REENVISION: FUTURE-GUIDED SELECTIVE ACTION REPAIR FOR LONG-HORIZON MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon robotic manipulation requires robust execution over sequences of interdependent actions. Existing methods typically handle execution deviations by replanning the remaining action chunk or correcting actions step by step as new observations arrive. However, a local deviation may affect only a few actions under distribution shift, yet replanning regenerates the entire chunk from an unfamiliar state, potentially introducing errors into otherwise valid actions. Step-wise correction may react too late, as it does not explicitly anticipate how the deviation affects subsequent actions. We propose ReEnvision, a future-guided selective action repair framework that updates a joint future-action belief using execution feedback. It then learns how the resulting belief revision affects each unexecuted action. The resulting belief revision enables selective repair of only the affected actions within a cached action chunk, preserving valid actions and reserving full replanning for severe deviations. On CALVIN, L-CALVIN, and RLBench, ReEnvision achieves 86.6% 5-step success, improves 10-step success from 56% to 63%, and reaches an average success rate of 88.8%, respectively. On five real-world manipulation tasks, it achieves an average success rate of 82.8%. Controlled repair experiments further demonstrate the benefits of localizing affected actions and preserving valid ones during execution. Our project page is available at https://reenvision.pages.dev.

1 INTRODUCTION

Long-horizon robotic manipulation requires robots to coordinate multiple dependent actions over an extended execution horizon, while preserving consistency across stages and recovering from intermediate deviations (Zhou et al., 2024; 2026). Such capability is important for real-world manipulation, where many tasks involve sustained, multi-stage interactions rather than isolated actions (Chen et al., 2024; Chai et al., 2026). However, reliable long-horizon execution remains challenging: an error at one stage can alter the state encountered by subsequent actions, causing later actions to become inappropriate and allowing errors to accumulate until the overall task fails (Tan et al., 2026).

Long-horizon manipulation has been approached from multiple perspectives, including hierarchical planning, predictive world modeling, and adaptive execution (Chai et al., 2025; Ma et al., 2026). Hierarchical planning and control methods decompose long tasks into subtasks, while world models anticipate future states and use them to guide actions (Sridhar et al., 2026; Li et al., 2026a). More recently, adaptive execution methods incorporate new observations during execution to adjust planned behavior: Future Forward Dynamics Causal Attention (FFDC) (Wang et al., 2026b) verifies whether the current rollout should continue or be replanned, while Asynchronous Action Chunk Correction (A2C2) (Sendai et al., 2025) uses each newly observed frame to predict a residual update to the actions being executed. In contrast, Real-Time Robot Execution with Masked Action Chunking (REMAC) (Wang et al., 2026a) preserves actions that are already committed and resamples only the remaining portion of the chunk. Despite these advances, existing methods primarily address execution deviations through rollout-level replanning, step-wise correction, or resampling the editable portion of an action chunk. They do not explicitly estimate which unexecuted actions are affected by an observed deviation. Under out-of-distribution (OOD) deviations, regenerating the entire remaining chunk from an unfamiliar intermediate state may disrupt established alignment and contact, amplifying local errors into downstream failures (Chen et al., 2026). Our analysis identifies pronounced replanning failures in approximately 11% of OOD sequences, with 55% occurring in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

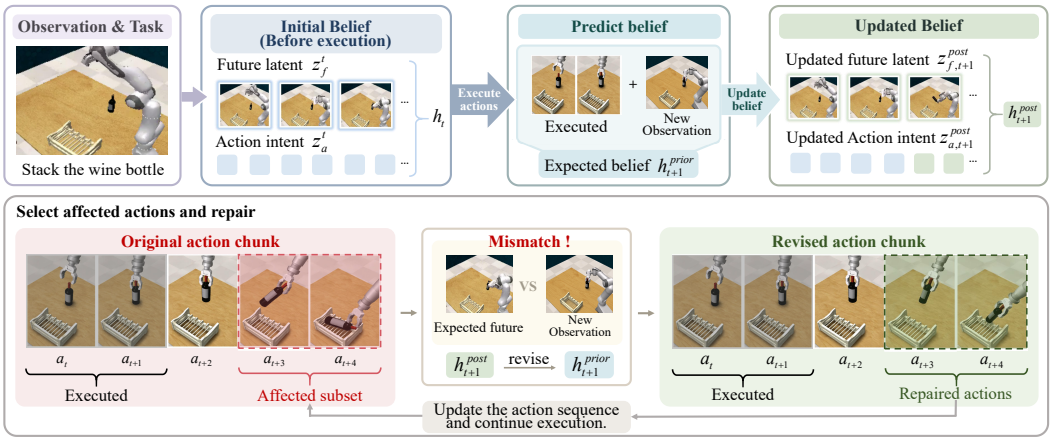


Figure 1: Overview of ReEnvision. ReEnvision maintains a joint future-action belief h_t and revises it with new observations. The revision identifies affected actions in the cached chunk, enabling selective repair. Unlike methods that rely on full replanning, ReEnvision repairs only affected actions within the current chunk while preserving unaffected ones.

direction-sensitive manipulation and 36% in contact- or pose-sensitive tasks. Meanwhile, step-wise correction may fail to anticipate how current deviations affect subsequent actions.

Addressing this limitation requires reasoning about the expected future and planned actions together: execution errors change what happens next, and these changes can in turn make some planned actions no longer suitable (Ye et al., 2026). By jointly modeling these two signals, the robot can reason about how an observed deviation changes what should happen next and how that change affects subsequent actions. This makes it possible to determine whether the remaining action chunk requires a global update or only a local adjustment. When only part of the chunk is affected, selectively repairing those actions preserves unaffected ones without regenerating the entire sequence.

Building on this idea, we propose ReEnvision, a future-guided selective action repair framework that revises a joint future-action belief and anticipates how the revision affects individual actions within a cached chunk as illustrated in Fig. 1. Given the current observation and task, a pretrained future generator predicts visual evolution, which is combined with the observed state to infer the corresponding action intent. Together, they form a joint future-action belief that conditions a diffusion policy to generate a multi-step action chunk. As new observations arrive, a latent dynamics model predicts how the joint belief should evolve following the executed action and updates the prediction using the observed outcome. The difference between the predicted and updated beliefs defines the belief revision. A temporal impact model then estimates how this belief revision affects each remaining action in terms of correction, validity, and uncertainty. These predictions enable selective repair of the affected actions while preserving unaffected ones, with full replanning reserved for broadly affected or highly uncertain suffixes. In experiments, ReEnvision achieves relative improvements of 2.5%, 12.5%, and 2.5% over the strongest reported baselines on CALVIN (Mees et al., 2022) 5-step success, L-CALVIN (Fan et al., 2025) 10-step success, and real-world manipulation, respectively, while achieving 88.8% average success on RLBench (James et al., 2019).

Our contributions are summarized as follows:

(i) We find that changes in the expected future typically affect only a local subset of actions within the remaining chunk; selectively repairing these actions is more effective than rewriting the entire chunk and enables correction before they are executed.

(ii) We propose ReEnvision, which learns how revisions in a joint future-action belief affect each unexecuted action and uses the predicted action-level impact to determine the scope of intervention, preserving unaffected actions while selectively repairing affected segments.

(iii) Extensive experiments demonstrate that ReEnvision achieves 86.6% 5-step success on CALVIN, 88.8% average success on RLBench, and 82.8% average success on five real-world manipulation tasks, with a 12.5% relative improvement in 10-step success on L-CALVIN.

2 RELATED WORK

2.1 HIERARCHICAL PLANNING AND CONTROL

Hierarchical planning and control methods decompose long-horizon manipulation into subgoals or execution phases, thereby reducing the effective decision horizon and improving temporal consistency (Zhao et al., 2023). Hierarchical Action Models for Open-World Robot Manipulation (HAMSTER) (Li et al., 2025b) first predicts a coarse end-effector trajectory, then conditions a low-level policy on this trajectory to generate precise 3D actions. Similarly, Long-VLA (Fan et al., 2025) instead decomposes each subtask into moving and interaction phases, and generates actions conditioned on the current phase to improve transitions between subtasks. Going beyond static decomposition, Reflective Contrastive Alignment and Planning Architecture (ReCAPA) (Zeng et al., 2026) further introduces predictive correction across action, subgoal, and trajectory levels, allowing local deviations to be corrected before they propagate into cascading failures. Despite these advances, such methods primarily operate over action-centric representations, without explicitly modeling how the physical world should evolve toward the task goal.

2.2 PREDICTIVE WORLD MODELS

Predictive world models forecast how the environment will evolve under candidate actions and use these predicted futures to guide subsequent control (Feng et al., 2025). Latent Diffusion World Model (LaDi-WM) (Huang et al., 2025) uses a diffusion-based world model to predict future states in a learned latent space, then conditions a diffusion policy on these predictions to refine actions. More recently, Mixture of Latent Actions (MoLA) (Li et al., 2026c) focuses on grounding imagined futures into executable control, converting future visual transitions into latent actions through inverse dynamics before decoding them into robot commands. In contrast, Decoupled Visual Forward and Inverse Dynamics Pretraining (DeFI) (Zhang et al., 2026) decouples the two learning objectives, separately training forward dynamics to predict futures and inverse dynamics to infer actions. However, these methods typically use predicted futures to guide action generation, so execution deviations may render the remaining actions inconsistent with the updated future (Gao et al., 2026).

2.3 ADAPTIVE ACTION EXECUTION

Adaptive action execution mitigates the mismatch between the updated future and the remaining committed actions by using execution feedback for online correction or replanning (Liang et al., 2026; Yao et al., 2026). FFDC (Wang et al., 2026b) compares predicted visual futures and actions with real observations to decide whether to continue the current rollout or replan, while A2C2 (Sendai et al., 2025) uses visual feedback at each step to predict a residual correction to the currently executed chunk action. More selectively, REMAC (Wang et al., 2026a) preserves already committed actions while resampling the remaining editable portion of the action chunk to maintain consistency during execution. These methods improve adaptive execution through rollout-level replanning decisions, step-wise residual correction, or masked action generation. However, they do not explicitly characterize how changes in the expected future affect individual unexecuted actions within a cached chunk (Bi et al., 2026). Consequently, these corrections may unnecessarily modify unaffected actions or fail to anticipate downstream effects on subsequent actions.

3 METHODOLOGY

3.1 FRAMEWORK OVERVIEW

To avoid both broad replanning from unfamiliar intermediate states and delayed step-wise correction, execution feedback should be used to anticipate how the expected future has changed and where its effects will appear in the remaining action chunk. This motivates localizing the affected actions in advance and revising only that portion while preserving the rest. To this end, we propose ReEnvision as shown in Fig. 2, which uses Stable Video Diffusion (SVD) (Blattmann et al., 2023) to predict visual evolution from the current observation and task instruction. Conditioned on the current state and anticipated future, an intent generator produces an action representation. Together with the future latent, it forms a joint belief that conditions a diffusion-based policy to generate an

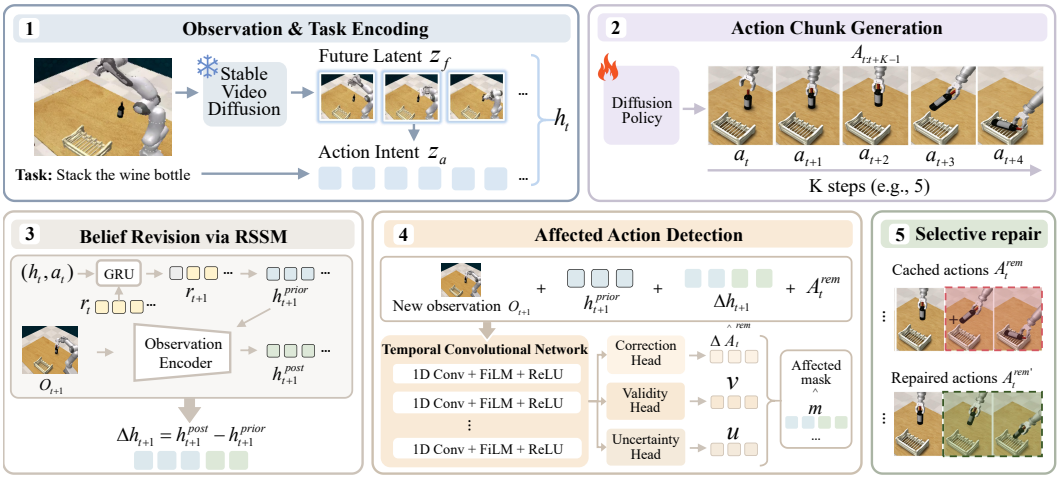


Figure 2: In ReEnvision, joint future-action belief h_t conditions action-chunk generation and is revised after each new observation. The resulting belief revision is used to predict per-action correction, validity, and uncertainty, enabling selective repair of the affected cached actions.

action chunk. As new observations arrive, a Recurrent State-Space Model (RSSM) (Hafner et al., 2019) updates the joint belief and measures its deviation from the expected belief transition. A Temporal Convolutional Network (TCN) (Bai et al., 2018) then uses this belief revision to estimate how each action in the remaining chunk is affected, leading to keep, selective correction, or replanning.

3.2 Joint Future-Action Belief

Given the current observation o_t and language instruction l, a pretrained SVD model extracts a base future representation $z_f^t \equiv z_f^{\text{base}}$ of the expected task-relevant evolution. During training, a target future representation z_f^{target} is constructed from successful trajectories to represent the desired execution stage. A Multilayer Perceptron (MLP)-based action-intent generator predicts an action-intent representation z_a^t, conditioned on the current observation, the base and target future representations, their difference $\Delta z_f = z_f^{\text{target}} - z_f^t$, and the task instruction. The generator is trained using an L_1 loss against action-intent targets derived from successful action chunks. During training, positive, identity, and negated revision pairs are sampled equally, with identity pairs mitigating the training–inference mismatch under zero revision.

At inference, the initial action intent is generated using a zero future revision, with $z_f^{\text{target}} = z_f^t$ and $\Delta z_f = 0$, without requiring a ground-truth target future. The future and action-intent representations are combined into a joint future-action belief,

$$h_t = (z_f^t, z_a^t), \tag{1}$$

which captures both the expected future and the corresponding action intent.

To generate executable actions, z_a^t is introduced as an additional conditioning signal to the pretrained diffusion policy. The original conditioning (o_t, l, z_f^t) is extended to (o_t, l, z_f^t, z_a^t), while the training objective remains the original diffusion action-prediction objective on ground-truth action chunks. The resulting policy produces a K-step action chunk $A_t = [a_t, \ldots, a_{t+K-1}]$.

3.3 Belief Revision

After generating an action chunk, the robot executes a_t and observes o_{t+1} as feedback on the resulting environment transition. Since the belief naturally evolves during execution, a change from h_t to the next belief does not by itself indicate an execution deviation. An RSSM therefore models the expected belief transition and incorporates execution feedback to capture discrepancies between the predicted and observation-updated beliefs. Given the current belief h_t and executed action a_t, the prior branch predicts the expected next belief h_{t+1}^{prior}. The posterior branch additionally incorporates the new observation o_{t+1} to infer the updated belief h_{t+1}^{post}.

During training, paired future–action latent representations are collected from consecutive execution steps of the same rollout. The current belief h_t and target belief h_{target} correspond to the current and next execution steps, respectively. Their difference defines the teacher latent displacement,

$$\Delta h^* = h_{\text{target}} - h_t. \tag{2}$$

Both RSSM branches are supervised to predict this displacement:

$$\mathcal{L}_{\text{RSSM}} = \ell(h_{t+1}^{\text{prior}} - h_t, \Delta h^*) + \ell(h_{t+1}^{\text{post}} - h_t, \Delta h^*), \tag{3}$$

where ℓ denotes the Smooth L_1 loss. The prior learns the expected belief transition without access to the new observation, while the posterior incorporates the observed outcome to refine the predicted update. During execution, the belief revision is defined as the difference between the posterior and prior beliefs, $\Delta h_{t+1} = h_{t+1}^{\text{post}} - h_{t+1}^{\text{prior}}$. This revision captures the observation-induced update to the expected belief transition and guides selective action repair. The posterior belief is then maintained for subsequent execution, $h_{t+1} \leftarrow h_{t+1}^{\text{post}}$.

3.4 Selective Action Repair

3.4.1 Belief-Guided Suffix Repair

After executing a_t, the remaining actions are cached as $A_t^{\text{rem}} = [a_{t+1}, \ldots, a_{t+K-1}]$. Since the belief revision Δh_{t+1} may affect only part of this suffix, a TCN S_ψ predicts an action correction $\Delta \hat{a}_k$, validity score v_k, and uncertainty u_k for each remaining action:

$$\left(\Delta \hat{A}_t^{\text{rem}}, \mathbf{v}, \mathbf{u}\right) = S_\psi\left(o_{t+1}, h_{t+1}^{\text{prior}}, A_t^{\text{rem}}, \Delta h_{t+1}\right), \tag{4}$$

where $k = 1, \ldots, K-1$ indexes the cached actions, larger v_k indicates higher action validity, and larger u_k indicates greater uncertainty in determining whether the action requires correction.

The affected actions are selected by optimizing a binary mask $\mathbf{m} \in \{0,1\}^{K-1}$:

$$\begin{aligned} \hat{\mathbf{m}} = \arg\min_{\mathbf{m}} \quad & \lambda_v \sum_{k=1}^{K-1} (1-m_k)(1-v_k) + \lambda_m \sum_{k=1}^{K-1} m_k \\ & + \lambda_c \sum_{k=1}^{K-2} |m_{k+1} - m_k| + \lambda_u \sum_{k=1}^{K-1} m_k u_k. \end{aligned} \tag{5}$$

The objective penalizes leaving low-validity actions unrepaired, modifying unnecessary actions, fragmented repair regions, and repairing highly uncertain actions. The corresponding weights are set to $\lambda_v = 1$, $\lambda_m = 0.5$, $\lambda_c = 0.15$, and $\lambda_u = 0.15$, favoring compact and reliable repair regions.

The optimized mask determines whether the cached suffix should be kept, selectively repaired, or discarded. The repair ratio $r = \frac{1}{K-1}\sum_{k=1}^{K-1} \hat{m}_k$ measures the fraction of affected actions, s denotes the number of disconnected affected segments, and $u_{\max} = \max_{1 \le k \le K-1} u_k$ measures the maximum uncertainty across the entire suffix. Full replanning is triggered when $u_{\max} \ge \eta$. Otherwise, the suffix is retained when no action is selected, and selective repair is applied when $r \le \rho$ and $s \le s_{\max}$. If either condition is violated, the cached suffix is discarded and fully replanned. Here, ρ, $s_{\max}$, and η control the maximum repair extent, fragmentation, and uncertainty, respectively.

For selective repair, only the selected actions are updated:

$$A_t^{\text{rem}\prime} = A_t^{\text{rem}} + \hat{\mathbf{m}} \odot \Delta \hat{A}_t^{\text{rem}}, \tag{6}$$

with the correction magnitude clipped to a bounded range and smoothing applied within the selected repair region to maintain trajectory continuity. Under full replanning, the cached suffix is discarded, and a new action chunk is generated by the diffusion policy conditioned on the posterior belief h_{t+1}^{post}.

3.4.2 Action Impact Model Training

S_ψ is trained on outcome-guided action pairs to approximate belief-induced corrections. Each pair consists of lower- and higher-return action chunks, A^- and A^+, generated by the same policy under matched observation, task, diffusion noise, and cache conditions.

$$A^- = A_{\text{neg}}, \qquad A^+ = A_{\text{pos}}. \tag{7}$$

Table 1: CALVIN (Mees et al., 2022) ABC→D results. We report the average success rate over 1,000 rollouts and the average number of consecutively completed tasks (Avg. Len.). ReEnvision demonstrates clear improvements over the baselines, particularly on longer-horizon sequences.

Method	**1**	**2**	**3**	**4**	**5**	**Avg. Len.** ↑
$\pi^*_{0.5}$ (Intelligence et al., 2025)	94.8	87.4	78.2	71.7	64.3	3.97
VPP (Hu et al., 2025)	96.5	90.9	86.6	82.0	76.9	4.33
REMAC (Wang et al., 2026a)	96.0	91.4	88.3	84.9	78.4	4.39
PALM (Liu et al., 2026)	96.9	93.8	89.3	85.9	82.0	4.48
DeFI (Zhang et al., 2026)	97.9	94.2	90.7	87.0	81.2	4.51
VITA (Ma et al., 2026)	**99.1**	**94.9**	91.2	87.8	84.5	4.57
ReEnvision	97.8	94.5	**93.0**	**90.1**	**86.6**	**4.62**

Both A^- and A^+ are full action chunks generated by the same frozen policy. Their first $K-1$ actions, denoted by A^-_{rem} and A^+_{rem}, are temporally aligned with the cached suffix A^{rem}_t. Their difference, $\Delta A^*_{\text{rem}} = A^+_{\text{rem}} - A^-_{\text{rem}}$, provides an outcome-guided correction from lower- to higher-return actions. At inference, pre-revision actions are retrieved directly from the cached suffix A^{rem}_t. For each remaining action k, the validity target indicates whether its outcome-guided correction magnitude remains within a predefined threshold:

$$v^*_k = \mathbb{I}\left[\|\Delta a^*_k\|_2 \le \tau\right], \tag{8}$$

where $k = 1, \ldots, K-1$ indexes the cached suffix, and $v^*_k = 1$ indicates that the outcome-guided correction magnitude does not exceed the threshold τ. An uncertainty target u^*_k is further constructed from the distance between $\|\Delta a^*_k\|_2$ and τ, assigning higher uncertainty to actions closer to the validity decision boundary and lower uncertainty to those farther away. The threshold τ is used only to construct offline supervision targets. During inference, predicted validity and uncertainty guide structured mask selection without directly thresholding predicted corrections.

S_ψ is jointly trained to predict outcome-guided action corrections, validity, and uncertainty:

$$\mathcal{L}_{\text{TCN}} = \left\| \Delta \hat{A}^{\text{rem}}_t - \Delta A^*_{\text{rem}} \right\|_1 + \lambda_{\text{val}} \,\text{BCE}\left(\mathbf{v}, \mathbf{v}^*\right) + \lambda_{\text{unc}} \left\|\mathbf{u} - \mathbf{u}^*\right\|_1 . \tag{9}$$

The loss weights are set to $\lambda_{\text{val}} = 1.0$ and $\lambda_{\text{unc}} = 0.1$. This outcome-guided supervision is used to approximate the action changes induced by belief revision. During execution, predicted corrections are applied only to the selected actions in Eq. 6, while unaffected actions are preserved.

3.5 Training and Inference

3.5.1 Training

The framework is trained in stages. Future representations from the pretrained SVD are first used to train the action-intent generator, after which the diffusion policy is further trained to generate action chunks conditioned on both future and action-intent representations. Rollouts from the resulting policy are then used to train the RSSM, with both prior and posterior branches supervised by the teacher latent displacement $\Delta h^* = h_{\text{target}} - h_t$. Finally, matched lower- and higher-return action chunks from the frozen diffusion policy provide outcome-guided correction targets for training S_ψ to predict per-action corrections, validity, and uncertainty.

3.5.2 Inference

At initialization, the current observation and task instruction are used to construct the initial joint belief and generate an action chunk. After each action is executed, the RSSM predicts the next belief and incorporates the new observation to obtain the posterior belief. The pre-revision actions are retrieved directly from the cached suffix A^{rem}_t, without regeneration. Given the belief revision Δh_{t+1} and cached suffix, S_ψ predicts per-action corrections, validity, and uncertainty. The suffix is retained when no action is selected and uncertainty is low, and selectively repaired when the affected region is sufficiently localized. Full replanning is triggered when the affected region is too large, fragmented, or uncertain, generating a new action chunk conditioned on h^{post}_{t+1}.

Table 2: Comparison with prior methods on the L-CALVIN (Fan et al., 2025), using 100 evaluation sequences of 10 consecutive instructions under the ABC→D setting.

Method	Tasks Completed in Sequence										Avg. Len.
	1	2	3	4	5	6	7	8	9	10	
VLAS (Zhao et al., 2025)	0.88	0.80	0.72	0.58	0.49	0.46	0.34	0.28	0.19	0.16	4.90
RoboVLMs (Li et al., 2026b)	0.90	0.80	0.78	0.70	0.64	0.60	0.51	0.41	0.36	0.34	6.04
REMAC (Wang et al., 2026a)	0.94	0.90	0.82	0.74	0.71	0.58	0.54	0.50	0.47	0.41	6.61
DeFI (Zhang et al., 2026)	0.98	0.96	0.92	0.86	0.75	0.71	0.59	0.57	0.52	0.44	7.30
Long-VLA (Fan et al., 2025)	**1.00**	**1.00**	**0.98**	0.91	0.85	0.82	**0.79**	0.70	0.63	0.56	8.24
ReEnvision	0.99	0.98	0.95	**0.92**	**0.89**	**0.83**	0.77	**0.72**	**0.68**	**0.63**	**8.36**

Table 3: Performance comparison on 18 RLBench (James et al., 2019) manipulation tasks, with 50 evaluation episodes per task. We report task success rates (%) and the average success rate (Avg. SR) across all 900 episodes. Best results are shown in bold.

Method	Avg. SR	Close Jar	Insert Peg	Light Bulb	Meat off Grill	Open Drawer	Place Cups	Shape Sorter	Place Wine
HDFlow (Gireesh et al., 2026)	87.5	96.0	93.3	88.7	98.7	82.0	56.7	**73.3**	94.7
TVVE (Bai et al., 2026)	86.6	**100.0**	**98.0**	86.0	94.0	90.0	54.0	62.0	92.0
SpatialActor (Shi et al., 2026)	87.4	94.0	93.3	88.7	98.7	82.0	56.7	**73.3**	94.7
BridgeVLA (Li et al., 2025a)	85.7	98.0	80.0	86.0	**100.0**	**98.0**	48.0	38.0	90.0
ReEnvision	**88.8**	98.0	82.0	**90.0**	**100.0**	90.0	60.0	50.0	**96.0**

Method	Push Buttons	Put Groceries	Put in Drawer	Put in Safe	Reach & Drag	Slide Block	Stack Blocks	Stack Cups	Sweep Dustpan	Turn Tap
HDFlow	**100.0**	72.0	98.7	96.7	**100.0**	91.3	56.0	81.3	**100.0**	95.3
TVVE	**100.0**	74.0	**100.0**	78.0	**100.0**	**100.0**	74.0	64.0	92.0	**100.0**
SpatialActor	**100.0**	72.0	98.7	96.7	**100.0**	91.3	56.0	81.3	**100.0**	95.3
BridgeVLA	**100.0**	80.0	88.0	**100.0**	**100.0**	98.0	76.0	80.0	90.0	92.0
ReEnvision	90.0	**92.0**	94.0	**100.0**	94.0	82.0	**94.0**	**90.0**	96.0	**100.0**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate ReEnvision on CALVIN (Mees et al., 2022), L-CALVIN (Fan et al., 2025), and RL-Bench (James et al., 2019), using DeFI (Zhang et al., 2026) as the backbone for CALVIN/L-CALVIN and BridgeVLA (Li et al., 2025a) for RLBench. On CALVIN, we follow the ABC→D protocol and evaluate 1,000 five-task sequences using 10-step, 7-DoF relative action chunks from RGB observations, proprioception, and language instructions. We report success rates for 1–5 consecutive tasks and average sequence length. L-CALVIN extends the same policy to ten-task sequences, reporting SR1–SR10 and average length. On RLBench, we evaluate 18 tasks with 50 episodes each and report per-task and mean success rates. All results are averaged over three random seeds.

For real-world evaluation, we use an ARX X5 robot on five tasks: pen placement, color sorting, sequential drawer placement, towel folding and placement, and sequential flower placement. We collect 150 kinesthetic demonstrations per task. Two Intel RealSense D405 cameras capture RGB observations at 30 Hz and 640×480 resolution. Each policy receives both RGB views, a language instruction, and a 7-D proprioceptive state.

4.2 ABLATION STUDIES

We use DeFI and BridgeVLA as the baselines. For belief representation, the diffusion-policy conditioning is varied while the rest of ReEnvision is kept fixed: z_f only for DeFI, z_a only for action-only, and $h_t = (z_f^t, z_a^t)$ for joint belief. Belief revision Δh, uncertainty u_k, and the full replanning are further ablated, with selective repair replaced by full replanning in the latter.

Table 4: Real-world manipulation performance on five tasks using an ARX X5 robot. Each method is evaluated over 50 trials per task.

Method	**Success Rate (%)**					
	Pen	Color	Drawer	Towel	Flower	**Average**
Diffusion Policy (Chi et al., 2024)	78	56	78	80	52	68.8
$\pi^*_{0.5}$ (Intelligence et al., 2025)	**90**	70	80	**94**	70	80.8
BridgeVLA (Li et al., 2025a)	88	70	**86**	74	68	77.2
DeFI (Zhang et al., 2026)	86	68	78	82	66	76.0
ReEnvision	**90**	**76**	84	88	**76**	**82.8**

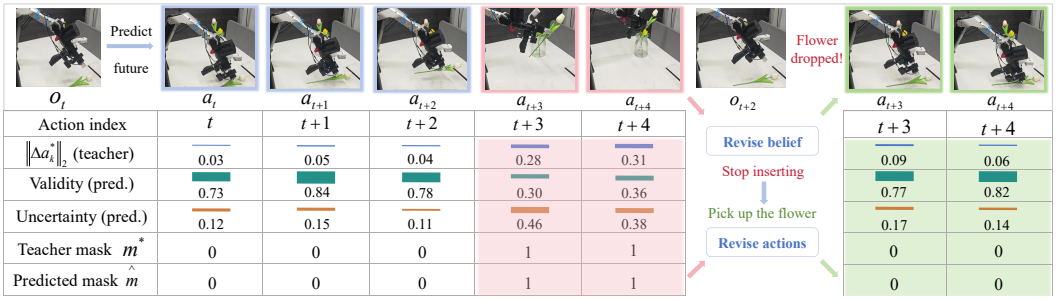


Figure 3: Visualization on a flower placement task. ReEnvision detects affected actions following an unexpected flower drop and selectively repairs the remaining action chunk through belief revision. For visualization, the teacher correction magnitudes $\|\Delta a_k^*\|_2$ are rescaled for readability, with actions exceeding 0.10 on the rescaled scale marked as affected.

To isolate the effect of repair location, we fix the observation, task condition, and diffusion noise while changing the belief from h_{t+1}^{prior} to h_{t+1}^{post}. Their paired policy responses define the reference correction ΔA^*_{rem}, with actions above threshold τ forming the reference mask m^*. No Repair leaves the cached suffix unchanged without applying ΔA^*_{rem}. For the remaining reference-mask controls, the same ΔA^*_{rem} is used while only the repair mask is varied: Selective w/ m^* repairs the affected positions, Random Mask repairs the same number of randomly selected positions, Complement Mask repairs positions outside m^*, and Whole Suffix repairs the entire remaining suffix. ReEnvision applies the TCN-predicted mask $\hat{m}$ together with the predicted correction $\Delta \hat{A}_t^{\text{rem}}$.

4.3 Analysis

As shown in Table 1, ReEnvision achieves the best performance from 3/5 onward and the highest Avg. Len. of 4.62, with larger gains at longer horizons. This trend is clearer on L-CALVIN (Table 2), where ReEnvision reaches an Avg. Len. of 8.36 and improves the later stages. On RLBench (Table 3), it improves the reproduced BridgeVLA baseline from 85.7% to 88.8% average success rate. In real-world experiments (Table 4), ReEnvision achieves the highest average success rate of 82.8% across five manipulation tasks, with leading or tied-leading results on Pen, Color, and Flower. Overall, selective repair reduces error accumulation by preserving valid actions: 86.8% of cached suffixes are kept unchanged, 11.1% selectively patched, and only 2.1% fully replanned.

Table 5 shows consistent gains over the corresponding backbone baselines across all three benchmarks, especially in longer-horizon settings. Joint (z_f, z_a) conditioning also outperforms either representation alone. Full replanning is slightly better on RLBench (88.9 vs. 88.8), while removing uncertainty gives a marginally higher L-CALVIN 5-step success (0.90 vs. 0.89); ReEnvision remains stronger on the longer-horizon metrics. In addition, Table 6 further highlights the importance of repair localization. On L-CALVIN, Selective w/ m^* reaches an Avg. Len. of 8.45, compared with 7.59 for Random Mask and 6.85 for Complement Mask, showing that repairing the affected actions is more effective than modifying random or unaffected ones. Fig. 3 illustrates how ReEnvision selectively repairs affected actions following an unexpected flower drop.

Table 5: Ablation study of ReEnvision on RLBench, CALVIN, and L-CALVIN.

Methods	RLBench	CALVIN		L-CALVIN		
	Avg. SR ↑	5-Step ↑	Avg. Len. ↑	Avg. Len. ↑	5-Step ↑	10-Step ↑
Baseline	85.7	81.2	4.51	7.30	0.75	0.44
z_f only	84.6	84.7	4.58	8.22	0.85	0.55
z_a only	88.4	85.2	4.57	8.26	0.80	0.56
w/o belief revision	87.4	82.6	4.54	7.96	0.83	0.50
w/o uncertainty	88.0	85.2	4.60	8.32	**0.90**	0.61
w/ full replanning	**88.9**	82.1	4.53	7.74	0.77	0.51
ReEnvision (z_f, z_a)	88.8	**86.6**	**4.62**	**8.36**	0.89	**0.63**

Table 6: Repair-location control study on RLBench, CALVIN, and L-CALVIN.

Methods	RLBench	CALVIN		L-CALVIN		
	Avg. SR ↑	5-Step ↑	Avg. Len. ↑	Avg. Len. ↑	5-Step ↑	10-Step ↑
No Repair	85.7	81.2	4.51	7.30	0.75	0.44
Random Mask	83.1	81.6	4.48	7.59	0.80	0.49
Complement Mask	77.4	78.8	4.39	6.85	0.69	0.38
Selective w/ m^*	**88.8**	85.9	**4.65**	**8.45**	**0.91**	**0.66**
Whole Suffix	81.2	80.7	4.50	7.15	0.80	0.43
ReEnvision	**88.8**	**86.6**	4.62	8.36	0.89	0.63

4.4 Discussion

Our results suggest that belief changes usually affect only a local part of the cached action suffix rather than making the whole suffix invalid. The repair-location study further shows that repair location is critical: Selective w/ m^* outperforms Random Mask, Complement Mask, and Whole Suffix, supporting selective modification of affected actions while preserving valid ones. The ablations also show complementary roles for future prediction and action intent: the former captures expected task evolution, while the latter represents how the policy realizes it. Their combination therefore provides a stronger basis for deciding which remaining actions require revision. Full replanning remains competitive on RLBench, where tasks are shorter and rebuilding the suffix can correct errors with less disruption. In longer-horizon settings, repeated replanning may discard useful actions and introduce unnecessary changes, making ReEnvision increasingly advantageous as the execution horizon grows. The learned ReEnvision policy also performs close to Selective w/ m^*, suggesting that these useful repair locations can be recovered without access to the reference mask.

ReEnvision estimates belief-induced action changes with a lightweight TCN and invokes full replanning only under severe deviations. Despite these optimizations, ReEnvision still introduces about 30% additional inference latency over the baseline. In addition, the outcome-guided action pairs provide action-level supervision, but they do not establish whether each individual repair improves task success. Task-level success is difficult to use directly because it gives only a final outcome, making action-level credit assignment hard. Future work could first localize a small repair region using belief changes, then use task outcomes to refine which actions should be modified.

5 Conclusion

We presented ReEnvision, which combines the expected future and action intent into a joint belief, updates this belief with new observations, and estimates how the resulting revision affects the remaining action sequence. This enables the policy to preserve valid actions, selectively repair affected ones, and reserve full replanning for severe deviations. Experiments on CALVIN, L-CALVIN, and RLBench, together with real-robot evaluations, show consistent improvements over the corresponding backbone policies, with larger gains over longer horizons. The repair-location study further shows that performance gains depend on correctly localizing belief-induced changes within the action chunk, so that affected actions are corrected while valid actions are preserved. Future work may improve affected-action estimation and extend selective repair to more diverse real-world settings.

AI USE STATEMENT

In this work, generative AI tools were used to assist with literature search and synthesis and to improve the clarity, readability, and wording of the manuscript. All AI-assisted content, including literature-related information and textual revisions, was reviewed and verified by the authors against the original sources where applicable. The authors take full responsibility for the final content, claims, and presentation of this work.

REPRODUCIBILITY STATEMENT

We provide the model architecture, training objectives, and staged training and inference procedures in Section 3. Section 4 details the benchmark protocols, evaluation metrics, and real-world experimental setup. Appendix A includes additional experimental analyses, runtime measurements, and hyperparameter sensitivity studies. All source code and pretrained model weights will be publicly released upon acceptance to facilitate reproducibility.

REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Yongjie Bai, Zhouxia Wang, Yang Liu, Kaijun Luo, Yifan Wen, Mingtong Dai, Weixing Chen, Ziliang Chen, Lingbo Liu, Guanbin Li, and Liang Lin. Learning to see and act: Task-aware virtual view exploration for robotic manipulation, 2026. URL https://arxiv.org/abs/2508.05186.

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 (CVPR)*, pp. 35101–35113, June 2026.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023. URL https://arxiv.org/abs/2311.15127.

Qi Chai, Zhang Zheng, Junlong Ren, Deheng Ye, Zichuan Lin, and Hao Wang. Causalmace: Causality empowered multi-agents in minecraft cooperative tasks, 2025. URL https://arxiv.org/abs/2508.18797.

Qi Chai, Wenhao Shen, Nanjie Yao, Yue Xia, Kaiyong Zhao, Jie Ma, Guosheng Lin, and Hao Wang. Evolvenav: Proactive preflection and self-evolving memory for zero-shot object goal navigation, 2026. URL https://arxiv.org/abs/2606.18235.

Zixuan Chen, Ze Ji, Jing Huo, and Yang Gao. Scar: Refining skill chaining for long-horizon robotic manipulation via dual regularization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 111679–111714. Curran Associates, Inc., 2024. doi: 10.52202/079017-3547. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ca92ff06d973ece92cecc561757d500e-Paper-Conference.pdf.

Zixuan Chen, Xiangrong Feng, Jieqi Shi, Lin Shao, Jing Huo, and Yang Gao. Agile: Learning robust long-horizon manipulation via affordance-grounded bidirectional latent planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6760–6769, June 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Yiguo Fan, Pengxiang Ding, Shuanghao Bai, Xinyang Tong, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, et al. Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. *arXiv preprint arXiv:2508.19958*, 2025.

Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. *arXiv preprint arXiv:2502.16707*, 2025.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, Qianli Ma, Seungjun Nah, Loic Magne, Jiannan Xiang, Yuqi Xie, Ruijie Zheng, Dantong Niu, You Liang Tan, K. R. Zentner, George Kurian, Suneel Indupuru, Pooya Jannaty, Jinwei Gu, Jun Zhang, Jitendra Malik, Pieter Abbeel, Ming-Yu Liu, Yuke Zhu, Joel Jang, and Linxi "Jim" Fan. Dreamdojo: A generalist robot world model from large-scale human videos, 2026. URL https://arxiv.org/abs/2602.06949.

Nandiraju Gireesh, Yuanliang Ju, Chaoyi Xu, Weiheng Liu, Yuxuan Wan, and He Wang. Hdflow: Hierarchical diffusion-flow planning for long-horizon tasks, 2026. URL https://arxiv.org/abs/2605.04525.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations, 2025. URL https://arxiv.org/abs/2412.14803.

Yuhang Huang, Jiazhao Zhang, Shilong Zou, Xinwang Liu, Ruizhen Hu, and Kai Xu. Ladi-wm: A latent diffusion-based world model for predictive manipulation. *arXiv preprint arXiv:2505.11528*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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, 2025. URL https://arxiv.org/abs/2504.16054.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment, 2019. URL https://arxiv.org/abs/1909.12271.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Luyao Zhang, Mingrui Yu, Zelin Gao, Nan Xue, Boyu Zhou, Xing Zhu, Mingyu Ding, Yujun Shen, and Yinghao Xu. Causal World Modeling for Robot Control. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026a. doi: 10.15607/RSS.2026.XXII.016.

Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models, 2025a. URL https://arxiv.org/abs/2506.07961.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, Tao Kong, Hanbo Zhang, and Huaping Liu. What matters in building vision-language-action models for generalist robots, 2026b. URL https://arxiv.org/abs/2412.14058.

Yajie Li, Bozhou Zhang, Chun Gu, Zipei Ma, Jiahui Zhang, Jiankang Deng, Xiatian Zhu, and Li Zhang. From imagined futures to executable actions: Mixture of latent actions for robot manipulation, 2026c. URL https://arxiv.org/abs/2605.12167.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 24040–24068, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/3bfee3bc6639c36e6e7b058db909f760-Paper-Conference.pdf.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models, 2026. URL https://arxiv.org/abs/2604.04161.

Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, Fangqiang Ding, and Ismini Lourentzou. Palm: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation, 2026. URL https://arxiv.org/abs/2601.07060.

Xiangkai Ma, Lekai Xing, Han Zhang, Wenzhong Li, and Sanglu Lu. Unifying perception and action: A hybrid-modality pipeline with implicit visual chain-of-thought for robotic action generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22380–22390, June 2026.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022. URL https://arxiv.org/abs/2112.03227.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks, 2025. URL https://arxiv.org/abs/2509.23224.

Hao Shi, Bin Xie, Yingfei Liu, Yang Yue, Tiancai Wang, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Spatialactor: Exploring disentangled spatial representations for robust robotic manipulation, 2026. URL https://arxiv.org/abs/2511.09555.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 97142–97166, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/9da515b1ad19d032a7398f00f5ff9b0c-Paper-Conference.pdf.

Huajie Tan, Peterson Co, Yijie Xu, Shanyu Rong, Yuheng Ji, Cheng Chi, Xiansheng Chen, Zhongxia Zhao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Action-sketcher: From reasoning to action via visual sketches for robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22433–22444, June 2026.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 120161–120179, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/c35834443b7881e782e52b3519fe27c7-Paper-Conference.pdf.

Rui Wang, Yue Zhang, Jiehong Lin, Kuncheng Luo, Jianan Wang, Zhongrui Wang, and Xiaojuan Qi. When to trust imagination: Adaptive action execution for world action models, 2026b. URL https://arxiv.org/abs/2605.06222.

Nanjie Yao, Hao Wang, Chong Cheng, Zhikang Chen, Wenzhe Li, et al. World models for embodied intelligence: From plausible to controllable to actionable. *arXiv preprint*, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie,

Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Xiyin Zeng, Yuyu Sun, Haoyang Li, Shouqiang Liu, and Hao Wang. Recapa: Hierarchical predictive correction to mitigate cascading failures, 2026. URL https://arxiv.org/abs/2604.21232.

Wenyao Zhang, Bozhou Zhang, Zekun Qi, Wenjun Zeng, Xin Jin, and Li Zhang. Disentangled robot learning via separate forward and inverse dynamics pretraining. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 74565–74595, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/793bfa8f8c8db6e33a7ecf410ae573a8-Paper-Conference.pdf.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

Wei Zhao, Pengxiang Ding, Min Zhang, Zhefei Gong, Shuanghao Bai, Han Zhao, and Donglin Wang. Vlas: Vision-language-action model with speech instructions for customized robot manipulation, 2025. URL https://arxiv.org/abs/2502.13508.

Pengfei Zhou, Liliang Chen, Shengcong Chen, Di Chen, Wenzhi Zhao, Rongjun Jin, Guanghui Ren, and Jianlan Luo. Act2Goal: From World Model To General Goal-conditioned Policy. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.015.

Zihan Zhou, Animesh Garg, Dieter Fox, Caelan Garrett, and Ajay Mandlekar. Spire: Synergistic planning, imitation, and reinforcement learning for long-horizon manipulation. *arXiv preprint arXiv:2410.18065*, 2024.

A Supplementary experiments

A.1 Mechanism Validation

We further validate two properties underlying selective action repair: (i) belief revision typically affects only a localized portion of the cached action suffix, and (ii) the TCN can accurately localize and predict the corresponding action changes.

Locality of affected actions. We use the paired policy responses under h_{t+1}^{prior} and h_{t+1}^{post} to obtain $\Delta A_{\text{rem}}^{*}$ and the corresponding reference affected-action mask m^{*}. Specifically, $m_k^{*} = 1$ indicates that the change of action k exceeds the threshold τ. We measure the fraction of affected actions and the number of contiguous affected segments within each cached suffix.

TCN localization accuracy. The TCN-predicted affected-action mask is evaluated against m^{*}. We report mask F1, temporal and segment IoU, and the prediction error of the action correction $\Delta \hat{A}_t^{\text{rem}}$.

As shown in Table 7, only 20.8% of the cached suffix is marked as affected by m^{*}, with 0.821 contiguous affected segments per suffix, indicating that belief-induced action changes are typically localized rather than distributed throughout the suffix. The TCN closely reproduces this structure, achieving a mask F1 of 0.861 and a non-empty segment IoU of 0.696 while predicting a similar affected fraction (20.6%). These results demonstrate that belief-induced action changes can be reliably localized. Notably, predicted action impact does not necessarily lead to modification, as the final intervention is determined by structured mask selection and the keep-repair-replan decision.

Table 7: Validation of action-change locality and TCN localization accuracy. m^* denotes the reference affected-action mask derived from paired policy responses. Non-empty IoU is computed only on suffixes containing at least one affected action.

Metric	**Value**
Affected fraction (m^*)	0.208
Pred. affected fraction	0.206
Segments per suffix (m^*)	0.821
Pred. segments per suffix	0.850
Mask F1	0.861
Temporal IoU (all)	0.785
Temporal IoU (non-empty)	0.690
Segment IoU (all)	0.788
Segment IoU (non-empty)	0.696
ΔA MAE	0.000167

Table 8: Controlled deviation recovery on L-CALVIN under perturbations of increasing severity.

Severity	Strategy	Recovery Success ↑	Avg. Len. ↑	Modified Actions	Replan Rate
Clean	ReEnvision	0.75	8.36	0.08	0.03
Small	No Repair	0.79	7.18	0.00	0.00
Small	Step-wise	0.75	7.56	0.03	0.00
Small	Full Replan	0.68	7.07	1.00	1.00
Small	Oracle Selective	0.93	8.25	0.08	0.05
Small	ReEnvision	0.84	7.93	0.18	0.11
Medium	No Repair	0.55	6.96	0.00	0.00
Medium	Step-wise	0.55	7.10	0.06	0.00
Medium	Full Replan	0.63	6.95	1.00	1.00
Medium	Oracle Selective	0.78	7.30	0.20	0.13
Medium	ReEnvision	0.70	7.14	0.30	0.17
Severe	No Repair	0.35	5.68	0.00	0.00
Severe	Step-wise	0.42	5.75	0.08	0.00
Severe	Full Replan	0.58	6.21	1.00	1.00
Severe	Oracle Selective	0.62	6.91	0.36	0.38
Severe	ReEnvision	0.60	6.62	0.37	0.52

A.2 Controlled Deviation Recovery

We construct a controlled deviation evaluation on L-CALVIN to examine how different recovery strategies respond to execution errors of increasing severity. During rollout, deviations are injected directly into the simulator state using two complementary perturbations. First, an end-effector deviation applies a controlled offset to the robot pose, providing a simple and reproducible execution error whose magnitude can be precisely varied. Second, object displacement perturbs the pose of a task-relevant object, representing an external state change that may invalidate only part of the cached action suffix. Both perturbations are applied at small, medium, and severe levels. This setup allows us to test whether localized deviations can be handled by selectively modifying the affected actions, while larger deviations increasingly require replanning. Recovery Success denotes the fraction of perturbed rollouts that successfully recover, Avg. Len. measures the average number of consecutively completed tasks, Modified Actions denotes the fraction of the cached suffix modified during recovery, and Replan Rate reports the frequency of discarding the cached suffix and generating a new action chunk. The clean setting corresponds to ReEnvision without injected perturbations and serves as a reference for the performance degradation caused by execution deviations.

As shown in Table 8, recovery becomes more difficult as deviation severity increases, while ReEnvision progressively shifts from suffix reuse and selective repair toward more frequent replanning. Interestingly, under small deviations, No Repair performs better than both Step-wise correction and Full Replan, suggesting that weak perturbations often leave much of the cached suffix valid and that

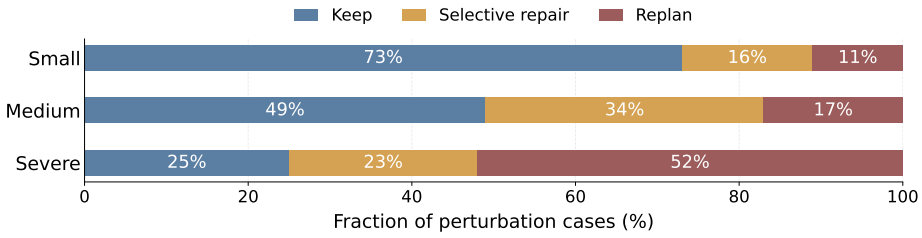


Figure 4: ReEnvision's execution strategy under controlled deviations of increasing severity. Each bar shows the fraction of perturbation cases handled by keeping the cached suffix, selectively repairing it, or triggering full replanning. As deviation severity increases, ReEnvision progressively shifts from suffix preservation toward selective intervention and full replanning.

Table 9: Runtime analysis of ReEnvision. Component latencies are measured individually, while the total latency reports end-to-end inference time per execution step. ReEnvision introduces approximately 32% additional latency over the backbone policy, while the belief revision and selective repair components add only a small computational overhead.

Component	**Latency**
Backbone policy	$\sim$110 ms
Future–action belief	$\sim$8 ms
Belief revision model	$\sim$2 ms
Temporal impact model	$\sim$1 ms
Mask optimization	$<$1 ms
ReEnvision (end-to-end)	$\sim$145 ms
Full diffusion replan	$\sim$110 ms

unnecessary intervention can disrupt otherwise successful execution. This also explains why selective repair provides the largest benefit when the deviation is large enough to invalidate part of the suffix but not severe enough to require complete regeneration. Under severe deviations, ReEnvision approaches the recovery performance of Full Replan while maintaining a higher Avg. Len., indicating that even in this regime, some cached actions can remain useful. Overall, these results support adapting the scope of intervention to the extent of the execution deviation.

Figure 4 further illustrates how ReEnvision adapts its execution strategy to deviation severity. Under small deviations, most cases are handled by preserving the cached suffix, with a keep rate of 0.73. As the deviation becomes moderate, selective repair becomes more frequent, increasing from 0.16 to 0.34, while the keep rate decreases. Under severe deviations, full replanning becomes the dominant response, accounting for 0.52 of the cases. This progression shows that ReEnvision adjusts the scope of intervention according to the extent of the execution deviation, preserving existing actions when possible and increasingly regenerating the suffix when the cached plan becomes unreliable.

A.3 Efficiency and Runtime Analysis

Table 9 reports the inference cost of the main ReEnvision components. The backbone diffusion policy requires approximately 110 ms per inference, while belief construction, RSSM updating, temporal impact prediction, and mask optimization are comparatively lightweight. In particular, the temporal impact model requires only about 1 ms, allowing belief-induced action changes to be estimated without an additional diffusion-policy evaluation. The complete ReEnvision pipeline requires approximately 145 ms per execution step, corresponding to about 32% additional end-to-end latency over the backbone. Full diffusion replanning, in contrast, requires an additional policy evaluation of approximately 110 ms when triggered. Since ReEnvision invokes full replanning only for severe deviations, most execution steps are handled through substantially cheaper keep or selective-repair operations.

Table 10: Hyperparameter sensitivity on CALVIN. Each parameter is varied independently while the others are fixed at their default values: $\tau = 10^{-4}$, $\rho = 0.6$, and $\eta = 0.9$. The default configuration is shown in bold.

Parameter	Value	Avg. Len. ↑	5-Step SR (%) ↑	Replan Rate (%)
τ	5×10^{-5}	4.50	85.0	8.0
	$\mathbf{1 \times 10^{-4}}$	**4.62**	**86.6**	**2.1**
	2×10^{-4}	4.55	82.0	3.0
ρ	0.4	4.59	87.0	12.0
	0.6	**4.62**	**86.6**	**2.1**
	0.8	4.57	86.0	4.0
η	0.7	4.53	83.0	15.0
	0.9	**4.62**	**86.6**	**2.1**
	1.0	4.53	85.0	3.0

A.4 Hyperparameter Sensitivity

We examine the sensitivity of ReEnvision to three parameters: the affected-action threshold τ, the maximum repair ratio ρ, and the uncertainty threshold η. The maximum number of repair segments is fixed at $s_{\max} = 1$. While τ controls offline supervision of the action impact model, ρ and η govern the online keep-repair-replan decision.

During training, τ defines the outcome-guided affected-action labels $m_k^* = \mathbb{I}[\|\Delta a_k^*\|_2 > \tau]$, the corresponding validity targets $v_k^* = 1 - m_k^*$, and the uncertainty targets. At inference, S_ψ predicts action corrections, validity, and uncertainty, which are used for structured mask selection without directly thresholding the predicted corrections using τ. The maximum uncertainty $u_{\max}$ is computed across the entire cached suffix. Full replanning is triggered when $u_{\max} \geq \eta$; otherwise, an empty mask retains the suffix, while a nonempty mask is selectively repaired when $r \leq \rho$ and $s \leq s_{\max}$. If either condition is violated, the suffix is fully replanned.

Sensitivity is evaluated on CALVIN using the same 5-step evaluation protocol as the main experiments. The default configuration is $\tau = 10^{-4}$, $\rho = 0.6$, $s_{\max} = 1$, and $\eta = 0.9$. For each value of τ, the validity and uncertainty targets are regenerated, and S_ψ is retrained using the same training protocol. All variants are evaluated using the same structured mask selection rule, with ρ and η fixed at their default values. To evaluate ρ and η, each inference threshold is varied independently while the action impact model trained with the default τ and all other parameters remain fixed. Specifically, we evaluate $\tau \in \{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\}$, $\rho \in \{0.4, 0.6, 0.8\}$, and $\eta \in \{0.7, 0.9, 1.0\}$.

As shown in Table 10, the default configuration achieves the highest average sequence length among the evaluated settings. Varying τ changes the offline supervision of S_ψ and consequently its predictions during execution. A smaller τ increases the replan rate to 8.0%, while a larger τ reduces the 5-step success rate to 82.0%. For the online decision thresholds, a stricter repair ratio of $\rho = 0.4$ increases the replan rate to 12.0%, with a slightly lower average sequence length despite a marginally higher 5-step success rate. Similarly, reducing η to 0.7 increases the replan rate to 15.0% and decreases the average sequence length to 4.53. These results illustrate how offline affected-action supervision and online intervention thresholds jointly influence repair behavior, with the default configuration achieving the highest average sequence length in this study.

B Theoretical Analysis

B.1 Uncertainty Target

For each future action, we define the magnitude of the outcome-guided correction as

$$s_k^* = \|\Delta a_k^*\|_2, \tag{10}$$

and determine whether the action is affected using

$$m_k^* = \mathbb{I}[s_k^* > \tau], \tag{11}$$

where $\tau > 0$ is the affected-action threshold.

Since predictions near the decision boundary $s_k^* = \tau$ are more ambiguous, we define the uncertainty target as

$$u_k^* = \exp\left(-\frac{|s_k^* - \tau|}{\tau}\right). \tag{12}$$

Proposition. The uncertainty target u_k^* reaches its maximum at $s_k^* = \tau$ and decreases monotonically as s_k^* moves away from the decision boundary.

Proof. Let

$$d_k = |s_k^* - \tau| \tag{13}$$

denote the distance from the decision boundary. Equation 12 can then be rewritten as

$$u_k^* = \exp\left(-\frac{d_k}{\tau}\right). \tag{14}$$

Because $\tau > 0$,

$$\frac{\partial u_k^*}{\partial d_k} = -\frac{1}{\tau}\exp\left(-\frac{d_k}{\tau}\right) < 0. \tag{15}$$

Therefore, u_k^* decreases monotonically with the distance d_k. When $s_k^* = \tau$, we have $d_k = 0$, yielding $u_k^* = 1$, which is the maximum value. Hence, actions closer to the affected-action decision boundary receive higher uncertainty targets, while actions farther from the boundary receive lower uncertainty targets. The uncertainty target therefore provides a monotonic measure of ambiguity around the affected-action decision boundary.

B.2 Controlled Belief Intervention

To isolate the effect of belief revision, we generate paired action chunks A^- and A^+ using the same frozen policy. The observation, task, diffusion noise, and cache state are held fixed, while only the belief changes from h^- to h^+. Their first $K-1$ actions, denoted by A^-_{rem} and A^+_{rem}, are temporally aligned with the cached suffix. For this analysis, the reference correction is defined as

$$\Delta A^*_{\text{rem}} = A^+_{\text{rem}} - A^-_{\text{rem}}. \tag{16}$$

Proposition. Suppose the paired actions are generated by the same frozen policy under identical observation, task, diffusion sampling condition, and cache state, with only the belief condition changing from h^- to h^+. Then, ΔA^*_{rem} isolates the policy's action response to the belief change $\Delta h = h^+ - h^-$. If the policy has a locally Lipschitz-continuous Jacobian with respect to its belief condition, this response admits a first-order approximation with a second-order remainder.

Proof. With all other policy inputs fixed, let P_{rem} select the first $K-1$ actions and define

$$F_{\text{rem}}(h) = P_{\text{rem}}\left(\pi(o_{t+1}, l, h; \xi)\right). \tag{17}$$

Under the stated matching conditions, the paired action difference satisfies

$$\Delta A^*_{\text{rem}} = F_{\text{rem}}(h^+) - F_{\text{rem}}(h^-). \tag{18}$$

Assuming a locally Lipschitz-continuous Jacobian, a first-order Taylor expansion gives

$$\Delta A^*_{\text{rem}} = J_h F_{\text{rem}}(h^-)\Delta h + \mathcal{O}\left(\|\Delta h\|_2^2\right). \tag{19}$$

For sufficiently small belief changes,

$$\Delta A^*_{\text{rem}} \approx J_h F_{\text{rem}}(h^-)\Delta h. \tag{20}$$

Thus, under the matching conditions, the paired target captures the frozen policy's sensitivity to belief changes. The temporal impact model S_ψ is trained on outcome-guided pairs to approximate belief-induced action changes without explicitly computing the policy Jacobian. During inference, pre-revision actions are retrieved directly from the cached suffix A_t^{rem}, and predicted corrections are applied only to selected actions without regenerating the pre-revision chunk.

B.3 Property of Selective Repair

Selective repair updates the cached action suffix according to

$$A_t^{\mathrm{rem}\prime} = A_t^{\mathrm{rem}} + \hat{\mathbf{m}} \odot \Delta \hat{A}_t^{\mathrm{rem}}, \tag{21}$$

where $\hat{\mathbf{m}} \in \{0,1\}^{K-1}$ denotes the selected repair mask, and $\Delta \hat{A}_t^{\mathrm{rem}}$ denotes the predicted correction for the cached suffix. Each $k = 1, \ldots, K-1$ indexes the corresponding future action a_{t+k}.

Proposition. Selective repair preserves all unselected actions and introduces no larger action perturbation than applying the same correction to the entire suffix. Specifically, for every k with $\hat{m}_k = 0$,

$$a'_{t+k} = a_{t+k}, \tag{22}$$

and, for any $p \geq 1$,

$$\|A_t^{\mathrm{rem}\prime} - A_t^{\mathrm{rem}}\|_p = \left\|\hat{\mathbf{m}} \odot \Delta \hat{A}_t^{\mathrm{rem}}\right\|_p \leq \left\|\Delta \hat{A}_t^{\mathrm{rem}}\right\|_p . \tag{23}$$

Proof. From Eq. 21, each remaining action is updated as

$$a'_{t+k} = a_{t+k} + \hat{m}_k \Delta \hat{a}_k. \tag{24}$$

When $\hat{m}_k = 0$, the correction vanishes and $a'_{t+k} = a_{t+k}$, so every unselected action is preserved.

Moreover, since $\hat{m}_k \in \{0,1\}$, masking cannot increase the magnitude of the applied correction. Therefore,

$$\left\|\hat{\mathbf{m}} \odot \Delta \hat{A}_t^{\mathrm{rem}}\right\|_p^p = \sum_{k=1}^{K-1} \hat{m}_k \, \|\Delta \hat{a}_k\|_p^p \leq \sum_{k=1}^{K-1} \|\Delta \hat{a}_k\|_p^p = \left\|\Delta \hat{A}_t^{\mathrm{rem}}\right\|_p^p . \tag{25}$$

Taking the p-th root gives Eq. 23. Thus, selective repair preserves unselected actions while restricting the action perturbation to the selected subset.