Stage-Verifiable Latent Chain-of-Thought with Local Repair for Long-Horizon Robotic Manipulation

Anonymous authors
Paper under double-blind review

Abstract

Long-horizon robotic manipulation requires intermediate reasoning to remain consistent with observed task progress. Continuous latent plans, however, often lack stage semantics that can be checked during execution. We present a vision-language-action framework that aligns latent segments with demonstration-derived object relations and subgoal completion, verifies their expected outcomes against observations, and repairs the affected suffix after execution deviations. The framework retains a verified prefix while its still-required conditions remain valid and conditions regeneration on the current scene and task history. All model calls, including verification and repair, are included in a task-level inference budget. The method achieves measured mean success rates of 98.0% on LIBERO overall and 96.8% on LIBERO-Long, and 58.0%/23.0% under clean/randomized conditions on 50 RoboTwin 2.0 tasks. On LIBERO-RECOVER, enabling repair increases macro-averaged recovery success from 22.8% to 26.0%. Relative to full replanning, local repair reduces cumulative inference time from 18.6 to 14.9 seconds per episode, a 19.9% reduction, with recovery success of 26.0% versus 26.4%. These results characterize the recovery–computation trade-off under the evaluated conditions.

1 Introduction

Long-horizon robotic manipulation requires a robot to maintain consistent task progress across interdependent operations. Opening a drawer, retrieving an object, and placing it in a container involves subgoals whose outcomes provide prerequisites for subsequent decisions. A missed grasp or displaced object can invalidate these prerequisites even when the task instruction remains unchanged. If later actions continue to assume that the intended outcome was achieved, a local execution error can propagate through the remaining task. Reliable continuation therefore requires intermediate reasoning to be checked against the state reached through execution.

The robot must distinguish intended transitions from observed outcomes and identify which reasoning remains applicable when they diverge. Reusing the entire plan can preserve invalid assumptions, whereas regenerating it after every deviation can discard useful computation. Effective reuse requires identifying completed progress whose consequences still support subsequent operations and reassessing those consequences as physical interactions change the scene, even when the language goal is unchanged.

Vision-language-action (VLA) models use intermediate representations to connect task understanding with control. Continuous latent reasoning offers a compact means of organizing this information without generating a complete textual plan at each decision. Recent work investigates latent planning, physical-state supervision, multimodal alignment, and adaptive reasoning depth (Huang et al., 2026; Liu et al., 2026b; Zhong et al., 2026; Lei & Shan, 2026). Predicting a plausible future state, however, does not establish that the corresponding subgoal has been achieved. Closed-loop execution also requires an observable interpretation of intermediate stages and a mechanism for revising the reasoning affected by a deviation. These requirements concern the relationship between a plan and its execution, beyond the quality of a prediction alone.

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.

Execution monitors and task-graph systems address failure diagnosis, recovery branches, and plan revision (Ying et al., 2026; Xu et al., 2026; Gu et al., 2026). Progress reuse and local recovery are established ideas. We study how observation-based verification selects continuous latent segments for reuse or regeneration within a VLA model. Representation learning, progress assessment, and repair-scope selection must share task-stage semantics, so retained context reflects valid progress rather than assumptions from an earlier plan.

Object relations and subgoal events in demonstrations provide a basis for this interpretation. An object being held or entering a target container describes a state change that can be assessed from observations; completion signals relate that change to the instruction. Aligning latent segments with these events associates intermediate representations with explicit expected outcomes, without requiring the full reasoning process to be expressed in text. The resulting interface compares anticipated progress with execution evidence. It does not treat the plausibility of a latent prediction as confirmation that the associated physical operation succeeded.

We propose a framework that aligns latent segments with demonstration-derived object relations and subgoal completion. A latent-state decoder predicts each stage's expected outcome, and an observation-based estimator assesses the executed state. Consistently supported completion advances the active stage; reliable, persistent deviations trigger regeneration from the earliest affected stage. A verified prefix is retained only while its still-required conditions remain valid. For example, a failed grasp need not invalidate an already opened drawer, whereas loss of an earlier prerequisite can require a broader repair. The regenerated suffix is conditioned on the current scene and valid history, allowing control to continue from the actual physical state.

Stage alignment makes expected outcomes available for checking, while feedback-driven repair updates the representations conditioning subsequent control. Repair depends on a meaningful correspondence between latent stages and observations; together, these components make regeneration responsive to execution. We measure complete-episode inference cost alongside recovery success to determine whether reuse saves computation after accounting for verification and repair.

Our evaluation uses LIBERO for language-conditioned manipulation, RoboTwin 2.0 for clean and randomized dual-arm tasks, and LIBERO-RECOVER for continuation from failure states. Controlled ablations distinguish additional state supervision, latent-stage alignment, and the use of verification to trigger repair. A separate comparison varies reconstruction scope while accounting for all model calls. Our contributions are a stage-supervised latent representation with an observation-based verification interface, a feedback-driven suffix update linked to valid task progress, and an empirical analysis of task performance, recovery, and inference cost. The method achieves 98.0% average success on LIBERO and 96.8% on Long tasks. Local repair reduces cumulative inference time by 19.9% relative to full replanning, with recovery success of 26.0% versus 26.4%, demonstrating the importance of assessing recovery and cost jointly.

2 Related Work

Reasoning representations for manipulation. Structured intermediate representations connect high-level task intent with executable behavior. Embodied-R1 uses spatial pointing as an embodiment-agnostic interface; Action-Sketcher expresses spatial intentions through editable visual sketches; and TCoT combines global and local trajectory reasoning with failure recovery (Yuan et al., 2026; Tan et al., 2026; Li et al., 2026). These approaches demonstrate that intermediate reasoning can support both action selection and execution adjustment. Our framework represents reasoning as continuous latent segments aligned with object relations and subgoal completion. This alignment gives the representations used for control an observable interpretation through which expected and executed progress can be compared. Spatial intentions specify desired behavior, whereas stage outcomes additionally provide conditions for assessing its result; these roles are complementary.

Latent planning and adaptive computation. Fast-ThinkAct distills language and visual planning into compact latent representations, while LaRA-VLA transitions from explicit text-visual supervision to latent reasoning for action generation (Huang et al., 2026; Bai et al., 2026). LaST_0 incorporates physical information through spatiotemporal latent modeling, and DualCoT-VLA combines visual and linguistic supervision through parallel reasoning (Liu et al., 2026b; Zhong et al., 2026). Adaptive early exit adjusts reasoning depth to the demands of individual decisions (Lei & Shan,

2026). These methods advance intermediate computation. We study reuse and revision across execution steps, accounting for verification overhead. Stage verification relates retained latent context to observed progress and subsequent prerequisites, determining which representations remain applicable afterward.

Execution verification and recovery. RoboFailRing uses retrieved information to support failure diagnosis, AgentChord prepares recovery branches within a task graph, and HarnessWAM maintains task and scene context for verification and plan revision (Ying et al., 2026; Xu et al., 2026; Gu et al., 2026). These approaches establish the value of execution evidence for preserving progress and adjusting subsequent behavior. We investigate how this evidence selects a latent suffix within the VLA for reconstruction. LIBERO-RECOVER complements standard task-success evaluation with continuation from failure states at multiple recovery levels (Liu et al., 2026a). Our controlled experiments examine stage supervision, semantic alignment, and repair scope. Published results provide comparison context; they do not constitute a controlled ranking against every monitoring or task-graph system.

3 Method

3.1 Problem Formulation and Overview

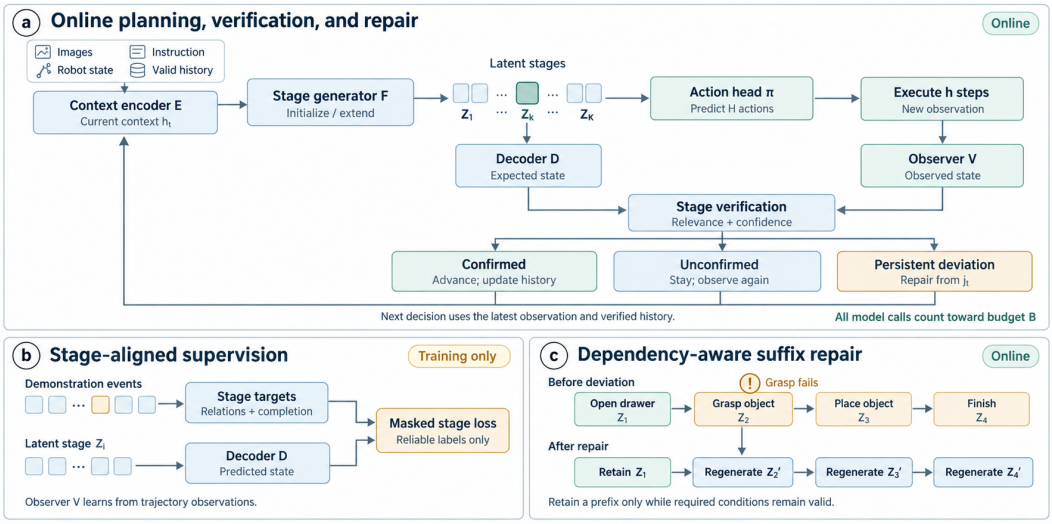

Figure 1: **Stage-verifiable latent reasoning and local repair.** **(a)** Context encoding and stage generation condition action chunks: the policy predicts H actions and executes h before observing again. Verification compares decoded endpoints with observation-based evidence. Confirmed completion advances the stage; insufficient evidence keeps it unconfirmed; persistent deviations trigger repair. **(b)** Demonstration-derived relations and subgoal completion supervise decoded stage targets through reliable-label masks; the observer learns from trajectory observations. **(c)** Repair regenerates the suffix from the earliest affected stage j_t using the current context, retaining a prefix only while its still-required conditions remain valid. A failed grasp therefore preserves the opening stage only if the drawer remains accessible. All model calls count toward the episode budget B.

Given a language instruction g, the robot observes $o_t = (I_t, \mathbf{q}_t)$ at environment step t, where I_t denotes images and $\mathbf{q}_t$ denotes proprioception. The history summary M_t records verified stages, relevant object states, and execution outcomes. A latent plan is an ordered sequence $\mathcal{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_K)$, with $\mathbf{Z}_i \in \mathbb{R}^{m \times d}$ containing m vectors of hidden dimension d. Let $\mathcal{Z}_{a:b}$ denote a subsequence and $\mathcal{Z}_{<i}$ its prefix before stage i. The encoder forms the current context:

$$\mathbf{h}_t = E(g, o_t, M_t). \tag{1}$$

The stage generator constructs the plan for $i = 1, \ldots, K$:

$$\mathbf{Z}_i = F_\theta(\mathbf{h}_t, \mathcal{Z}_{<i}). \tag{2}$$

For active stage k, the action head predicts an H-step chunk:

$$\widehat{\mathbf{A}}_t = \pi_\phi(o_t, g, M_t, \mathcal{Z}_{k:K}). \tag{3}$$

The chunk $\widehat{\mathbf{A}}_t \in \mathbb{R}^{H \times d_a}$ covers steps $t, \ldots, t + H - 1$, where d_a is the action dimension. The robot executes $1 \leq h \leq H$ actions before observing again. Execution uses images, language, and proprioception; simulator states serve only offline annotation and independent evaluation.

A stage can span multiple action chunks. Training stage counts follow demonstration events, with padding masked out. At inference, the model predicts stages and their termination within an extendable planning window. Finishing a window does not certify physical task completion. Figure 1 summarizes the loop; Algorithm 1 in Appendix A gives the pseudocode.

3.2 Stage Alignment and Training

Semantic stage targets. Demonstration boundaries correspond to stable, task-relevant events, including grasp establishment, placement, containment, and container opening. At boundary τ_i, object relations $\mathbf{r}^\star_{\tau_i}$ and subgoal-completion signals $\mathbf{c}^\star_{\tau_i}$ define the stage target

$$\mathbf{s}^\star_i = \begin{bmatrix} \mathbf{r}^\star_{\tau_i} \\ \mathbf{c}^\star_{\tau_i} \end{bmatrix}, \tag{4}$$

Here $\mathbf{s}^\star_i \in \mathbb{R}^P$, with P state components. Relations describe physical changes; completion signals establish their relevance to the instruction. Containment, for example, satisfies a placement subgoal only for the instructed container. The mask $\mu_{i,p} \in \{0, 1\}$ marks reliable labels for component p of stage i; uncertain components are excluded, not labeled negative.

Predicted and observed states. A decoder maps each latent segment to its expected endpoint:

$$\widehat{\mathbf{s}}_i = D_\psi(\mathbf{Z}_i). \tag{5}$$

An observation estimator independently derives execution evidence from $\mathcal{O}_t = o_{\max(0,t-W+1):t}$, a window of at most $W \geq 1$ observations:

$$\widetilde{\mathbf{s}}_t = V_\eta(\mathcal{O}_t, g). \tag{6}$$

Both estimates belong to $\mathbb{R}^P$. The observation estimator trains on states before and after completion; temporal input supports grasp and release recognition. Shared visual features do not make a latent prediction evidence that its expected outcome occurred.

The stage-alignment loss compares decoded endpoints with the available target components:

$$\mathcal{L}_{\text{stage}} = \sum_{i=1}^{K} \frac{\sum_{p=1}^{P} \mu_{i,p}\, \ell_{\text{state}}(\hat{s}_{i,p}, s^\star_{i,p})}{\sum_{p=1}^{P} \mu_{i,p} + \varepsilon}. \tag{7}$$

Here $\hat{s}_{i,p}$ denotes component p, ℓ_{state} is its loss, and $\varepsilon > 0$ stabilizes the denominator. Each stage is normalized over labeled components; fully masked stages contribute zero. The joint objective is

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda_{\text{stage}}\mathcal{L}_{\text{stage}} + \lambda_{\text{obs}}\mathcal{L}_{\text{obs}}. \tag{8}$$

The weights are nonnegative. Action loss uses ℓ_1 regression; discrete predicates use masked binary cross-entropy. Observation loss supervises the estimator on trajectory observations. Stage loss trains

the decoder and control representations, while action loss connects them to demonstrated behavior. Conditional suffix training uses these same objectives. "Verifiable" denotes comparison with observed task states, not a formal correctness guarantee.

3.3 Observation-Based Stage Verification

Verification focuses on candidate stage endings, persistent stagnation, and violations of conditions required by subsequent operations. For active stage k, the discrepancy between its expected endpoint and the observed state is

$$e_{t,k} = \frac{\sum_{p=1}^{P} \omega_{t,k,p}\, \ell_{\text{ver}}(\hat{s}_{k,p}, \tilde{s}_{t,p})}{\sum_{p=1}^{P} \omega_{t,k,p} + \varepsilon}, \tag{9}$$

where ℓ_{ver} is a nonnegative component-wise discrepancy. Nonnegative weights $\omega_{t,k,p}$ combine task relevance and observation confidence, whereas training masks $\mu_{i,p}$ identify reliable labels.

Endpoint discrepancies during normal motion do not alone justify repair. Completion requires consecutive observations supporting the relevant conditions and subgoal; reliable persistent contradictions trigger repair. Insufficient evidence leaves the stage **unconfirmed** until later observations resolve it. Thus, a small discrepancy with zero or negligible observation weight cannot certify completion: the regularizer supplies numerical stability, not evidence. Missing information is not treated as a confirmed execution failure.

3.4 Dependency-Aware Local Repair

After a repair trigger is confirmed, the model identifies the earliest stage requiring revision. Let $\mathcal{A}_t \subseteq \{1, \ldots, k-1\}$ contain earlier stages whose established conditions are reliably contradicted and remain necessary for the unfinished task. The repair starts at

$$j_t = \min\big(\{k\} \cup \mathcal{A}_t\big). \tag{10}$$

Using the latest context from Equation (1), the same stage generator as Equation (2) regenerates stages $i = j_t, \ldots, K'$:

$$\mathbf{Z}'_i = F_\theta\big(\mathbf{h}_t, \mathcal{Z}_{<j_t} \oplus \mathcal{Z}'_{j_t:i-1}\big). \tag{11}$$

The updated plan joins the retained prefix and regenerated suffix:

$$\mathcal{Z}^{\text{new}} = \mathcal{Z}_{<j_t} \oplus \mathcal{Z}'_{j_t:K'}. \tag{12}$$

The operator $\oplus$ concatenates stage sequences, K' is the terminal stage of the repaired plan, and an empty index interval denotes an empty sequence. If $\mathcal{A}_t$ is empty, repair starts at k; otherwise, it extends to the earliest affected prerequisite. The retained prefix is checked against conditions still needed by future execution, rather than requiring every historical relation to persist. Releasing an object after successful placement, for example, does not invalidate the completed grasp.

Stage outcomes define prerequisites for later operations. A failed grasp preserves an opening stage if the drawer remains accessible; otherwise, repair revisits that prerequisite and its dependents. Retention preserves valid history without repeating completed actions. Regeneration starts from the observed state, with suffix length adapted to the remaining task and recovery requirements.

Conditional suffix training samples demonstration stage boundaries. The aligned prefix, boundary observation, instruction, and history condition reconstruction of remaining stages and actions. The separately evaluated recovery-data augmentation applies the same construction to recovery trajectories.

3.5 Budgeted Closed-Loop Execution

All model operations contribute to the episode-level inference budget. With N_{rep} triggered repairs, the cumulative cost is

$$C_{\text{total}} = C_{\text{enc}} + C_{\text{init}} + C_{\text{roll}} + C_{\text{ver}} + C_{\text{act}} + \sum_{n=1}^{N_{\text{rep}}} C_{\text{rep}}^{(n)}. \tag{13}$$

The budget requires $C_{\text{total}} \leq B$. The nonnegative terms cover shared encoding, initial planning, rolling extensions, verification, action decoding, and repair, in the units of B. Encoding is charged once and excluded downstream; the repair sum is zero without repairs. The same accounting applies to successful and failed episodes.

Remaining budget constrains calls and regeneration length. Repair prioritizes the current stage and direct dependents; more distant planning extends during execution. Reconstruction need not cover the full task horizon. We measure the complete loop, including verification and action computation, because regenerated-vector counts alone do not determine inference time.

4 Experiments

4.1 Benchmarks and Data

We evaluate LIBERO-Spatial, Object, Goal, and Long (LIBERO-10), each containing ten tasks (Liu et al., 2023); RoboTwin 2.0 under clean and randomized conditions (Chen et al., 2025); and LIBERO-RECOVER at four recovery levels (Liu et al., 2026a). RoboTwin's 50-task and Fast-ThinkAct ten-task protocols are reported separately. Recovery averages use Spatial, Object, Goal, and LIBERO-100, rather than Long.

Demonstration replay maps stable object-relation changes and subgoal events to predicates, with unreliable labels masked. Splits preserve complete trajectories; recovery samples are grouped by failure scenario and source trajectory. The default uses demonstrations and stage annotations. Additional recovery data and dedicated initial-frame input are evaluated independently against this configuration.

4.2 Model and Controlled Comparisons

The policy fine-tunes OpenVLA using OpenVLA-OFT continuous action chunks and ℓ_1 loss (Kim et al., 2024; 2025). Each stage has four latent vectors; windows contain up to eight stages and extend during execution. The model predicts $H = 8$ actions and executes $h = 4$. Each benchmark is fine-tuned separately, with normalization fitted only on its training data. Optimization settings appear in Appendix A.

Ablations match initialization, task splits, training trajectories, observations, and execution intervals. Retrained configurations use three independent seeds, with initial states and failure scenarios paired for evaluation. Repair strategies reuse weights, the observation estimator, and triggers where possible. Table 4 uses our controlled base-policy implementation; Table 1 gives published OpenVLA-OFT scores as an external reference.

4.3 Metrics and Reporting

Task success averages per-task success rates, with completion determined by environment goals within the evaluation limits. Stage-verification F_1 is measured on a fixed diagnostic observation set using environment-derived labels. Recovery evaluation includes all predefined failure instances: undetected failures, unsuccessful repairs, and budget exhaustion count as failures.

Let $n_{s,\ell} > 0$ and $u_{s,\ell}$ denote the numbers of evaluated and successful instances for suite s and level ℓ, respectively. With $S = 4$ suites and $L = 4$ levels, Table 3 reports the per-level macro-average:

$$R_\ell = \frac{1}{S}\sum_{s=1}^{S}\frac{u_{s,\ell}}{n_{s,\ell}}. \tag{14}$$

Tables 4–6 report the overall suite macro-average:

$$R_{\text{overall}} = \frac{1}{S}\sum_{s=1}^{S}\frac{\sum_{\ell=1}^{L} u_{s,\ell}}{\sum_{\ell=1}^{L} n_{s,\ell}}. \tag{15}$$

Neither rate pools suites into one denominator; averaging R_ℓ generally differs from R_{overall}. Equations define proportions; tables show percentages. Appendix B defines additional metrics. Success and verification report three-seed measured means without uncertainty estimates or significance claims. Timing reports episode-level mean cumulative inference time, p95 decision latency, and mean regenerated vectors; efficiency conclusions apply to this setting.

4.4 Experimental Results and Comparisons

Tables 1–3 report our measured means and published results. Their differing data, inputs, and budgets preclude causal cross-paper rankings; Section 4.5 provides controlled component ablations.

Table 1: Task success on LIBERO (%): published results and our measured means.

Method	Source	Spatial	Object	Goal	Long	Average
Diffusion Policy	A	78.3	92.5	68.3	50.5	72.4
Octo	A	78.9	85.7	84.6	51.1	75.1
DiT Policy	A	84.2	96.3	85.4	63.8	82.4
OpenVLA	A	84.7	88.4	79.2	53.7	76.5
π_0-FAST	A	96.4	96.8	88.6	60.2	85.5
π_0	A	96.8	98.8	95.8	85.2	94.2
OpenVLA-OFT	A	97.6	98.4	97.9	94.5	97.1
CoT-VLA	B	87.5	91.6	87.6	69.0	83.9
ThinkAct	B	88.3	91.4	87.1	70.9	84.4
MolmoAct	B	87.0	95.4	87.6	77.2	86.8
Fast-ThinkAct	B	92.0	97.2	90.2	79.4	89.7
LaRA-VLA	C	96.4	99.8	98.6	96.6	97.9
Ours	This work	98.0	98.8	98.4	96.8	98.0

A: Kim et al. (2025); B: Huang et al. (2026); C: Bai et al. (2026). Published averages retain their source definitions; our average equally weights the four suites. Cross-paper settings differ.

LIBERO. Overall/Long success is 98.0%/96.8%, compared with published scores of 97.9%/96.6% for LaRA-VLA and 97.1%/94.5% for OpenVLA-OFT. The 0.1-point overall difference from LaRA-VLA does not establish statistical significance. Our Object/Goal scores, 98.8%/98.4%, remain below LaRA-VLA's 99.8%/98.6%, indicating variation across suites despite strong aggregate performance.

RoboTwin 2.0. On 50 tasks, Easy/Hard success is 58.0%/23.0%, exceeding the highest listed published scores by 2.8/6.7 percentage points. On the ten-task subset, success is 68.0%/31.0%, or 2.3/4.6 points above Fast-ThinkAct. These averages refer to different task sets. The substantial degradation under randomization remains a limitation. Moreover, DP3 uses point clouds rather than image-only inputs, so the ranking does not isolate the contribution of reasoning.

Failure recovery. The L1–L4 macro-averages are 52.0%, 32.0%, 20.0%, and 9.0%. The L4 score exceeds the highest listed published reference, 5.6%, by 3.4 percentage points, but remains low in absolute terms. Recovery depends on action capability, observation quality, and failure localization as well as verification. Detecting a discrepancy does not by itself provide the actions needed to resolve it.

Inference cost. Table 5 includes encoding, verification, action decoding, and all planning and repair calls. Local repair uses 14.9 seconds per episode versus 18.6 for full replanning (19.9% less), with

Table 2a: RoboTwin 2.0: 50 tasks (%).

Method	Easy	Hard
DP	28.0	0.6
ACT	29.7	1.7
RDT	34.5	13.7
π_0	46.4	16.3
DP3	55.2	5.0
Ours	58.0	23.0

Published baselines: Chen et al. (2025). The protocol uses 50 clean demonstrations per task and 100 tests per condition.

Table 2b: RoboTwin 2.0: 10-task subset (%).

Method	Easy	Hard
DP	43.1	0.6
ACT	45.5	3.5
π_0	52.9	16.3
RDT	56.4	22.8
ThinkAct	62.4	24.7
Fast-ThinkAct	65.7	26.4
Ours	68.0	31.0

Published baselines: Huang et al. (2026). Task coverage differs from Table 2a; the two averages are not directly comparable.

Table 3: LIBERO-RECOVER: macro-average success at each recovery level (%).

Method	L1	L2	L3	L4
π_0	46.1	24.4	13.5	4.6
π_0-FAST	40.0	22.9	16.1	3.7
GR00T-N1.5	30.7	25.3	15.2	1.5
OpenVLA-OFT	47.7	26.6	14.2	5.6
Wan2-Policy	17.8	16.2	12.5	3.9
Cosmos-Predict2-Policy	15.5	14.5	9.9	0.0
Ours	52.0	32.0	20.0	9.0

Published values: Liu et al. (2026a). Each level equally weights Spatial, Object, Goal, and LIBERO-100. L1–L4: action retry, action adaptation, object-state recovery, and environment-state recovery.

recovery of 26.0% versus 26.4%. This measured trade-off applies to the evaluated setting and does not establish a recovery advantage across budgets. Appendix C separately discusses published Fast-ThinkAct latencies.

4.5 Ablations and Analysis

Stage alignment and supervision. The auxiliary-supervision baseline uses the same state labels without constructing a staged latent chain. The no-alignment variant retains the chain, decoder, and observation estimator, but blocks stage-supervision gradients into the latent representations and upstream encoder while continuing to train the decoder. The repair-disabled variant retains observation inputs and closed-loop action updates but prohibits deviation-triggered latent reconstruction.

Table 4: Component ablations: task success, recovery, and verification F_1 (%).

Configuration	LIBERO Avg.	LIBERO Long	Recovery	Stage F1
Base policy: no latent reasoning	96.8	94.2	20.5	N/A
Auxiliary state supervision only	97.1	94.8	21.3	N/A
Without latent stage alignment	97.2	95.0	22.1	81.6
Execution-feedback repair disabled	97.5	95.8	22.8	92.4
Complete method	98.0	96.8	26.0	92.4

All rows report measured means from our controlled experiments. Recovery uses the suite macro-average of Table 6. N/A indicates that a latent-stage verification interface is not defined. The base-policy implementation is distinct from the published OpenVLA-OFT row in Table 1.

The complete method improves average success, Long success, and recovery over the base policy by 1.2, 2.6, and 5.5 percentage points. Auxiliary supervision alone yields gains of 0.3, 0.6, and 0.8 points, leaving the complete method 0.9, 2.0, and 4.7 points higher. Removing alignment reduces verification F_1 from 92.4% to 81.6% and recovery from 26.0% to 22.1%. Disabling repair reduces recovery to 22.8%. Its unchanged 92.4% F_1 follows from evaluating shared verifier weights on the

same diagnostic observations, distinguishing verification accuracy from the use of verification during execution.

Repair scope. Given the same observation, instruction, and valid history, full replanning regenerates the remaining latent plan without repeating completed actions. Current-stage-only reconstruction starts at the active stage and cannot revisit earlier prerequisites; dependency-aware repair checks those prerequisites before selecting the suffix.

Table 5: Repair strategies: recovery success and episode-level inference cost.

Repair strategy	Recovery (%)	Time (s/episode)	p95 latency (ms)	Vectors (/episode)
Repair disabled	22.8	11.8	180	0
Full replanning	26.4	18.6	820	62.4
Current-stage-only suffix reconstruction	24.6	14.2	360	23.6
Dependency-aware local repair	26.0	14.9	430	31.2

All strategies use the same recovery instances, step limits, and task-level budget. Time includes all model calls and both successful and failed episodes; shared encoding is counted once. Vector counts include triggered repairs but exclude ordinary rolling extensions. All entries are means except p95 latency.

Relative to full replanning, local repair reduces cumulative inference time by 19.9%, p95 latency from 820 to 430 ms (47.6%), and regenerated vectors from 62.4 to 31.2 (50.0%), with recovery lower by 0.4 percentage points. Current-stage-only repair is 0.7 seconds faster but 1.4 points lower in recovery than dependency-aware repair. Enabling local repair increases recovery by 3.2 points over disabling it, at an additional 3.1 seconds per episode. These results characterize the trade-off among repair scopes; aggregate timing does not isolate prefix-reuse savings from effects associated with episode termination.

Recovery data and observation history. Table 6 evaluates augmentations independently. Recovery trajectories augment training data; initial-frame input changes observations while preserving the trajectory set. Additional encoding counts toward inference cost.

Table 6: Recovery data and initial-frame input: suite macro-average success (%).

Method	Original	Recovery data	Initial frame
OpenVLA-OFT	20.8	25.4	26.8
GR00T-N1.5	17.8	21.2	18.9
Ours	26.0	31.0	28.0

Published baselines: Liu et al. (2026a). Each augmentation is evaluated independently against the original configuration. Recovery instances are pooled within each suite before averaging the four suite rates. Averaging the level columns in Table 3 yields a different statistic.

Recovery increases from 26.0% to 31.0% with recovery data and to 28.0% with initial-frame input. These independent gains of 5.0 and 2.0 percentage points cannot be added to infer the effect of combining the augmentations. Equation (15) explains why the 26.0% overall recovery score need not equal the arithmetic mean of the four level scores in Table 3.

5 Conclusion

We presented a VLA framework that integrates stage-supervised latent reasoning, observation-based verification, and selective suffix repair. The method achieves 98.0% average LIBERO success, 96.8% on Long tasks, and 26.0% macro-averaged recovery success. Relative to full replanning, local repair reduces cumulative inference time by 19.9%, with recovery lower by 0.4 percentage points. Controlled ablations support stage alignment and execution feedback under the evaluated conditions. Limitations include weak environment-state recovery, reliance on simulation-derived annotations, and evaluation at the reported budget setting. Future work includes real-robot transfer, weaker stage supervision, and evaluation of recovery–cost trade-offs across budgets.

Reproducibility Statement

Section 3 and Algorithm 1 specify the interfaces for latent reasoning, verification, and repair. Section 4.3 and Appendix B define the evaluation metrics and aggregation procedures. Appendix A documents the recorded training and annotation settings. Tables 4 and 5 describe the controlled comparisons; published reference scores retain the scope of their original studies.

AI Use Statement

OpenAI Codex assisted with manuscript editing and polishing, including Chinese–English translation, organization, mathematical notation, clarification of reported results, reference lookup and checking, and LaTeX formatting. Generative AI also refined the layout of the author-provided method figure. The method and experimental measurements were supplied by the authors; this assistance did not include conducting experiments or inspecting their implementation. The authors retain responsibility for the final manuscript.

References

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent Reasoning VLA: Latent Thinking and Prediction for Vision-Language-Action Models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.01166v2.

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. *arXiv preprint arXiv:2506.18088v2*, 2025. doi: 10.48550/arXiv.2506.18088. URL https://arxiv.org/abs/2506.18088v2.

Zhaopeng Gu, Bingke Zhu, Tianxi Lin, Guibo Zhu, Yingying Chen, Kai Wang, Tingyu Yuan, Chaoyang Zhao, Zhaowen Li, Peng Su, and Jinqiao Wang. HarnessWAM: Bridging Prediction and Deliberation in World Action Models. *arXiv preprint arXiv:2608.09516*, 2026. doi: 10.48550/arXiv.2608.09516. URL https://arxiv.org/abs/2608.09516.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, April 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chi-Pin Huang, Yunze Man, Zhiding Yu, Min-Hung Chen, Jan Kautz, Yu-Chiang Frank Wang, and Fu-En Yang. Fast-ThinkAct: Efficient Vision-Language-Action Reasoning via Verbalizable Latent Planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5070–5081, June 2026. URL https://arxiv.org/abs/2601.09708v2.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024. doi: 10.48550/arXiv.2406.09246. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.017. URL https://arxiv.org/abs/2502.19645v2.

Dianqiao Lei and Lianlei Shan. Think Less, Act Early: Reinforced Latent Reasoning with Early Exit in Vision-Language-Action Models. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.15099.

Xiang Li, Ya-Li Li, Yuan Wang, Huaqiang Wang, and Shengjin Wang. TCoT: Trajectory Chain-of-Thoughts for Robotic Manipulation with Failure Recovery in Vision-Language-Action Model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(8):6486–6494, 2026. doi: 10.1609/aaai.v40i8.37577. URL https://ojs.aaai.org/index.php/AAAI/article/view/37577.

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. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://arxiv.org/abs/2306.03310.

Lin Liu, Zhicheng Bao, Lu Zhang, Ziying Song, Wu Yang, Yuzheng Zhuang, Shuai Tao, Wulong Liu, Caiyan Jia, and Huchuan Lu. LIBERO-RECOVER: Beyond Task Success Towards Failure Recovery in Robotic Manipulation Models. *arXiv preprint arXiv:2609.05178v2*, 2026a. doi: 10.48550/arXiv.2609.05178. URL https://arxiv.org/abs/2609.05178v2.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, Zhengping Che, Jian Tang, Pheng-Ann Heng, and Shanghang Zhang. LaST_0: Latent Spatio-Temporal Chain-of-Thought for Robotic Vision-Language-Action Model. *arXiv preprint arXiv:2601.05248v4*, 2026b. doi: 10.48550/arXiv.2601.05248. URL https://arxiv.org/abs/2601.05248v4.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Huajie Tan, Peterson Co, Yijie Xu, Shanyu Rong, Yuheng Ji, Cheng Chi, Xiansheng Chen, Qiongyu Zhang, Zhongxia Zhao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Action-Sketcher: From Reasoning to Action via Visual Sketches for Long-Horizon Robotic Manipulation. *arXiv preprint arXiv:2601.01618v1*, 2026. doi: 10.48550/arXiv.2601.01618. URL https://arxiv.org/abs/2601.01618v1.

Sheng Xu, Ruixing Jin, Huayi Zhou, Bo Yue, Guanren Qiao, Yunxin Tai, Yueci Deng, Kui Jia, and Guiliang Liu. From Reaction to Anticipation: Proactive Failure Recovery through Agentic Task Graph for Robotic Manipulation. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.180. URL https://arxiv.org/abs/2605.11951.

Chenduo Ying, Linkang Du, Yuanchao Shu, and Peng Cheng. RoboFailRing: Retrieval-Augmented and Language Grounding Failure Detection for VLM-enabled Robotic Manipulation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13188–13202, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.602. URL https://aclanthology.org/2026.acl-long.602/.

Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, Hongyao Tang, and Jianye Hao. Embodied-R1: Reinforced Embodied Reasoning for General Robotic Manipulation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.13998v2.

Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, Yingcong Chen, Liuqing Yang, and Haoang Li. DualCoT-VLA: Visual-Linguistic Chain of Thought via Parallel Reasoning for Vision-Language-Action Models. *arXiv preprint arXiv:2603.22280*, 2026. doi: 10.48550/arXiv.2603.22280. URL https://arxiv.org/abs/2603.22280.

A Training, Annotation, and Verification Details

Algorithm 1 Stage verification and latent-suffix repair

Require: Instruction g, initial observation o_0, history M_0, budget B, and trained modules
Initialize $\mathcal{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_K)$ and $t \leftarrow 0$, $k \leftarrow 1$; check budget before every model call.
while the task is unfinished and the next call is within budget **do**
    Observe o_t; estimate $\widetilde{\mathbf{s}}_t$; assess stage completion and persistent deviations.
    if required evidence is insufficient **then**
        Keep the stage unconfirmed; do not advance or certify the prefix from missing evidence.
    else if consecutive valid observations support completion **then**
        Update M_t and advance k; extend the planning window if needed.
    else if reliable persistent evidence confirms a repair trigger **then**
        Form $\mathcal{A}_t$ by checking earlier conditions still required by the remaining task.
        Set $j_t \leftarrow \min(\{k\} \cup \mathcal{A}_t)$ and retain the verified prefix $\mathcal{Z}_{<j_t}$.
        Regenerate the suffix using Equation (11); set $\mathcal{Z} \leftarrow \mathcal{Z}^{\text{new}}$, $K \leftarrow K'$, and $k \leftarrow j_t$.
    end if
    If execution is still required and within budget, predict a chunk, execute h steps, and update time and cost.
end while

The policy is fine-tuned separately for each benchmark. Action dimensions follow the corresponding single-arm or bimanual interface, and normalization statistics are computed only from the relevant training set. The recorded optimizer is AdamW with an initial learning rate of 10^{-4}, an effective batch size of 64, weight decay of 0.01, and BF16 training (Loshchilov & Hutter, 2019). LoRA adapters use rank 32; the latent, state-decoding, observation-estimation, and action modules are trained (Hu et al., 2022). The initial stage and observation loss weights are both 1. Checkpoint and loss-weight selection use validation data. Efficiency comparisons hold hardware, numerical precision, batch size, warm-up, and synchronization consistent across repair strategies.

Stage boundaries are defined by stable changes in task-relevant predicates, including transitions from support to grasp, object release or placement, containment, and container opening or closing. Targets combine physical relations with their relevance to the instructed subgoal. Uncertain annotations are masked rather than labeled negative. Splits preserve complete demonstrations; recovery samples are grouped by failure scenario and source trajectory to prevent near-duplicate variants from crossing splits. Simulator states are restricted to offline annotation and independent evaluation.

Each stage consists of four latent vectors with the backbone hidden dimension. Masks accommodate varying stage counts during training. At inference, the model predicts subsequent stages and their termination within a planning window, whose documented limit is eight stages. Longer tasks extend the plan as execution advances. The action head conditions on the current observation, instruction, valid history, and remaining latent segments. Its output $\widehat{\mathbf{A}}_t$ in Equation (3) is an H-step action chunk; the first h steps are executed before the next observation.

The observation estimator uses recent images and proprioception. Candidate completion, persistent stagnation, and violations of required conditions initiate verification. The recorded initial completion rule requires three consecutive valid confirmations; thresholds and observation settings are selected using validation data. Verification distinguishes supported completion, reliable persistent contradiction, and insufficient evidence. A low discrepancy is insufficient for completion when required predicates remain unobserved. The regularizer ε in Equation (9) prevents a zero denominator and does not supply missing evidence.

Dependency checks concern conditions still required by the unfinished task. Releasing an object after successful placement does not invalidate a completed grasp. If an earlier stage's still-required condition becomes invalid, repair starts from that stage. Equation (10) selects the start from $\mathcal{A}_t$ and the active stage. Equation (12) concatenates the retained prefix and regenerated suffix. The stage generator has the same input and output types during initial planning and suffix reconstruction.

Suffix training samples split positions at demonstration stage boundaries. The aligned prefix, boundary observation, instruction, and history serve as inputs; the remaining stages and actions provide supervision. For the recovery-data augmentation, actual recovery trajectories supply suffix and action targets from failure states. The original configuration in Tables 4–6 excludes these additional

trajectories. All training and verification notation refers to the same task-state components; hidden dimension d, state dimension P, label masks $\mu_{i,p}$, window length W, and verification weights $\omega_{t,k,p}$ are distinct quantities.

B Metric Definitions and Accounting

Let N_{task} be the number of tasks, $N_i > 0$ the number of test episodes for task i, and $y_{i,n} \in \{0, 1\}$ its success indicator for episode n. Task-averaged success is

$$\text{SR} = \frac{1}{N_{\text{task}}} \sum_{i=1}^{N_{\text{task}}} \left(\frac{1}{N_i} \sum_{n=1}^{N_i} y_{i,n} \right). \tag{16}$$

The LIBERO average gives equal weight to Spatial, Object, Goal, and Long, each containing ten tasks. RoboTwin's 50-task and ten-task averages are computed over their respective task sets and are not combined into a single ranking. The 50-task protocol uses 50 clean demonstrations per task and 100 tests under each clean/randomized condition. Ten-task reference results follow the FastThinkAct protocol. Success and recovery rates, and verification F_1, are defined as proportions and multiplied by 100 for percentage reporting.

Stage verification is evaluated using environment-derived completion labels on fixed observations before and after completion. True positives (TP) correctly confirm completed stages; false positives (FP) confirm unfinished stages; and false negatives (FN) fail to confirm completed stages. For a nonzero denominator,

$$F_{1,\text{ver}} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}}. \tag{17}$$

For $N_{\text{rec}} > 0$ predefined recovery instances with success indicators $z_n \in \{0, 1\}$, the pooled instance-level recovery rate is

$$\text{RSR}_{\text{pooled}} = \frac{1}{N_{\text{rec}}} \sum_{n=1}^{N_{\text{rec}}} z_n. \tag{18}$$

This pooled rate differs from the suite macro-averages in Equations (14) and (15). Table 3 gives equal weight to the four suites at each recovery level. For the overall scores in Tables 4–6, instances are first pooled within each suite, after which the four suite rates are averaged. When level proportions differ across suites, equally averaging the level columns yields a different statistic. Consequently, the level scores of 52.0%, 32.0%, 20.0%, and 9.0% need not average to the 26.0% overall score. Both procedures retain undetected failures and budget-exhausted episodes in the denominator.

Cumulative inference time sums the model operations in Equation (13) over a test episode. Environment dynamics are excluded from model computation. Visual encoding shared by action prediction and verification is timed once. The regenerated-vector count includes only deviation-triggered reconstruction; ordinary rolling extensions contribute to inference time but are excluded from this auxiliary count. Mean episode time includes successful and failed episodes. Decision latency measures the interval from observation to an executable action, and Table 5 reports its 95th percentile. Hardware, precision, batching, warm-up, and synchronization are matched across repair strategies.

Aggregate timing does not distinguish all possible sources of savings. Full replanning achieves the highest recovery score in Table 5; local repair regenerates fewer vectors but can also change episode-termination patterns. The evidence therefore supports the reported recovery–cost trade-off at the evaluated setting. A success-versus-budget frontier or a comparison at matched success rates would require additional measurements. Likewise, measured means without uncertainty estimates do not establish the statistical significance of small differences.

C Published Context and Scope of the Evidence

The LIBERO reference rows retain the source groups in Table 1: A denotes OpenVLA-OFT, B Fast-ThinkAct, and C LaRA-VLA. Columns consistently follow Spatial, Object, Goal, and Long, while published averages retain their original definitions. Cross-paper differences may reflect training data, observation access, or inference budgets. These comparisons provide context for the controlled component and repair-scope experiments in Tables 4 and 5.

Fast-ThinkAct reports inference latencies of 805, 5674, and 7513 ms for Fast-ThinkAct-3B, ThinkAct-3B, and ThinkAct-7B, respectively (Huang et al., 2026). The corresponding reductions of approximately 85.8% and 89.3% describe those published comparisons. They are distinct from our measured 19.9% reduction in cumulative episode-level inference time between local repair and full replanning and use different timing settings.

Published ablations also motivate controlling supervision. Removing Fast-ThinkAct's verbalization preference loss changes LIBERO success from 89.7% to 88.6%; removing both that loss and visual distillation yields 86.3% (Huang et al., 2026). LaRA-VLA reports 55.2% and 68.8% on SimplerEnv for its no-CoT configuration and joint latent text-vision supervision, respectively (Bai et al., 2026). These experiments do not test our modules. Our auxiliary-state baseline and no-alignment variant instead assess additional supervision and the stage-semantic constraint under the stated controls.

The complete method combines a staged latent representation with execution-feedback repair. The no-alignment comparison examines stage-semantic supervision while retaining a trained verification decoder. Disabling repair evaluates the use of verifier outputs during execution, with verifier weights and diagnostic observations held fixed. These comparisons support the roles of the proposed components but do not establish superiority over an explicit task-graph system implemented with otherwise identical supervision and feedback.

Full replanning may be beneficial when prerequisites fail broadly or earlier verification is incorrect. Current-stage-only reconstruction can cost less when failures remain local, but it cannot revisit earlier conditions. Observation errors under domain randomization can induce false triggers, while environment-state recovery requires action capabilities beyond discrepancy detection. The 9.0% L4 score and the decline from 58.0% to 23.0% under scene randomization define limitations of the present evidence. All evaluations are conducted in simulation; these results do not establish generalization to physical robots.