EVENT-ALIGNED VISUAL ACTION REASONING FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World-Action Models (WAMs) utilize future visual prediction as an intermediate reasoning process to guide action generation. However, existing WAMs typically structure visual imagination according to predefined temporal intervals, without explicitly accounting for the different roles of task-critical interactions and connecting transitions. We argue that effective visual foresight should align directly with task-relevant interactions and their corresponding reasoning demands. To this end, we introduce an event-aligned visual action reasoning framework that organizes visual-action prediction around interaction events. Through event-aligned visual-action supervision, WAM learns to generate event-aligned visual context in each imagined rollout, placing greater emphasis on critical state changes that inform action generation. This shapes the visual reasoning granularity according to the underlying interaction dynamics, with detailed reasoning around task-critical events and coarser progression through connecting transitions. Furthermore, we introduce an execution validity head that identifies the valid portion of each predicted action sequence, avoiding redundant actions during chunked inference. Experiments demonstrate a 10.26 percentage point improvement in DOMINO success rate over baseline and competitive performance on RoboTwin 2.0. It also transfers from DOMINO Level 1 to Levels 2 and 3 without target-level adaptation.

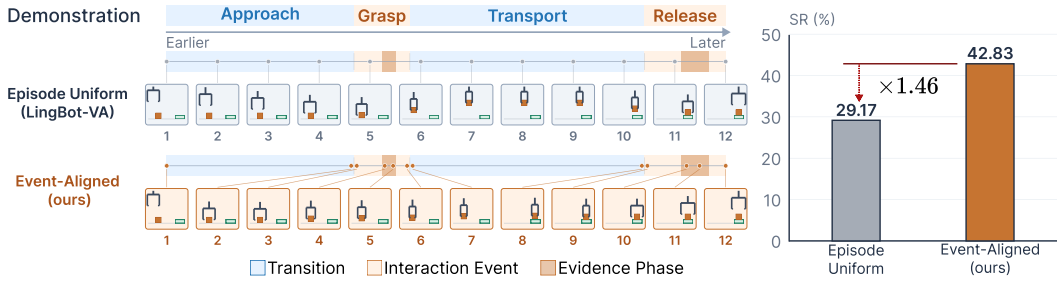
1 INTRODUCTION

Recent World Action Models (WAMs) have emerged as a promising paradigm for robot manipulation by coupling action generation with future visual prediction (Li et al., 2026a; Aditi et al., 2026; Ye et al., 2026). Unlike conventional vision-language-action (VLA) policies that map current observations and instructions directly to actions (Kim et al., 2025; Black et al., 2026; Bjorck et al., 2025; Black et al., 2025; Team, 2025; Zheng et al., 2025), WAMs additionally model how the scene may evolve and leverage the predicted future for action generation. Such visual prediction can be viewed as an intermediate reasoning process that simulates future scene evolution and grounds action generation in predicted visual states. From this perspective, a fundamental question is how a WAM should structure its visual imagination to best inform action generation.

A key observation is that visual states along a manipulation trajectory contribute differently to action generation. Consider grasping an object: physical contact between the gripper and the object provides direct evidence of the interaction, while the preceding gripper closure and the subsequent lift provide the local context for how the grasp begins and proceeds. Meanwhile, the initial reach serves as a connective phase linking earlier task stages to the grasp itself. These phases serve distinct functions: The core interaction (`grasp`) evidence confirms whether the intended robot–object state transition has occurred; the surrounding motion (`closing` and `lifting`) provides the context of this change; and the connecting motion (`approaching`) determines how the robot transitions to it. However, the manipulation trajectory rarely reflects these functional task stages. Informative interaction events often span only a few frames, while connecting motions can take up the majority of the trajectory. Therefore, visual imagination based on fixed temporal intervals may risk fragmenting critical, fine-grained interactions across distinct prediction horizons.

Despite rapid progress in WAMs, current paradigms largely overlook this functional gap. Existing works predominantly focus on how visual predictions should be integrated with actions. Li et al.

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.



(a) Visual-Action Chunks Granularity (b) DOMINO success rate

Figure 1: **Interaction-dependent temporal granularity of visual reasoning.** (a) Our event-aligned visual action reasoning framework organizes visual-action targets around interaction events and transitions. (b) DOMINO success rates under a matched sequence budget of training visual-action slots.

(2026a) explicitly interleaves future visual prediction and action generation, (**?**) retains video modeling only during training, and (Zhang et al., 2026) replaces dense future videos with task-relevant visual transformations. However, as shown in Fig. 1a, these methods universally structure visual-action horizons uniformly across the demonstration timeline (denoted as *Episode Uniform*). They largely inherit the sampled trajectory structure from training data, without adapting visual imagination to different interaction events.

To address the above limitations, we propose an event-aligned visual action reasoning framework that organizes visual imagination around task-critical interactions, rather than directly following the sampled trajectory structure. The key idea is to shape the temporal progression of imagined states according to the underlying interaction dynamics. We construct event-aligned visual-action supervision by first segmenting training trajectories according to interaction events and then building visual-action prediction chunks from the resulting event boundaries. Within this event-aligned organization, we reorganize training trajectories such that task-critical events preserve fine-grained visual-action evolution, while transitions toward the relevant states are represented with fewer intermediate states, as shown in Fig. 1. Through this supervision, the model learns to advance its visual imagination in alignment with interaction progress, generating visual context that captures how critical interactions unfold under actions and supports action generation.

However, event-aligned visual action reasoning also introduces a practical challenge for action execution. Since interaction segments may span different durations, their corresponding action sequences can have different valid lengths. When mapped to fixed-size action chunks following Li et al. (2026a), short segments use repeated or padded actions to fill the remaining prediction slots during training. Executing these padding-induced actions can unnecessarily prolong the segment before the next prediction. To address this issue, we further introduce an execution validity head that identifies the valid portion of each predicted action sequence. At inference time, the validity head determines which portion to execute, avoiding redundant actions while retaining actual actions associated with interactions.

Experiments on DOMINO (Fang et al., 2026) and RoboTwin 2.0 (Chen et al., 2025) demonstrate improved manipulation performance, including a 10.26% gain in DOMINO success rate over LingBot-VA (Li et al., 2026a) and competitive performance on RoboTwin 2.0. The framework also demonstrates successful transfer from DOMINO Level 1 to Level 2/3 without target-level adaptation. Our contributions are summarized as below,

- We proposed an event-aligned visual action reasoning framework that organizes future prediction with task-critical interactions as focus to reason more carefully over crucial state changes. Instead of inheriting a predefined temporal structure from sampled trajectories, our formulation enables the reasoning process to adapt its granularity to the interaction dynamics.
- We further introduce an execution validity head to identify the valid portion of each predicted action sequence under event-aligned reasoning. By avoiding the execution of redundant actions, it improves action continuity and leads to smoother action execution.
- Our comprehensive experiments demonstrate that our method achieves superior performance, especially on the more challenging dynamic manipulation dataset DOMINO, with significantly better success rate and outstanding transfer performance.

2 Related Works

Vision-Language-Action Models. VLA models transfer knowledge from pretrained VLMs to robot control by generating actions conditioned on vision observations and language instructions. OpenVLA (Kim et al., 2025) formulates control as action-token prediction, while π_0 (Black et al., 2026) and GR00T N1 (Bjorck et al., 2025) combine pretrained VLM representations with generative action models. Recent works further improve generalization across tasks and embodiments through heterogeneous data and scalable adaptation, as in $\pi_{0.5}$ (Black et al., 2025), Gemini Robotics (Team, 2025), and X-VLA (Zheng et al., 2025). Other works investigate how pretrained representations should be adapted for action generation, including explicit spatial modeling Qu et al. (2025), lightweight feature adaptation Wang et al. (2026), representation insulation Driess et al. (2025), and persistent object-centric representations Ren et al. (2026). Despite their strong generalization, these approaches primarily map current observations and instructions directly to actions. Recent methods such as MEM (Torne et al., 2026) and LingBot-VLA 2.0 (Wu et al., 2026) begin to move beyond this formulation by incorporating temporal memory and future prediction, respectively.

World-Action Models. Beyond directly mapping observations to actions, recent WAMs leverage video generative models to jointly model future visual dynamics and robot actions. Ye et al. (2026); Li et al. (2026a); Aditi et al. (2026); Xu et al. (2026) couple visual prediction with action generation, using predicted future observations as an intermediate representation for reasoning. These approaches demonstrate that modeling the evolution of the visual world can provide richer structure than direct action prediction. Recent works further explore whether dense video prediction is necessary. Fast-WAM (**?**) reduces explicit visual generation at inference, while ImageWAM (Zhang et al., 2026) focuses on single target-image prediction instead of video rollout, suggesting that uniformly predicting every intermediate frame introduces substantial redundancy. In parallel, approaches explore temporal representation in world modeling. Li et al. (2026b) models trajectories around action-relevant events, while He et al. (2026); Yang et al. (2025) use sparse keyframes or subgoals to represent long-horizon dynamics. These methods highlight the importance of selecting task-relevant temporal information rather than modeling all future states uniformly.

3 Interaction-Structured Visual Reasoning for WAM

We propose an event-aligned visual action reasoning framework that organizes visual prediction according to the reasoning demands of robot manipulation. Rather than treating visual prediction as a uniform continuation of the sampled trajectory, we structure visual reasoning around task-relevant interactions and the visual context needed to support their corresponding actions. Our framework consists of event-aligned visual-action reasoning and execution validity prediction.

3.1 World-Action Models

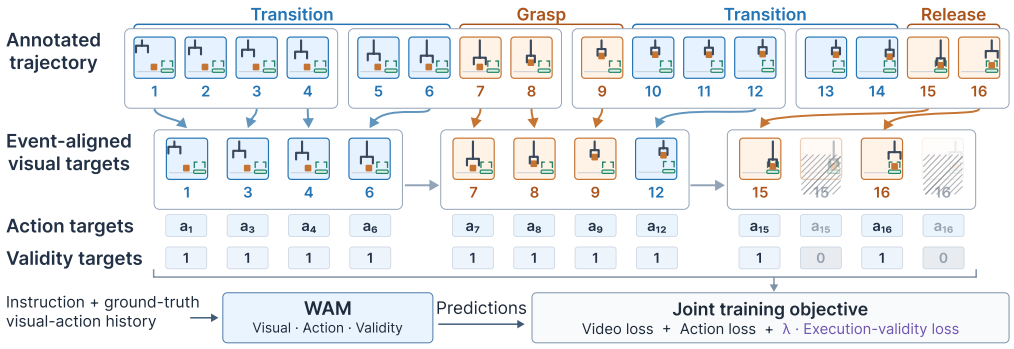
Problem Formulation. We build on autoregressive latent-diffusion World-Action Models (WAMs) (Li et al., 2026a), which represents the episode as a sequence of visual-action chunks $\{(\mathcal{V}_n, \mathcal{A}_n)\}_{n=1}^{N}$. Visual observations are encoded by a causal video VAE, where each latent visual frame is temporally aligned with a group of τ actions, following Li et al. (2026a) sampling scheme.

Given the initial visual context $\mathcal{V}_0$ encoded from o_0 and the instruction l, the model is trained to autoregressively generate the interleaved sequence $\mathcal{V}_1, \mathcal{A}_1, \ldots, \mathcal{V}_N, \mathcal{A}_N$, producing each visual or action chunk through iterative denoising conditioned on its preceding context in the sequence. It is optimized with teacher forcing, conditioning each prediction on ground-truth visual-action history.

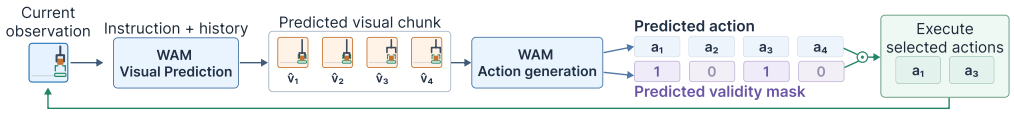
At inference time, the model first predicts a visual chunk and then generates its associated action chunk conditioned on the imagined scene evolution. Therefore, *visual predictions act as a form of reasoning for action*. The imagined scene provides the context where the action is derived. As execution proceeds, the latest observations will replace their corresponding predicted visual states in the context history, grounding subsequent generation in the observed environment.

Trajectory-Based Visual Reasoning. Under the standard WAM sampling scheme, visual-action prediction inherits the temporal organization of the demonstration episode. For a chunk containing K latent visual states,

$$\mathcal{V}_n = (z_{n,1}, \ldots, z_{n,K}), \qquad \mathcal{A}_n = (\mathbf{a}_{n,1_\tau}, \ldots, \mathbf{a}_{n,K_\tau}), \tag{1}$$



(a) **Event-aligned visual-action chunk construction and joint training.**



(b) **Visual-guided action generation with execution validity prediction.**

Figure 2: **Interaction-Structured Visual Reasoning framework.** Event-aligned chunks allocate visual reasoning according to interaction structure, while execution validity prediction identifies which generated actions to execute.

where $z_{n,k}$ denotes the k-th latent visual state and $\mathbf{a}_{n,k_\tau}$ its temporally aligned group of τ actions. These visual-action pairs are determined by the sampled trajectory, and the chunks with fixed size are unaware of interaction boundaries. Consequently, a task-relevant interaction may be split into neighboring chunks, or a chunk may contain multiple scattered events. This information presented to the action model may lead to potential performance loss. We therefore seek a structure that organizes visual reasoning according to task-relevant interactions rather than trajectory position alone.

3.2 Event-Aligned Visual Reasoning

A manipulation task progresses through *interactions* (such as grasping, releasing, or pressing), connected by *transitions* (such as reaching and carrying), with different demands on visual prediction. Interactions contain the state changes that decide task success and call for detailed reasoning, while transitions follow regular observation–action dynamics. Inspired by human perception and motor control, which adapt processing to the demands of the current task state (Todorov & Jordan, 2002; Zacks et al., 2007), we organize visual reasoning structured by interaction. We first define it as a taxonomy over the events of a demonstration, and then build visual action chunks from it.

Interaction Taxonomy. A demonstration is represented as a sequence of contiguous events, where each event corresponds to a recognizable unit of behavior with a clear beginning and end. Formally, events $E_0, \ldots, E_M$ partition the episode, where the i-th event covers frames s_i through e_i:

$$E_i = \{(o_t, a_t)\}_{t=s_i}^{e_i}, \quad e_M = T. \tag{2}$$

Each event has a type $c_i \in \{\texttt{interaction}, \texttt{transition}\}$. An *interaction event* changes a robot–object or robot–environment relationship that matters for task success. A *transition event* moves the robot between interactions and leaves these relationships unchanged. Two interaction events may be adjacent with no transition in between them.

Within an interaction event, we further distinguish two phases as shown in Fig. A1. The *evidence phase* is the part in which the relationship change becomes visually observable, providing direct visual evidence of task progress. The *support phase* is the remaining motion within the same event that occurs before, after, or on both sides of the evidence phase. The first and last frame of the evidence phase are its *evidence boundaries* s_i^{ev} and e_i^{ev}, with $s_i \le s_i^{\text{ev}} < e_i^{\text{ev}} \le e_i$. The support phase is the complement $[s_i, s_i^{\text{ev}}) \bigcup (e_i^{\text{ev}}, e_i]$. Transition events contain no task-progress evidence and are

Algorithm 1 Event-aligned visual-action chunk construction

Require: Events $\{(E_i, c_i)\}_{i=0}^{M}$ with evidence boundaries of interaction events; chunk length K
Ensure: Sequence of visual-action chunks $\mathcal{C}$

$\mathcal{C} \leftarrow [\,]$
for $i = 0, \ldots, M$ **do**
  if $c_i =$ `transition` **then**
    $\mathcal{C}.\mathrm{append}(\mathcal{S}_K(E_i))$ ▷ One chunk per transition, endpoints not anchored
  else
    $P, V, Q \leftarrow$ leading support, evidence, trailing support of E_i
    $F \leftarrow \lceil |V|/K \rceil \cdot K - |V|$ ▷ Slots the evidence leaves free
    if $F = 0$ **then** ▷ Evidence fills whole chunks; Support phases standalone
      if $P \neq \emptyset$ **then**
        $\mathcal{C}.\mathrm{append}(\bar{\mathcal{S}}_K(P))$
      end if
      $\mathcal{C}.\mathrm{extend}(\mathrm{Split}_K(V))$
      if $Q \neq \emptyset$ **then**
        $\mathcal{C}.\mathrm{append}(\bar{\mathcal{S}}_K(Q))$
      end if
    else ▷ Support fills the free slots around the evidence
      $(n_P, n_Q) \leftarrow \mathrm{Allocate}(F, P, Q)$ ▷ ≥ 1 slot per non-empty side, rest split evenly
      $\mathcal{C}.\mathrm{extend}\big(\mathrm{Split}_K(\bar{\mathcal{S}}_{n_P}(P) \oplus V \oplus \bar{\mathcal{S}}_{n_Q}(Q))\big)$
    end if
  end if
end for
return $\mathcal{C}$

therefore not further divided. [1] Event and evidence boundaries come from the event annotation of the source demonstration and are fixed offline as shown in Sec. D.

Event-Aligned Interaction Structure. We now build the visual-action chunk of Sec. 3.1 structured from events instead of fixed temporal positions. Three rules connect the taxonomy to the chunks.

RULE 1. *Event Boundaries*: A chunk never crosses an event boundary. Each transition event is resampled into exactly one chunk, while an interaction event produces one or more chunks.

RULE 2. *Phase-dependent density*: Within an interaction event, the evidence phase keeps its native temporal resolution and is never subsampled. The support phase is resampled into slots that the evidence leaves free in its chunks. If the evidence leaves no slot, the support before or after it forms a chunk of its own.

RULE 3. *Interaction Boundaries*: The first and last frame of an interaction event are always kept. Where a support phase lies on the side, its resampling anchors the boundary frame.

Therefore, both levels of the taxonomy enter the construction. Event boundaries decide where a chunk may start and end, and phases decide which frames inside an interaction are kept densely. Support can be compressed while keeping its boundaries, indicating how the event begins and ends around the evidence. Sec. 4.3 shows that reducing it hurts even when evidence itself is protected. Algorithm 1 shows the construction algorithm. It uses three operators: $\mathcal{S}_r(\cdot)$ resamples a segment to r frames, $\bar{\mathcal{S}}_r(\cdot)$ does the same while keeping the first and last frame of the segment, and $\mathrm{Split}_K(\cdot)$ cuts a sequence into consecutive chunks of K frames. Each action chunk consists of the actions paired with the selected frames. If a segment is shorter than its allocated slot count, frames are repeated as needed. These repeated frames carry no new observation and are exactly the positions that the execution validity head of Sec. 3.3 learns to skip. More details are discuss in Sec. A.

Aligning chunk boundaries with events also matters at inference time. Because a chunk never mixes two events, an interaction that does not complete as expected is revisited by the next prediction rather than averaged into a chunk that already contains the following motion, so prediction drift is corrected where it arises instead of propagating across interactions.

[1] In a grasp, the evidence phase is the gripper making contact with the object, the support phase is the gripper starting to close before contact and lifting off after it, and the reach before the grasp is a transition event.

3.3 Execution Validity Prediction

Event-aligned chunks have a fixed length K, but the segments they encode do not. A segment shorter than its slots is padded by repeating frames before VAE encoding (Sec. 3.2), and each repeated frame carries a copy of the action group of the frame it repeats. At inference, the event boundaries are unknown, so the model can not know how many of the K slots of a chunk carry new observations. Executing the copies would make the robot pause at the same target, which we call a *stall*. Therefore, for every generated action group, we let the model predict whether it should be executed.

Joint Action and Validity Prediction. Let h_n denote the context available before predicting chunk n, including the instruction l, initial context $\mathcal{V}_0$, and all previous visual-action chunks. Conditioned on predicted visual chunk $\widehat{\mathcal{V}}_n$ and h_n, the model generates action chunk $\widehat{\mathcal{A}}_n = (\widehat{a}_{n,1}, \ldots, \widehat{a}_{n,K_\tau})$ together with a validity probabilities $p_n = (p_{n,1}, \ldots, p_{n,K})$,

$$\big(\widehat{\mathcal{A}}_n, \widehat{\mathbf{p}}_n\big) = f_\theta\big(\widehat{\mathcal{V}}_n, h_n\big), \tag{3}$$

where $\widehat{\mathbf{p}}_n = (\widehat{p}_{n,1}, \ldots, \widehat{p}_{n,K_\tau})$ denotes the probabilities that each action should be executed. The probabilities are predicted by a projection layer g_ϕ applied to the action-stream features of each of the K positions after the final transformer block, which outputs a logit ℓ_{n_k} with $p_{n,k} = \sigma(\ell_{n,k})$. The head is trained jointly with the backbone. The ground-truth mask $\mathbf{m}_n$ comes from chunk construction, where $m_{n,k} = 1$ if the k-th frame of action is the first occurrence of an observation frame, and $m_{n,k} = 0$ if it is a repeated copy. Validity is defined per frame and shared by the τ actions aligned with it. The head is supervised with cross-entropy over all chunks and positions, $\mathcal{L}_{\text{val}} = -\frac{1}{K\tau}\sum_{j=1}^{K\tau}[m_{n,j}\log p_{n,j} + (1 - m_{n,j})\log(1 - p_{n,j})]$.

Stall-Free Execution. At inference time, we set $\widehat{m}_{n,j} = \mathbf{1}[\sigma(\ell_{n,j}) > 0.5]$ and execute only the valid action groups, in their original order,

$$\widehat{\mathcal{A}}_n^{\text{exec}} = \big(\widehat{a}_{n,j}\big)_{1 \le j \le K,\ \widehat{m}_{n,j}=1}. \tag{4}$$

This mechanism allows the executed action sequence to adapt to the interaction represented by each chunk while the model keeps a fixed-length output structure.

3.4 Training Objective

Following Li et al. (2026a), we train the visual and action streams on the event-aligned chunks using flow-matching losses $\mathcal{L}_{\text{dyn}}$ and $\mathcal{L}_{\text{inv}}$ for visual and action prediction, respectively. $\mathcal{L}_{\text{dyn}}$ is applied to all frames, including repeated frames, so the visual stream learns to reproduce the padded chunk structure. $\mathcal{L}_{\text{inv}}$ is restricted and averaged only over valid positions, and invalid action loss is zeroed out. The overall objective is $\mathcal{L} = \mathcal{L}_{\text{dyn}} + \mathcal{L}_{\text{inv}} + 0.1\mathcal{L}_{\text{val}}$.

4 Experiments

4.1 Experimental setup

Benchmarks and baselines. We evaluate on DOMINO (Fang et al., 2026) and RoboTwin 2.0 (Chen et al., 2025). DOMINO contains 35 dynamic-manipulation tasks. We use clean Level 1 scenes with the Aloha-AgileX embodiment and a motion coefficient of 0.1, evaluating 100 episodes per task. RoboTwin 2.0 contains 50 bimanual-manipulation tasks, which we evaluate separately in clean and randomized environments. We compare against LingBot-VA (Li et al., 2026a), Fast-WAM (**?**), and ImageWAM (Zhang et al., 2026). We retain each baseline's original architecture and preprocessing pipeline and train all baselines for a comparable number of epochs. We additionally evaluate our framework on real-world manipulation tasks, with the experimental setup and results provided in Sec. C.1.

Implementation and metrics. Our model is initialized from pretrained LingBot-VA (Li et al., 2026a). We report Success Rate (SR) and Manipulation Score (MS) on DOMINO, and SR under clean and random settings on RoboTwin 2.0. MS measures terminal spatial progress toward the target with penalties for workspace violations and clutter collisions. We implement the execution head as a zero-initialized token-wise linear projection, with an execution-loss weight of $\lambda_{\text{exec}} = 0.1$ and an inference threshold of $\tau_{\text{exec}} = 0.5$. Dataset preparation, model-specific hyperparameters, and evaluation details are provided in Sec. B.

Table 1: **Evaluation on DOMINO and RoboTwin 2.0.** For LingBot-VA on RoboTwin, the upper row reports published results, while the lower row (†) reports our re-evaluation of the released checkpoint (3 steps for video tokens to $s = 0.6$, 10 steps for action tokens to $s = 1.0$).

Methods	DOMINO		RoboTwin 2.0		
	SR	MS	Clean SR	Rand. SR	Avg. SR
π_0 (Black et al., 2026)	8.17	23.96	65.92	58.40	62.16
$\pi_{0.5}$ (Black et al., 2025)	9.63	26.17	82.74	76.76	79.75
PUMA (Fang et al., 2026)	17.20	34.97	–	–	–
DynamicWAM[1] (Lou et al., 2026)	38.20	53.20	–	–	–
ABot-M0 (Yang et al., 2026)	–	–	81.20	80.40	80.80
Motus (Bi et al., 2026)	–	–	88.66	87.02	87.80
ImageWAM (Zhang et al., 2026)	18.86	35.47	**93.20**	**93.56**	**93.38**
Fast-WAM (**?**)	19.09	35.33	91.88	91.78	91.83
LingBot-VA† (Li et al., 2026a)	32.57	45.55	92.93 92.18†	91.55 90.24†	92.24 91.21†
Ours	**42.83**	**55.92**	93.02	91.58	92.30

Table 2: Cross-level transfer on the 10-task subset of DOMINO.

Method	L2 transfer		L3 transfer	
	SR	MS	SR	MS
PUMA	10.5	30.28	4.6	20.16
ImageWAM	13.3	30.44	2.8	15.09
Fast-WAM	13.5	31.56	2.4	15.34
LingBot-VA	16.9	30.37	6.8	19.36
Ours	**29.4**	**45.42**	**9.7**	**24.23**

Table 3: **Execution behavior on DOMINO.** N_{exec} and N_{roll} are the average number of action commands issued and visual rollouts requested per episode, measured only on episodes where both LingBot-VA and our model succeed.

Method	SR↑	$N_{\text{exec}}\downarrow$	$N_{\text{roll}}\downarrow$
LingBot-VA	32.57	167.18	6.22
Ours	42.83	106.46	4.49

4.2 Benchmark Performance

DOMINO. As shown in Tab. 1, on DOMINO, our method achieves 42.83% SR and 55.92 MS, outperforming the LingBot-VA backbone (32.57% SR and 45.55 MS) with non-marginal improvements of 10.26 in SR and 10.37 in MS. Note that although Fast-WAM and ImageWAM can achieve above 90% SR on RoboTwin 2.0, their SR on DOMINO are below 20%, highlighting the challenges of dynamic manipulation on DOMINO. Our method can significantly improve performance on dynamic manipulation, demonstrating the effectiveness of event-aligned visual reasoning. Complete task-level results are provided in Sec. C.2.

RoboTwin 2.0. As shown in Tab. 1, on RoboTwin 2.0, our method obtains 93.02% SR in clean scenes and 91.58% in randomized scenes, with an average of 92.30%, outperforming the corresponding LingBot-VA backbone. Compared with ImageWAM and Fast-WAM, our method leads to competitive performance, while performing much better than $\pi_{0.5}$ and ABot-M0.

Transfer across DOMINO levels. We further evaluate the transfer performance where the model is trained on the Level 1 data of DOMINO and evaluated on the ten-task Level 2/3 subset without target-level additional adaptation. As shown in Tab. 2, our method achieves the best SR and MS performance with non-marginal improvements, such as our 45.42 MS *vs*. 31.56 MS from Fast-WAM on L2, and our 24.23 MS *vs*. 20.16 MS from PUMA on L3. Note that our zero-shot model performs much better than PUMA, which finetunes the L1 model with LoRA for L2/L3, demonstrating our outstanding generalization and robustness performance. The detailed results are shown in Sec. C.2.

4.3 Analysis of event-aligned visual action reasoning

We examine how the temporal organization of supervision affects event-aligned visual action reasoning, considering *visual reasoning granularity*, the *placement of interaction evidence*, and the *sur-*

[1]DynamicWAM is trained with 150 clean and 150 randomized demonstrations per task (10,500 in total).

Table 4: **Ablations of temporal sampling scheme and execution validity prediction on DOMINO.** Action slots count all sampled positions, including repetitions, across all retained demonstrations. All variants except the backbone reference and the no-head ablation use the execution validity head during training and inference. Full sampling statistics are reported in Tab. C4.

Sampling	Granularity	Placement	Slots	SR↑	MS↑
LingBot-VA (reference)	Episode	Uniform	291,296	32.57	45.55
Episode Uniform	Episode	Uniform	211,664	29.17	41.32
Event Uniform	Event	Uniform	211,664	39.60	52.26
Reduced Evidence	Event	Transition-Protected	268,592	36.17	48.36
Reduced Context	Event	Evidence-protected	173,216	37.83	52.99
Event-aligned (Ours)	Event	Evidence-protected	211,664	**42.83**	**55.92**
Event-aligned w/o execution validity head			211,664	39.11	52.33
Event-aligned w/ execution validity head			211,664	**42.83**	**55.92**

rounding context. Our Event-aligned sampling serves as the reference scheme: it organizes demonstrations into event-aligned granularity, explicitly preserves interaction evidence, and retains slots from the surrounding non-evidence context. The sampling variants in Tab. 4 use the same 1,693 retained DOMINO demonstrations and share the same training and execution head configuration. We additionally assess the execution validity head through a separate comparison of models trained with and without the head. LingBot-VA without the head is included as a backbone reference. The composition of sampled sequences relative to evidence phases is detailed in Sec. C.3.

Event alignment at a matched sequence budget. Episode Uniform samples uniformly across each demonstration's initial-to-final range, preserving the endpoints and total action-slot budget of Event-aligned. Event Uniform instead preserves its event boundaries and per-event budgets, while sampling uniformly within each event without explicitly reserving interaction evidence. Event Uniform reaches 39.60% SR, much higher than 29.17% from Episode Uniform, under exactly the same slot budget. It demonstrates that event-aligned granularity and their associated budget allocation provide a more effective temporal organization than episode-wide uniform sampling.

Interaction evidence placement. Explicitly preserving interaction evidence further improves SR from 39.60% with Event Uniform to 42.83% with Event-aligned, while maintaining the same event boundaries and per-event budgets. To further examine the role of evidence placement, Reduced Evidence retains the event boundaries but shifts sampling toward transition event and support phase, away from interaction evidence. Despite using 268,592 action slots (26.9% more than Event-aligned), it achieves only 36.17% SR, demonstrating the importance of preserving fine-grained interaction evidence.

Context surrounding protected interaction evidence. Reduced Context preserves the same protected interaction evidence as Event-aligned while decreasing the sampling budget for surrounding non-evidence context, resulting in less action slots. It reaches 37.83% SR, lower than Event-aligned, indicating the effectiveness of additional context around protected interaction evidence. Our framework benefits from both evidence and surrounding support.

Execution validity prediction. We assess the execution validity head under Event-aligned sampling by comparing separately trained models with and without the head. The model without the head directly executes the complete predicted action sequence. As shown in Tab. 4, the model with the head improves SR from 39.11% to 42.83%, showing the effectiveness of distinguishing action validity with the proposed execution validity head.

Visualization of event-aligned visual reasoning. Fig. 3 compares LingBot-VA and our method on a *Scan Object* episode, exhibiting video attention to the previous observation (Fig. 3a), and action attention to the visual prediction (Fig. 3c). Both methods start from the same initial observation, and later chunks follow each policy's own rollout. At chunk 0, our method's video attention already focuses on the moving object, while LingBot-VA's is largely diffuse. Our predicted visual futures in Fig. 3c depict an ordered interaction sequence: approaching and grasping the moving object (chunks 0–1), approaching and grasping the scanner (chunks 2–3), and reaching a scanning

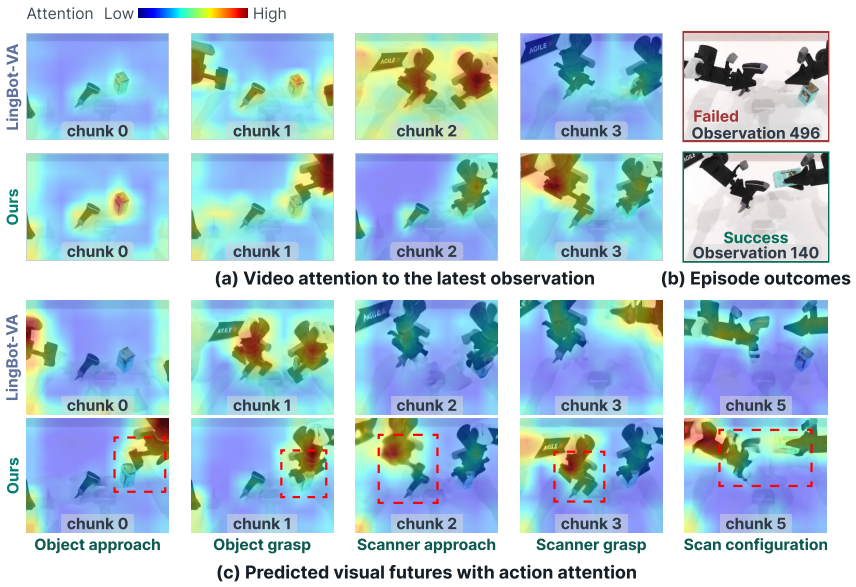


Figure 3: **Visual predictions and attention on a *Scan Object* demonstration episode.** (a) Video attention to the latest observation. (b) Episode outcomes. (c) The last predicted frame of each displayed chunk, overlaid with action attention. Red boxes highlight interaction regions. Both methods start from the same initial observation. Later chunks follow each policy's own rollout, so frames in the same column can differ.

configuration (chunk 5). Action attention shifts toward the corresponding interaction regions, highlighted by the red boxes. In contrast, LingBot-VA splits its attention across both arms and attempts to grasp both objects simultaneously, missing the moving target. Our method completes the task at observation 140, while LingBot-VA remains unsuccessful at observation 496 (Fig. 3b). These observations are consistent with our finding that task-critical events demand detailed reasoning and suggest that organizing visual prediction around interaction events benefits task completion. Additional visualization cases are given in Sec. C.5.

4.4 Execution Behavior

We examine the number of action commands and visual-rollout requests under our policy and the original LingBot-VA, on identical episodes where both can succeed. As demonstrated in Tab. 3, across 738 identical successful episodes spanning all 35 tasks, the average issued command count decreases from 167.18 to 106.46, while the average visual-rollout request count drops from 6.22 to 4.49. These results together show our method is not only more effective with higher SR, but also more efficient with less commands and visual predictions. Task-level breakdowns and the scope of this comparison are given in Sec. C.4.

5 Conclusion

We presented an event-aligned visual action reasoning framework that structures visual imagination around task-relevant interactions and their corresponding reasoning demands. Through event-aligned visual-action supervision, our model learns to generate visual context for action generation with detailed state evolution around interaction events and coarser progression through connecting transitions, adapting visual reasoning granularity to the underlying interaction dynamics. An execution validity head identifies the valid portion of predicted action sequences to avoid redundant commands. Experiments demonstrate state-of-the-art success rate on DOMINO and competitive performance on RoboTwin. On shared successful episodes, it also reduces the average number of issued action commands and imagination requests compared with the baseline. These findings highlight interaction structure as an effective organizing principle for visual imagination in WAMs.

AI USE STATEMENT

In this work, we used generative AI tools for *cleaning and reformatting dataset* (as shown in Sec. D).

We have not used generative AI tools for proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, supporting qualitative and thematic data analysis and interpreting results, and generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs are not applicable to this work.

Additionally, we used generative AI tools for *edit a research paper to improve readability*.

We have reviewed all AI-assisted work. We have checked and filtered the LLM-suggested event annotation specifications to retain only those critical to each specific robotic task. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

N/A

REPRODUCIBILITY STATEMENT

We detail the event annotation procedure for the DOMINO and RoboTwin training datasets in Sec. D and the experimental setup and evaluation protocols in Sec. B. To facilitate reproducibility, we will publicly release the event-annotated datasets, our code, and all relevant model checkpoints, including the baselines we fine-tuned on DOMINO and our models on both DOMINO and RoboTwin.

REFERENCES

Aditi, Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson G. Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, Aarti Basant, Mukesh Beladiya, Mohammad Qazim Bhat, Zaid Pervaiz Bhat, Dan Blick, Vanni Brighella, Han Cai, Tiffany Cai, Eric Cameracci, Jiaxin Cao, Yulong Cao, Mark Carlson, Carlos Casanova, Ting-Yun Chang, Yan Chang, Yu-Wei Chao, Prithvijit Chattopadhyay, Roshan Chaudhari, Chieh-Yun Chen, Junyu Chen, Ke Chen, Qizhi Chen, Wenkai Chen, Xiaotong Chen, Yu Chen, An-Chieh Cheng, Click Cheng, Xiu Chia, Jeana Choi, Chaeyeon Chung, Wenyan Cong, Yin Cui, Magdalena Dadela, Nalin Dadhich, Wenliang Dai, Joyjit Daw, Alperen Degirmenci, Rodrigo Vieira Del Monte, Robert Denomme, Sameer Dharur, Marco Di Lucca, Ke Ding, Wenhao Ding, Yifan Ding, Yuzhu Dong, Nicole Drumheller, Yilun Du, Aigul Dzhumamuratova, Aleksandr Efitorov, Hamid Eghbalzadeh, Naomi Eigbe, Imad El Hanafi, Hassan Eslami, Benedikt Falk, Jiaojiao Fan, Jim Fan, Amol Fasale, Sergiy Fefilatyev, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Vikram Fugro, Prashant Gaikwad, TJ Galda, Katelyn Gao, Yihuai Gao, Wenhang Ge, Sreyan Ghosh, Arushi Goel, Vivek Goel, Akash Gokul, Rama Govindaraju, Jinwei Gu, Miguel Guerrero, Elfie Guo, Aryaman Gupta, Siddharth Gururani, Hugo Hadfield, Song Han, Ankur Handa, Zekun Hao, Mohammad Harrim, Ali Hassani, Nathan Hayes-Roth, Yufan He, Chris Helvig, and Cyrus Hogg. Cosmos 3: Omnimodal world models for physical AI. *CoRR*, abs/2606.02800, 2026. doi: 10.48550/ARXIV.2606.02800. URL https://doi.org/10.48550/arXiv.2606.02800.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith LLontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: an open foundation model for generalist humanoid robots. *CoRR*, abs/2503.14734, 2025. doi: 10.48550/ARXIV.2503.14734. URL https://doi.org/10.48550/arXiv.2503.14734.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

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*, 2025.

Danny Driess, Jost Springenberg, Brian Ichter, LILI YU, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowl-

edge insulating vision-language-action models: Train fast, run fast, generalize better. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 102867–102888. Curran Associates, Inc., 2025. doi: 10.52202/085713-3439. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/94e936034d12bcd04834ec2773f02aff-Paper-Conference.pdf.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. In *European Conference on Computer Vision (ECCV)*, 2026.

Ziheng He, Yixiang Chen, Ning Yang, Zhanqian Wu, Qisen Ma, Yuan Xu, Jiabing Yang, Peiyan Li, Xiangnan Wu, Xiaofeng Wang, Zheng Zhu, Jing Liu, Nianfeng Liu, and Yan Huang. Skip: Sparse keyframe interpolation paradigm for efficient embodied world models, 2026. URL https://arxiv.org/abs/2606.00664.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

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

Shalfun Li, Victor Yao, Charles Yang, Truth Qu, Regis Cheng, Ryan Yu, Howard Lu, Newton Von, Vincent Chen, Yohann Tang, Maeve Zhang, Ellie Ma, Gody Li, Starrick Liu, Sage Yang, Lorien Shu, J. W. Gao, Ethan Chen, Colin Ye, Yu Sun, Elise Mon, PS Zhang, Neo Li, Lily Li, James Wang, Ping Yang, Chris Pan, Lucy Liang, Hang Su, Roy Gan, Hao Wang, and Qian Wang. Wallwm: Carving world action modeling at the event joints, 2026b. URL https://arxiv.org/abs/2606.01955.

Yunfan Lou, Hewen Gao, Xiyu Zhu, Zhuoran Qiao, Xuan Han, Yifan Yang, Yifan Ye, Boxian Yao, and Zhibo Pang. Dynamicwam: Dual-path motion conditioning for world-action models in dynamic manipulation, 2026. URL https://arxiv.org/abs/2608.00793.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model, 2025. URL https://arxiv.org/abs/2501.15830.

Peng Ren, Haoyang Ge, Jiang Zhao, Cong Huang, Yukun Shi, Pei Chi, and Kai Chen. Closing the loop in humanoid vla: Persistent 3d object tokens for verifiable loco-manipulation, 2026. URL https://arxiv.org/abs/2607.18016.

Gemini Robotics Team. Gemini robotics: Bringing AI into the physical world. *CoRR*, abs/2503.20020, 2025. doi: 10.48550/ARXIV.2503.20020. URL https://doi.org/10.48550/arXiv.2503.20020.

Emanuel Todorov and Michael I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, November 2002. ISSN 1546-1726. doi: 10.1038/nn963. URL https://doi.org/10.1038/nn963.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models, 2026. URL https://arxiv.org/abs/2603.03596.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. Vla-adapter: an effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'26/IAAI'26/EAAI'26. AAAI Press, 2026. ISBN 978-1-57735-906-7. doi: 10.1609/aaai.v40i22.38931. URL https://doi.org/10.1609/aaai.v40i22.38931.

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

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

Liudi Yang, Yang Bai, George Eskandar, Fengyi Shen, Mohammad Altillawi, Dong Chen, Soumajit Majumder, Ziyuan Liu, Gitta Kutyniok, and Abhinav Valada. Roboenvision: A long-horizon video generation model for multi-task robot manipulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 21281–21288, 2025. doi: 10.1109/IROS60139.2025.11246352.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Jeffrey M. Zacks, Nicole K. Speer, Khena M. Swallow, Todd Samuel Braver, and Jeremy R. Reynolds. Event perception: a mind-brain perspective. *Psychological bulletin*, 133 2:273–93, 2007. URL https://api.semanticscholar.org/CorpusID:10494362.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. Imagewam: Do world action models really need video generation, or just image editing?, 2026. URL https://arxiv.org/abs/2606.19531.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

A Details of Event-Aligned Chunk Construction

This section completes Algorithm 1 with the definitions and cases omitted from Sec. 3.2. Throughout, a segment is a contiguous run of frames, each paired with its group of τ actions (Sec. 3.1), and K is the chunk length.

Taxonomy. Fig. A1 shows how we break a demonstration into parts, with a few examples at each level. At the top is the episode, which we split into a sequence of events. In an *interaction event*, the robot changes its relationship with an object or with the environment. A *transition event* covers the motion in between. We split each interaction event further into two phases. The *evidence phase* consists of the frames that show the change happening. The *support phase* is the motion leading into or out of the evidence.

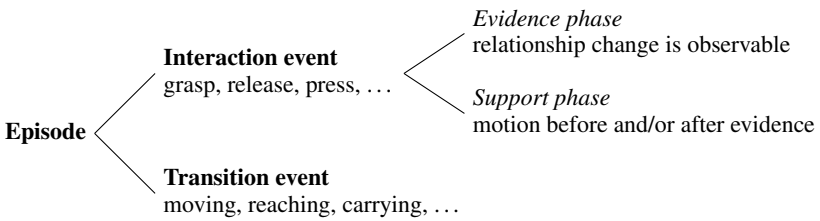


Figure A1: Interaction taxonomy of a demonstration. An episode is partitioned into interaction and transition events. Interaction events are further structured into evidence and support phases.

Operators. For a segment $X = (x_1, \ldots, x_{|X|})$ and a target length $r \geq 1$, the resampler $\mathcal{S}_r(X)$ returns r frames taken at uniformly spaced positions of X, so that frames repeat when $|X| < r$; $\mathcal{S}_0(X) = \emptyset$. The anchored resampler $\bar{\mathcal{S}}_r(X)$ keeps x_1 and $x_{|X|}$ and fills the remaining $r-2$ positions with $\mathcal{S}_{r-2}$ applied to the interior $(x_2, \ldots, x_{|X|-1})$; for $r = 1$ it keeps the frame at the event boundary, so that Rule 3 holds. $\mathrm{Split}_K(X)$ cuts X into consecutive chunks of K frames; if $|X|$ is not a multiple of K, the final chunk is filled to K frames by repetition. Repeated frames carry no new observation; they are the positions that the execution validity head (Sec. 3.3) marks as invalid.

Transition events. A transition event E_i becomes the single chunk $\mathcal{S}_K(E_i)$. Its first and last frame are not anchored.

Interaction events. Let P, V, and Q be the leading support, the evidence, and the trailing support of an interaction event, with $|V| \geq 1$ and either support possibly empty. The evidence keeps every frame and occupies $\lceil |V|/K \rceil$ chunks, leaving $F = \lceil |V|/K \rceil K - |V| \in \{0, \ldots, K-1\}$ free slots.

Case $F > 0$. The free slots are given to the support, $(n_P, n_Q) = \mathrm{Allocate}(F, P, Q)$ with $n_P + n_Q = F$: if only one support is non-empty, it receives all F slots; if both are non-empty, each receives one slot and the remaining $F-2$ are split evenly, $(n_P, n_Q) = (\lfloor F/2 \rfloor, F - \lfloor F/2 \rfloor)$ for $F \geq 2$; if neither exists, $(n_P, n_Q) = (0, 0)$ and Split_K fills the free slots by repetition. The sequence $\bar{\mathcal{S}}_{n_P}(P) \oplus V \oplus \bar{\mathcal{S}}_{n_Q}(Q)$ then has exactly $\lceil |V|/K \rceil K$ frames and is cut by Split_K.

Case $F = 0$. The evidence fills its chunks exactly. Each non-empty support becomes a standalone chunk, $\bar{\mathcal{S}}_K(P)$ placed before the evidence chunks and $\bar{\mathcal{S}}_K(Q)$ after them.

Worked example. Let $K = 4$ and consider a grasp with $|P| = 6$, $|V| = 10$, and $|Q| = 4$, preceded by a reach of 30 frames. The reach is a transition event and becomes one chunk of four frames sampled from its interior. The evidence needs $\lceil 10/4 \rceil = 3$ chunks and leaves $F = 2$ free slots, so $(n_P, n_Q) = (1, 1)$: the first frame of the grasp, the ten evidence frames, and the last frame of the grasp form a 12-frame sequence, cut into three chunks. If instead $|V| = 12$, then $F = 0$: P becomes the standalone chunk $\bar{\mathcal{S}}_4(P)$ (its first and last frame and two interior frames), the evidence fills three chunks, and Q, already four frames long, is kept as is. In both cases the first and last frame of the grasp appear in its chunks (Rule 3), and no chunk contains frames from both the reach and the grasp (Rule 1).

B EXPERIMENTAL DETAILS

B.1 BENCHMARKS AND TRAINING DATA

Implementation and metrics. Our model is initialized from pretrained LingBot-VA (Li et al., 2026a). Event-based preparation retains 1,693 of the 1,750 source demonstrations on DOMINO and 24,844 of the 27,500 source demonstrations on RoboTwin 2.0. The training schedules use 10k and 50k optimizer updates, respectively, with a learning rate of 10^{-5} and an effective batch size of 32. We report Success Rate (SR) and Manipulation Score (MS) on DOMINO, and SR under clean and random settings on RoboTwin 2.0. MS measures terminal spatial progress toward the target with penalties for workspace violations and clutter collisions. We implement the execution head as a zero-initialized token-wise linear projection, with an execution-loss weight of $\lambda_{\text{exec}} = 0.1$ and an inference threshold of $\tau_{\text{exec}} = 0.5$. All inference, following LingBot-VA, use Euler solver with 3 steps for video tokens (integrating to s = 0.6) and 10 steps for action tokens (integrating to s = 1.0). Execution cost is measured by the number of action vectors issued to the controller.

Benchmark settings. Tab. B1 summarizes the benchmark settings and our training schedules. DOMINO evaluation uses clean Level 1. RoboTwin 2.0 evaluation covers both clean and randomized environments.

Table B1: Benchmark settings and training schedules for our model.

Setting	**DOMINO**	**RoboTwin 2.0**
Tasks	35	50
Source demonstrations	1,750 clean Level 1	2,500 clean + 25,000 randomized
Retained demonstrations	1,693	2,465 clean + 22,379 randomized
Optimizer updates	10,000	50,000
Learning rate	10^{-5}	10^{-5}
Effective batch size	32	64
Evaluation condition	Clean Level 1	Clean / randomized
Evaluation episodes per task	100	100

Dataset preparation. All event-based models use the same retained demonstration dataset for each benchmark. DOMINO demonstrations come from the benchmark release of Fang et al. (2026): 50 clean Level 1 demonstrations for each of the 35 tasks, 1,750 in total. Simulator replay of the released trajectory fails for 57 of them, which are excluded, while the remaining 1,693 are retained. On RoboTwin 2.0, the 27,500 source demonstrations have two origins. The 25,000 randomized demonstrations (500 per task) and the 50 clean demonstrations of `put_bottles_dustbin` are the same source episodes as the released LingBot-VA post-training data (Li et al., 2026a). The released clean demonstrations of the other 49 tasks do not have source trajectories for simulator replay, so we collected 2,450 clean demonstrations (50 per task) ourselves with the RoboTwin 2.0 data generator (Chen et al., 2025). We use 24,844 demonstrations with successful event annotations (2,465 clean and 22,379 randomized). The remaining 2,656 source demonstrations are excluded from this pool because replay failures. Sec. D specifies the event annotation process.

B.2 MODEL IMPLEMENTATION

Initialization and optimization. Our model is initialized from the pretrained LingBot-VA base checkpoint. For DOMINO, we use bf16 training with 8 GPUs, a per-gpu batch size of 1, and 4 gradient-accumulation steps, giving an effective batch size of 32. The learning rate is 10^{-5}, the optimizer moment coefficients are $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay is 0.1, and the warm-up lasts 10 updates. The DOMINO schedule comprises 10,000 updates and the RoboTwin schedule uses 50,000 updates and 64 effective batch size with the same learning rate. We follow LingBot-VA to use three RGB views: one high-mounted camera and two wrist cameras, with configured image dimensions of 256×320 (height $\times$ width). We retain the backbone's 30-channel action representation and quantile-based normalization.

Execution head. The execution head operates on the normalized, timestep-modulated final action-token features shared with the action output projection. It applies a single linear layer from d features

Table B2: **DOMINO baseline training budgets.** The last column reports the estimated total number of non-padding training action target occurrences during training, in millions.

Method	Updates	Global batch	Sampling unit	Action targets (M)
LingBot-VA	10,000	32	Episode segment	≈ 55.21
Fast-WAM	15,000	128	32-action window	≈ 56.19
ImageWAM	10,000	384	16-action window	≈ 58.90

to one execution logit, with parameters shared across all action slots. For $d = 3072$, the head introduces only $d + 1 = 3{,}073$ parameters and no additional Transformer blocks.

Temporal grouping. During training, the number of latent temporal groups per chunk is randomized over $C \in \{1, 2, 3, 4\}$. During inference, we follow LingBot-VA to use $C = 2$ for fair comparison. Each future latent temporal group corresponds to 16 action slots. The first inference request has a nominal capacity of 16 future action slots, and subsequent requests have 32.

Noise augmentation and inference. Following the training setup described in the main experiment, we apply noise augmentation with probability $p = 0.5$ and $s_{\text{aug}} \sim \mathcal{U}[0.5, 1.0]$. The video and action SNR shifts are 5.0 and 1.0, respectively. For evaluation, we use Euler solver with 3 steps for video tokens (integrating to s = 0.6) and 10 steps for action tokens (integrating to s = 1.0), following the paper statement of LingBot-VA. The classifier-free guidance scales are 5.0 for video and 1.0 for actions.

B.3 Baselines training configurations

Training configurations. Tab. B2 reports the DOMINO training budgets of the WAM baselines. ImageWAM and Fast-WAM both use a learning rate of 10^{-4} with a cosine schedule, weight decay of 0.01, a maximum gradient norm of 1.0, bf16 mixed precision, and random seed 42. ImageWAM uses a per-worker batch size of 12 with four accumulation steps; Fast-WAM uses a per-worker batch size of 16 without gradient accumulation. With eight workers, their effective batch sizes are 384 and 128, respectively.

ImageWAM. We follow the official ImageWAM setting with the FLUX.2-klein-base-4B backbone. Training uses 16-action windows with endpoint-frame visual supervision. The three camera views are packed into a 288×256 input, and action/state vectors are 14-dimensional with z-score normalization. The visual and action loss weights are 0.5 and 1.0, respectively. Both flow schedulers use a shift of 5.0 during training and inference.

Fast-WAM. We follow the official Fast-WAM setting with the Wan2.2-TI2V-5B backbone. Training uses 32-action windows with an action–video sampling ratio of $4{:}1$. The three camera views are packed into a 384×320 input, and action/state vectors are 14-dimensional with z-score normalization. The video and action flow schedulers use shifts of 5.0 and 1.0, respectively, during both training and inference.

C Additional Results

This appendix reports real-world experiments, task-level results, cost breakdowns, and qualitative examples.

C.1 Real World Experiments

We conduct simulation to real-world experiments on a Unitree G1 humanoid robot equipped with Unitree Dex1-1 grippers. We consider three complex manipulation tasks requiring bimanual coordination: (1) *Tennis Ball Insertion*, where the robot holds a tube with one arm, inserts a tennis ball with the other, and places the tube upright on a coaster; (2) *Bimanual Cup Placement*, where the robot sequentially places blue and green cups on their corresponding coasters; and (3) *Drawer Manipulation*, where the robot opens a drawer, places a cube inside, and closes the drawer. The policy

Table C1: SR(%) on three real world tasks.

Method	Tennis	Cup	Drawer
LingBot-VA	44%	24%	64%
Ours	52%	28%	56%

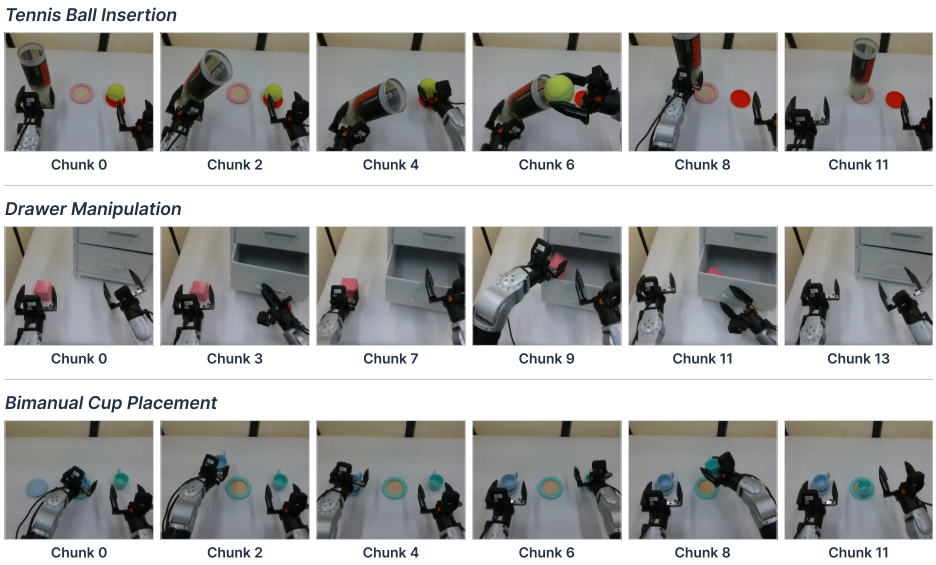


Figure C1: Real world task exmaples.

uses three RGB camera views and predicts 16-dimensional actions comprising 14 arm joint targets and two gripper commands.

We collect 45, 44, and 90 teleoperated demonstrations for the three tasks, respectively. We fine-tune a separate policy for each task for 4k steps, using a learning rate of 10^{-4} and a global batch size of 8. The LingBot-VA baseline is initialized from the official RoboTwin-posttrained checkpoint. Our method uses event-based full-evidence sampling and is initialized from our 50k-step RoboTwin checkpoint, including its execution head. We maintain the same 3-step video (integrating to s=0.6) and 10-step action (integrating to s=1.0) and $C = 2$ for inference. The evaluation for each method covers 25 independent real-robot rollouts per task.

C.2 Complete task-level results

DOMINO task-level performance. Tab. C2 reports SR and MS for all 35 DOMINO tasks under the Level 1 setting used in the main evaluation. It provides the task-level breakdown of Tab. 1, where our method achieves 42.83% SR and 55.92 MS, compared with 32.57% and 45.55 for LingBot-VA.

Zero-shot transfer across dynamic levels. Tab. C3 provides the individual-task results on the ten-task Level 2/3 subset. The policies are trained on Level 1 and evaluated without target-level adaptation. At Level 2, our method improves SR from 16.5% to 26.7% and MS from 30.94 to 44.03. At Level 3, SR increases from 6.8% to 8.7% and MS increases from 19.08 to 24.08. The gains of our method generalize beyond the training level, although the low absolute SR at Level 3 show that task completion under these dynamics remains challenging.

C.3 Training-sequence composition

Tab. C4 characterizes the prepared training sequences for the sampling variants in Tab. 4. An action slot is counted each time a position of the demonstration trajectory is sampled, including repetitions.

Table C2: Detailed per-task performance comparison on DOMINO Clean Level 1.

Task	ImageWAM		Fast-WAM		LingBot-VA		Ours	
	SR	MS	SR	MS	SR	MS	SR	MS
Adjust Bottle	71	80.09	63	77.53	**81**	**87.51**	48	62.78
Beat Block Hammer	18	31.89	12	25.45	33	45.56	**49**	**57.98**
Click Alarmclock	11	16.77	9	13.74	9	15.24	**22**	**29.51**
Click Bell	2	11.65	2	10.62	3	14.20	**7**	**18.03**
Dump Bin Bigbin	**36**	45.86	**36**	44.27	33	**47.48**	32	38.67
Grab Roller	48	62.63	49	64.77	59	68.55	**65**	**77.02**
Handover Block	0	38.37	9	51.90	**19**	56.44	13	**59.65**
Handover Mic	17	42.80	37	52.83	**48**	**60.06**	38	52.45
Hanging Mug	7	37.74	14	44.07	**39**	**61.24**	27	58.35
Move Can Pot	31	54.89	45	**68.03**	**48**	63.07	32	56.99
Move Pillbottle Pad	17	26.41	10	18.79	40	45.92	**72**	**75.14**
Move Playingcard Away	6	27.27	11	27.08	**30**	**41.55**	21	32.75
Move Stapler Pad	3	15.31	1	12.44	10	21.12	**20**	**33.49**
Place A2B Left	5	33.23	8	34.05	**16**	**38.18**	13	34.35
Place A2B Right	7	33.12	10	37.72	**24**	**46.17**	21	41.24
Place Bread Basket	22	31.35	13	23.43	41	47.67	**91**	**91.76**
Place Bread Skillet	15	31.32	11	27.78	11	27.75	**26**	**47.85**
Place Can Basket	15	40.64	17	47.06	**37**	**52.66**	24	46.63
Place Container Plate	37	42.19	34	38.62	60	63.31	**91**	**91.56**
Place Empty Cup	9	19.37	5	15.03	37	42.65	**62**	**65.51**
Place Fan	6	13.30	0	9.24	14	19.88	**50**	**54.90**
Place Mouse Pad	7	21.35	2	17.11	19	34.00	**64**	**70.19**
Place Object Basket	15	39.62	18	39.53	40	51.36	**72**	**77.49**
Place Object Scale	7	19.57	2	14.26	21	30.90	**28**	**39.24**
Place Object Stand	2	11.93	3	13.37	17	23.45	**48**	**52.42**
Place Phone Stand	9	23.94	8	24.62	14	29.03	**31**	**41.45**
Place Shoe	26	35.40	27	36.95	27	39.71	**54**	**63.36**
Press Stapler	9	18.41	8	15.62	13	22.19	**47**	**52.56**
Put Bottles Dustbin	22	36.46	35	45.53	**47**	**56.99**	35	45.35
Put Object Cabinet	29	59.05	**36**	63.87	26	53.05	**36**	**64.03**
Rotate QRcode	20	37.49	15	32.17	26	36.60	**34**	**44.34**
Scan Object	3	29.36	4	28.99	11	34.77	**22**	**46.08**
Shake Bottle	52	67.45	53	67.06	72	79.56	**73**	**80.79**
Shake Bottle Horizontally	63	78.28	55	71.69	**71**	82.85	68	**83.44**
Stamp Seal	13	26.93	6	21.49	44	53.51	**63**	**70.00**
Average	18.86	35.47	19.09	35.33	32.57	45.55	**42.83**	**55.92**

Slots are classified as inside or outside according to whether their source positions fall within any evidence phase within the interaction event, and percentages use each row's own total.

C.4 Execution cost breakdown

Tab. C5 expands the execution comparison in Tab. 3. For each task, we use the same evaluation episodes on which both LingBot-VA and our model succeed. N_{exec} counts complete action commands actually issued to the controller. N_{roll} counts completed future visual-rollout requests. The paired set contains 738 episodes across all 35 tasks. For each task, we compute each method's mean number of issued action commands over these episodes and the percentage reduction relative to the baseline using these means.

Table C3: Zero-shot transfer to unseen DOMINO dynamic levels.

Task	ImageWAM		Fast-WAM		LingBot-VA		Ours	
	SR	MS	SR	MS	SR	MS	SR	MS
Level 2								
Beat Block Hammer	2	13.52	1	13.29	2	13.40	**8**	**20.13**
Grab Roller	40	56.81	43	59.28	37	50.30	**47**	**65.30**
Handover Block	0	36.83	8	50.49	**30**	**54.91**	3	51.36
Move Pillbottle Pad	10	19.01	4	13.68	10	18.52	**14**	**22.34**
Place A2B Left	1	30.47	1	27.48	**14**	32.53	**14**	**32.66**
Place Bread Basket	4	16.37	3	15.87	5	15.68	**41**	**47.11**
Place Can Basket	13	36.96	**14**	**43.48**	8	26.91	6	32.25
Place Container Plate	16	22.81	14	20.32	13	19.60	**79**	**81.03**
Press Stapler	1	12.76	1	12.48	3	14.18	**25**	**34.06**
Shake Bottle	46	58.90	46	59.23	47	57.67	**57**	**67.98**
Average	13.3	30.44	13.5	31.56	16.9	30.37	**29.4**	**45.42**
Level 3								
Beat Block Hammer	2	8.03	1	7.78	2	10.20	**3**	**12.81**
Grab Roller	9	23.70	7	23.14	8	23.50	**18**	**33.82**
Handover Block	0	28.38	4	31.09	**9**	36.64	4	**41.16**
Move Pillbottle Pad	0	5.57	0	5.88	4	10.47	**9**	**17.75**
Place A2B Left	3	17.00	2	16.65	**5**	18.23	4	**20.82**
Place Bread Basket	2	17.59	0	16.51	6	21.64	**12**	**26.62**
Place Can Basket	1	16.34	2	19.43	2	17.65	**7**	**25.07**
Place Container Plate	4	9.23	3	9.00	11	16.61	**17**	**23.84**
Press Stapler	0	8.41	0	8.41	**3**	11.57	**3**	**12.81**
Shake Bottle	7	16.63	5	15.52	18	27.07	**20**	**27.55**
Average	2.8	15.09	2.4	15.34	6.8	19.36	**9.7**	**24.23**

Table C4: Slot composition of sampling configurations in Tab. 4.

Sampling	Granularity	Slots	Inside evidence phases	Outside evidence phases
LingBot-VA (reference)	Episode	291,296	94,233 (32.35%)	197,063 (67.65%)
Episode Uniform	Episode	211,664	80,953 (38.25%)	130,711 (61.75%)
Event Uniform	Event	211,664	100,653 (47.55%)	111,011 (52.45%)
Reduced Evidence	Event	268,592	61,913 (23.05%)	206,679 (76.95%)
Reduced Context	Event	173,216	116,114 (67.03%)	57,102 (32.97%)
Event-aligned (Ours)	Event	211,664	116,114 (54.86%)	95,550 (45.14%)

C.5 Additional Visualizations of Event-Aligned Visual Reasoning

Additional visual prediction and attention visualizations in Fig. C2, Fig. C3 and Fig. C4 show how visual prediction and executed behavior evolve through manipulation. For visualization of the visual prediction, we integrate to s=1.0 for video tokens here. Predicted visual states describe the model's anticipated scene evolution, while real observations show the interaction realized during execution. As shown in the predicted visual states, which are the last frame for each chunk, our method proceed the visual reasoning following the interaction dynamics. For each chunk, its visual imagination focuses on one transition or interaction event, which is benefit for action prediction and task completion.

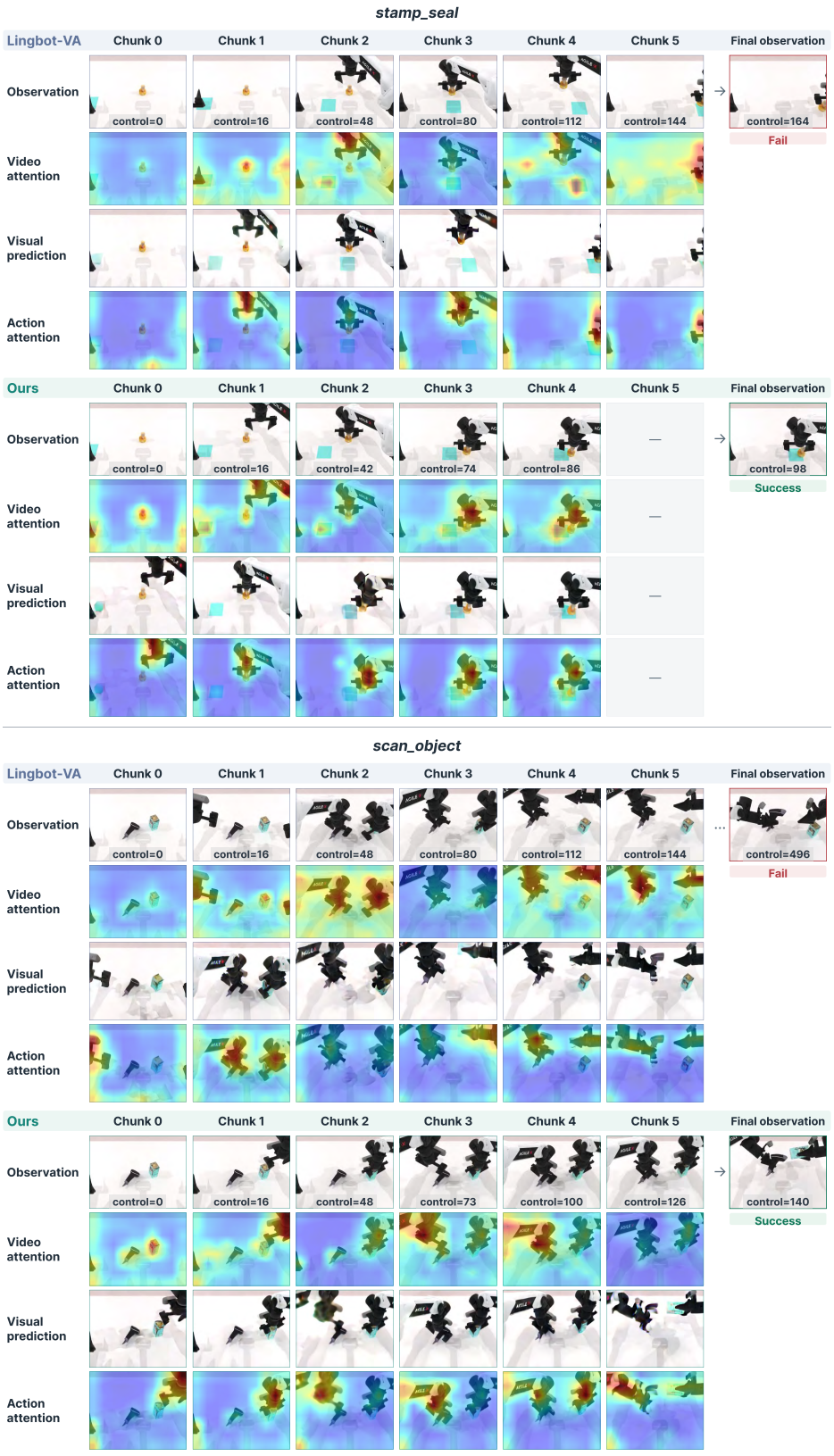


Figure C2: Visual prediction and attention during task execution on *stamp seal* and *scan object*.

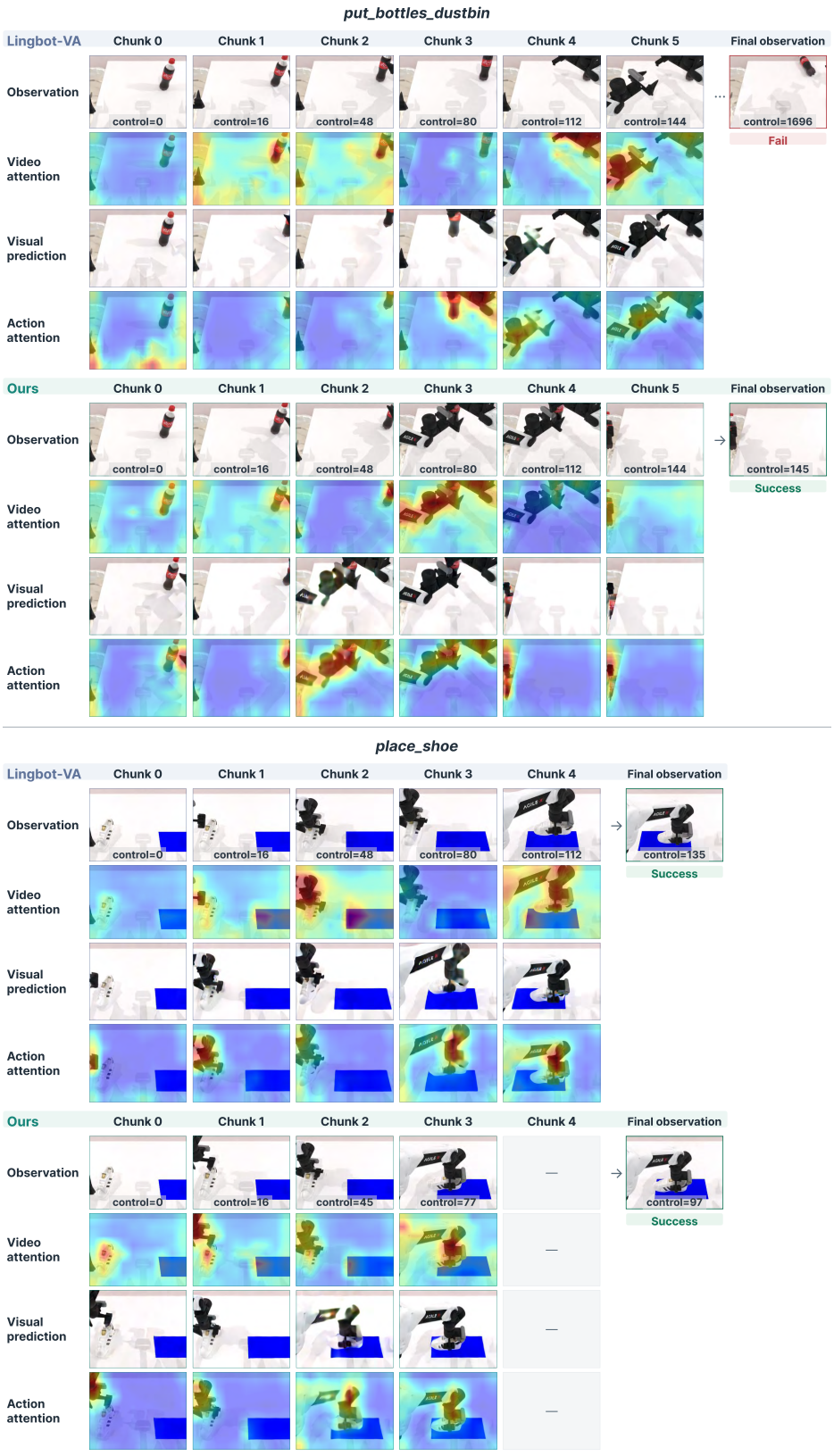


Figure C3: Visual prediction and attention during task execution on *put bottle dustbin* and *place shoe*.

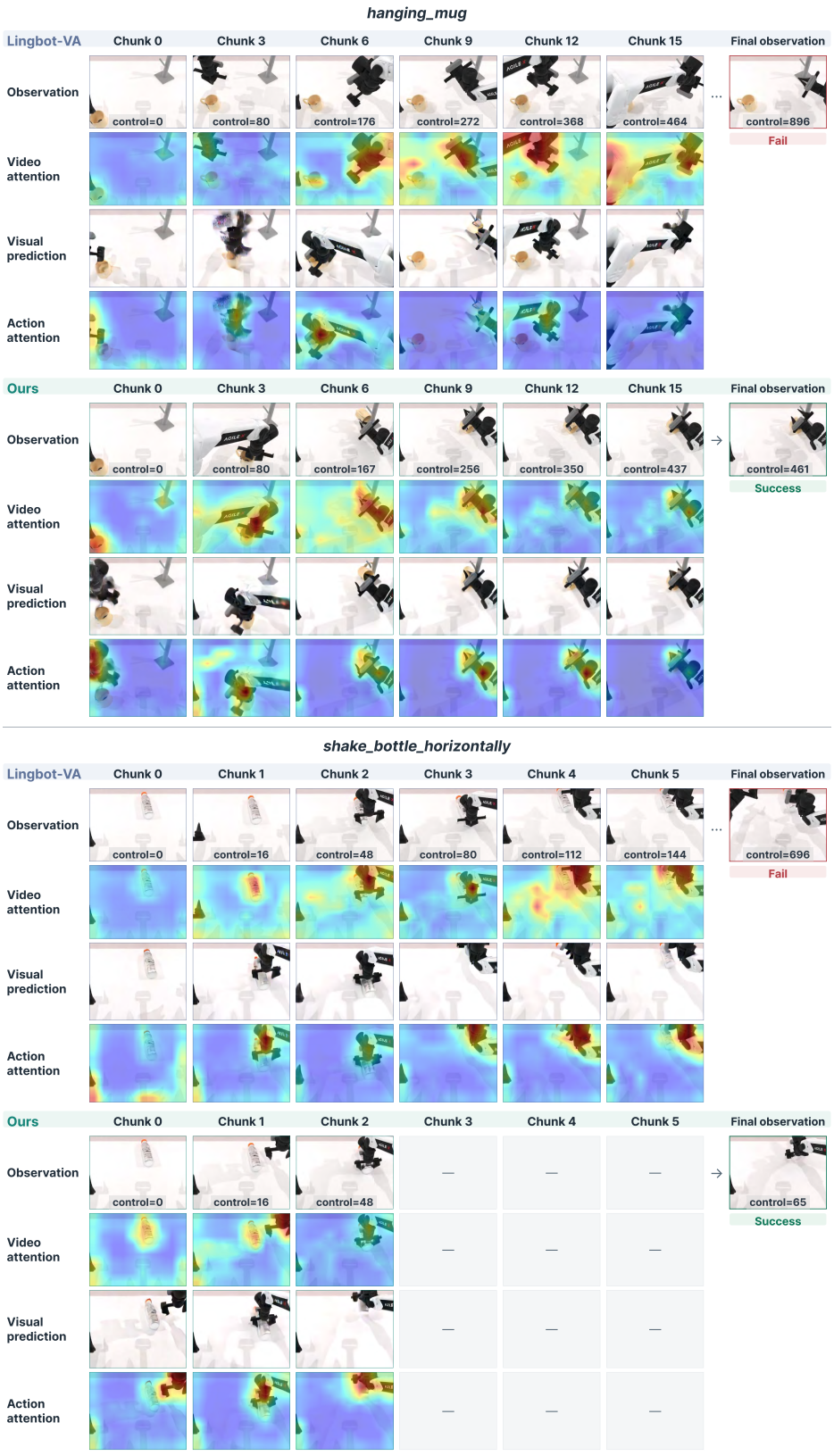


Figure C4: Visual prediction and attention during task execution on *hanging mug* and *shake bottle horizontally*.

Table C5: Execution and visual-rollout requests on common successes.

Task	**Paired** n	$N_{\text{exec}}^{\text{base}}$	$N_{\text{exec}}^{\text{ours}}$	**Reduction** (%)	$N_{\text{roll}}^{\text{base}}$	$N_{\text{roll}}^{\text{ours}}$
adjust_bottle	42	114.40	86.02	24.81	4.83	3.50
beat_block_hammer	27	111.44	77.26	30.67	4.19	3.26
click_alarmclock	6	68.33	75.17	-10.00	3.00	3.17
click_bell	1	216.00	77.00	64.35	8.00	3.00
dump_bin_bigbin	13	196.85	123.38	37.32	7.15	4.85
grab_roller	45	110.64	62.60	43.42	4.69	3.09
handover_block	8	276.50	138.00	50.09	9.25	5.00
handover_mic	25	243.84	119.56	50.97	8.56	4.56
hanging_mug	12	312.92	205.67	34.27	11.00	7.92
move_can_pot	13	157.46	90.15	42.75	5.92	4.00
move_pillbottle_pad	37	139.43	92.11	33.94	5.22	4.00
move_playingcard_away	13	164.46	91.54	44.34	5.92	4.00
move_stapler_pad	4	143.50	94.75	33.97	5.50	4.25
place_a2b_left	5	171.40	89.80	47.61	6.20	4.00
place_a2b_right	11	156.45	102.18	34.69	5.73	4.45
place_bread_basket	35	128.54	85.97	33.12	5.00	4.00
place_bread_skillet	5	126.20	86.20	31.70	5.00	4.00
place_can_basket	7	361.57	216.00	40.26	12.57	8.00
place_container_plate	57	135.65	88.42	34.82	5.14	4.07
place_empty_cup	33	149.21	87.52	41.35	5.85	4.09
place_fan	9	154.89	93.89	39.38	6.00	4.00
place_mouse_pad	15	165.07	129.60	21.49	6.07	5.40
place_object_basket	30	282.87	182.53	35.47	9.57	7.50
place_object_scale	14	136.36	97.64	28.39	5.00	4.36
place_object_stand	16	130.88	87.69	33.00	5.00	4.00
place_phone_stand	12	119.25	91.92	22.92	5.00	4.00
place_shoe	16	140.81	93.50	33.60	5.19	4.00
press_stapler	11	92.73	61.55	33.63	3.91	2.64
put_bottles_dustbin	24	170.21	111.58	34.44	6.25	4.54
put_object_cabinet	14	256.36	254.79	0.61	9.00	9.86
rotate_qrcode	15	182.60	90.27	50.57	6.80	4.00
scan_object	6	201.00	117.17	41.71	7.33	5.17
shake_bottle	62	90.05	67.34	25.22	3.97	3.16
shake_bottle_horizontally	56	92.00	67.96	26.13	4.02	3.29
stamp_seal	39	151.59	89.36	41.05	5.92	4.18
Task mean	738	167.18	106.46	34.63	6.22	4.49

Table D1: **Inputs to the event annotation pipeline.** Reviewed task specifications select the entities and detection rules. Measurements and recorded execution determine the times for each demonstration.

Input	Provenance	Use in annotation
Physical states and contacts	Recorded robot values and replayed simulator state	Identify gripper changes, object contact and motion, relative-pose stability, and articulation changes.
Task-success predicates	Native `check_success()` in benchmark task code	Localize task-specific state changes using the original predicate and our temporal confirmation rule.
Expert execution intervals	Recorded execution of benchmark expert routines	Localize designated task motions, such as the DOMINO shaking routines.
Task specifications and detection rules	Our annotation protocol, with LLM-assisted specifications reviewed task by task	Select actors, contact pairs, detector mappings, and the rules used to localize and verify interactions.

D Event Annotation for Event-Aligned Visual Reasoning

This appendix describes the offline annotation pipeline for the event-aligned visual action reasoning framework in Sec. 3.2. The pipeline first applies reviewed task specifications to each demonstration to localize task-relevant interactions and record their temporal evidence. The resulting event and phase annotations then provide the inputs to event-aligned visual-action chunk construction. We describe this process in the order of task specifications, episode-level temporal annotation, and the interface to chunk construction.

D.1 Task specifications and annotation inputs

Task specifications. We used an LLM (Claude-Opus 5) to draft task-level event specifications from simulator source code, including scene construction, expert execution procedures, and success predicates. Each draft was manually inspected and corrected before use. The reviewed specifications cover 35 DOMINO and 50 RoboTwin 2.0 tasks and identify the relevant objects and arms, native-predicate mappings, and applicable detection rules, with functional contact pairs specified where applicable. Each specification is reused across demonstrations of the same task. Episode-specific event timestamps were subsequently extracted during simulator replay by applying event detectors configured using the reviewed specifications. The prompt template used to draft the specifications is provided in Sec. D.5.

Annotation inputs. The simulator supplies physical states and contact reports, and the benchmark task implementations supply native success predicates and expert routines. Our annotation protocol defines the derived contact and hold conditions, detector labels, and temporal confirmation rules applied to these inputs. Native success predicates retain the conditions specified by each benchmark. Tab. D1 summarizes the provenance and use of these inputs.

D.2 Event localization and temporal annotation

Source timeline. Let L_{src} denote the number of observations in a raw demonstration, indexed by $t \in \{0, \ldots, L_{\mathrm{src}} - 1\}$. Replay records robot and scene states at the original observation-capture boundaries and evaluates the native task-success predicate on the same timeline. All annotation times and temporal windows refer to these source observations, before resampling or VAE encoding.

Localizing interactions. The reviewed specification selects the detectors applied to each demonstration. Their inputs and outputs are summarized below:

- *Acquisition and release.* Shared rules use recorded gripper values, replayed contacts, relative poses, and object motion to identify object grasps and releases. Handover and bimanual records combine the corresponding arm-specific detections.

Table D2: **Detector evidence windows in DOMINO.** Windows retain the source observations used for verification. Their relation to the method's event and phase annotations is described in Sec. D.3.

Detector source	Temporal evidence retained
Stable closed contact	From onset through the saved post-evidence end, including the complete qualifying stability window.
Merged bimanual grasp	From onset through the latest of the merged confirmation row and both component post-evidence windows.
Handover receive; object release / handover give	$[t_{\text{on}}, t_{\text{conf}} + 1)$, including the detector's stability or release-confirmation observations.
Task-specific body-pair contact	From contact onset through confirmation and the associated task-success or object-release effect, as specified by the detector.
Handle hold and drawer motion	The linked drawer record supplies the held-run window for handle verification; drawer-motion evidence spans the saved baseline-to-peak-search interval.
Native success or expert motion	$[t_{\text{on}}, t_{\text{conf}} + 1)$, using the predicate-confirmation interval or the designated expert-method record.

- *Functional contact.* Contacts between configured body pairs localize interactions such as placement, hanging, and stacking. Native task predicates can also be mapped to this detector category. Each record retains its source.
- *Task-specific state changes.* Native task predicates identify conditions such as reaching a target region or satisfying an orientation requirement. Articulated-object detections additionally use contact, pose, and joint-state measurements.
- *Recorded task motions.* Designated expert-execution intervals provide temporal anchors for motions such as DOMINO's vertical and horizontal shaking routines.

The configured detector subtypes for each task are listed in Sec. D.6.

Event times. Each detector record identifies the detected change, its participating objects and arms, and three temporal anchors: onset t_{on}, effect t_{eff}, and confirmation t_{conf}, with

$$0 \leq t_{\text{on}} \leq t_{\text{eff}} \leq t_{\text{conf}} < L_{\text{src}}. \tag{5}$$

Onset marks the beginning of the qualifying condition, effect anchors the detector-specific change, and confirmation records its verification anchor. For ordinary and handle grasps, the saved confirmation time marks the start of a qualifying stability window. For releases, handover receives, contacts, and success predicates, it marks the final confirmation row. The record therefore also retains the complete verification window and the measurements used by the detector.

Detector evidence windows. For a detector record η, we recover a half-open window $[u_\eta, v_\eta)$ containing the source observations used to verify the detected interaction. These windows follow the saved contact, stability, motion, or expert-execution records, rather than a fixed-radius neighborhood around the effect time. In particular, grasp verification can extend beyond the saved confirmation anchor. Tab. D2 summarizes the window sources in the DOMINO annotation protocol.

D.3 From temporal annotations to chunk construction

Event and phase annotations. Detector times provide temporal anchors for the event annotations in Sec. 3.2. Gripper opening and closing starts provide additional anchors for the surrounding manipulation motion. Records can overlap or describe different aspects of the same interaction, so detector records are distinct from the contiguous event partition used by the method. In particular, an effect time alone does not specify both boundaries of an interaction event.

The annotation interface to chunk construction consists of the events (E_i, c_i) with boundaries (s_i, e_i), together with evidence boundaries $(s_i^{\text{ev}}, e_i^{\text{ev}})$ for each interaction event. The detector windows retain the measurements used to verify a change, while the evidence and support phases specify the temporal organization of visual states within the interaction. The chunk constructor receives these event and phase boundaries as fixed annotations on the source timeline, following the interaction taxonomy in Sec. 3.2.

Retained trajectory. In the DOMINO sampling protocol, the retained trajectory includes the last event anchor and any later observations required by the saved detector evidence. Extending this final sampling range preserves the original annotation times. Row 0 remains the initial observation, and unused trailing observations are omitted. The method and its matched uniform sampling controls use the same retained endpoint. The final retained frame is T in the notation of Sec. 3.2; L_{src} denotes the raw observation count.

Visual-action targets. The annotated events and evidence boundaries are passed to Algorithm 1 to construct the visual-action training targets. Event boundaries determine chunk membership, and the evidence and support phases determine the temporal sampling within an interaction. Actions remain paired with their selected visual frames, and repeated positions provide supervision for the execution validity head in Sec. 3.3. The construction operators and cases are detailed in Sec. A.

D.4 Annotation checks and dataset selection

Consistency checks. We check replay alignment with the source demonstration and verify that detector labels and temporal records are consistent with the reviewed task specification and recorded execution. Replay failures and insufficient evidence are recorded explicitly. The annotation manifest retains episode status and any replay discrepancies used in downstream data selection.

DOMINO data selection. The audited DOMINO manifest contains $35 \times 50 = 1{,}750$ source demonstrations, of which 1,693 are retained. These include 1,364 with successful canonical annotations, 328 with recorded end-effector replay discrepancies, and one with an explicitly recorded short-confirmation exception using two terminal true observations instead of the default three. The remaining 57 demonstrations have replay failures and are excluded.

Use of annotation inputs. Simulator states, contacts, native task predicates, and expert-method markers are used offline to construct supervision. Their role is to localize interactions in the source demonstrations. Inference uses the learned visual-action model and execution validity predictions described in the main text.

D.5 Prompt for task-level event specifications

We used the following prompt with Claude Opus 5 to draft the task-level event specifications.

Prompt for drafting task-level event specifications

```
You are assisting in constructing task-level event specifications for
  robotic manipulation datasets. Your task is to draft a structured
  specification from the supplied simulator source code.

INPUTS
- Platform: {DOMINO or RoboTwin 2.0}
- Task source files: {envs/<task_name>.py}
- Supported specification schema and event-detector conventions:
  {schema_and_detector_contract}

SOURCE ANALYSIS
For each task, inspect:
1. Scene construction, such as load_actors(), to identify actor
  attributes and their meanings.
2. The expert execution procedure, such as play_once(), including
  conditional branches, handovers, regrasps, and bimanual actions.
3. check_success() and any helper functions it calls to determine the
  actual success conditions.

SPECIFICATION RULES
1. Use exact environment attribute names for:
   - grasp_actor_attributes: candidate actors for grasp detection;
   - primary_actor_attributes: actors associated with the main event.
2. Set expected_grasp_min using the expert procedure and the supplied
  detector conventions.
```

```
3. Define primary_event using the supported event vocabulary:
   - type: event category;
   - subtype: a concise description of the transition or interaction;
   - role: its semantic role in task execution;
   - predicate_definition: a faithful English summary of
  check_success().
4. Preserve the predicate's actual logic, including thresholds,
  coordinate frames, logical operators, branch conditions, and
  persistent or latched state. Do not add contact, release, stability,
  orientation, or displacement requirements unless the source checks
  them.
5. Add optional fields, such as functional_pairs, grasp_mode,
  handover, or branch-specific requirements, only when supported by
  both the source and the supplied detector contract.
6. Do not invent event timestamps or infer rules from task names
  alone.

OUTPUT EXAMPLE
{
  "source": {
    "platform": "RoboTwin 2.0",
    "task_module_root": "envs"
  },
  "event_contract": {
    "index_space": "episode-local source observation row",
    "model_independent": true
  },
  "tasks": {
    "adjust_bottle": {
      "source_file": "envs/adjust_bottle.py",
      "grasp_actor_attributes": ["bottle"],
      "primary_actor_attributes": ["bottle"],
      "expected_grasp_min": 1,
      "primary_event": {
        "type": "state_transition",
        "subtype": "high_at_instructed_side",
        "role": "COMPLETE",
        "predicate_definition": "Bottle functional point 0 is above
  world Z 0.9 and is left of world X -0.15 for qpose_tag 0 or right of
  world X 0.15 for qpose_tag 1; gripper state, contact, and
  displacement are not checked."
      }
    }
  }
}
```

D.6 TASK-SPECIFIC DETECTOR INVENTORY

Tab. D3 lists the DOMINO task mappings, and Tabs. D4 and D5 list the RoboTwin 2.0 mappings. Prefixes S, F, and G denote detector categories: `state_transition`, `functional_contact`, and `grasp`, respectively. These categories identify detected changes. The method uses the interaction and transition event types defined in Sec. 3.2. In particular, the detector label `state_transition` can mark a task-relevant interaction. Generic object-grasp, release, and gripper-change labels are shared across tasks and are omitted from individual rows. Additional G entries specify bimanual, handle, or handover-receive detections, with receives paired with `release/handover_release`. The inventory lists configured subtypes.

Table D3: DOMINO task-to-detector-subtype mapping. Prefixes S, F, and G identify detector categories.

Task(s)	Configured detector subtype(s) and interpretation
adjust_bottle	S: high_at_side Bottle reaches the required side and height.
beat_block_hammer	F: hit Hammer satisfies the task-specific hit condition.
click_alarmclock, click_bell, press_stapler	F: press Gripper satisfies the task-specific press condition.
dump_bin_bigbin	S: contents_height_band Bin reaches the required height; waste enters the height band.
grab_roller	S: lifted_bimanually; G: bimanual_grasp Two-arm acquisition and the roller-lift predicate.
handover_block	S: placed_at_target; G: handover_receive Handover followed by placement at the target.
handover_mic	S: handover_terminal_configuration; G: handover_receive Handover reaches the required terminal configuration.
hanging_mug	S: hung; F: hang_contact Mug reaches the rack, is released, and contacts it.
move_can_pot, place_a2b_left, place_a2b_right	S: placed_relative Relative-position and release conditions hold.
move_pillbottle_pad, place_container_plate, place_empty_cup, place_object_scale, place_object_stand, place_phone_stand, stamp_seal	S: placed; F: place_contact Placement succeeds with contact at the designated target.
move_playingcard_away	S: moved_away Card leaves the central X band and satisfies release and applicable displacement conditions.
move_stapler_pad, place_fan, place_mouse_pad, place_shoe	S: placed_oriented; F: place_contact Placement satisfies position, orientation, and contact conditions.
place_bread_basket	S: contacting_target_region; F: place_contact Bread satisfies basket-region, contact, and release conditions.
place_bread_skillet	S: positioned_at_target; F: place_contact Bread reaches the skillet target region; the primary predicate does not require release.
place_can_basket, place_object_basket	S: contacting_lifted_target; F: place_contact Object and lifted basket satisfy spatial and contact conditions.
put_bottles_dustbin	S: fixed_region_entry; G: handover_receive Bottle enters the target region; handover depends on the source branch.
put_object_cabinet	S: target_region_entry; S: drawer_opened; G: handle_grasp Drawer motion followed by entry into the cabinet target region.
rotate_qrcode	S: oriented QR-code object meets orientation, height, and applicable position conditions.
scan_object	S: aligned Object satisfies scanner-axis alignment and distance conditions.
shake_bottle	S: lift_proxy; S: shaken Lift proxy plus the recorded completion of the expert vertical-shake method.
shake_bottle_horizontally	S: lift_proxy; S: shaken_horizontally Lift proxy plus the recorded completion of the expert horizontal-shake method.

Table D4: RoboTwin 2.0 task-to-detector-subtype mapping (1/2). Prefixes S, F, and G identify detector categories.

Task(s)	Configured detector subtype(s)
adjust_bottle	S: high_at_instructed_side
beat_block_hammer	F: hit; F: hit_contact
blocks_ranking_rgb	S: ordered_red_green_blue
blocks_ranking_size	S: ordered_large_medium_small
click_alarmclock, click_bell, press_stapler	F: press
dump_bin_bigbin	S: contents_height_band
grab_roller	S: lifted_with_closed_grippers; G: bimanual_grasp
handover_block	S: placed_at_target; F: support_contact; G: handover_receive
handover_mic	S: handover_terminal_configuration; G: handover_receive
hanging_mug	S: hung_geometry; F: hang_contact
lift_pot	S: lifted_upright_between_tcps; G: bimanual_grasp
move_can_pot	S: placed_relative
move_pillbottle_pad	S: placed_on_pad_geometry; F: support_contact
move_playingcard_away	S: outside_center_x_band
move_stapler_pad, place_fan, place_mouse_pad	S: placed_on_pad_oriented; F: support_contact
open_laptop	S: joint_threshold_with_tcp_proximity
open_microwave	S: joint_threshold
pick_diverse_bottles, pick_dual_bottles	S: dual_bottles_at_lift_targets
place_a2b_left	S: placed_relative_left
place_a2b_right	S: placed_relative_right
place_bread_basket	S: all_bread_in_basket_region; F: containment_contact
place_bread_skillet	S: positioned_over_lifted_skillet; F: support_contact

Table D5: RoboTwin 2.0 task-to-detector-subtype mapping (2/2). Prefixes S, F, and G identify detector categories.

Task(s)	Configured detector subtype(s)
place_burger_fries	S: both_items_at_tray_regions; F: support_contact
place_can_basket, place_object_basket	S: contacting_lifted_basket; F: containment_contact
place_cans_plasticbox	S: both_cans_in_box_regions; F: containment_contact
place_container_plate	S: placed_on_plate_geometry; F: support_contact
place_dual_shoes	S: paired_shoes_in_box; F: containment_contact
place_empty_cup	S: placed_on_coaster_geometry; F: support_contact
place_object_scale	S: placed_on_scale_geometry; F: support_contact
place_object_stand	S: placed_on_stand_geometry; F: support_contact
place_phone_stand	S: placed_on_phone_stand_geometry; F: support_contact
place_shoe	S: placed_at_fixed_target_oriented; F: support_contact
put_bottles_dustbin	S: all_bottles_in_fixed_region; G: handover_receive
put_object_cabinet	S: cabinet_target_region_entry; S: expert_drawer_pull; G: handle_grasp
rotate_qrcode	S: oriented_below_height
scan_object	S: scanner_axis_alignment
shake_bottle, shake_bottle_horizontally	S: lift_height_proxy
stack_blocks_three	S: three_block_stack_geometry; F: stack_contact
stack_blocks_two	S: two_block_stack_geometry; F: stack_contact
stack_bowls_three	S: three_bowl_stack_geometry; F: stack_contact
stack_bowls_two	S: two_bowl_stack_geometry; F: stack_contact
stamp_seal	S: seal_at_visual_target_xy
turn_switch	S: joint_near_upper_limit