TRACEWAM: TASK-STATE REFERENCE-GUIDED ACTION CANDIDATE EVALUATION FOR LONG-HORIZON ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models generate robot actions from visual and language context, but generation alone does not establish active-stage alignment. We study *task-state alignment drift*: deterioration of task-critical poses, contacts, and spatial relations relative to stage goals during closed-loop execution. We propose *Task-State Reference-Guided Action Candidate Evaluation for World Action Models (TraceWAM)*, which represents the consequences of a base action chunk and a bounded residual candidate as structured physical deviations from the active stage goal. This object-centric representation connects consequence learning with pre-execution evaluation. Stage references and progress memory define the context; a shared verifier predicts each candidate's deviation and uncertainty. Training-only branches from shared simulator states supervise alternative outcomes. The residual is accepted only when reference validity, risk advantage, and predicted-progress conditions hold. On RoboTwin 2.0 and LIBERO, TraceWAM achieves average success rates of 93.0% and 98.1%, exceeding Fast-WAM by 1.6 and 1.4 percentage points, respectively, and reaches 41.6% overall success on RMBench. Matched-context comparisons assess structured consequence evaluation; ablations examine stage references, action-conditioned verification, and uncertainty weighting, while execution diagnostics characterize stage-relative state changes.

1 INTRODUCTION

Reliable multi-stage manipulation requires more than actions that follow instructions: execution must maintain the physical relations required at each stage. Consider placing a yellow-and-white mug on a green coaster (Fig. 1(a)). Grasping establishes contact, transport maintains the grasp and relative pose, and placement establishes the target spatial relation. Even with contact maintained, transport motion can leave the mug misaligned for stable placement. We define *task-state alignment drift* as deterioration of task-critical poses, contacts, and spatial relations relative to the active stage goal during closed-loop execution. The decision problem is therefore not only how to generate actions from new observations, but how to assess whether a candidate will improve or worsen stage alignment before retaining the base action or applying a local correction.

Vision–language–action (VLA) policies map observations and instructions to robot actions (Zitkovich et al., 2023; Kim et al., 2025; Black et al., 2025b); stage modeling and intermediate-goal reasoning further organize multi-stage execution (Fan et al., 2025; Zhao et al., 2025). World action models (WAMs) provide predictive representations through future observation prediction, latent dynamics, or joint observation–action modeling (Wu et al., 2024; Ye et al., 2026; Yuan et al., 2026). We study how these representations can guide action selection against the active stage's physical requirements. Evaluating candidates against a common stage goal and the same task-relevant components makes their physical consequences comparable even when their action sequences differ.

We propose **TraceWAM**, a task-state-reference-guided framework that uses stage-relative structured physical deviations for consequence learning and pre-execution candidate evaluation. The instruction, initial observation, and compositional schema library define stage references; progress memory selects the active stage. A shared verifier predicts alignment deviations and uncertainty for a base action chunk and a bounded residual candidate in an object-centric task-state space (Fig. 1(c)). Branches

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.

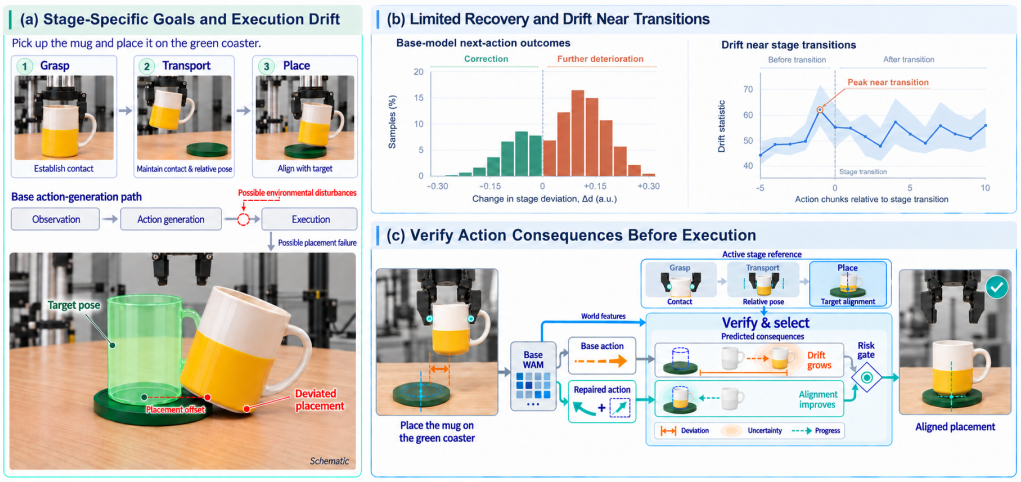

Figure 1: Action generation does not establish stage alignment. (a) Stage-specific physical relations can deteriorate during execution. (b) Transition-neighborhood drift motivates checking whether the next action corrects or worsens misalignment. (c) TraceWAM compares base and bounded-residual consequences under the same stage reference before selecting an action.

from the same simulator state provide alternative outcomes to supervise candidate differences. At execution, the residual is accepted only with a valid reference, risk advantage over the base action, and predicted progress from the current state; otherwise, the policy retains the base action.

TraceWAM improves success over Fast-WAM on RoboTwin 2.0, LIBERO, and RMBench, with larger RoboTwin gains under randomization, improvements across all four LIBERO suites, and gains on memory-dependent RMBench tasks. nTAD, DriftAUC, and ablations examine task-state changes and dependence on key designs. A real-world case study on a single-arm robot illustrates candidate–reference mismatch and action realignment after the target position changes.

Our contributions are threefold:

- We define *task-state alignment drift* as task-state deterioration relative to the active stage goal during closed-loop execution, and construct a task-relevant comparison space comprising object poses, contacts, spatial relations, and gripper state.
- We propose **TraceWAM**, whose common stage-relative structured deviation representation connects consequence learning and candidate evaluation. Shared-state branches supervise outcome differences between base actions and bounded residual candidates; risk-advantage and predicted-progress conditions translate these predictions into action selection.
- We demonstrate full-system execution gains on RoboTwin 2.0, LIBERO, and RMBench, use nTAD, DriftAUC, and ablations to analyze task-state changes and dependence on stage references and candidate evaluation, and illustrate candidate–reference mismatch and action realignment after target displacement in a single-arm real-world case study.

2 Related Work

Vision–Language–Action Models (VLA) and Stage References. VLA models learn instruction-conditioned control from vision–language pretraining and demonstrations. π_0 (Black et al., 2025b) generates continuous actions through flow matching; $\pi_{0.5}$ (Black et al., 2025a) combines semantic subtask prediction and low-level control. ReKep (Huang et al., 2025) instead optimizes relational keypoint constraints to satisfy stage subgoals and path requirements, using a locally rigid keypoint forward model. These approaches condition generation or optimize explicit constraints rather than learn stage-relative outcome differences between a base action and its local correction. TraceWAM learns these differences under shared pose, contact, and spatial references for candidate selection.

World Action Models (WAM) and Predictive Control. WAMs model visual dynamics and actions to learn predictive control representations. Motus (Bi et al., 2026) integrates understanding, video, and action experts; LingBot-VA (Li et al., 2026) jointly learns video and action prediction with observation feedback. Fast-WAM (Yuan et al., 2026) retains video co-training but omits explicit future generation at test time. These generation mechanisms do not specify when a local correction should replace the base action under the active stage's physical goal. TraceWAM instead uses world representations to evaluate candidate consequences before deciding which action to execute.

Candidate-Outcome Evaluation. Cosmos Policy (Kim et al., 2026) ranks candidates by predicted value rather than explicit stage-relative physical deviation. EV-WM (Wang et al., 2026) uses event-based and uncertainty-aware verification; its feature-space planner requires future-feature rollouts and event decoding. TraceWAM directly predicts stage-relative structured physical deviations from current world representations and candidate actions. Shared-state branches supervise these deviations, which aggregate into the physical distance used for candidate scoring. The active reference links consequence learning and bounded-correction acceptance through risk advantage over the base and predicted progress from the current physical task state.

3 METHODOLOGY

3.1 PROBLEM ANALYSIS

Contact or relative-pose deterioration can make continued goal-directed motion unsuitable for the active stage. Recognizing misalignment does not determine which candidate will improve it. TraceWAM therefore learns candidate outcomes as structured physical deviations from a common stage reference to compare a base action with a bounded correction. Shared-state branches supervise outcome differences, while risk advantage and predicted progress determine correction acceptance.

3.2 PROBLEM FORMULATION

At round t, the robot selects $a_t \in \mathcal{A}$ using instruction l, visual observations o_t, proprioception p_t, and pre-decision history $\mathcal{H}_t$. Here $\mathcal{A}$ is the feasible action-chunk space; $H, T_{\max}$ bound the control steps per chunk and decisions per episode. The simulated state after h steps of candidate a is $\bar{x}^a_{t,h}$, with shared initial state $\bar{x}^a_{t,0} = \bar{x}_t$.

Task state $x = \{T^o, T^e, c, r, q\}$ comprises object pose relative to a task anchor, end-effector pose relative to the object, contact and spatial-relation predicates, and gripper state. $\Delta_{\mathcal{Q}}(x, g)$ concatenates residuals from reference g under representation and aggregation rules $\mathcal{Q}$. For $T, T^* \in SE(3)$, the rigid-pose residual is $\rho(T, T^*) = S_{\mathrm{SE}(3)} \operatorname{Log}((T^*)^{-1}T)^\vee$; $^\vee$ converts the logarithm to six coordinates and $S_{\mathrm{SE}(3)}$ balances translation and rotation (Barfoot, 2024). For indices $\mathcal{C} = \{o, e, c, r, q\}$, binary mask v, weights $w_j \geq 0$, and stabilizer $\varepsilon > 0$, define

$$d_{\mathcal{Q}}(x, g; v) = \frac{\sum_{j \in \mathcal{C}} v_j w_j d_j(x_j, g_j)}{\sum_{j \in \mathcal{C}} v_j w_j + \varepsilon} = \|\Delta_{\mathcal{Q}}(x, g)\|_{\mathcal{Q};v}. \tag{1}$$

For component j, d_j is a normalized pose, predicate, or gripper ℓ_1 error; $\|\cdot\|_{\mathcal{Q};v}$ uses the same scales, weights, and aggregation. The mask selects the physical quantities relevant to the comparison. With offline stage label $\bar{z}_t$, reference $\bar{g}_{\bar{z}_t}$, relevance mask $s_{\bar{z}_t}$, and actual duration $H_t(a) \leq H$, we define signed drift by the change in stage-relative distance:

$$D^{H,*}_t(a) = d_{\mathcal{Q}}\left(\bar{x}^a_{t,H_t(a)}, \bar{g}_{\bar{z}_t}; s_{\bar{z}_t}\right) - d_{\mathcal{Q}}(\bar{x}_t, \bar{g}_{\bar{z}_t}; s_{\bar{z}_t}). \tag{2}$$

Both terms fix the starting reference and mask, separating physical changes from changes in stage goals or visibility. Positive values define *task-state alignment drift* (deterioration); negative values indicate improvement, distinct from current misalignment $d_{\mathcal{Q}}$. Offline labels and references serve training and evaluation, with the same distance applied before and after candidate execution.

3.3 MODEL ARCHITECTURE

Overview. TraceWAM connects consequence learning and action selection through stage-relative structured physical deviations. A shared verifier predicts deviations and uncertainty for base and

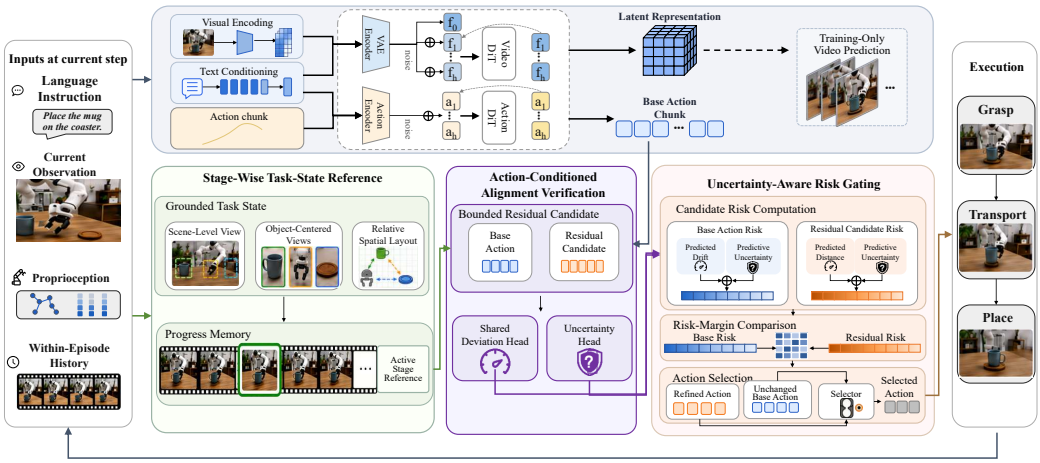

Figure 2: **Overview.** TraceWAM verifies a base action and a bounded residual candidate under a shared active stage reference. A shared verifier predicts candidate alignment deviation and uncertainty, and the selector executes the residual only when validity, risk, and progress conditions are satisfied; otherwise, it retains the base action. Same-state training provides consequence supervision, while post-execution observations update the next state and stage.

bounded-residual candidates sharing the reference, progress memory, and reliable task-relevant components. Corrections require reference validity, risk advantage, and predicted progress; otherwise, the base action is retained. Execution feedback updates the state and stage for the next decision.

Stage References and Progress Tracking. From l, o_1 and frozen schemas $\mathcal{S}$, the deterministic parser $(G, \mathcal{B}, \Gamma) = \mathrm{ParseGoal}(l, o_1; \mathcal{S})$ constructs K references $G = \{g_k\}_{k=1}^{K}$ and object-role bindings $\mathcal{B}$. Γ specifies relevance masks s_k, nonempty critical-component sets J_k, and tolerances $\epsilon_{k,j}$. Schemas encode predicate structures and ordering; training demonstrations supply pose prototypes and tolerances. Each g_k gives the target values of the task-state components for stage k. The frozen reader R_ξ, parameterized by ξ, estimates task state and a binary reliability mask: $(\tilde{x}_t, m_t) = R_\xi(o_t, p_t; \mathcal{B})$. Object-pose goals remain fixed relative to task anchors, while current observations update the anchor-relative poses in $\tilde{x}_t$. Set $v_{t,k} = s_k \odot m_t$ (elementwise product). With $B_k \in \{0, 1\}$ indicating complete bindings and a defined reference, $\mathrm{Valid}_{t,k} = B_k \prod_{j \in J_k} v_{t,k,j}$ permits comparison; stage completion requires each critical component to meet its tolerance:

$$C_{t,k} = \begin{cases} \mathbf{1}\left[\bigwedge_{j \in J_k} d_j(\tilde{x}_{t,j}, g_{k,j}) \le \epsilon_{k,j}\right], & \mathrm{Valid}_{t,k} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Here $\mathbf{1}[\cdot]$ is an indicator. Binary advancement and recovery events $\chi_t^{\mathrm{adv}}, \chi_t^{\mathrm{rec}}$ require, respectively, L_{adv} consecutive completion checks or L_{rec} consecutive violations of a previously satisfied predicate that remains critical to the stage being monitored. Stage references determine which predicates are required; reliability determines which components support the comparison. A gated recurrent unit (Cho et al., 2014) integrates observations and execution history into progress memory b_t:

$$b_t = \mathrm{GRU}_\omega(b_{t-1}, [\phi(o_t), p_t, \tilde{x}_t, m_t]) \,. \tag{4}$$

The GRU parameters are ω; $\phi(o_t)$ is a stop-gradient backbone feature, and brackets denote concatenation. The transition mask permits staying, advancing, or rolling back one stage:

$$M_t(k) = \begin{cases} 0, & k = z_{t-1}, \\ 0, & k = z_{t-1} + 1,\ \chi_t^{\mathrm{adv}} = 1, \\ -\kappa, & k = z_{t-1} - 1,\ \chi_t^{\mathrm{rec}} = 1, \\ -\infty, & \text{otherwise.} \end{cases} \tag{5}$$

The active stage is $z_t = \arg\max_{1 \le k \le K}(W_b b_t + M_t)_k$; W_b is the stage-readout matrix and $\kappa \ge 0$ penalizes rollback. Initialize $b_0 = 0, z_0 = 1$. Both candidates use g_{z_t} and $v_t = s_{z_t} \odot m_t$.

Action-Conditioned Alignment Verification. The backbone produces world features h_t^w and base chunk a_t^{bc}. A bounded residual supplies a local alternative (Johannink et al., 2019) from shared context $u_t = [\mathrm{sg}(h_t^w), g_{z_t}, b_t, v_t]$, where sg stops gradients. Let P_c be a fixed diagonal chunk mask selecting continuous coordinates and excluding discrete gripper commands:

$$\begin{aligned} \tilde{r}_t &= P_c \, \mathrm{clip}(r_\psi(u_t), -\eta, \eta)\,, \\ a_t^r &= \mathrm{Proj}_{\mathcal{A}}\big(a_t^{bc} + \tilde{r}_t\big)\,. \end{aligned} \tag{6}$$

Here ψ parameterizes the residual head and $\eta > 0$ bounds corrections. Projection clips continuous coordinates to the action range and retains the base chunk's discrete gripper sequence. These local motion corrections are evaluated through their consequences for pose, contact, and spatial alignment. The shared verifier with parameters θ_d and full action encoding $e_a(a)$ predicts

$$\Big(\hat{\delta}_{t,H}(a), \hat{\sigma}_{t,H}(a)\Big) = f_{\theta_d}^{\mathrm{dev}}([u_t, e_a(a)])\,. \tag{7}$$

Outputs are the candidate's terminal stage-relative deviation under the stopping rule and uncertainty $\hat{\sigma}_{t,H}(a) = \sigma_{\min} + \mathrm{softplus}(s_t(a))$, with raw scalar $s_t(a)$ and positive floor $\sigma_{\min}$. Shared context fixes comparison conditions, while action encodings distinguish outcomes. Unlike current-state distance, these predictions support prospective comparison of action-dependent physical outcomes.

Risk-and-Progress Action Selection. An advantage over the base action does not itself imply progress from the current state. We therefore compare both, using uncertainty-weighted risk

$$\mathcal{R}_t(a) = \|\hat{\delta}_{t,H}(a)\|_{\mathcal{Q};v_t} + \beta \hat{\sigma}_{t,H}(a), \qquad \beta \ge 0, \tag{8}$$

where β weights uncertainty. Subtracting the shared current distance $d_t = d_{\mathcal{Q}}(\tilde{x}_t, g_{z_t}; v_t)$ preserves predicted-distance rankings; risk also accounts for uncertainty. We accept corrections by

$$\lambda_t = \begin{cases} \mathbf{1}\big[\mathcal{R}_t(a_t^r) < \min\{\mathcal{R}_t(a_t^{bc}) - m_g, d_t - m_p\}\big]\,, & \mathrm{Valid}_{t,z_t} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Nonnegative m_g, m_p specify risk and progress margins. The first bound requires a sufficient advantage over the base candidate; the second requires predicted progress relative to the current state. The policy executes the residual when $\lambda_t = 1$, otherwise the base; invalid references bypass scoring. Execution updates the state-check record until H steps, reliable stage completion, or episode termination; active episodes then update observations and history for the next comparison (Algorithm 1).

3.4 Training Objective

Shared-State Branch Supervision. A demonstration provides executed actions and observed outcomes; same-state simulator branches additionally supply outcomes for alternative candidates. $\mathcal{A}_t \subseteq \mathcal{A}$ contains the base action and M additional proposals: the current residual, bounded random perturbations, and $\Delta a_t^{\mathrm{exp}} = P_c \, \mathrm{clip}(a_t^{\mathrm{demo}} - a_t^{bc}, -\eta, \eta)$, where a_t^{demo} is the training state's expert chunk. All proposals use Eq. (6)'s subspace and projection; branches share the initial state, seed, maximum horizon, and stopping rule. Their terminal states yield

$$\delta_{t,H}^*(a) = \Delta_{\mathcal{Q}}\Big(\bar{x}_{t,H_t(a)}^a, \bar{g}_{\bar{z}_t}\Big)\,. \tag{10}$$

Prediction and risk training use $\bar{g}_{\bar{z}_t}$ and $v_{t,\bar{z}_t} = s_{\bar{z}_t} \odot m_t$; correction targets and preferences use $D_t^{H,*}$ under the full stage mask. Online selection uses estimated states and policy-side references. The branch terminal state supervises the deviation from the active reference, while differences between branches establish which candidate has the better realized outcome from their common starting state.

Auxiliary Supervision. For aggregated error $e_{t,a} = \|\hat{\delta}_{t,H}(a) - \delta_{t,H}^*(a)\|_{\mathcal{Q};v_t,\bar{z}_t}$, heteroscedastic regression learns deviations and error scales (Kendall & Gal, 2017):

$$\mathcal{L}_{\mathrm{align}} = \frac{1}{|\mathcal{A}_t|} \sum_{a \in \mathcal{A}_t} \left[\frac{e_{t,a}^2}{2\hat{\sigma}_{t,H}^2(a)} + \frac{1}{2} \log \hat{\sigma}_{t,H}^2(a) \right]. \tag{11}$$

Algorithm 1 TraceWAM Inference

Require: Instruction l; initial observations (o_1, p_1); schema library $\mathcal{S}$
Require: Trained backbone, R_ξ, memory and prediction heads; P_c; $H, T_{\max}$; defined hyperparameters
$(G, \mathcal{B}, \Gamma) \leftarrow \mathrm{ParseGoal}(l, o_1; \mathcal{S})$
Initialize $\mathcal{H}_1 \leftarrow \emptyset$, $b_0 \leftarrow 0$, $z_0 \leftarrow 1$, and the state-check record
for $t = 1$ to $T_{\max}$ **do**
  $(\tilde{x}_t, m_t) \leftarrow R_\xi(o_t, p_t; \mathcal{B})$; obtain $\phi(o_t), h_t^w, a_t^{bc}$ from the backbone
  Compute $\{v_{t,k}, \mathrm{Valid}_{t,k}, C_{t,k}\}_{k=1}^K$; obtain $\chi_t^{\mathrm{adv}}, \chi_t^{\mathrm{rec}}$ from the state-check record
  Update b_t using Eq. (4); form M_t using Eq. (5)
  $z_t \leftarrow \arg\max_{1 \le k \le K} (W_b b_t + M_t)_k$; $a_t \leftarrow a_t^{bc}$; $\lambda_t \leftarrow 0$
  if $\mathrm{Valid}_{t,z_t} = 1$ **then**
    $v_t \leftarrow s_{z_t} \odot m_t$; $d_t \leftarrow d_{\mathcal{Q}}(\tilde{x}_t, g_{z_t}; v_t)$
    $u_t \leftarrow [\mathrm{sg}(h_t^w), g_{z_t}, b_t, v_t]$; construct a_t^r using Eq. (6)
    for $a \in \{a_t^{bc}, a_t^r\}$ **do**
      Predict $(\hat{\delta}_{t,H}(a), \hat{\sigma}_{t,H}(a))$ with Eq. (7); compute $\mathcal{R}_t(a)$ with Eq. (8)
    end for
    Evaluate λ_t using Eq. (9); $a_t \leftarrow \lambda_t a_t^r + (1 - \lambda_t) a_t^{bc}$
  end if
  Execute a_t with state checks until H steps, reliable stage completion, or episode termination
  Update the state-check record; stop if the episode has terminated
  Obtain (o_{t+1}, p_{t+1}) and update $\mathcal{H}_{t+1}$
end for

Stage loss uses unmasked logits, $\mathcal{L}_{\mathrm{prog}} = -\log[\mathrm{Softmax}(W_b b_t)]_{\bar{z}_t}$. For masked stage selection, previous-stage inputs use true labels initially, then scheduled sampling. Let $a_t^+ = \arg\min_{a \in \mathcal{A}_t} D_t^{H,*}(a)$. Set $r_t^* = a_t^+ - a_t^{bc}$ when $D_t^{H,*}(a_t^{bc}) - D_t^{H,*}(a_t^+) \ge \gamma_{\mathrm{res}}$ (the minimum improvement threshold), otherwise zero. Residual loss $\mathcal{L}_{\mathrm{res}} = \|P_c(\tilde{r}_t - r_t^*)\|_1$ supervises modifiable coordinates. Better–worse pairs $(a^+, a^-) \in \mathcal{P}_t$ satisfy $D_t^{H,*}(a^-) - D_t^{H,*}(a^+) > \gamma_{\mathrm{pair}}$, where γ_{pair} is the pairing threshold. Candidate risk ranking uses

$$\mathcal{L}_{\mathrm{rank}} = \frac{1}{\max\{1, |\mathcal{P}_t|\}} \sum_{(a^+, a^-) \in \mathcal{P}_t} \left[\mathcal{R}_t(a^+) - \mathcal{R}_t(a^-) + m_r\right]_+ . \tag{12}$$

Here m_r is the ranking margin, $[x]_+ = \max(x, 0)$, and empty sums are zero. The objectives learn deviations and proposals while aligning candidate risk with realized outcome preferences. Regression supervises the physical error components, residual learning supervises a local proposal, and ranking connects the ordering of observed branch outcomes to the score used for selection.

We combine these objectives with the backbone video–action flow-matching loss $\mathcal{L}_{\mathrm{base}}$ (Lipman et al., 2023) and auxiliary weights $\lambda_\delta, \lambda_z, \lambda_r, \lambda_g$:

$$\mathcal{L} = \mathcal{L}_{\mathrm{base}} + \lambda_\delta \mathcal{L}_{\mathrm{align}} + \lambda_z \mathcal{L}_{\mathrm{prog}} + \lambda_r \mathcal{L}_{\mathrm{res}} + \lambda_g \mathcal{L}_{\mathrm{rank}}. \tag{13}$$

The pretrained reader is frozen and ParseGoal is non-learned. We first train the backbone, then freeze it and train progress, verifier, and residual branches on branched data.

4 Experiments

4.1 Implementation Details

We build TraceWAM on the pretrained Wan2.2-5B video DiT, text encoder, and video VAE (Team Wan et al., 2025; Team Wan, 2025). The action expert follows the video architecture with hidden size 1024 and about 1B parameters; the two backbones total about 6B. Fixed-order multi-view observations are VAE-encoded, and instructions condition both branches through cross-attention. The state reader is pretrained on training-environment annotations and frozen; schemas and reference parameters are fixed before evaluation and unchanged across test episodes.

The maximum action horizon is 32 control steps; factor-four subsampling gives nine frames including the initial observation. Inference uses 10 denoising steps, guidance scale 1.0, and Eq. (9). Chunk execution follows the stopping rule in Algorithm 1. Training uses three A100 GPUs and AdamW

with learning rate 10^{-4}, weight decay 0.01, cosine decay, mixed precision, and gradient clipping at 1.0. After backbone training, auxiliary modules use the shared-state branches in Sec. 3.4.

4.2 Experimental Setup

We evaluate on LIBERO (Liu et al., 2023), RoboTwin 2.0 (Chen et al., 2026a), and RMBench (Chen et al., 2026b). These benchmarks examine task goals, scene randomization, and memory-dependent execution. Success rate measures final completion; task-state metrics describe intermediate deterioration, and ablations examine the selection design.

For $N \geq 1$ non-overlapping executed chunks $\mathcal{V} = \{t_1, \ldots, t_N\}$, let $\bar{d}_t = d_{\mathcal{Q}}(\bar{x}_t, \bar{g}_{\bar{z}_t}; s_{\bar{z}_t})$ and $\nu_t = \max\{\bar{d}_t, \epsilon^{\text{scale}}_{\bar{z}_t}, \epsilon_0\}$, where $\epsilon^{\text{scale}}_{\bar{z}_t}$ is a stage-specific scale floor and $\epsilon_0 > 0$. With $e_i = \max\{0, D^{H,*}_{t_i}(a_{t_i})\}/\nu_{t_i}$, $\text{nTAD} = \frac{1}{N}\sum_{i=1}^{N} e_i$ measures mean normalized positive deterioration, and $\text{DriftAUC} = \frac{2}{N(N+1)}\sum_{i=1}^{N}(N-i+1)e_i$ weights earlier deterioration more heavily. Unlike the current distance $d_{\mathcal{Q}}$ or signed chunkwise change $D^{H,*}_t$, these aggregates do not offset deterioration with later recovery. Both use post-execution simulator states and offline stage references.

LIBERO. Spatial, Object, Goal, and Long each contain 10 tasks with 50 demonstrations per task. Training lasts 20,000 steps; 50 closed-loop trials per task with distinct environment seeds yield 2,000 episodes. We report per-suite success rates and their unweighted average across the four task suites.

RoboTwin 2.0. One policy is trained for 30,000 steps on 50 bimanual tasks with 27,500 demonstrations: 2,500 clean and 25,000 strongly randomized. Each task has 100 evaluation episodes per setting. We average across tasks and then across clean and randomized settings. Controlled variants share tasks, evaluation budgets, and seeds; branch outcomes supervise auxiliary training.

RMBench. Five $M(1)$ tasks require one critical historical state; four $M(n)$ tasks require repeated exploration, trial-and-error, or accumulated feedback. Each uses 50 synthetic expert demonstrations and 100 test episodes, totaling 450 training trajectories and 900 test episodes. We report group means and the mean over all nine tasks, weighting groups by their respective task counts.

Matched-Context Protocol. Table 2 compares direct execution of demonstration-trained and branch-trained residual candidates with scalar and structured verification. Branch-based variants share the backbone, state reader, stage references, progress memory, residual generator, and branch data. Verifiers use matched training budgets and the same acceptance-rule form, predicting either aggregate distance directly or structured physical deviations before aggregation.

Real-World Case Study. We examine action realignment on a single-arm robot placing a red block in a target bowl. Displacing the bowl during transport changes its current position while preserving the instruction and desired block–bowl relation. The grasp–transport–place sequence provides a qualitative case study of candidate evaluation following a change in the physical scene.

4.3 Main Results

Table 1 compares TraceWAM with π_0 (Black et al., 2025b), $\pi_{0.5}$ (Black et al., 2025a), Motus (Bi et al., 2026), LingBot-VA (Li et al., 2026), and Fast-WAM (Yuan et al., 2026). Bold and underlined entries mark the best and second-best listed results; "–" denotes unreported results. All baseline results are reproduced under our evaluation protocol. Table 1 compares full systems; Sec. 4.5 controls stage context, residual proposals, and branch supervision to examine candidate evaluation.

On RoboTwin 2.0, TraceWAM achieved 92.8% clean, 93.1% randomized, and 93.0% average success, exceeding Fast-WAM by 0.9, 2.3, and 1.6 percentage points. It ranked first on randomized and average success; clean success was 0.1 point below LingBot-VA. Randomized and average gains over LingBot-VA were 1.6 and 0.8 points; gains over Fast-WAM were larger under randomization.

On LIBERO, TraceWAM averaged 98.1%, exceeding Fast-WAM by 1.4 points. Gains covered Goal, Spatial, Long, and Object by 2.4, 1.8, 1.2, and 0.2 points, respectively. TraceWAM led Spatial and Goal and ranked second overall to LingBot-VA (98.5%). Improvements over Fast-WAM thus spanned all four suites, with the largest differences on goal- and spatial-relation tasks.

On RMBench, TraceWAM achieved 49.8% on $M(1)$, 31.3% on $M(n)$, and 41.6% overall, ranking first on $M(1)$ and overall and second on $M(n)$ among the methods in Table 1. It exceeded LingBot-

Table 1: Success rates (%) on RoboTwin 2.0, LIBERO, and RMBench under our evaluation protocol.

	RoboTwin 2.0			LIBERO					RMBench		
Method	Clean	Rand.	Avg.	Spatial	Object	Goal	Long	Avg.	$M(1)$	$M(n)$	Overall
π_0	65.9	58.4	62.2	97.8	98.4	94.6	81.0	93.0	–	–	–
$\pi_{0.5}$	82.7	76.8	79.8	97.2	**99.8**	96.0	94.2	96.8	14.4	5.5	10.4
Motus	88.7	87.0	87.9	96.8	**99.8**	96.6	97.6	97.7	–	–	–
LingBot-VA	**92.9**	91.5	92.2	98.5	99.6	97.2	**98.5**	**98.5**	34.2	**35.5**	34.8
Fast-WAM	91.9	90.8	91.4	96.8	99.4	95.4	95.2	96.7	2.8	5.5	4.0
TraceWAM (Ours)	92.8	**93.1**	**93.0**	**98.6**	99.6	**97.8**	96.4	98.1	**49.8**	31.3	**41.6**

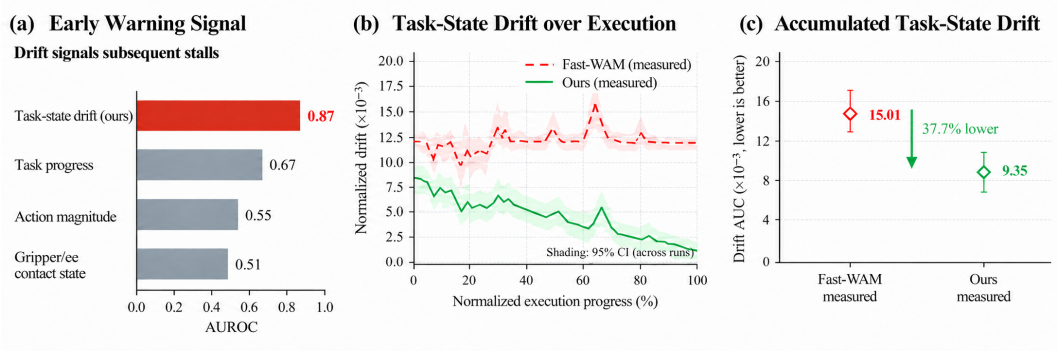

Figure 3: Task-state alignment diagnostics on RoboTwin 2.0. (a) Discrimination of subsequent progress stalls. (b) Normalized deterioration over execution progress, with 95% confidence intervals across runs. (c) Trajectory-level DriftAUC for both methods.

VA by 15.6 and 6.8 points on $M(1)$ and overall, but was 4.2 points lower on $M(n)$. Both groups improved over Fast-WAM, whose overall score was 4.0%, with gains on tasks requiring one historical state as well as repeated exploration.

4.4 Task-State Alignment Analysis

Figure 3(a) reports an AUROC of 0.87 for subsequent-stall discrimination, compared with 0.67 for task progress, 0.55 for action magnitude, and 0.51 for gripper/contact state. In panel (b), TraceWAM shows lower normalized deterioration than Fast-WAM over the displayed execution progress. Panel (c) reports DriftAUC values of 15.01×10^{-3} for Fast-WAM and 9.35×10^{-3} for TraceWAM, a reduction of 37.7%. These diagnostics show lower task-state deterioration in the analyzed trajectories.

Real-World Case Study. Figure 4 follows the robot through target displacement and recovery. After grasping the red block (a–b), the robot began transport toward the bowl. The bowl was displaced during this motion (c), while the base candidate remained directed toward its former position. The active goal still required placing the block in the bowl: its anchor-relative definition stayed fixed, and the new observation changed the measured relation between the carried block and the target. TraceWAM evaluated the base and bounded-residual candidates against this same reference and selected a correction toward the displaced bowl (d). The robot then placed and released the block at the updated target location (e–f). This case connects the physical change in the scene to the candidate–reference mismatch and the resulting motion correction, showing action realignment within a complete grasp–transport–place sequence.

4.5 Candidate Evaluation and Ablation Studies

Candidate Evaluation. With context, branch data, and candidate generation held fixed (Table 2), Branch residual directly executes the candidate a_t^r in Eq. (6), bypassing the base-versus-residual comparison in Eq. (9). It achieved success rates of 90.6% on randomized RoboTwin and 30.8% overall on RMBench. Scalar verification increased these rates to 92.0% and 37.6%. Structured verification

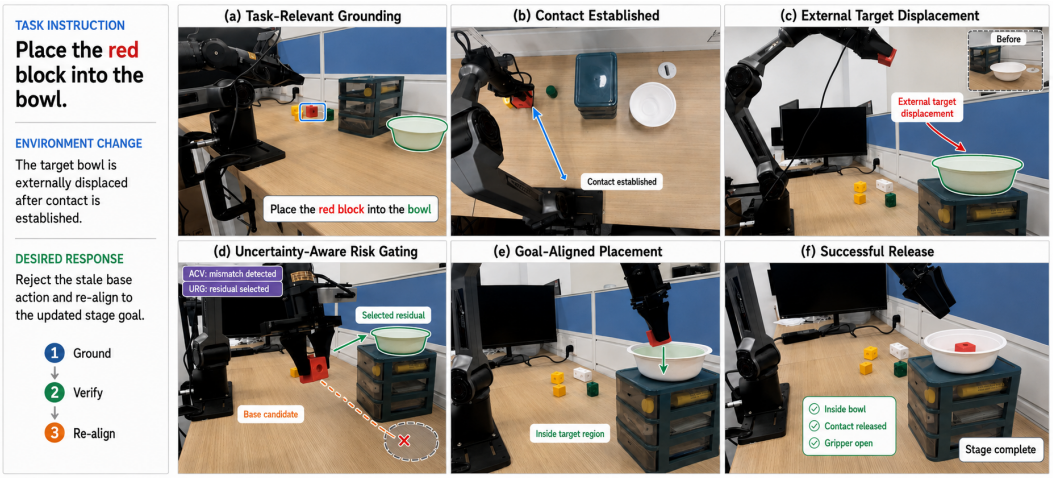

Figure 4: Real-world case study of action realignment after target displacement. The robot grasps the red block (a–b), and the bowl is moved during transport (c). Evaluation against the active placement reference selects a correction toward the bowl's current position (d), followed by placement and release (e–f). The sequence illustrates the change in action selection within one execution.

Table 2: Matched context (%).

Variant	RT-R	RM
Fast-WAM	90.8	4.0
Demo residual	89.9	23.8
Branch residual	90.6	30.8
Scalar verifier	92.0	37.6
Structured (ours)	**93.1**	**41.6**

Table 3: RoboTwin ablations (%).

Variant	Clean	Rand.	Avg.	Drop
Full	**92.8**	**93.1**	**93.0**	–
w/o STR	86.5	82.7	84.6	−8.4
w/o ACV	85.1	80.8	83.0	−10.0
w/o UW	90.2	87.8	89.0	−4.0

RT-R: RoboTwin randomized; RM: RMBench overall. Full: TraceWAM; Drop: percentage points.

provided further gains of 1.1 and 4.0 percentage points, reaching 93.1% and 41.6%. These gains support using stage-relative physical deviations for consequence learning and candidate evaluation beyond direct prediction of aggregate distance. The structured predictor retains pose, contact, and relation errors until the stage mask aggregates them into a common score. Candidate selection therefore uses the same physical comparison that supervises outcome learning.

Component Ablations. Table 3 examines dependence on stage-wise task-state references (STR), action-conditioned alignment verification (ACV), and uncertainty weighting (UW). Removing ACV reduced average success from 93.0% to 83.0%, a 10.0-point drop; removing STR and UW yielded 84.6% and 89.0%, drops of 8.4 and 4.0 points, respectively. ACV and STR removal caused larger drops under randomization (12.3 and 10.4 points) than in clean scenes (7.7 and 6.3), showing stronger dependence on stage references and candidate-specific verification under scene variation.

5 Conclusion

TraceWAM represents candidate action consequences as structured physical deviations from the active task-stage goal, using this unified representation to connect consequence learning with pre-execution action selection. Shared-state branches provide supervision for the physical deviations caused by base actions and bounded corrections. At execution, these predicted deviations allow both candidates to be compared against the same stage goal, with risk advantage and predicted progress guiding the choice between them. Experiments on RoboTwin 2.0, LIBERO, and RMBench show higher closed-loop success than Fast-WAM. Matched-context comparisons show further gains from structured evaluation over scalar verification. Component ablations assess the roles of stage references, action-conditioned verification, and uncertainty weighting in closed-loop execution, while diagnostics show lower task-state deterioration during task execution.

AI USE STATEMENT

We used generative AI tools primarily to improve the language, clarity, and presentation of the manuscript and to assist with formatting and typesetting. This assistance included revising English phrasing, organizing the presentation of technical details, and formatting equations and tables. The authors reviewed and revised the AI-assisted content and take full responsibility for the final manuscript, including its technical claims, reported results, and references.

REPRODUCIBILITY STATEMENT

The formulation, training objectives, and inference procedure of TraceWAM are described in Sec. 3 and Algorithm 1. Appendix A summarizes the backbone and optimization settings, and Appendix B specifies the task-state representation, stage references, and validity checks. Appendix C details shared-state branch supervision and the two-stage training procedure. Benchmark budgets, metric aggregation, and the matched-context comparison are described in Sec. 4.2 and Appendix D. Appendix E explains the execution diagnostics, while Appendix F describes the real-world demonstration. Training uses three NVIDIA A100 GPUs.

REFERENCES

Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, Cambridge, UK, 2nd edition, 2024. doi: 10.1017/9781009299909. URL https://doi.org/10.1017/9781009299909.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Bi_Motus_A_Unified_Latent_Action_World_Model_CVPR_2026_paper.html.

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 *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

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 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2506.18088.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *CoRR*, abs/2603.01229, 2026b. doi: 10.48550/ARXIV.2603.01229. URL https://doi.org/10.48550/arXiv.2603.01229.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/V1/D14-1179. URL https://aclanthology.org/D14-1179/.

Yiguo Fan, Shuanghao Bai, Xinyang Tong, Pengxiang Ding, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, Zhaoxin Fan, Badong Chen, and Donglin Wang. Long-VLA: Unleashing long-horizon capability of vision language action model for robot manipulation. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2018–2037. PMLR, 2025. URL https://proceedings.mlr.press/v305/fan25a.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp.

4573–4602. PMLR, 2025. URL https://proceedings.mlr.press/v270/huang25g.html.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation*, pp. 6023–6029. IEEE, 2019. doi: 10.1109/ICRA.2019.8794127. URL https://doi.org/10.1109/ICRA.2019.8794127.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, volume 30, pp. 5574–5584, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html.

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 *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/748becc400a57c0e31cfe6a2e7951467-Abstract-Conference.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. *CoRR*, abs/2601.21998, 2026. doi: 10.48550/ARXIV.2601.21998. URL https://arxiv.org/abs/2601.21998v1.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Team Wan. Wan2.2: Official model and code release. Official software and model release, 2025. URL https://github.com/Wan-Video/Wan2.2. Released July 28, 2025. Includes the Wan2.2-TI2V-5B checkpoint.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *CoRR*, abs/2503.20314, 2025. doi: 10.48550/ARXIV.2503.20314. URL https://doi.org/10.48550/arXiv.2503.20314.

Kailin Wang, Haoxiang Jie, Yaoyuan Yan, Jiacheng Zhou, and Zhiyou Heng. EV-WM: Event-verified world models for long-horizon robotic manipulation. *CoRR*, abs/2606.13053, 2026. doi: 10.48550/ARXIV.2606.13053. URL https://arxiv.org/abs/2606.13053v2.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NxoFmGgWC9.

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. *CoRR*, abs/2602.15922, 2026. doi: 10.48550/ARXIV.2602.15922. URL https://doi.org/10.48550/arXiv.2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *CoRR*, abs/2603.16666, 2026. doi: 10.48550/ARXIV.2603.16666. URL https://doi.org/10.48550/arXiv.2603.16666.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Tsung-Yi Lin, Gordon Wetzstein, Ming-Yu Liu, and Donglai Xiang. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1702–1713, 2025. doi: 10.1109/CVPR52734.2025.00166. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhao_CoT-VLA_Visual_Chain-of-Thought_Reasoning_for_Vision-Language-Action_Models_CVPR_2025_paper.html.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Additional Implementation Details

A.1 Backbone Configuration

TraceWAM is initialized from Wan2.2-5B together with its pretrained text encoder and video variational autoencoder (VAE). The action expert follows the video-transformer architecture with a hidden dimension of 1024 and approximately one billion parameters. The video and action backbones therefore contain approximately six billion parameters in total; this count excludes the task-state reader and TraceWAM's auxiliary modules.

The backbone retains video–action co-training. At inference, the clean tokens of the current observation provide the world representation for action generation, without sampling a future video. TraceWAM uses these world features to generate a local residual and predict candidate-specific task-state deviations. Its verifier predicts structured consequences directly, rather than reconstructing candidate videos.

A.2 Input and Observation Processing

At decision round t, the policy receives the instruction l, multi-view RGB observations o_t, and proprioception p_t. Camera views are concatenated in a fixed order before VAE encoding. The instruction is encoded by the pretrained text encoder and supplies cross-attention conditioning to the video and action branches. Training chunks contain nine video frames, including the initial observation, obtained by factor-four temporal subsampling over a maximum horizon of 32 control steps.

The frozen task-state reader separately maps (o_t, p_t) and the object-role bindings to the estimated task state $\tilde{x}_t$ and reliability mask m_t. The progress memory receives the backbone observation feature, proprioception, estimated state, and reliability mask. Backbone features are detached during auxiliary training, as specified in Sec. 3.3.

A.3 Action Representation and Bounded Correction

Let a chunk be $a \in \mathbb{R}^{H \times d_a}$, with maximum horizon $H = 32$ and benchmark-dependent action dimension d_a. The fixed diagonal mask P_c operates on the flattened chunk and selects continuous motion coordinates. It sets the entries corresponding to discrete gripper commands to zero. The residual and feasible candidate are

$$\tilde{r}_t = P_c \,\mathrm{clip}(r_\psi(u_t), -\eta, \eta), \qquad a_t^r = \mathrm{Proj}_{\mathcal{A}}(a_t^{\mathrm{bc}} + \tilde{r}_t). \tag{A.1}$$

Projection enforces the continuous action range and preserves the base chunk's discrete gripper sequence. Thus, the candidate can adjust the approach, transport, or placement motion while retaining the proposed opening and closing commands. These motion changes can affect contact geometry and relative pose; introducing an additional gripper opening or closing event requires a subsequent base-policy decision.

The full chunk, including its gripper commands, is encoded for consequence prediction. Both candidates share $u_t = [\mathrm{sg}(h_t^w), g_{z_t}, b_t, v_t]$; only their action encodings differ.

A.4 Inference and State Updates

Action generation uses ten denoising steps and classifier-free guidance scale 1.0. At each decision round, the reader estimates the current state, the progress module selects the active stage, and the backbone provides the base action chunk. When the reference is valid, the shared verifier evaluates the base and bounded-residual candidates. The selector accepts the residual only if its uncertainty-weighted risk satisfies both the risk-advantage and predicted-progress conditions in Eq. (9).

Chunk execution ends at the maximum horizon, reliable stage completion, or episode termination. The state-check record is updated during execution, and the next decision uses the new observations and history. An invalid reference retains the base candidate. Simulator branching is used for training supervision; online selection uses the learned verifier.

A.5 Optimization and Training Hardware

Table 4 summarizes the backbone and optimization settings specified in Sec. 4.1. Training uses three NVIDIA A100 GPUs. The backbone is trained first; it is then frozen while the progress, residual, and verifier modules are optimized using shared-state branch data. The pretrained reader remains frozen during these two stages.

B Task-State Representation and Stage References

B.1 Task-State Components and Coordinate Frames

The task state is $x = \{T^o, T^e, c, r, q\}$. The pose T^o represents an object relative to its task anchor; T^e represents the end effector relative to that object. The vectors c and r describe contact predicates and spatial relations, respectively, and q describes the gripper state. Each component may contain the entries required by the bound task objects. A fixed ordering aligns reader outputs, stage references, relevance masks, and verifier targets.

Table 4: Backbone and optimization settings. Training-step budgets refer to backbone training; auxiliary training is described in Appendix C.

Setting	Value
Video backbone	Pretrained Wan2.2-5B
Text encoder and video VAE	Associated pretrained Wan2.2 components
Action-expert hidden dimension	1024
Backbone parameter count	Approximately 5B video + 1B action
Maximum action horizon	32 control steps
Video temporal subsampling	Factor four; nine frames per chunk
Inference denoising steps	10
Classifier-free guidance scale	1.0
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	0.01
Learning-rate schedule	Cosine decay
Numerical precision	Mixed precision
Gradient clipping	1.0
LIBERO backbone training	20,000 steps
RoboTwin backbone training	30,000 steps
Training hardware	Three NVIDIA A100 GPUs

Writing T_W^A, T_W^O, and T_W^E for the world poses of the anchor, object, and end effector gives the coordinate interpretation

$$T^o = (T_W^A)^{-1} T_W^O, \qquad T^e = (T_W^O)^{-1} T_W^E. \tag{B.1}$$

This representation distinguishes movement toward a task anchor from changes in the grasp geometry. When an anchor moves, current anchor-relative coordinates change even if the desired object-to-anchor relation remains fixed.

B.2 Component Residuals and Aggregation

For a pose T and reference T^*, TraceWAM uses the scaled rigid-pose residual

$$\rho(T, T^*) = S_{SE(3)} \operatorname{Log}\left((T^*)^{-1} T\right)^{\vee}. \tag{B.2}$$

The scale matrix balances translation and rotation. Contact, spatial-relation, and gripper residuals compare the corresponding encoded entries with their reference values. The concatenated representation $\Delta_Q(x, g)$ uses the same coordinate ordering and normalization in training and evaluation.

For $j \in \mathcal{C} = \{o, e, c, r, q\}$, d_j is the normalized ℓ_1 error for that component. The aggregation rule is

$$d_Q(x, g; v) = \frac{\sum_{j \in \mathcal{C}} v_j w_j d_j(x_j, g_j)}{\sum_{j \in \mathcal{C}} v_j w_j + \epsilon} = \|\Delta_Q(x, g)\|_{Q;v}. \tag{B.3}$$

The same scales, component weights, and mask convention are applied when a predicted deviation is converted to a candidate distance. Consequently, the verifier's structured output and the current-state distance are evaluated in the same units. Predicate entries contribute only through their encoded mismatch; an inactive component contributes zero weight.

B.3 Stage Relevance and Critical Components

Each stage k has a relevance mask s_k and a nonempty critical-component set J_k. The relevance mask determines which physical quantities enter candidate comparison. The critical set identifies the quantities that must be reliable to validate the reference and must meet their tolerances to establish completion. A relevant component can contribute to comparison without being a completion prerequisite.

The observation-dependent mask is $v_{t,k} = s_k \odot m_t$. Both candidates use this same mask, preventing a candidate from appearing preferable because it was evaluated on a different subset of physical quantities. During offline drift measurement, the complete stage mask $s_{\tilde{z}_t}$ is used for both endpoints.

B.4 Stage Reference Construction

The deterministic parser $(G, \mathcal{B}, \Gamma) = \mathrm{ParseGoal}(l, o_1; \mathcal{S})$ combines the instruction, initial observation, and frozen schema library. The bindings $\mathcal{B}$ associate task roles with objects and anchors; $G = \{g_k\}_{k=1}^{K}$ contains the ordered stage references; and Γ specifies relevance masks, critical sets, and component tolerances.

Schemas supply predicate structure and stage ordering, while training demonstrations supply pose prototypes and tolerances. The resulting schema and reference parameters are fixed before evaluation. Pose goals remain defined relative to the bound anchors; new observations update the estimated current state rather than replacing the desired physical relation.

B.5 Validity, Completion, and Progress Memory

Let B_k indicate that bindings are complete and the reference is defined. Reference validity is

$$\mathrm{Valid}_{t,k} = B_k \prod_{j \in J_k} v_{t,k,j}. \tag{B.4}$$

Stage completion requires validity and $d_j(\tilde{x}_{t,j}, g_{k,j}) \le \epsilon_{k,j}$ for every critical component. A temporarily unobserved critical quantity therefore cannot establish completion.

Progress memory integrates observations and execution history through the GRU in Eq. (4). The transition mask in Eq. (5) permits staying at the current stage, advancing after L_{adv} consecutive completion checks, or rolling back by one stage after L_{rec} consecutive violations of a previously satisfied predicate that remains critical. The rollback penalty is κ. Memory is initialized with $b_0 = 0$ and $z_0 = 1$, and transitions are restricted to valid stage indices.

C Training and Shared-State Branch Construction

C.1 Shared-State Branch Generation

Each training example begins at a simulator state $\bar{x}_t$ with its observation, expert action chunk, and offline stage label $\bar{z}_t$. Candidate branches start from the same simulator state and random seed and use the same maximum horizon and stopping rule. A branch stops after H control steps, reliable stage completion, or episode termination. Its realized duration $H_t(a)$ can therefore differ across candidates even though the maximum horizon and rule are shared.

The starting reference $\bar{g}_{\bar{z}_t}$ is retained when constructing all outcomes for that example. The terminal state of a branch is compared with this reference even when execution reaches the next stage. This separates the consequence of an action from a change in the goal used to judge it.

C.2 Candidate Set

The branch set $\mathcal{A}_t$ contains the base chunk and M additional proposals drawn from the current residual, bounded random perturbations, and expert-derived corrections. For the demonstration action a_t^{demo}, the expert-derived correction is

$$\Delta a_t^{\mathrm{exp}} = P_c \,\mathrm{clip}(a_t^{\mathrm{demo}} - a_t^{\mathrm{bc}}, -\eta, \eta). \tag{C.1}$$

It is added to the base chunk and projected into $\mathcal{A}$ using the same rule as the online residual candidate. All proposals obey the same continuous-coordinate mask and action bounds. The branch set consequently evaluates alternative local motions while retaining the base chunk's discrete gripper commands.

C.3 Structured Consequence Targets

For each candidate, the supervision target is its terminal deviation from the starting stage reference:

$$\delta^*_{t,H}(a) = \Delta_Q\big(\bar{x}^a_{t,H_t(a)}, \bar{g}_{\bar{z}_t}\big). \tag{C.2}$$

Prediction and risk training use the offline reference with $v_{t,\bar{z}_t} = s_{\bar{z}_t} \odot m_t$. The structured regression error is $e_{t,a} = \|\hat{\delta}_{t,H}(a) - \delta^*_{t,H}(a)\|_{Q;v_{t,\bar{z}_t}}$. The uncertainty output is positive by construction, $\hat{\sigma}_{t,H}(a) = \sigma_{\min} + \mathrm{softplus}(s_t(a))$, and the heteroscedastic objective in Eq. (11) jointly learns deviations and their error scale.

Realized candidate preferences use the signed change $D^{H,*}_t(a)$ under the full stage mask. Since all branches share their initial state and reference, their initial distances cancel:

$$D^{H,*}_t(a) - D^{H,*}_t(a') = d_Q\big(\bar{x}^a_{t,H_t(a)}, \bar{g}_{\bar{z}_t}; s_{\bar{z}_t}\big) - d_Q\big(\bar{x}^{a'}_{t,H_t(a')}, \bar{g}_{\bar{z}_t}; s_{\bar{z}_t}\big). \tag{C.3}$$

Thus, branch preferences compare action-dependent terminal alignment under a common physical goal. A preferred branch need not produce absolute progress; that additional requirement is imposed by the online progress gate.

C.4 Residual Supervision

The best realized candidate is $a^+_t = \arg\min_{a\in\mathcal{A}_t} D^{H,*}_t(a)$. Its difference from the base chunk is used as the residual target only if the realized improvement over the base is at least γ_{res}; otherwise, the target is zero. The loss $\mathcal{L}_{\mathrm{res}} = \|P_c(\tilde{r}_t - r^*_t)\|_1$ supervises the modifiable coordinates. The minimum-improvement test avoids fitting a nonzero correction when the sampled alternatives offer no sufficient advantage over the base action.

C.5 Ranking Supervision and Selection

A pair (a^+, a^-) contributes to ranking supervision when $D^{H,*}_t(a^-) - D^{H,*}_t(a^+) > \gamma_{\mathrm{pair}}$. The hinge loss in Eq. (12) then encourages $R_t(a^+) + m_r \le R_t(a^-)$. An empty pair set contributes zero ranking loss.

At deployment, both candidates are scored as $R_t(a) = \|\hat{\delta}_{t,H}(a)\|_{Q;v_t} + \beta\hat{\sigma}_{t,H}(a)$. The residual must improve over the base by m_g and over the current distance by m_p. The two comparisons have distinct roles: relative advantage chooses between the alternatives, whereas predicted progress rejects a residual that is preferable to the base but still insufficiently aligned with the current stage goal.

C.6 Two-Stage Optimization

The backbone is first trained with the video–action flow-matching objective. The second stage freezes the backbone and optimizes the progress, residual, and verifier modules using the branch data and auxiliary terms of Eq. (13). Since backbone parameters are frozen in this stage, $\mathcal{L}_{\mathrm{base}}$ does not update them.

Progress supervision applies cross-entropy to the unmasked stage logits. Previous-stage inputs for masked stage selection initially use the offline labels, followed by scheduled sampling. The frozen reader supplies the estimated states and reliability masks used by the auxiliary modules; the parser remains deterministic. The second stage therefore learns stage tracking, candidate proposals, and structured consequence evaluation while retaining the trained backbone.

D Additional Experimental Protocols

D.1 LIBERO

We evaluate on LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Each suite contains ten tasks with 50 demonstrations per task, yielding 500 demonstrations per suite and 2,000 across the four suites. Backbone training runs for 20,000 steps. Each task is evaluated in 50 closed-loop episodes with distinct environment seeds, giving 2,000 evaluation episodes in total. A suite score is the mean task success rate, and the overall score is the unweighted mean of the four suite scores.

D.2 RoboTwin 2.0

A single multi-task policy is trained on 50 bimanual tasks using 2,500 demonstrations from clean scenes and 25,000 demonstrations from strongly randomized scenes. Backbone training runs for 30,000 steps. Each task has 100 evaluation episodes in each of the clean and randomized settings: 5,000 episodes per setting and 10,000 across the two settings. We first average task success rates within each setting and then average the clean and randomized scores. Methods and ablations share task sets, evaluation budgets, and environment seeds.

D.3 RMBench

RMBench contains five $M(1)$ tasks, which require a critical historical state, and four $M(n)$ tasks, which require accumulated feedback, repeated exploration, or trial-and-error. Each task supplies 50 synthetic expert demonstrations and is evaluated over 100 episodes, giving 450 training trajectories and 900 evaluation episodes.

Let S_i denote the success rate of task i. The overall score averages all nine tasks and can equivalently be computed from the group means:

$$S_{\text{overall}} = \frac{1}{9}\sum_{i=1}^{9} S_i = \frac{5S_{M(1)} + 4S_{M(n)}}{9}. \tag{D.1}$$

Scores are aggregated before rounding. The two group means receive weights corresponding to their task counts.

D.4 Matched-Context Evaluation

The comparison in main-text Table 2 isolates the candidate evaluator within the shared branch-trained system. The branch-residual, scalar-verifier, and structured-verifier variants share the backbone, frozen state reader, stage references, progress memory, residual generator, and branch data. The scalar and structured verifiers receive matched training budgets and use the same form of the acceptance rule.

For scalar verification, the supervised consequence is the terminal aggregate distance

$$y^*_{t,H}(a) = d_Q\big(\bar{x}^a_{t,H_t(a)}, \bar{g}_{\bar{z}_t}; v_{t,\bar{z}_t}\big), \tag{D.2}$$

predicted directly. Structured verification instead predicts $\delta^*_{t,H}(a)$ before applying the same distance aggregation. Both formulations evaluate the candidate against the active stage reference. The direct branch-residual variant executes its proposal without this candidate-verification decision. The demonstration-residual variant uses demonstration-based residual supervision and provides a separate comparison for the supervision source.

Holding these components fixed makes the scalar-to-structured comparison specific to the consequence representation and its use in candidate evaluation.

E Additional Results and Diagnostics

E.1 Chunkwise Deterioration and Trajectory Aggregation

The nTAD and DriftAUC metrics are defined in Sec. 4.2. Diagnostics use the simulator state after actual execution and the offline stage label at the start of each chunk. Each chunk uses its starting reference and full stage mask at both endpoints, so a stage transition does not change the goal used to measure that chunk's deterioration. The normalization floors prevent division by a vanishing initial distance. Positive clipping prevents later improvement from cancelling earlier deterioration, and DriftAUC gives larger weight to earlier deterioration.

The trajectory-level DriftAUC values in Fig. 3(c) are 15.01×10^{-3} for the baseline method and 9.35×10^{-3} for TraceWAM, corresponding to a relative reduction of approximately 37.7%.

F REAL-WORLD SETUP

F.1 PLATFORM, OBSERVATION, AND TASK

The real-world demonstration uses a single AgileX Robotics PiPER six-axis manipulator equipped with a gripper, together with visual observations of the workspace and robot proprioception.

The task is to grasp a red block and place it into a bowl. The bowl serves as the target anchor, and its position is externally changed during transport. The reader updates the block–bowl relation from the current observations while the desired placement relation remains defined relative to the bowl.

F.2 EXECUTION PROCEDURE

Figure 4 records the following sequence. The robot first approaches and grasps the red block. During transport, the bowl is displaced from its original location. The base candidate remains directed toward the previous target position, creating a mismatch with the current anchor-relative state. TraceWAM evaluates the base and bounded-residual candidates against the active reference, accepts a correction toward the displaced bowl, and completes placement and release.

The displacement changes the observed target geometry. The resulting correction is a continuous motion adjustment selected by the same validity, risk, and progress conditions as in simulation. The residual preserves the gripper commands in the base chunk, consistent with Appendix A.