations for 200 samples from each evaluation set. A prediction is correct when the selected object matches the reference instance specified by the instruction: the simulator object ID for RoboTwin or the annotated object for web images. Recognizing only the object category is insufficient when multiple instances are present. Incorrect selections, missed objects, and unresolved predictions are counted as errors and remain in the denominator. Accuracy is $100N_{\text{correct}}/N_{\text{eval}}$, where N_{correct} is the number of correctly identified samples and N_{eval} is the total number of evaluated samples. Table 6 reports 1,984 correct predictions out of 2,000 RoboTwin samples (99.20%) and 921 out of 1,000 web-image samples (92.10%). This metric assesses instruction-guided object selection; it does not measure the accuracy of the complete endpoint graphs or object-motion estimates. These measurements do not establish module accuracy on the seven held-out RoboTwin tasks used for policy evaluation. The web-image assessment concerns static images and is separate from the real-robot execution study in Appendix B. The ablations in Table 4 additionally measure the contribution of the supplied motion summaries and target bindings to execution performance. The annotation-derived examples in Figure 3 illustrate the representation and are not predictions from the perception pipeline.

Table 6: **Instruction-conditioned target-object identification accuracy.** Both evaluation sets are held out from module development and hyperparameter tuning. N_{eval} counts scored instruction–image pairs.

Data source	N_{eval}	N_{correct}	Accuracy (%)
RoboTwin	2,000	1,984	99.20
Web images	1,000	921	92.10

A.3 Auxiliary Future-Chunk Prediction

IFP provides training-only supervision beyond the main predicted chunk, as summarized in Figure 2(c). The construction follows Zero-WAM (Zhou et al., 2026); the full branch inputs and losses are specified below.

Auxiliary future prediction. While the main branch predicts V_{i+1}^r, we collect its robot-video hidden states $\{H_m^{i+1}\}_{m=1}^M$ from selected intermediate layers and fuse them as

$$\Phi^{i+1} = P_{\text{fuse}}\big(\text{Concat}\left(H_1^{i+1}, \ldots, H_M^{i+1}\right)\big), \tag{13}$$

where P_{fuse} is an MLP projection. These hidden states represent the noised current prediction chunk after conditioning on h_i^r, $\mathbf{Z}$, and c_{lang}. We attach K auxiliary branches targeting $V_{j_k}^r$, using the indices and chunk stride defined in Equation 7. Each branch receives its noised future target, the fused representation, and causally available robot context and language. ORTS tokens are not directly appended to the auxiliary inputs; the future losses supervise representations produced by the main model. The branches predict their respective targets in parallel, without using one another's predictions.

Future-chunk supervision. For each target, we sample standard Gaussian noise ϵ_k and a noise level $\sigma_k \in [0,1]$, forming $V_{j_k,\sigma_k}^r = (1-\sigma_k)V_{j_k}^r + \sigma_k\epsilon_k$. The k-th branch predicts $\widehat{u}_k$ as in Equation 7, where $\mathcal{K}_i$ denotes the permitted causal backbone context and language c_{lang}. Its history features may already carry information from $\mathbf{Z}$ through the main video stream; the auxiliary branch does not directly attend to $\mathbf{Z}$. We supervise the auxiliary branches with the future-chunk flow-matching loss

$$\mathcal{L}_{\text{IFP}} = \mathbb{E}\left[\sum_{k=1}^{K} w_k \left\|\widehat{u}_k - \left(\epsilon_k - V_{j_k}^r\right)\right\|_{\Omega_k}^2\right], \tag{14}$$

where w_k weights each prediction horizon and $\|\cdot\|_{\Omega_k}^2$ denotes the mean squared error over valid future entries, with the configured flow-matching time weights. Targets beyond the trajectory boundary are masked out. The auxiliary gradients propagate through the fused features into the main backbone, providing supervision for longer-term visual prediction.

A.4 Main Flow-Matching Losses and Optimization

For stream $x \in \{v, a\}$, let $X_v = V_{i+1}^r$ and $X_a = A_{i+1}^r$ denote the video latent and normalized action targets. Using standard Gaussian noise ϵ_x and noise level $\sigma_x \in [0,1]$, we form $X_{x,\sigma_x} = (1-\sigma_x)X_x + \sigma_x\epsilon_x$. The video flow $\widehat{u}_v$ uses history features, $\mathbf{Z}$, and c_{lang}; the action flow $\widehat{u}_a$ uses robot-domain features, including the teacher-forced next visual chunk V_{i+1}^r, and c_{lang}. We supervise each stream with

$$\mathcal{L}_x = \mathbb{E}\Big[\|\widehat{u}_x - (\epsilon_x - X_x)\|_x^2\Big], \qquad x \in \{v, a\}, \tag{15}$$

where $\|\cdot\|_x^2$ denotes the stream-specific mean squared error with flow-matching time weights; the action reduction excludes history slots, invalid entries, and unused channels. These losses and the auxiliary objective in Equation 14 form the policy objective in Equation 8.

Starting from pretrained world–action weights, the current configuration updates the task encoder and action branch while keeping video weights fixed. The frozen video group includes the auxiliary fusion and video-side prediction modules; action-side auxiliary parameters follow the action branch's optimization scope. Gradients through the video backbone remain enabled so that visual supervision can update the condition encoder. The pretrained perception and language models and the extraction and grounding rules remain fixed throughout policy optimization.

A.5 Deployment

ORTS, target-role bindings, and $\mathbf{Z}$ are computed once from c_{demo}, c_{lang}, and I_0^r. The policy then predicts visual futures and executes action chunks, updating h_i^r with observations and executed actions. All parameters remain fixed, and IFP branches are omitted.

B Real-World Experimental Protocol

We evaluate spatial generalization in bimanual candle lighting after adaptation on data collected with the same Nero platform. The shared robot adaptation corpus contains 400 trajectories from four other manipulation tasks and 20 candle-lighting trajectories collected at one layout, denoted L_1. The main comparison evaluates all four methods in fresh rollouts at five layouts: the seen layout L_1 and four held-out layouts L_2–L_5. This experiment measures generalization to unseen spatial layouts of an adapted task.

B.1 Hardware and Task Configuration

The platform consists of two Nero robot arms equipped with their standard grippers, performing a collaborative manipulation task: *lighting a cake candle with a match*. The setup has three Gemini 335L cameras: one wrist-mounted camera on each arm and one external third-person camera (Figure 6). Robot visual observations comprise the three RGB video streams, with input images of 224×224 pixels. The match is initially placed on top of the matchbox. The right arm picks up the match, strikes it against the matchbox, and uses the burning match to light the candle. The left arm grasps the matchbox and cooperates with the right arm during striking. These operations describe the task sequence; the outcome is evaluated by whether the candle is lit, as defined in Appendix B.6.

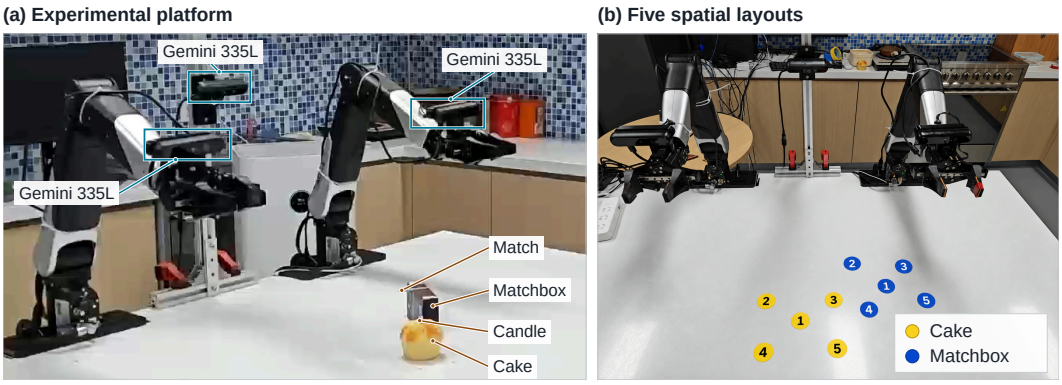

Figure 6: **Real-world platform and spatial layouts.** (a) Nero setup with three boxed Gemini 335L cameras and labeled task objects. (b) Cake (yellow) and matchbox (blue) positions; matching numbers define L_1–L_5. In (b), the tabletop is digitally cleaned and numbered circles replace tape markers.

B.2 Adaptation Data and Human Demonstrations

Robot adaptation corpus. The four auxiliary tasks contribute 100 trajectories each, and the target task contributes $K = 20$ complete trajectories at L_1, giving a total corpus of 420 robot trajectories (Table 7). All four methods use these 420 trajectories for training, with no validation subset withheld from this corpus. The 400 auxiliary-task trajectories and 20 target-task trajectories are mixed during fine-tuning for all four methods. Training and inference configurations are consolidated in Table 9.

Table 7: **Shared Nero robot adaptation corpus.** Four auxiliary tasks provide 400 trajectories, and all 20 target-task trajectories come from layout L_1.

Task	Trajectories
Object-to-container	100
Block arrangement	100
Bottle reorientation	100
Stacking two blocks	100
Candle lighting at L_1	20
Total	420

Target-task support set. We denote the 20 candle-lighting trajectories by $\mathcal{D}_{\text{target}}$. Each covers the full bimanual task, from match pickup to candle ignition, with robot observations, proprioceptive states, and actions. The budget follows RICL's use of 20 target-task robot demonstrations for retrieval and further fine-tuning (Sridhar et al., 2025); it is a common data budget rather than an established optimal sample count for candle lighting. Every method uses this same target-task set for fine-tuning. RICL additionally reuses it as the evaluation retrieval buffer. Thus, RICL receives 20 unique target-task robot trajectories, not separate sets for training and retrieval. The buffer contains no target-task robot demonstrations from L_2–L_5.

Human demonstrations. IntentWAM uses filmed human demonstrations to provide task conditions and robot trajectories to provide execution supervision. Each of the four auxiliary tasks has 10 human demonstration videos, all used for training. Candle lighting has 10 human videos in total, with two recorded at each of L_1–L_5. The two L_1 videos are used for training and reused as demonstration inputs during evaluation at L_1. The eight videos from L_2–L_5 are reserved for evaluation at their respective layouts and are excluded from parameter training and model selection. Thus, the human corpus contains 50 unique videos: 42 used for training and eight used only for held-out-layout evaluation. Evaluation at L_1 uses both a seen layout and seen human demonstrations, whereas evaluation at L_2–L_5 uses held-out layouts and human demonstrations. All robot evaluation rollouts are new executions, separate from the training trajectories and human source recordings.

During training, each robot trajectory is paired with one randomly sampled human video from the same task's training pool: 10 videos for each auxiliary task and the two L_1 videos for candle lighting. For each trial in the main comparison (Table 3), one human video is randomly selected from the two videos associated with the target layout and used as that trial's demonstration condition. Videos may be reused across trials; the 10 trials per layout are not constrained to a five–five allocation between the two videos.

B.3 Adaptation Layout and Spatial Generalization

Paired spatial layouts. The setup uses five paired layouts. Each layout $i \in \{1, \ldots, 5\}$ associates one candle position $\mathbf{p}_i^{\text{candle}}$ with one matchbox position $\mathbf{p}_i^{\text{box}}$. These pairs define five joint layouts rather than all 25 combinations. Figure 6(b) shows both arms and the workspace, with yellow cake-position markers and blue matchbox-position markers. Matching numbers identify each layout pair: in each color group, 1 is central, 2 and 3 are upper left and upper right, and 4 and 5 are lower left and lower right, respectively, in the displayed view. The match starts on the matchbox in each layout, and the arm responsibilities remain fixed.

One adaptation layout and five evaluation layouts. We label the layout used to collect all 20 target-task demonstrations as L_1, and the remaining four as L_2–L_5. The layout sets are

$$\mathcal{C}_{\text{adapt}} = \{L_1\}, \qquad \mathcal{C}_{\text{held}} = \{L_2, L_3, L_4, L_5\}, \qquad \mathcal{C}_{\text{test}} = \mathcal{C}_{\text{adapt}} \cup \mathcal{C}_{\text{held}}. \tag{16}$$

All four methods are evaluated in fresh robot rollouts at all five layouts. Evaluation at L_1 measures seen-layout execution; it does not replay or score the collected training trajectories. Evaluation at L_2–L_5 measures spatial transfer beyond the target-task adaptation layout.

Scope and data separation. The configuration-level evaluation follows the principle of adapting on seen task configurations and testing on held-out configurations used in Zero-WAM's real-world experiments (Zhou et al., 2026). Here, the specified change is spatial layout, and candle lighting is included in robot adaptation training. The held-out-layout claim is relative to the target-task adaptation data; it does not assert that these workspace positions never occur in the four auxiliary tasks. The target task at L_2–L_5 is excluded from parameter training, retrieval support, and model selection. Formal evaluation rollouts at all five layouts are excluded from model selection. Human videos from L_2–L_5 are used only as execution-time inputs; L_1 evaluation reuses the two training videos.

B.4 Compared Methods and Test Inputs

We compare π_0, LingBot-VA (Li et al., 2026), RICL (Sridhar et al., 2025), and IntentWAM after adaptation using the shared robot corpus. Every method receives the four auxiliary tasks and the 20 target-task trajectories at L_1. All parameters remain fixed during evaluation. Table 8 distinguishes the robot training budget from the additional demonstration information available at execution.

The comparison measures seen-layout execution and held-out-layout generalization at a common robot data budget. Demonstration inputs and initialization weights differ across methods, as summarized in Tables 8 and 9. The main system comparison alone does not isolate the effect of the ORTS representation from other training or input differences.

Table 8: **Real-world method and data conditions.** All methods use the same 400 auxiliary-task robot trajectories and 20 candle-lighting trajectories at L_1. All receive task language and their configured robot observations. RICL's retrieval buffer reuses its 20 target-task training trajectories. Human-video counts for IntentWAM are documented separately.

Method	Auxiliary trajectories	Target training trajectories	Target retrieval trajectories	Test demonstration
π_0	400	20	0	None
LingBot-VA	400	20	0	None
RICL	400	20	20	Robot observations, states, and actions
IntentWAM	400	20	0	Human video

B.5 Model Adaptation and Baseline Configurations

Robot interfaces and optimization. All four methods are fine-tuned on the shared Nero corpus and adapted to its observation, state, and bimanual action interfaces using their respective training objectives. We initialize the RICL baseline from pretrained π_0-FAST and follow the official RICL training procedure, mixing all 400 auxiliary-task trajectories with the 20 L_1 candle-lighting trajectories during retrieval-conditioned fine-tuning. IntentWAM associates same-task demonstration conditions with robot execution supervision. All four methods are fine-tuned for 30,000 optimizer updates. We retain the official learning-rate and batch-size settings for π_0, LingBot-VA, and RICL; IntentWAM adopts the corresponding Zero-WAM settings. Normalization statistics and fitted preprocessing use only training data. Checkpoint selection excludes the held-out target layouts and all formal test rollouts. Table 9 lists the public implementation references and the real-world training and inference settings.

Retrieval and intent conditioning. RICL retrieves four groups of observations, states, and action chunks from the fixed L_1 support set, following the context size in the original paper (Sridhar et al., 2025, Appendix C). These four retrieved groups are distinct from the $K = 20$ complete trajectories in the retrieval buffer. For IntentWAM, pretrained visual perception, DeepSeek reasoning conditioned on the instruction, and semantic matching construct ORTS from the human source video and bind its roles to objects in the robot's scene, without per-trial manual correction of the resulting outputs. Role inference and the unique-match rule are described in Appendix A.2.

B.6 Execution and Success Criterion

All four methods use a common task instruction and a shared evaluation initialization list covering L_1–L_5. Trial-level records, including the method, checkpoint, layout and initialization identifiers, source video where applicable, and binary outcome, will be released with the code.

A trial is successful if the robot lights the cake candle within three minutes, and unsuccessful otherwise. A human evaluator determines whether the candle has been lit; candle ignition is the sole task outcome. The manipulation sequence describes execution rather than separate scored milestones.

The policy generates predictions at approximately 10 Hz, while the robot controller operates at 100 Hz. All trials have a common 180-second execution budget. A trial ends successfully when the human evaluator confirms candle ignition; if the candle has not been lit when the budget expires, the trial ends and is counted as unsuccessful. Repeated striking attempts with the same match are allowed within a trial, but replacing the match is not allowed. Repeated strikes count toward the same 180-second budget and do not reset the timer or increase the trial count. The same time limit, human assessment criterion, and retry rules apply to all four methods.

B.7 Repetitions and Statistical Reporting

For Table 3, each method is evaluated in 10 fresh robot trials at each of L_1–L_5, giving 50 trials per method and 200 trials across the four methods. Table 3 reports candle-lighting success at L_1 over

Table 9: **Real-world training and inference configuration.** All four methods mix 400 auxiliary-task and 20 target-task Nero trajectories during fine-tuning. RICL's initialization and training procedure are listed separately from the public checkpoint references for the other methods.

Setting	Configuration
Public implementations	π_0: Physical-Intelligence/openpi; LingBot-VA: Robbyant/lingbot-va; RICL: ricl-vla/ricl_openpi. IntentWAM follows Section 3.
Robot training data	Four auxiliary tasks: 4×100 trajectories; candle lighting: 20 trajectories at L_1. All 420 are mixed during fine-tuning for each method.
Robot visual input	Three RGB streams: left wrist, right wrist, and third-person view; 224×224 input images.
RICL context	Four retrieved groups, following the original paper; retrieval buffer: the same 20 L_1 robot training trajectories.
RICL initialization/training	Pretrained π_0-FAST; retrieval-conditioned fine-tuning on the 420-trajectory mixture following `ricl-vla/ricl_openpi`.
IntentWAM conditioning	Pretrained visual perception, instruction-conditioned DeepSeek role reasoning, and semantic matching; no per-trial manual correction.
Evaluation parameters	Fixed during all evaluation rollouts.
Robot control frequency	100 Hz.
Initialization references	π_0: `pi0_base` from OpenPI. LingBot-VA: `robbyant/lingbot-va-base`. IntentWAM backbone: `zero-wam-pretrain`, consistent with the backbone initialization described in Appendix C.3.
Fine-tuning budget	30,000 optimizer updates for each method.
Learning rate / effective batch	π_0: 2.5×10^{-5} (peak) / 32; OpenPI training defaults and learning-rate configuration. LingBot-VA: 10^{-5} / 8; official post-training configuration. RICL: 2.5×10^{-5} (peak) / 16; official RICL training configuration. IntentWAM: 10^{-4} / 8.
Robot interface	State inputs and predicted actions comprise the end-effector positions and orientations of both arms, together with gripper values normalized to $[0, 1]$. Policy predictions are generated at approximately 10 Hz.

Effective batch size includes data-parallel workers and gradient accumulation. For LingBot-VA and IntentWAM, the value 8 follows the official eight-worker launch settings with per-worker batch size 1 and one accumulation step (LingBot-VA; Zero-WAM). The 30,000-update budget is shared across methods and overrides their original training-run lengths.

10 trials and pools successful and total trials across L_2–L_5 for a held-out aggregate of 40 trials per method. For either aggregate, the success rate is $100\, n_{\text{lit}}/N$, with $N = 10$ or 40, respectively. These summaries use the same binary outcome. Each method uses the same allocation of trials to layouts. Trials that fail during demonstration processing, grounding, or execution remain in the denominator. All five layouts belong to one real-world task.

B.8 Cross-Layout Demonstration Reuse

Purpose. This experiment tests whether the adapted policy can reuse existing human demonstrations when the execution layout changes, without a new demonstration recorded at each layout. It complements the current-layout demonstration setting in Table 3. The task goal remains candle ignition; only the source-demonstration condition changes within each target layout. We evaluate IntentWAM with no additional training or new baseline evaluation.

Conditions and frozen configuration. Let $d_i^{(1)}$ and $d_i^{(2)}$ denote the two existing human videos recorded at layout L_i. At each held-out execution layout L_i, $i \in \{2, 3, 4, 5\}$, we compare:

- **Current-layout:** uses $d_i^{(1)}$ and $d_i^{(2)}$, recorded at the execution layout.
- **Reused L_1:** uses the same two videos $d_1^{(1)}$ and $d_1^{(2)}$ across all four execution layouts.

Both conditions use the same adapted checkpoint, task instruction, robot and camera setup, preprocessing, and inference settings. Policy parameters remain fixed. The source-video sets are fixed, and no video is selected according to success at a particular test layout.

Table 10: **Real-world demonstration-reuse configuration.** Both conditions use the same frozen IntentWAM checkpoint at each execution layout L_2–L_5.

Condition	Source videos at L_i	Trials per video	Trials per layout
Current-layout	$d_i^{(1)}$, $d_i^{(2)}$	40	80
Reused L_1	$d_1^{(1)}$, $d_1^{(2)}$	40	80

Source reuse and target-scene grounding. Source ORTS is extracted automatically from the selected video and fixed instruction. Target-role bindings are predicted anew from the current execution layout's initial image after each reset, using the same automatic pipeline in both conditions. Intent tokens combine the selected source ORTS with these current-scene bindings. Reusing an L_1 video does not reuse the bindings or final intent tokens of an L_1 execution scene. No target terminal observation or manual correction of ORTS or bindings is supplied.

Trial allocation and collection. At each layout and under each condition, each of the two source videos is used for 40 trials, giving 80 trials per condition and layout. The total is $4 \times 2 \times 80 = 640$ robot trials: 320 with current-layout videos and 320 with reused L_1 videos. The two conditions are evaluated contemporaneously under the same protocol, with their executions interleaved. They share the specified object placements and robot reset configurations for each target layout. Physical resets are matched rather than assumed to reproduce exactly identical states. Trial records identify the layout, condition, source video, and binary outcome.

Outcome and termination. A human evaluator assesses candle ignition within the same three-minute limit as in Appendix B.6. Each trial ends when ignition is confirmed or the 180-second execution budget expires; failure to light the candle within this budget is counted as unsuccessful. Repeated strikes with the same match count toward the same time budget without resetting the timer; replacing the match during a trial is not allowed. Both conditions use the same time limit, human assessment criterion, and retry rules. Unsuccessful trials, including demonstration-processing, grounding, and execution failures, remain in the denominator.

Relationship to the main comparison. The current-layout condition is collected contemporaneously with the reused-L_1 condition. Its aggregate is $152/320 = 47.50\%$, numerically equal to the earlier held-out score in Table 3. These are separate evaluations: the earlier score uses 40 held-out

trials with random source-video selection, whereas this comparison uses 320 trials per condition with 40 trials per video at each layout. The earlier trials are not substituted for the current-layout condition in this comparison.

Metrics and reporting. Let $s_{i,c}$ be successful candle ignitions among the 80 trials at layout L_i under condition c. The per-layout rate is $100s_{i,c}/80$, and the aggregate and difference are

$$S_c = \frac{100}{320}\sum_{i=2}^{5} s_{i,c}, \qquad \Delta S = S_{\text{reused } L_1} - S_{\text{current}}. \tag{17}$$

Pooling the 320 outcomes is equivalent to averaging the four layout rates equally. The current-layout success counts for L_2–L_5 are $(39, 41, 35, 37)$ out of 80 each; the reused-L_1 counts are $(35, 38, 36, 34)$. Totals are 152/320 and 143/320, respectively. Table 11 reports percentages, with differences in percentage points computed from the unrounded rates and rounded to two decimal places. A negative difference indicates lower success with reused L_1 demonstrations.

Table 11: **Cross-layout demonstration reuse (IntentWAM).** Success rates (%) under the same adapted checkpoint.

Demonstration context	L_2	L_3	L_4	L_5	Mean
Current-layout	48.75	51.25	43.75	46.25	47.50
Reused L_1	43.75	47.50	45.00	42.50	44.69
Difference (pp)	-5.00	-3.75	$+1.25$	-3.75	-2.81

Each condition has 80 trials per layout and 320 in total; each source video is used for 40 trials per layout. Mean averages the four layouts equally. Differences are reused L_1 minus current-layout rates in percentage points.

Results and scope. Reusing the two L_1 demonstrations achieves 44.69% success across the held-out layouts, compared with 47.50% for current-layout demonstrations, a difference of -2.81 percentage points. Reuse is lower at L_2, L_3, and L_5, and higher at L_4. These results support using existing demonstrations across the evaluated layouts without recording new layout-specific videos, while execution reliability remains limited. Both L_1 videos were used during adaptation, whereas the current-layout videos were excluded from training. The comparison therefore evaluates practical reuse of known demonstrations; it does not isolate source geometry from video familiarity or establish generalization to unseen source videos. It evaluates the complete pipeline rather than attributing the outcome solely to target grounding. The evidence concerns one adapted task, four held-out layouts, and two reused source videos; the point estimates alone do not establish equivalence or absence of degradation.

C RoboTwin Simulation Protocol

This appendix details the task split, demonstration conditions, model configurations, scene variations, and evaluation procedure underlying Tables 1 and 2. Both tables evaluate held-out tasks under R1, R2, and H demonstrations; the second additionally evaluates controlled target-scene changes.

C.1 Task Split and Scope of Generalization

Benchmark version and task partition. We use RoboTwin 2.0 (Chen et al., 2025b) from the `main` branch of the official `RoboTwin-Platform/RoboTwin` repository, with the benchmark checkout fixed at commit `96c1fea` (September 5, 2026).[1] The XPolicyLab submodule is checked out at `fa431ec` (September 12, 2026), rather than the `c37109c` gitlink recorded by that parent commit. Reproducing this environment requires both the parent revision and the actual submodule revision. The task universe is the 50-task suite listed in the official RoboTwin 2.0 documentation.[2] We reserve the seven tasks in Table 12 for evaluation and use all remaining 43 tasks for training. Figure 7 illustrates the scenes of these seven held-out tasks. Thus, the official task list together

[1] `https://github.com/RoboTwin-Platform/RoboTwin/tree/96c1fea`

[2] `https://robotwin-platform.github.io/doc/tasks/index.html`

with the seven held-out identifiers fully specifies the task partition. The partition is applied by task identity across robot embodiments, rather than by randomly splitting trajectories of the same task.

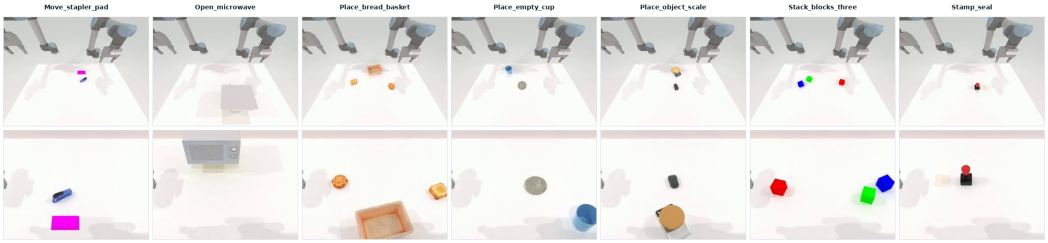

Figure 7: **Scenes of the seven held-out RoboTwin 2.0 tasks.** Columns show, from left to right: move stapler to pad, open microwave, place bread into basket, place empty cup, place object on scale, stack three blocks, and stamp seal. The top row shows scene overviews; the bottom row provides closer views of the task objects. UR5-WSG is the execution robot for all seven tasks.

Training data. Robot training data cover the 43 seen tasks, with task-dependent embodiment coverage. Module-specific uses of these trajectories and baseline training subsets are specified separately in Appendix C.3. The training-data manifest, including actual trajectory counts by task and embodiment, the total used for policy training, and human-demonstration counts, will be released with the code. Task-level association permits reuse of same-task demonstrations without trajectory-specific correspondence.

Collection randomization and acceptance. Collection uses the scene-randomization settings of the official RoboTwin randomized preset, with head-camera distance randomization disabled. The table-height offset is sampled from $[-0.03, 0]$ m; light colors are randomized, with the extreme-lighting option enabled at probability 0.02. Background textures are drawn from the 10,000-image `seen` pool; the 1,000-image `unseen` pool is reserved for evaluation. The clean-background probability is 0.02. Clutter generation requests ten distractors from the available asset pool after excluding task objects, within $x \in [-0.59, 0.59]$ m and $y \in [-0.34, 0.34]$ m; the configured clean-scene branch can omit clutter. Our collection procedure first searches for seeds satisfying planner success, scene stability, and the unmodified official task-success checker, and then renders the accepted episodes. This two-stage acceptance procedure is distinct from the official collection workflow: matching randomization settings does not establish equality of the distributions of accepted episodes. Collection seeds are deterministically derived from the master seed, task, configured embodiment name, episode index, and search epoch/attempt, and are stored with the delivered episodes. Collection-stage infrastructure errors are tracked separately from scene rejection; policy-evaluation failures follow Appendix C.5.

Training embodiments and roles. IntentWAM training uses five robot embodiments: UR5-WSG, Aloha-AgileX, ARX-X5, Franka-Panda, and Piper. Each embodiment can serve as either the execution robot or the source demonstrator for a training task, subject to task-specific data availability. The execution trajectory supplies robot history and future video/action supervision; the source demonstration supplies the task specification for the same task. Source and execution embodiments may be the same or different. For robot-source training samples, we randomly pair two trajectories from the same task's available robot data, assigning one as the execution target and the other as the source demonstration. Human-source samples instead associate a human demonstration with a robot execution of the same task. Robot–robot pairing allows the available embodiments to serve in either role; it does not require every task to support all five embodiments or every possible embodiment pair to occur in the sampled training data. The restriction to UR5-WSG as the execution robot applies to the held-out-task simulation evaluation protocol.

Checkpoint selection on seen tasks. All collected trajectories from the 43 training tasks are used for training, without holding out a separate validation set of trajectories. We evaluate candidate checkpoints through closed-loop simulation rollouts on these 43 seen tasks and select the checkpoint with the highest macro-averaged task success rate. These seen-task rollouts provide the model-selection signal; they are separate from the held-out-task evaluation. The seven held-out tasks are

reserved for final testing and are not used for hyperparameter tuning, early stopping, or checkpoint selection. The selected IntentWAM checkpoint remains fixed across the reported demonstration-source and scene conditions.

Scope of task exclusion. Trajectories from the seven held-out tasks are excluded from policy training across all embodiments, including their use as execution targets or task-demonstration conditions. Derived ORTS specifications and corresponding human videos of those tasks are likewise excluded from policy training. This exclusion includes the trainable intent-token encoder. The pretrained perception and language models are used without task-specific fine-tuning; the extraction and grounding rules remain fixed during evaluation. The task split concerns training performed in this work and does not characterize the original pretraining data of these external models.

Table 12: Held-out RoboTwin tasks and environment identifiers.

Task	Environment identifier
Place object on scale	`place_object_scale`
Stamp seal	`stamp_seal`
Open microwave	`open_microwave`
Move stapler to pad	`move_stapler_pad`
Place bread into basket	`place_bread_basket`
Place empty cup	`place_empty_cup`
Stack three blocks	`stack_blocks_three`

At deployment, an independent demonstration of a held-out task is permitted as context, including a robot video in the robot-source condition. This protocol excludes target-task parameter training, not target-task demonstrations supplied at inference. The source demonstration is separate from the evaluated execution; future observations and actions of that execution are unavailable to the policy. All parameters remain fixed during testing. Pretraining exposure is documented separately: exclusion from our post-training split does not establish that a backbone or perception model has never encountered related tasks. Evaluation across source conditions does not establish transfer to an unseen source domain or embodiment if those sources were present in training.

C.2 Demonstration Sources and Test Inputs

Demonstration conditions. Each held-out task has separate source banks for the three conditions in Table 13. At simulation test time, we fix the execution embodiment to UR5-WSG; IntentWAM training allows all five robot embodiments to serve in both execution and demonstration roles. R1 uses independent UR5-WSG demonstrations. For each test task, R2 includes every embodiment among Aloha-AgileX, ARX-X5, Franka-Panda, and Piper for which successful simulation demonstrations can be generated. H uses generated human demonstration videos, following the video-generation protocol of Zero-WAM (Zhou et al., 2026). Each robot-source video depicts an execution separate from the target rollout, including in R1. The execution embodiment, target action interface, and each policy's parameters remain fixed across all three conditions; changing the demonstrator does not trigger policy adaptation. The R1/R2 distinction concerns the demonstrator's embodiment, not a change in object instance, scene layout, or camera alone.

R2 tests cross-embodiment demonstration transfer on held-out tasks while the execution robot remains UR5-WSG. The four R2 source robot embodiments are part of the training embodiment pool; the held-out dimension is task identity, not robot embodiment. IntentWAM training uses random within-task trajectory pairing as described in Appendix C.1; Zero-WAM's source-specific checkpoints and training are described in Appendix C.3. For each task, R2 is evaluated separately for every included source embodiment, using the same source set and target initializations across compared methods. Source eligibility is determined by successful simulator demonstration collection, independently of the evaluated policies; the resulting source set is fixed before evaluation. Per-embodiment success rates are averaged with equal weight to obtain the task's R2 score; task scores are then macro-averaged over the evaluated tasks. This aggregation prevents a source embodiment with more demonstration videos or rollout attempts from receiving greater weight. Per-embodiment R2 results and task-specific coverage will be released with the code; the aggregation is specified in Appendix C.6.

Table 13: **Demonstration conditions and evaluation coverage.** All conditions use complete external demonstration videos and UR5-WSG as the target execution embodiment. Source robot action annotations are not policy inputs.

Condition	Source demonstrator	Table 1	Table 2
R1	UR5-WSG; independent trajectory	Standard scenes	Standard and shifted scenes
R2	Source pool: Aloha-AgileX, ARX-X5, Franka-Panda, Piper	Standard scenes	Standard and shifted scenes
H	Human (generated video)	Standard scenes	Standard and shifted scenes

Human-video provenance. The H condition on all seven held-out tasks uses generated human videos rather than filmed human demonstrations. Human-video policy conditions include videos obtained through video generation or robot-to-human video conversion, processed by the fixed ORTS pipeline. These training videos and their derived specifications obey the task-level exclusion in Appendix C.1: none of the seven held-out tasks is used to train the policy or intent-token encoder. The generated videos of held-out tasks are supplied only as inference-time demonstrations, with all model parameters fixed.

Source–target pairing. The R1 and R2 scene and camera settings will be documented in the released evaluation configurations. Source and target share the task but can differ in object positions, camera, or background, with identifiable task-role correspondences. Sources are selected using predefined quality criteria rather than downstream policy success, and their assignment to target initializations is shared across compared methods.

Demonstration counts and selection. R1 and H each cover all seven held-out tasks, with one fixed demonstration per task and condition. For R2, we retain one fixed demonstration per task and eligible source embodiment, using the eligibility rule above. We retain robot demonstrations that pass the simulator's task-success checker and show the complete task-relevant interaction and final state. We retain generated human demonstrations that preserve the task goal, object identities, and required action order, and show completion without task-altering artifacts such as object disappearance, teleportation, or interpenetration. If multiple candidates satisfy these criteria, we select one using a predefined random seed before policy evaluation. Selection does not depend on policy success or the success of ORTS extraction or grounding. The selected source videos, including the robot trajectories used to generate H videos, are collected separately from target evaluation rollouts using distinct initialization seeds.

Camera views. R1 and R2 source videos use RGB observations from the source robot's `head_camera`. H videos are generated from head-camera robot videos. During execution, all compared policies receive UR5-WSG RGB observations from the head, left-wrist, and right-wrist cameras, using the same target camera configuration. Thus, the single-view source demonstration and the multi-view target observations serve distinct roles. Source and target camera configurations, including resolution and simulator camera parameters, will accompany the code release; cameras remain fixed across the reported scene interventions.

Demonstration assignment. The selected demonstration remains fixed across all rollouts and evaluation seeds for a given task and source condition; in R2, it remains fixed separately for each source embodiment. Zero-WAM and IntentWAM receive the same source video for each matched target initialization. The R1, R2, and H demonstrations are also held fixed between standard scenes and their paired scene variations. For each task, the R2 source-embodiment set is identical across all scene conditions. The evaluation manifest will be released with the code, specifying the task, source condition and embodiment, source-video identifier, source collection or generation record, evaluation seed, target initialization identifier, and scene condition for each rollout. Source-to-target assignments are fixed before evaluation and shared across compared methods.

Policy inputs. All methods receive their specified target observations and causally available execution history. WAN-Action and LingBot-VA use task language without a source video. Zero-WAM receives the source video through its visual-context interface. IntentWAM derives ORTS from the

same video and estimates role bindings from the initial target observation. Source robot action labels, joint trajectories, and end-effector commands are not demonstration inputs. Simulator annotations used for evaluation do not enter predicted-interface policy inputs. Extraction and grounding failures remain in the denominator of end-to-end task success.

Language controls. All methods use the official RoboTwin task description (the *Description* field) as the language instruction. For IntentWAM, this same instruction also conditions DeepSeek's role reasoning in the ORTS pipeline. For each task, the instruction text is identical across methods and remains fixed across demonstration sources and scene conditions. WAN-Action and LingBot-VA receive this instruction without a source demonstration; Zero-WAM receives it together with the source video, and IntentWAM uses it alongside the video-derived ORTS condition. Language conditioning is enabled for Zero-WAM in all R1, R2, and H comparisons. This language-matched configuration is denoted Zero-WAM† in the result tables: the original reported Zero-WAM ICL inference disables language, although its formulation supports optional language (Zhou et al., 2026). The exact instruction text will be included in each task's released evaluation configuration. Robot-video prompting extends Zero-WAM's published human-video setup; the R1 and R2 models are separately fine-tuned with robot demonstrations before evaluation. These comparisons evaluate complete systems with different source mixtures and checkpoint histories, rather than isolating the effect of the ORTS interface.

C.3 Baseline Initialization and Training

WAN-Action follows the baseline definition in Zero-WAM (Zhou et al., 2026): a causal video–action model with a mixture-of-transformers architecture, initialized from Wan-2.2-TI2V-5B and trained on the 43 seen RoboTwin tasks. LingBot-VA (Li et al., 2026) uses robot video–action pretraining followed by post-training on the same seen-task set. WAN-Action and LingBot-VA use the same 43-task robot execution corpus as IntentWAM, covering UR5-WSG, Aloha-AgileX, ARX-X5, Franka-Panda, and Piper. For Zero-WAM R1 and R2, we use the official implementation at revision `08e2c4a` and initialize separate post-training runs from the released `zero-wam-pretrain` checkpoint.[3] IntentWAM initializes its world–action backbone from the same pretrained release and adds the ORTS interface described in Section 3. The pretrained snapshot is `7040c4195df2`; the full revision identifies the model components used for initialization.[4] IntentWAM's robot training pairs draw both execution and source roles from the five-embodiment pool described in Appendix C.1; UR5-WSG-only execution is an evaluation setting, not a restriction on training supervision.

Source-specific Zero-WAM checkpoints. For R1 and R2, Zero-WAM is separately fine-tuned on our 43 training tasks using robot demonstration videos as source conditions for the corresponding setting; robot execution supervision covers the task-dependent five-embodiment pool. The seven evaluation tasks are excluded from both fine-tuning runs. The reported R1 and R2 means of 32.34% and 26.61% are obtained with these respective fine-tuned checkpoints. For H, we use the official RoboTwin inference checkpoint linked by the Zero-WAM repository, `zero-wam-posttrain-robotwin`, without additional local fine-tuning.[5] The official HumanGen construction pairs each generated human video with its originating robot trajectory; this describes the released human-video pipeline, not the robot-demonstration pairing in our R1/R2 runs. Zero-WAM therefore uses different checkpoints for R1, R2, and H, whereas IntentWAM uses one checkpoint across all three sources. Within each source condition, the selected checkpoint remains fixed across standard and perturbed scenes, with no test-time policy updates.

Post-training configuration. For IntentWAM and our Zero-WAM R1/R2 runs, we adopt the optimizer and schedule from the released RoboTwin training configuration (Table 14). The training budget is 50,000 optimizer updates, with candidate checkpoints saved every 1,000 updates and selected by seen-task rollouts as described in Appendix C.1. These local training settings do not

[3] `https://github.com/robbyant-research/Zero-WAM/tree/08e2c4ae41e2b63573a299825cebe6753481407c`

[4] `https://huggingface.co/Robbyant-Research/zero-wam-pretrain/tree/7040c4195df216c900334ef62d5fdcf05c0601aa`

[5] `https://huggingface.co/Robbyant-Research/zero-wam-posttrain-robotwin`

describe the training history of the official H checkpoint, which the repository reports was post-trained on a mixture of Task-diverse VA, pre-training ICL, and RoboTwin ICL data. The public data release contains only part of that mixture; using the official H checkpoint does not imply matching its training corpus to our local runs. Our RoboTwin data retain the 43-task exclusion and collection protocol in Appendix C.1. Sharing optimizer settings does not imply identical pretrained weights, trainable parameter counts, or training compute across methods.

Table 14: **Policy post-training settings adopted from the Zero-WAM release.** Settings apply to IntentWAM and our Zero-WAM R1/R2 runs; H uses the official checkpoint. The optimizer and schedule follow `va_robotwin_train_cfg.py`. The pretrained perception and language models remain fixed.

Setting	Value
Optimizer	AdamW
Learning rate	10^{-4}
Adam coefficients / epsilon	$(0.9, 0.95)$ / 10^{-8}
Weight decay	0.01
Learning-rate schedule	200-step warmup, then constant
Gradient clipping norm	1.0
Per-device batch / accumulation steps	1 / 1
Parameter precision	BF16
Training budget / checkpoint interval	50,000 / 1,000 updates
Auxiliary prediction modules / stride	4 / 2
Auxiliary horizon weights	$(0.5, 0.25, 0.15, 0.10)$

Trainable components and conditioning. Our Zero-WAM R1/R2 runs follow its released joint video–action and auxiliary-prediction training recipe, with source-video dropout 0.1 and target-text dropout 0.4. IntentWAM retains the optimization scope in Section 3.5: policy training updates the task encoder and action branch while keeping video weights fixed and preserving gradients through the video backbone to the condition encoder. ORTS extraction and role grounding combine fixed pretrained perception, instruction-conditioned DeepSeek reasoning, and semantic matching, as detailed in Appendix A.2. During policy post-training and evaluation, IntentWAM predicts ORTS from the supplied demonstration and grounds task roles in the initial target observation; simulator-provided ORTS or role bindings are not substituted for these predictions. Its action decoder uses the predicted next visual chunk, robot history, and language without directly attending to ORTS tokens (Section 3.4); auxiliary future-prediction branches are omitted at deployment.

Inference configuration and robot adaptation. We follow the released sampling configuration with 50 video denoising steps and 50 action denoising steps, an inference `frame_chunk_size` of 2, and action guidance scale 1. For Zero-WAM, source-video guidance remains 5; we set `TARGET_TEXT_CFG=1` to retain the official task instruction in the target stream, implementing the language-matched comparison in Appendix C.2. The released negative default instead selects the empty-text embedding. Target camera images use the released 224×288 preprocessing resolution (height $\times$ width); Zero-WAM source videos use 320×480 at 12 fps. Robot-source videos in R1/R2 enter the same visual-context interface used for H. The released robot configuration contains Franka-specific action channels and normalization statistics; for our embodiment pool, action-channel mappings and inverse decoding follow each robot's control interface, and normalization statistics are computed from training trajectories only. UR5-WSG execution therefore uses its corresponding action mapping and statistics.

C.4 Scene Variations

We construct layout, appearance, and distractor variations using RoboTwin's scene-generation and randomization mechanisms with specified random seeds. The task-specific configurations, seed lists, standard–perturbed scene assignments, and code and asset version identifiers will be released with the code. Demonstrations and policy parameters remain fixed across conditions. It covers R1, R2, and H on the seven held-out tasks. For each R2 source embodiment, the same demonstration is used across standard, layout, appearance, and distractor conditions, and the eligible source set remains fixed. Standard scenes use the selected benchmark configuration with randomized evaluation initializations. The Standard reference in Table 2 uses the same task set, checkpoints, source assignments, and target initializations as Table 1. Each shifted scene is paired with a standard initialization

through a recorded base scene identifier. Perturbations preserve the intended outcome, identifiable object roles, and physical feasibility; their ranges are fixed before inspecting test results.

Layout. Positions of selected task-relevant objects are changed while object identity and the required state change remain fixed. Layouts remain reachable and collision-feasible.

Appearance. Lighting and background textures are changed through the scene configuration. The appearance configuration also changes the table-height offset, moving the table and objects together; it therefore includes geometric as well as photometric variation. Role-defining attributes remain interpretable from the task inputs.

Distractors. Distractor identities and placements are resampled using RoboTwin's clutter-generation mechanism, while preserving an identifiable correct role assignment and feasible execution. The primary condition excludes unavoidable obstructions. Success does not require reproducing incidental background relations in the demonstration.

Relation to collection randomization. The training collection already includes background, lighting, table-height, and clutter variation, as described in Appendix C.1. Using the disjoint `unseen` background pool provides held-out texture identities, but does not by itself establish a distribution shift for other scene factors. Table 2 evaluates the layout, appearance, and distractor interventions separately using paired standard and perturbed scenes.

A mean drop across perturbations is computed only on a common task set.

C.5 Execution and Success Criteria

All methods use the official task-success checkers from the fixed RoboTwin revision in Appendix C.1, with identical target initializations and reset rules. Success is determined by the target environment, independently of the ORTS extracted from the source demonstration. Each rollout is limited to 1,200 environment action steps across all tasks, methods, and scene conditions. One step corresponds to one environment action call; individual actions within a predicted chunk count separately toward this limit. Execution stops upon success or exhaustion of the 1,200-step budget, with any remaining actions in a predicted chunk discarded. Rollouts that reach the limit without success, time out, or encounter policy-side ORTS extraction, target-grounding, or execution failures are counted as unsuccessful and retained in the success-rate denominator.

C.6 Repetitions and Statistical Reporting

For Tables 1 and 2, we use ten evaluation seeds, with 100 closed-loop rollouts per task, source condition, and scene condition within each seed. The 100-rollout budget within each seed follows the per-task evaluation budget of Zero-WAM (Zhou et al., 2026). For R2, this budget applies separately to each eligible source embodiment in each scene condition. Each task therefore has 1,000 rollouts per R1 or H scene condition, and 1,000 rollouts per R2 source embodiment per scene condition; language-only references use the same 1,000-rollout budget per task and scene condition. The single demonstration selected for each task and source remains fixed throughout these rollouts, as specified in Appendix C.2. These are evaluation seeds, not independent training runs. For R1, H, and language-only evaluation, the 1,000 binary outcomes per task yield success rates in increments of 0.1 percentage points, naturally producing a trailing zero when displayed to two decimal places. Averaging across tasks or R2 source embodiments can produce finer increments; this per-task resolution does not prescribe the precision of those averages. Methods share target initializations and, where applicable, source assignments. Perturbations are paired with standard-scene counterparts. Locally trained checkpoints are selected by the highest macro-averaged success rate on the 43 seen tasks, as described in Appendix C.1; Zero-WAM H uses the fixed official checkpoint. No demonstration-source or scene condition uses a checkpoint selected from the seven held-out tasks.

Let $y_{m,d,\xi,t,s,n} \in \{0, 1\}$ denote success for method m, source condition d, scene condition ξ, task t, evaluation seed s, and rollout n. For language references, d denotes no source demonstration. With

N attempts per task and seed and $J = 7$ tasks, we compute

$$\begin{aligned} \mathrm{SR}_{m,d,\xi,t,s} &= \frac{100}{N} \sum_{n=1}^{N} y_{m,d,\xi,t,s,n}, \\ \overline{\mathrm{SR}}_{m,d,\xi,s} &= \frac{1}{J} \sum_{t=1}^{J} \mathrm{SR}_{m,d,\xi,t,s}. \end{aligned} \tag{18}$$

For R2, the per-task score is first aggregated across source embodiments rather than pooled over all attempts. Let $\mathcal{E}_t^{\mathrm{R2}}$ be the nonempty set of included source embodiments for task t, and let $\mathrm{SR}_{m,e,\xi,t,s}$ denote the success rate for source embodiment e, computed with its own rollout count. We use

$$\mathrm{SR}_{m,\mathrm{R2},\xi,t,s} = \frac{1}{|\mathcal{E}_t^{\mathrm{R2}}|} \sum_{e \in \mathcal{E}_t^{\mathrm{R2}}} \mathrm{SR}_{m,e,\xi,t,s} \tag{19}$$

as the task-level input to the macro average in Equation 18. This R2 aggregation applies separately to standard, layout, appearance, and distractor conditions, using an identical source-embodiment set for each task across all conditions. A task without an eligible R2 demonstration is marked unavailable, not assigned zero success; source comparisons use the common task subset. The released evaluation manifest will identify task-specific R2 source sets and any coverage exclusions; evaluation uses one fixed demonstration per source embodiment and 100 rollouts per source embodiment, scene condition, and evaluation seed. Per-task scores are summarized by their mean across evaluation seeds. In Table 1, Average is computed from the seven listed task scores with equal weights and rounded to two decimal places. Evaluation across seeds is conditional on the fixed source demonstrations and does not measure variation across independently sampled demonstrations.

For Table 2, let $S_{m,d,\xi}$ denote the displayed seven-task mean success rate for method m, demonstration source d, and scene condition ξ. Mean drop in percentage points is computed from these reported scene means:

$$\mathrm{MeanDrop}_{m,d} = S_{m,d,\mathrm{standard}} - \frac{1}{3} \sum_{\xi \in \mathcal{P}} S_{m,d,\xi}, \tag{20}$$

where $\mathcal{P} = \{\mathrm{layout}, \mathrm{appearance}, \mathrm{distractors}\}$. The final difference is rounded to two decimal places without rounding intermediate calculations. Negative values indicate improvement and are not clipped. Absolute success accompanies drops because a small decrease from a low standard score does not establish robustness.

C.7 Ablation Protocol and Reporting

Shared evaluation setting. Table 4 evaluates the seven held-out tasks in Table 1 with UR5-WSG in standard RoboTwin scenes. The source condition is H, with one fixed generated human demonstration per task shared across variants. Full retains object dynamics, target grounding, and IFP; the three ablated variants each remove one of these components while retaining the H demonstration condition. The three ablated variants receive the same task language and share their target-initialization list. Each ablated variant uses three evaluation seeds and 100 rollouts per task per seed, totaling 300 rollouts per task, with the official success checkers and 1,200-action-step limit specified in Appendix C.5. Full reuses the ten-seed H evaluation reported in Table 1. Demonstration-processing and execution failures remain in the denominator.

Single-component changes. **w/o Object Dynamics (OD)** removes the motion-summary input to the condition encoder while retaining task roles, both endpoint graphs, target grounding, and IFP. This comparison measures the contribution of the encoded motion statistics, rather than establishing temporal-order understanding. **w/o Target Grounding (TG)** removes Γ_0^r and its target-object identity and geometry features while retaining source ORTS, source-role features, robot history, and IFP. Target-scene information remains available through robot observations, so this variant removes explicit correspondence rather than all access to the target scene. **w/o IFP** sets the auxiliary loss weight to zero while preserving the main video and action losses and their weights. The IFP auxiliary branches are omitted at deployment for all variants.

Table 15: **Per-task RoboTwin ablations (%).** Full and all ablated variants use the same H demonstrations and standard scenes; variants differ in the retained components. Average weights all seven tasks equally; Table 4 summarizes these results.

Task	Full	w/o OD	w/o TG	w/o IFP
Place object on scale	23.40	16.67	4.67	17.33
Stamp seal	47.90	18.67	11.67	21.67
Open microwave	53.80	13.67	9.33	18.33
Move stapler to pad	74.20	38.00	32.33	51.67
Place bread into basket	36.30	26.00	21.67	32.00
Place empty cup	86.10	51.33	46.33	77.33
Stack three blocks	14.10	3.33	2.33	6.67
Average ↑	47.97	23.95	18.33	32.14

Training controls. Each ablated policy is trained with its component change from the same pretrained initialization as Full, rather than masking a component only at test time. Training trajectories, demonstration-source mixture, same-task pairing rule, optimizer settings, update budget, and checkpoint-selection rule are shared. Fixing H at evaluation does not imply H-only policy training. The trainable parameter groups of retained components follow the full-model configuration.

Aggregation and interpretation. In Table 15, ablated-variant task entries report mean success across three evaluation seeds, and Average weights the seven tasks equally. Ablated-variant success rates and task averages are rounded to two decimal places, while Full retains the main table's displayed values. Drop in Table 4 is computed from the reported mean success rates, using 47.97% for Full, and displayed to two decimal places. Evaluation seeds are not independent training runs, and the fixed demonstration per task does not measure variation across demonstration samples. The table reports means only and makes no statistical-significance claim. The IFP comparison concerns execution performance; it does not directly measure visual-prediction accuracy or reliance on demonstration content.

C.8 Switching Demonstrated Goals in Three-Block Stacking

Purpose. We evaluate demonstration-goal switching on RoboTwin's `stack_blocks_three`. The diagnostic tests whether changing the demonstrated stacking goal affects execution under fixed language and matched initial execution scenes. The diagnostic varies the final color order, not merely the trajectory used to reach an unchanged goal. Table 5 reports aggregate Goal and Tower rates, separately from the standard seven-task evaluation and component ablations.

Fixed instruction and goal permutations. For the evaluation in Table 5, we use the following identical instruction across all six demonstration conditions:

> There are three blocks on the table. Move them to the center of the table and stack them into a single tower.

The instruction specifies the shared manipulation task but leaves the color order unspecified. With this instruction and each initial execution scene held fixed, the target color order is supplied only by the demonstration. Let $\pi = (b, m, t)$ denote the bottom, middle, and top colors, respectively. The six permutations in Table 16 define equally weighted targets. The same fixed instruction is used throughout ORTS extraction, role inference, target grounding, and policy inference. The target permutation identifier, evaluator goal, and any order-specific collection instruction are unavailable to the evaluated policy and its perception pipeline.

Frozen models and matched demonstrations. The diagnostic uses R1 robot demonstrations and UR5-WSG execution, with the existing IntentWAM checkpoint fixed throughout. Zero-WAM's R1 checkpoint provides a video-conditioned comparison under the same inputs and execution settings. No stacking-order examples are added to policy training, and neither model is selected or adapted using these diagnostic outcomes. Each matched demonstration set contains one valid video per permutation. Within each set, the source robot, camera, block appearances, initial block positions,

Table 16: **Three-block goal permutations.** R/G/B denote red/green/blue. Orders are always listed from bottom to top; P_1 is the original task goal. Each adjacent row pair keeps the bottom color fixed and exchanges the middle and top colors.

Goal	Bottom	Middle	Top	Required support relations
P_1	R	G	B	Green on red; blue on green
P_2	R	B	G	Blue on red; green on blue
P_3	G	R	B	Red on green; blue on red
P_4	G	B	R	Blue on green; red on blue
P_5	B	R	G	Red on blue; green on red
P_6	B	G	R	Green on blue; red on green

and background remain fixed; only the demonstrated stacking goal and the actions needed to achieve it change. Each video starts with three separate blocks and ends with a released, stable tower in its designated order. Demonstrations are accepted by task validity and visibility, not by downstream policy success; no rendered text or filename-derived goal information is supplied as context.

Validation of demonstration-goal information. We verified the demonstrated stacking orders using the initial graph, goal graph, and object-motion information jointly. The initial graph records the starting object configuration, the goal graph records the terminal object relations, and tracked motion links object identities across the demonstrated manipulation. Together, these components distinguish the six demonstrated stacking orders and supply the goal information encoded in the intent condition. All test inputs remain automatically predicted, with no manual role correction or replacement by simulator-derived ORTS. Unresolved colors, failed extraction, and ambiguous bindings remain evaluation failures rather than grounds for dropping trials.

Execution pairing and budget. Execution starts from feasible scenes with three separate blocks and varied color-to-position assignments, selected independently of evaluated-model outcomes. For each matched comparison, we evaluate all six permutations from the same saved simulator state, with identical robot initialization, cameras, lighting, distractors, and action budget. We reset policy history between rollouts, reuse the corresponding policy-sampling seed across the six conditions, and randomize their run order. Source and target layouts are sampled independently. Extraction and language-model sampling settings remain fixed, with predictions cached consistently across repeated uses of the same input. Evaluation uses ten evaluation seeds and ten executions per color order within each seed. Across six orders, this yields $10 \times 6 \times 10 = 600$ attempts per method. Evaluation seeds are not independent training runs.

Aggregation and precision. With ten trials per seed and order, equal averaging over six orders and ten seeds is equivalent to pooling all 600 binary outcomes. Success rates are rounded to two decimal places. Percentage-point differences and Tower-minus-Goal values are computed from the displayed rates.

Goal-aware scoring and termination. The evaluator recognizes all six orders using a task checker parameterized by the bottom, middle, and top object roles. The center-region, support, gripper-release, and stability criteria use identical geometric tolerances across permutations. Each rollout terminates at the first complete, released tower at the table center that satisfies the common checker for 20 consecutive environment action steps, or at the 1,200-action-step limit. The confirmation steps count toward that limit. A stable tower in another order is classified as a wrong-order completion; it is not counted as success for the requested goal. The legacy original-order success signal is excluded from rollout termination and outcome labeling. Simulator object identities and poses are used only by the evaluator to determine the final order, never to construct the policy's conditions.

Metrics and reporting. Let $o_{s,\pi,r}$ denote the observed valid tower order, or $\emptyset$ for failure, on trial r of evaluation seed s with requested order $\pi \in \Pi$. For J seeds and n trials per seed and order, the

equally weighted aggregation is

$$S_{\text{goal}} = \frac{100}{6Jn} \sum_{s=1}^{J} \sum_{\pi \in \Pi} \sum_{r=1}^{n} \mathbf{1}[o_{s,\pi,r} = \pi],$$
$$S_{\text{tower}} = \frac{100}{6Jn} \sum_{s=1}^{J} \sum_{\pi \in \Pi} \sum_{r=1}^{n} \mathbf{1}[o_{s,\pi,r} \neq \emptyset]. \tag{21}$$

Here, $J = 10$ and $n = 10$, giving 600 attempts per method. Both metrics retain all attempts, including extraction, grounding, and execution failures, in the denominator; Goal is not conditional on successful tower completion. Wrong-order completion is $S_{\text{tower}} - S_{\text{goal}}$, in percentage points of all attempts. The displayed scores give improvements of 12.66 points in Goal and 6.17 points in Tower for IntentWAM, with wrong-order completion at 4.17% versus 10.66%.

Interpretation. The intervention changes the demonstrated goal while holding language and the initial execution scene fixed; Goal and Tower distinguish demonstrated-order completion from general stacking reliability. IntentWAM's higher Goal rate and lower wrong-order completion are consistent with demonstrated goal information guiding execution. The aggregate results do not establish reliable switching for every individual color order. This experiment concerns color-specific terminal relations; it does not establish understanding of arbitrary temporal action sequences or broad task generalization.