Feedback World Model Enables Precise Guidance of Diffusion Policy

Tuo An* **Jindou Jia*** **Gen Li** **Jingliang Li**
Chuhao Zhou **Pengfei Liu** **Bofan Lyu** **Jiaqi Bai**
Xinying Guo **Geng Li** **Jianfei Yang**

MARS Lab, Nanyang Technological University
antu0001@e.ntu.edu.sg, jianfei.yang@ntu.edu.sg
*Equal contribution.

Project page: https://ntumars.github.io/Feedback_World_Model/

Abstract

World models aim to improve robotic decision making by predicting the consequences of actions. However, in practice, their predictions often become unreliable once the robot encounters states outside the training distribution, limiting their effectiveness at deployment. We observe that execution itself provides a natural but underutilized signal: after each action, the robot directly observes the true next state, revealing the mismatch between predicted and actual outcomes. Building on this insight, we propose **feedback world model**, an inference-time mechanism that closes the loop between prediction and observation at inference time. Instead of treating the world model as a static open-loop predictor, our method maintains a lightweight feedback state that is updated online to iteratively correct future predictions, compensating for model errors using real-time observations without additional training data or parameter updates. We show that this process can be interpreted as a latent-space observer and admits convergence guarantees under bounded, asymptotically convergent dynamics residuals and a stable feedback gain. We further introduce action-aware guidance to better translate corrected predictions into control by emphasizing action-controllable components while suppressing irrelevant variations. Experiments on LIBERO-Plus, Robomimic, and real-world manipulation tasks demonstrate that our method substantially improves both prediction accuracy and policy performance under distribution shift, reducing world model prediction error by up to 76.4%. The same feedback mechanism improves the predictions of the pretrained foundation model Fast-WAM-Joint without updating its parameters, demonstrating its plug-and-play applicability. These results show that incorporating real-time feedback at inference time provides a simple yet powerful alternative to static world modeling.

1 Introduction

World models, which predict how environment states evolve under actions, have emerged as a promising tool for improving the robustness and generalization of robotic policies [4, 11, 22, 23, 28, 30]. Existing approaches primarily incorporate world models in two ways. One line integrates predicted dynamics, preferences, or rewards into policy training, using them as auxiliary supervision or optimization signals [16, 23, 24, 30]. The other leverages world models at inference time to evaluate or plan over candidate actions, directly influencing decision making during execution [25, 27]. Across both paradigms, the key idea is to move beyond pure behavior cloning by grounding policy learning

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and execution in predicted environment dynamics, encouraging actions that lead to plausible and task-consistent future states [19, 25].

However, these benefits become difficult to realize when policies are deployed beyond the training distribution. In robotic manipulation tasks, even small variations in initial pose, object configuration, or visual observation can shift the system into out-of-distribution (OOD) regimes. Under such shifts, the reliability of the world model becomes critical: inaccurate predictions can directly mislead both policy learning and test-time guidance [19]. A common strategy to improve robustness is to scale up data or model capacity, for example through large pretrained video models [28] or additional real-world rollouts for finetuning [16, 25]. While effective, this approach substantially increases training cost and is often impractical in data-limited robotic settings [28, 30]. Notably, deployment itself provides an underexplored signal: after each action, the robot observes the true next state, which directly exposes the mismatch between predicted and actual transitions. This real-time feedback offers a natural opportunity to improve prediction reliability without additional data collection or model scaling.

For guidance based on a frozen world model, conditioning each prediction on the latest observation leaves a further opportunity: retaining past prediction residuals to correct subsequent predictions. Such online correction can help the predictor track deployment conditions as an episode unfolds. This motivates our central question: **can a world model actively exploit real-time feedback during inference to maintain reliable prediction under distribution shift, without requiring additional data or larger models?**

To this end, we propose a feedback world model that leverages real-time observations during policy inference to correct future predictions online. Instead of operating as a static open-loop predictor, our model maintains a lightweight feedback state that is updated after each environment interaction. The observed mismatch between predicted and observed transitions is used to iteratively correct subsequent predictions, mitigating error accumulation without requiring additional training data or parameter updates. This mechanism can be interpreted as a latent-space observer with a convergence guarantee under the residual and stability assumptions. Furthermore, we introduce action-aware guidance strategy for diffusion policy inference. Rather than uniformly comparing predicted observations, we emphasize components that are more directly influenced by the robot's actions, such as end-effector motion and object pose. This focuses guidance on controllable, task-relevant changes while reducing interference from action-irrelevant variations.

We evaluate our method on LIBERO-Plus [6], Robomimic [17], and real-world manipulation tasks, with distribution shifts spanning robot initial states, lighting, backgrounds, object layouts, and sensor noise. The experiments examine both world model prediction accuracy and policy performance, and include feedback correction of a frozen pretrained Fast-WAM-Joint model across the full LIBERO-10 suite. Without additional training data or parameter updates, feedback reduces world-model prediction error by up to 76.4%. When combined with action-aware guidance, it improves policy robustness in simulation and on real robots, matching or exceeding the base policy's success rate in 18 of 20 task–perturbation settings on LIBERO-Plus.

To summarize, our contributions are threefold: (1) We propose a feedback world model that incorporates real-time observations as online corrective signals, correcting frozen predictions with low incremental feedback cost and a guarantee under explicit residual and stability conditions. (2) We introduce an action-aware guidance strategy that emphasizes controllable, action-relevant components to improve policy guidance. (3) Experiments in simulation and on real robots demonstrate improved prediction accuracy and policy performance under distribution shift, with further gains when transferring our feedback mechanism to the pretrained Fast-WAM-Joint model.

2 Related work

Efficient and accurate world models. Recent work on robotic world models has improved model quality along three directions: stronger latent representations, more accurate predictive dynamics, and more efficient architectures. Representation learning spans geometric and foundation-model approaches in robotics [2]. VLA-JEPA uses latent predictive learning [23], while DINO-WM learns dynamics over pretrained visual features [31]. Another improves predictive accuracy through iterative or closed-loop refinement, where world models are adapted using rollout data or coupled with downstream learning [7, 16]. In parallel, efficiency-oriented designs such as Fast-WAM [30],

DreamZero [28], and DDP-WM [29] reduce the cost of world or world-action modeling through latent-space prediction, architectural simplification, or faster inference.

Latent filtering and belief models. Alongside improvements in representations and dynamics, latent filtering reconciles model predictions with new observations. DVBF [15] learns state-space models through variational inference. PlaNet and Dreamer [8–10] jointly learn recurrent latent dynamics and observation-conditioned inference, using observations to update their latent state during execution. Our analytic feedback update adds residual correction to an already trained predictor without training an additional inference network or changing model parameters. This enables direct reuse of pretrained world models, while their corrected predictions provide more reliable guidance for diffusion policies.

Inference-time policy steering. Recent efforts steer pretrained policies at inference time for improved robustness. Since diffusion policies generate actions through iterative denoising [1], they naturally allow external objectives to be injected during sampling. Value-Guided Policy Steering [18] re-ranks sampled actions using a learned value function, DynaGuide [5] injects guidance from an external latent dynamics model, and latent policy barrier (LPB) [25] uses future latent predictions relative to an expert manifold. More structured guidance signals include contact-guided generation in hierarchical diffusion policy [26] and task-progress guidance in ProgressVLA [27].

3 Preliminaries

3.1 Action-conditioned world model for robot learning

In robot learning, world models are commonly used as action-conditioned predictive models that estimate how the robot-environment state evolves under candidate actions. In this work, we consider prediction in a latent state space. Let O_t denote the observation context available at time t, which may include the current observation as well as a finite history of previous observations. We encode this context into a latent state

$$z_t = \psi(O_t), \tag{1}$$

where ψ is the context encoder. For notational brevity, we denote a candidate future action sequence from time t as A_t. The latent world model f_θ then predicts the next latent state as

$$\hat{z}_{t+1} = f_\theta(z_t, A_t). \tag{2}$$

This formulation captures action-conditioned latent dynamics and enables rollout-based reasoning about the consequences of candidate robot actions [8–10].

3.2 Diffusion policy

Generative policies model conditional action distributions through diffusion [1] or flow matching [14]. We use a diffusion policy that generates action sequences through iterative denoising. The policy learns a time-dependent score function that estimates the gradient of the log-probability of actions conditioned on the observation and optional language instruction [20, 21].

Score function learning. A diffusion policy π_ϕ parameterized by ϕ learns the score function:

$$s_\phi(A_t^{(\tau)}, O_t, \ell, \tau) \approx \nabla_{A_t^{(\tau)}} \log p_\tau(A_t^{(\tau)} \mid O_t, \ell), \tag{3}$$

where $A_t^{(\tau)}$ is the noisy action at diffusion step τ, and ℓ denotes an optional language instruction. In practice, this is implemented via a denoising objective that trains the model to predict the score (or equivalently, the injected noise) at each diffusion step.

Guided denoising. Since the diffusion policy explicitly models the score function, additional objectives can be naturally incorporated at inference time via score guidance [3]. Given a task-oriented guidance term $g(A_t^{(\tau)}, O_t)$, the guided score is:

$$\tilde{s}(A_t^{(\tau)}, O_t, \ell, \tau) = s_\phi(A_t^{(\tau)}, O_t, \ell, \tau) + \lambda\, g(A_t^{(\tau)}, O_t), \tag{4}$$

where λ controls the strength of guidance. This formulation enables integrating auxiliary signals, such as those derived from world models, into the action generation process [12, 25].

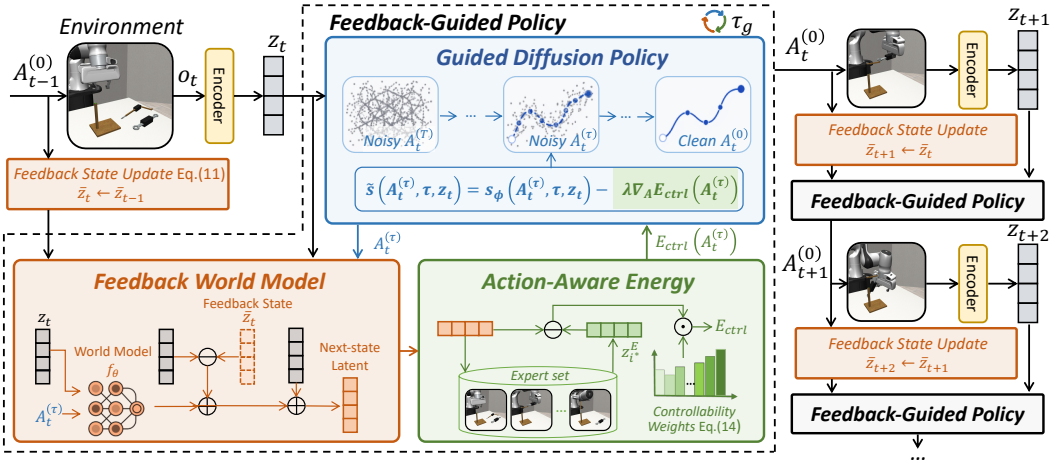


Figure 1: **Overview of the Feedback-Guided Policy.** During denoising, the feedback world model predicts the future latent outcome of the current action trajectory, and an action-aware energy guides the policy toward expert-like, action-relevant states. After each execution, the new observation updates the feedback state, which corrects subsequent feedback world model predictions and forms an outer loop for reducing prediction drift during inference.

4 Methodology

4.1 Framework overview

We first provide an overview of the proposed *Feedback-Guided Policy* (Fig. 1), an inference framework that uses online observation feedback to construct reliable and action-aware policy guidance. At each denoising step τ, a candidate action sequence $A_t^{(\tau)}$ is evaluated through a learned world model. Following prior world model guidance methods [25], the predicted next latent state $\hat{z}_{t+1}(A_t^{(\tau)})$ is matched to the nearest expert latent in $\mathcal{Z}^E = \{z_i^E\}_{i=1}^N$ with N samples:

$$i^\star = \arg\min_i \left\| \hat{z}_{t+1}(A_t^{(\tau)}) - z_i^E \right\|_2^2. \tag{5}$$

The corresponding guidance energy is then defined as the squared latent distance to this nearest expert state:

$$E(A_t^{(\tau)}) = \left\| \hat{z}_{t+1}(A_t^{(\tau)}) - z_{i^\star}^E \right\|_2^2. \tag{6}$$

This energy encourages candidate actions whose predicted outcomes are closer to expert-like latent states. It is used to guide diffusion sampling as

$$\tilde{s} = s_\phi - \lambda \nabla_{A_t^{(\tau)}} E(A_t^{(\tau)}), \tag{7}$$

where λ controls the guidance strength, and $\nabla_{A_t^{(\tau)}} E(A_t^{(\tau)})$ denotes the gradient of the guidance energy with respect to the candidate action sequence.

This standard formulation is limited by two factors: the predicted latent state may drift under distribution shift, and uniform latent matching may include action-irrelevant components in the guidance signal. We address these issues with two complementary designs. First, a *feedback world model* uses online observations to correct latent predictions during inference. Second, an *action-aware energy* reweights latent dimensions according to their estimated action controllability. Together, they produce guidance based on predictions that are both dynamically reliable and action-relevant. Algorithm 1 in Appendix E summarizes the full inference pipeline. In the following subsections, we detail the feedback world model and action-aware energy, respectively.

4.2 Feedback world model

As discussed in Sec. 4.1, world-model-based guidance relies on the predicted latent $\hat{z}_{t+1}$ to construct the energy signal. In standard usage, the pretrained world model f_θ predicts in an open-loop manner according to Eq. (2); prediction errors under distribution shift can therefore bias guidance.

To improve prediction reliability, we introduce a *feedback world model* that uses real observations to correct latent predictions *without updating* the pretrained world model f_θ. Following [13], we reinterpret the learned transition model in velocity form:

$$v_t(A_t^{(\tau)}) \triangleq \frac{f_\theta(z_t, A_t^{(\tau)}) - z_t}{\delta t} = \frac{\hat{z}_{t+1} - z_t}{\delta t}, \tag{8}$$

where δt is the environment update interval. In addition to the observed latent state z_t, we maintain an auxiliary feedback state $\bar{z}_t$ (initialized as z_0) that tracks the model's internal *belief* about where the system should be after past executed actions. The discrepancy

$$e_t = z_t - \bar{z}_t \tag{9}$$

therefore measures the accumulated prediction discrepancy up to the current environment step.

During diffusion guidance at step t, this discrepancy is held fixed and used to correct the score function for every noisy action $A_t^{(\tau)}$:

$$\begin{cases} \hat{v}_t(A_t^{(\tau)}) = v_t(A_t^{(\tau)}) + Le_t, \\ z_{t+1}^{\mathrm{fb}}(A_t^{(\tau)}) = z_t + \delta t \cdot \hat{v}_t(A_t^{(\tau)}), \end{cases} \tag{10}$$

where L is the positive-definite feedback gain. The corrected prediction $z_{t+1}^{\mathrm{fb}}(A_t^{(\tau)})$ replaces the open-loop prediction $\hat{z}_{t+1}(A_t^{(\tau)})$ in the guidance energy of Eq. (6). After the guided clean action $A_t^{(0)}$ is executed in the environment, the feedback state is propagated using the same action-conditioned corrected velocity:

$$\bar{z}_{t+1} = \bar{z}_t + \delta t \cdot \hat{v}_t(A_t^{(0)}). \tag{11}$$

After the environment returns the next observation o_{t+1}, we obtain the corresponding observed latent state z_{t+1}. This latent state reflects the actual transition reached after executing $A_t^{(0)}$, and is used to form the next feedback residual:

$$e_{t+1} = z_{t+1} - \bar{z}_{t+1}, \tag{12}$$

which is carried forward and used to correct predictions at step $t{+}1$. Thus, the action $A_t^{(0)}$ and the observation z_{t+1} are aligned across one environment transition: $A_t^{(0)}$ advances the internal *belief* to $\bar{z}_{t+1}$, and the subsequently observed z_{t+1} calibrates the residual signal used at the next decision step.

Convergence guarantee. Assume a symmetric positive-definite gain L, a bounded latent-velocity residual $\|\Delta_t\| \le \gamma$ with $\Delta_t \to \Delta_\infty$, and, for the discrete update, $\rho(I - \delta t L) < 1$. Then

$$\lim_{t\to\infty} \|z_t - \bar{z}_t\| \le \frac{\gamma}{\lambda_{\min}(L)}. \tag{13}$$

For $L = lI$, the bound is γ/l. The proof and the corrected one-step prediction result appear in Appendix A.

Gain selection. We use $L = lI$ and compare a coarse set of gains using held-out prediction MSE. For an update interval δt, the non-oscillatory homogeneous range is $0 < l \le 1/\delta t$; the effective gain is $\delta t l$. Appendix M reports task settings and sensitivity of prediction accuracy to l (Table 12).

4.3 Action-aware guidance

World-model-based guidance in Eq. (7) depends not only on prediction accuracy, but also on whether the guidance objective emphasizes action-relevant state changes. Uniform latent matching is suboptimal because latent representations often entangle controllable factors, such as robot-object geometry, with weakly controllable factors, such as background appearance and lighting. We therefore propose an *action-aware energy* that reweights latent dimensions according to their controllability.



Figure 2: **Simulated tasks.** For LIBERO-Plus, we evaluate four representative single-task settings from the *Kitchen* and *Living Room* (LR) scenes, with further details provided in Appendix C. For Robomimic tasks, we evaluate three representative tasks including Transport, Square and Tool-Hang.

Specifically, we estimate the controllability of the j-th latent dimension by its *counterfactual variance* under the learned world model:

$$w_j = \frac{1}{M} \sum_{i=1}^{M} \mathrm{Var}_{a \sim \mathcal{A}_{\mathrm{demo}}}([f_\theta(z_i,\, a)]_j)\,, \quad j \in \{1, \ldots, D\}, \tag{14}$$

where $\{z_i\}_{i=1}^{M}$ are observations drawn at uniformly spaced indices along the demonstration trajectory, and the variance is taken over actions sampled from the demonstration action pool $\mathcal{A}_{\mathrm{demo}}$. A larger w_j indicates that the corresponding latent dimension is more responsive to action perturbations and is therefore more relevant for action guidance. In practice we approximate (14) with M=200 states and 32 action samples per state.

We normalize the weights and apply a soft interpolation to preserve the overall scale of the energy:

$$\bar{w}_j = \frac{w_j}{\frac{1}{D}\sum_{k=1}^{D} w_k + \epsilon}, \qquad w_j^{(\beta)} = 1 + \beta\,(\bar{w}_j - 1), \tag{15}$$

where $\epsilon = 10^{-8}$ is a small constant added for numerical stability that prevents division by zero in the unlikely event that the average counterfactual variance vanishes, and $\beta \in [0, 1]$ controls the strength of controllability modulation. The final guidance loss replaces the standard MSE with the controllability-weighted form

$$E_{\mathrm{ctrl}}(A_t^{(\tau)}) = \sum_{j=1}^{D} w_j^{(\beta)}\, (z_{t+1,j}^{\mathrm{fb}}(A_t^{(\tau)}) - z_{i^\star,j}^{E})^2. \tag{16}$$

The gradient $\nabla_A E_{\mathrm{ctrl}}$ is then used as the guidance term in Eq. (7), so action-sensitive latent dimensions contribute more strongly to denoising guidance, while action-invariant dimensions are down-weighted.

5 Experiments

We conduct experiments mainly to answer two questions. First, can the proposed feedback world model reduce prediction error online in OOD settings? Second, when combined with controllability-aware guidance, can the improved feedback world model further enhance the OOD success rate of diffusion policies? In simulation, we evaluate robustness to perturbations in robot initial states, lighting, background textures, object layouts, and sensor noise. On real robots, we test pick-and-place and drawer-opening tasks under initial-pose shifts. We also apply feedback to the frozen pretrained foundation model Fast-WAM-Joint across the full LIBERO-10 suite in LIBERO-Plus to assess its plug-and-play applicability.

5.1 Experimental setup

Simulation benchmarks. We evaluate our method on seven simulated manipulation tasks from Robomimic [17] and LIBERO-Plus [6], as shown in Fig. 2. For Robomimic, we use three representative tasks, Square, Tool-Hang, and Transport, and train the diffusion policy in a low-data regime using 20% of the official demonstrations. OOD evaluation on Robomimic perturbs the robot's initial state. For LIBERO-Plus, we use four single-task settings from LIBERO-10 and evaluate all five

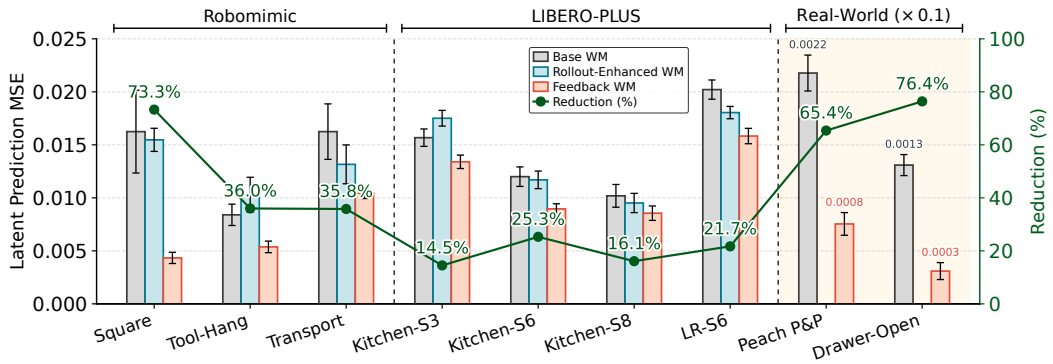


Figure 3: **Latent prediction MSE under robot-initial-state OOD.** Simulation: step-pooled means $\pm$ episode-cluster bootstrap SE; real world: episode-equal means $\pm$ SEM. The green curve shows the relative reduction from Base to Feedback. Real-world bars and error bars are magnified $10\times$ for display. Counts and estimators are detailed in Appendix G.

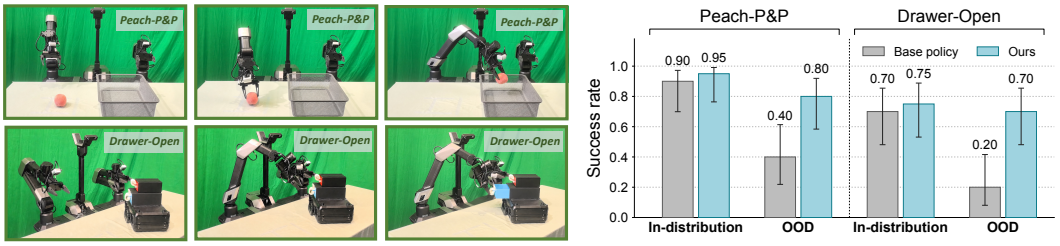


Figure 4: **Real-world tasks and results.** We deploy the policy on Peach-P&P and Drawer-Open tasks under both in-distribution and out-of-distribution initial states. Each point uses 20 binary trials; error bars are 95% Wilson intervals.

perturbation categories listed above. For each simulated task, we train a base world model on the corresponding expert demonstrations and keep its parameters fixed during evaluation. Task selection and perturbation protocols are detailed in Appendices B, C, and H; statistical reporting and the pretrained-model setup appear in Appendices G and J, respectively.

Real-world tasks. We further evaluate on two real-world manipulation tasks, Peach-P&P and Drawer-Open, as shown in Fig. 4. Peach-P&P requires picking a peach from the table and placing it into a target box, while Drawer-Open requires handle localization and contact-rich pulling with a deformable rubber handle. We collect 50 and 65 expert trajectories for the two tasks, respectively, and train both the diffusion policy and world model from scratch. At test time, we reset the robot arm to abnormal initial poses outside the training distribution, yielding a real-world robot-initial-state OOD setting.

5.2 Experimental results

5.2.1 Feedback world model improves online prediction.

We first evaluate whether the proposed feedback mechanism improves world model prediction under robot initial-state OOD shifts. Fig. 3 shows latent prediction MSE under simulated and real-world OOD settings. The base world model incurs large errors under perturbed initial states. By correcting its internal predictive state with online observations, our feedback world model reduces mean prediction error without extra training data or parameter updates. The reduction reaches 73.3% on Robomimic Square and 65.4%/76.4% on the real-world Peach-P&P and Drawer-Open tasks, confirming improved prediction reliability under OOD conditions.

Table 1: **Success rates on simulated OOD manipulation tasks.** Baselines include *Base* policy, *Mixed BC* using expert and rollout data, *Filtered BC* removing failed rollouts, *Std. Guid.* using the base world model, and *LPB-style Rollout Guidance* using a rollout-enhanced world model. Best and second-best results are bolded and underlined. Rollout details appear in Appendix D.

Benchmark	Task	Base	Mixed BC	Filtered BC	Std. Guid.	Rollout Guid.	Ours
Robomimic	Square	0.36	0.08	0.12	0.40	<u>0.44</u>	**0.46**
	Tool-Hang	0.27	0.22	0.22	0.22	<u>0.31</u>	**0.36**
	Transport	0.34	0.26	0.36	0.46	<u>0.52</u>	**0.60**
LIBERO-Plus	Kitchen-S3	0.55	0.57	**0.70**	0.61	0.50	<u>0.68</u>
	Kitchen-S6	0.45	0.48	<u>0.54</u>	0.48	0.50	**0.55**
	Kitchen-S8	0.59	0.42	0.35	<u>0.62</u>	<u>0.62</u>	**0.65**
	LR-S6	<u>0.26</u>	0.24	0.24	0.21	0.24	**0.36**
Average		0.40	0.32	0.36	0.43	<u>0.45</u>	**0.52**

Controlled action interventions and physical-state probes assess the action dependence and physical relevance of these predictions (Appendix I). Comparisons with Kalman and learned filtering baselines appear in Appendix K; Appendix L analyzes how the model residual varies during execution.

5.2.2 Action-aware guidance improves controllable policy guidance.

While the feedback world model improves prediction reliability, effective policy guidance also requires focusing the guidance objective on action-controllable state components. To examine this, we visualize the distribution of action controllability in the world model latent space (Fig. 5). Across three Robomimic tasks, the sorted weights show a clear long-tailed pattern: the top 5% of dimensions contribute about 20–25$\%$ of the total controllability weight, while many dimensions have much smaller weights. The per-latent-observation variance map further shows that high-weight dimensions remain consistently action-responsive across observations. These results reveal strong action anisotropy in the latent representation, supporting our design choice to emphasize action-controllable dimensions during guidance.

We further ablate action-aware guidance on Robomimic tasks, as shown in Table 2. Compared with guidance using only the feedback world model, adding action-aware weighting further improves policy success rates. By downweighting action-irrelevant visual components, action-aware guidance focuses action optimization on task-relevant state changes and improves downstream policy success.

5.2.3 Combined guidance improves OOD performance across benchmarks.

Having verified the effectiveness of feedback correction and action-aware weighting separately, we next evaluate their combined effect on policy performance across broader simulated and real-world OOD settings. Table 1 and Fig. 4 report policy success rates in simulated and real-world OOD settings. The simulated results average 52% for our method, compared with 40% for the base policy, 43% for standard guidance and 45% for LPB-style rollout guidance. Although Filtered BC slightly outperforms ours on Kitchen-S3, rollout-based augmentation is less stable overall: Mixed BC often degrades performance, and rollout-enhanced guidance improves the average success rate only moderately compared with standard guidance. This suggests that simply adding rollout data does not necessarily resolve OOD generalization, since rollout trajectories may contain failure cases, biased state distributions, or insufficient coverage of the final test-time perturbations.

Real-world results further confirm the benefit under robot initial-state shifts. Under in-distribution states, our method slightly improves the already strong base policy, from 90% to 95% on Peach-P&P and from 70% to 75% on Drawer-Open. Under OOD states, the gains become much larger, improving success from 40% to 80% and from 20% to 70%, respectively. Overall, the results suggest that reliable next-state prediction from the feedback world model, together with fine-grained action-aware guidance, leads to more robust policy performance under distribution shift.

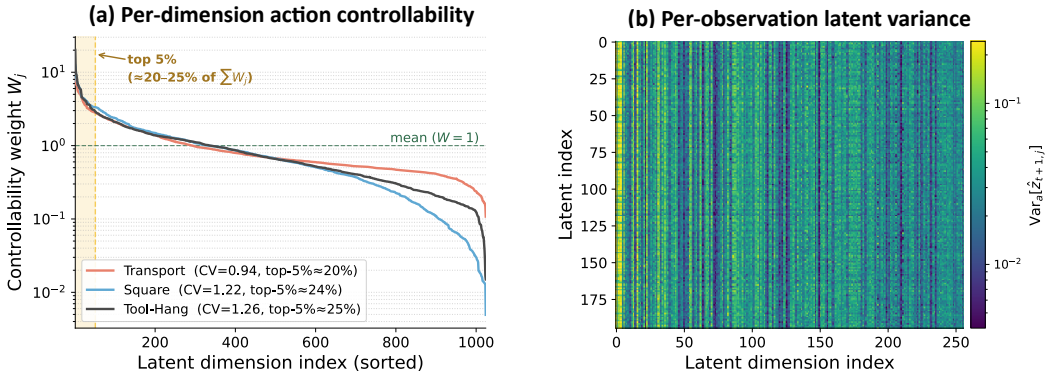


Figure 5: **Action controllability of the world model latent.** **(a)** Sorted controllability weights w_j on three Robomimic tasks, shown in log scale. **(b)** Counterfactual action variance $\mathrm{Var}_a[\hat{z}_{t+1,j}]$ on Transport, computed over sampled observations and 256 selected latent dimensions. Persistent vertical bands indicate dimensions that remain action-responsive across observations, supporting the use of an offline-frozen W at inference time.

Table 2: **Ablation study.** + *Feedback WM* uses the feedback world model for guidance without per-dimension weighting; + *Action-aware* additionally uses the offline counterfactual-variance score W.

Method	Square	Transport	Tool-Hang
Base	0.36	0.34	0.27
+ Feedback WM	0.40	0.54	0.32
+ Action-aware	**0.46**	**0.60**	**0.36**

5.2.4 Robustness across LIBERO-Plus perturbations.

We evaluate the four LIBERO-Plus tasks under perturbations to robot initial states, lighting, background textures, object layout, and sensor noise. Across these five categories, pooled success increases by 4.5–9.9 percentage points and latent MSE decreases by 6.8–47.8% (Table 3). Our method matches or exceeds the base policy in 18 of the 20 task–perturbation settings. Appendix H reports the results for each setting.

5.2.5 Transfer to a pretrained world-action model.

To test the plug-and-play applicability of feedback, we apply our feedback mechanism to Fast-WAM-Joint [30], built on the pretrained Wan2.2-5B backbone. The model uses two-camera observations and a single multi-task policy across the full LIBERO-10 suite. We keep its architecture and learned parameters frozen, introducing feedback at inference time without retraining or additional training data.

World model prediction. We evaluate 7,553 paired predictions over all 393 robot-initial-state OOD variants and 1,203 ID predictions from 100 episodes. Fig. 6 shows that feedback reduces latent MSE under both ID and OOD conditions at all three tested gains. At gain 0.5, ID MSE decreases from 0.1772 to 0.1635 (7.8%), and OOD MSE decreases from 0.3015 to 0.2464 (18.3%). Against VAE-reconstructed observations, OOD pixel MSE decreases by 18.0% and PSNR increases by 0.86 dB. Together, these results show that inference-time feedback improves the predictions of a pretrained foundation model in both latent and image space, while preserving its learned parameters.

Policy performance. On the LIBERO-Plus Robot Initial States OOD evaluation across LIBERO-10, our method improves the success rate of Fast-WAM-Joint from 77.1% to 81.4%, a gain of 4.3 percentage points. The prediction and policy results demonstrate that our method can be applied to an existing pretrained world-action model without updating its parameters. Appendix J provides the evaluation details.

Table 3: **LIBERO-Plus evaluation across five perturbation categories.** Success rates are pooled within perturbation categories; MSE reduction is relative to the base world model. ΔSR is in percentage points (pp).

Perturbation	Base SR	Ours SR	ΔSR	MSE reduction
Robot initial states	45.7%	55.6%	+9.9	19.4%
Lighting	42.2%	46.7%	+4.5	47.8%
Background textures	59.5%	64.9%	+5.4	42.2%
Object layout	67.2%	75.2%	+8.0	12.3%
Sensor noise	81.8%	86.6%	+4.8	6.8%

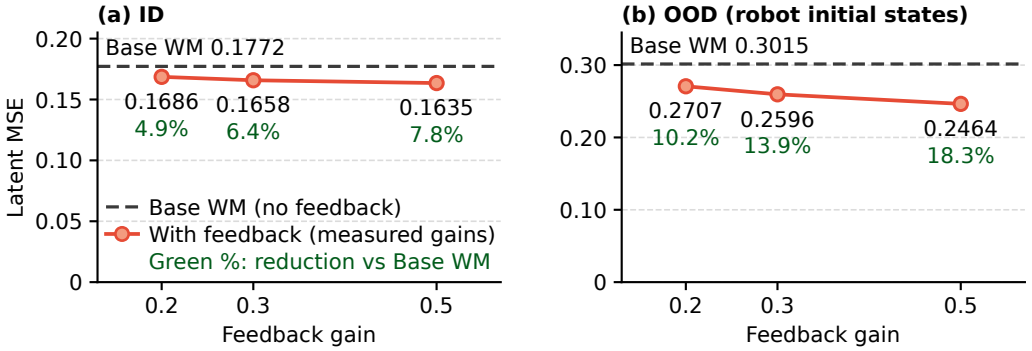


Figure 6: **Feedback improves frozen Fast-WAM-Joint predictions.** Latent MSE on LIBERO-10 under ID and robot-initial-state OOD conditions. Dashed lines show the base model; markers show feedback at each tested gain. Green labels give the relative MSE reduction.

6 Conclusion

In this work, we introduce a feedback world model for robust test-time guidance of diffusion policies. Instead of using a pretrained world model as a static open-loop predictor, our method exploits real observations collected during policy execution to correct the model's latent predictive state online, improving prediction reliability without additional training data or parameter updates. We further propose action-aware guidance, which weights latent prediction errors by offline-estimated controllability to focus guidance on action-sensitive state components. Extensive simulated and real-world experiments show that combining feedback correction with action-aware guidance leads to more robust policy execution in OOD manipulation settings.

Limitations and future work. Similar to prior inference-time guidance methods for diffusion policies [25, 27], our approach has higher inference latency than an unguided base diffusion policy because guidance is applied during denoising. Since the feedback update is analytic and the controllability weights are computed offline, most overhead comes from the guided-diffusion paradigm rather than from our proposed components. Future work can apply these ideas to more efficient guidance frameworks and latency-sensitive robotic tasks.

Acknowledgments and Disclosure of Funding

This work is supported by MOE Singapore Tier 1 Grant RG83/25, RS36/24, and a Start-up Grant from Nanyang Technological University. The authors declare no competing interests.

References

[1] 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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[2] Tianchen Deng, Yue Pan, Shenghai Yuan, Dong Li, Chen Wang, Mingrui Li, Long Chen, Lihua Xie, Danwei Wang, Jingchuan Wang, Javier Civera, Hesheng Wang, and Weidong Chen. What is the best 3d scene representation for robotics? from geometric to foundation models. *arXiv preprint arXiv:2512.03422*, 2025.

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Proceedings of Advances in Neural Information Processing Systems*, pages 8780–8794, 2021.

[4] Jiahua Dong, Qi Lyu, Baichen Liu, Xudong Wang, Wenqi Liang, Duzhen Zhang, Jiahang Tu, Hongliu Li, Hanbin Zhao, Henghui Ding, Yulun Zhang, Zhi Han, Nicu Sebe, Fahad Shahbaz Khan, Salman Khan, Mubarak Shah, Philip Torr, Ming-Hsuan Yang, and Dacheng Tao. Learning to model the world: A survey of world models in artificial intelligence. *TechRxiv*, 2026.

[5] Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance. *arXiv preprint arXiv:2506.13922*, 2025.

[6] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[7] Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world. In *Proceedings of Conference on Robot Learning*, pages 425–445. PMLR, 2023.

[8] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[9] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[10] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *Proceedings of International Conference on Learning Representations*, 2021.

[11] Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, Oier Mees, Marc Pollefeys, Zhuang Liu, Jiajun Wu, Pieter Abbeel, Jitendra Malik, Yilun Du, and Jianfei Yang. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.

[12] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[13] Jindou Jia, Zihan Yang, Meng Wang, Kexin Guo, Jianfei Yang, Xiang Yu, and Lei Guo. Feedback favors the generalization of neural ODEs. In *Proceedings of International Conference on Learning Representations*, 2025.

[14] Jindou Jia, Gen Li, Xiangyu Chen, Tuo An, Yuxuan Hu, Jingliang Li, Xinying Guo, and Jianfei Yang. Action-to-action flow matching. *arXiv preprint arXiv:2602.07322*, 2026.

[15] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1605.06432.

[16] Xiaokang Liu, Zechen Bai, Hai Ci, Kevin Yuchen Ma, and Mike Zheng Shou. World-vla-loop: Closed-loop learning of video world model and vla policy. *arXiv preprint arXiv:2602.06508*, 2026.

[17] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of Conference on Robot Learning*, pages 1678–1690. PMLR, 2022.

[18] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Proceedings of Conference on Robot Learning*, pages 4996–5013. PMLR, 2025.

[19] Julian Quevedo, Percy Liang, and Sherry Yang. Evaluating robot policies in a world model. *arXiv preprint arXiv:2506.00613v1*, 2025. URL https://arxiv.org/abs/2506.00613v1.

[20] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of Advances in Neural Information Processing Systems*, 2019.

[21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of International Conference on Learning Representations*, 2021.

[22] Yue Su, Sijin Chen, Haixin Shi, Mingyu Liu, Zhengshen Zhang, Ningyuan Huang, Weiheng Zhong, Zhengbang Zhu, Yuxiao Liu, and Xihui Liu. World guidance: World modeling in condition space for action generation. *arXiv preprint arXiv:2602.22010*, 2026.

[23] Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

[24] Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, et al. Atomvla: Scalable post-training for robotic manipulation via predictive latent world models. *arXiv preprint arXiv:2603.08519*, 2026.

[25] Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. In *Proceedings of Advances in Neural Information Processing Systems*, 2025.

[26] Dexin Wang, Chunsheng Liu, Faliang Chang, and Yichen Xu. Hierarchical diffusion policy: Manipulation trajectory generation via contact guidance. *IEEE Transactions on Robotics*, 41: 2086–2104, 2025.

[27] Hongyu Yan, Qiwei Li, Jiaolong Yang, and Yadong Mu. Progressvla: Progress-guided diffusion policy for vision-language robotic manipulation. *arXiv preprint arXiv:2603.27670*, 2026.

[28] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[29] Shicheng Yin, Kaixuan Yin, Weixing Chen, Yang Liu, Guanbin Li, and Liang Lin. Ddp-wm: Disentangled dynamics prediction for efficient world models. *arXiv preprint arXiv:2602.01780*, 2026.

[30] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

[31] Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of International Conference on Machine Learning*, pages 79115–79135. PMLR, 2025.

A Proof of the convergence guarantee

We provide the convergence analysis of the proposed feedback world model. Let $z_t = \psi(O_t)$ denote the observed latent state and let $\bar{z}_t$ denote the auxiliary feedback state. For the executed action $A_t^{(0)}$, we write the true latent transition in velocity form as

$$z_{t+1} = z_t + \delta t \, v_t^\star(A_t^{(0)}), \tag{17}$$

where $v_t^\star(A_t^{(0)})$ is the true latent velocity induced by the environment. The learned world model gives

$$v_t(A_t^{(0)}) = \frac{f_\theta(z_t, A_t^{(0)}) - z_t}{\delta t}. \tag{18}$$

We assume that the learned latent velocity has a bounded and asymptotically convergent residual error:

$$v_t^\star(A_t^{(0)}) = v_t(A_t^{(0)}) + \Delta_t, \qquad \|\Delta_t\| \le \gamma, \qquad \Delta_t \to \Delta_\infty, \tag{19}$$

where $\gamma > 0$ is an unknown constant and $\|\Delta_\infty\| \le \gamma$. This assumption states that the learned latent dynamics may have a nonzero residual error, but the residual remains bounded and approaches a steady limiting disturbance along the deployment trajectory.

The feedback world model corrects the learned velocity by

$$\hat{v}_t(A_t^{(0)}) = v_t(A_t^{(0)}) + L(z_t - \bar{z}_t), \tag{20}$$

where $L = L^\top \succ 0$ is the feedback gain. The auxiliary feedback state is updated as

$$\bar{z}_{t+1} = \bar{z}_t + \delta t \, \hat{v}_t(A_t^{(0)}). \tag{21}$$

Define the feedback estimation error as

$$e_t = z_t - \bar{z}_t. \tag{22}$$

Combining Eq. (17), Eq. (19), Eq. (20), and Eq. (21), we obtain

$$\begin{aligned} e_{t+1} &= z_{t+1} - \bar{z}_{t+1} \\ &= z_t + \delta t\big(v_t(A_t^{(0)}) + \Delta_t\big) - \Big[\bar{z}_t + \delta t\big(v_t(A_t^{(0)}) + L(z_t - \bar{z}_t)\big)\Big] \\ &= (I - \delta t L)e_t + \delta t \, \Delta_t. \end{aligned} \tag{23}$$

Thus, the feedback residual follows a stable linear system driven by the latent dynamics residual.

We first analyze the continuous-time counterpart of Eq. (23), which is

$$\dot{e}(t) = -Le(t) + \Delta(t), \qquad \|\Delta(t)\| \le \gamma, \qquad \Delta(t) \to \Delta_\infty. \tag{24}$$

Since $L \succ 0$, the homogeneous system $\dot{e}(t) = -Le(t)$ is exponentially stable. The solution of Eq. (24) is

$$e(t) = e^{-Lt}e(0) + \int_0^t e^{-L(t-s)}\Delta(s)\, ds. \tag{25}$$

The first term converges to zero because all eigenvalues of L are positive:

$$\lim_{t\to\infty} e^{-Lt}e(0) = 0. \tag{26}$$

For the second term, since $\Delta(t) \to \Delta_\infty$, the forced response of the stable linear system converges to the equilibrium induced by the limiting residual:

$$\lim_{t\to\infty} e(t) = L^{-1}\Delta_\infty. \tag{27}$$

Therefore,

$$\lim_{t\to\infty} \|e(t)\| = \|L^{-1}\Delta_\infty\| \le \|L^{-1}\|_2\|\Delta_\infty\| \le \frac{\gamma}{\lambda_{\min}(L)}. \tag{28}$$

Since $e(t) = z(t) - \bar{z}(t)$, we obtain

$$\lim_{t\to\infty} \|z(t) - \bar{z}(t)\| \le \frac{\gamma}{\lambda_{\min}(L)}. \tag{29}$$

For the scalar-gain case $L = lI$ with $l > 0$, this reduces to

$$\lim_{t\to\infty} \|z(t) - \bar{z}(t)\| \le \frac{\gamma}{l}. \tag{30}$$

For the discrete-time update used in our implementation, Eq. (23) can be written as

$$e_{t+1} = Ae_t + \delta t\, \Delta_t, \qquad A = I - \delta t L. \tag{31}$$

If the feedback gain satisfies

$$\rho(A) = \rho(I - \delta t L) < 1, \tag{32}$$

where $\rho(\cdot)$ denotes the spectral radius, then the discrete error system is exponentially stable in the absence of residual input. Unrolling Eq. (31) gives

$$e_t = A^t e_0 + \delta t \sum_{k=0}^{t-1} A^{t-1-k} \Delta_k. \tag{33}$$

Because $\rho(A) < 1$, we have $A^t e_0 \to 0$. Moreover, since $\Delta_t \to \Delta_\infty$, the input response converges to the steady-state response of the stable linear system:

$$\lim_{t\to\infty} e_t = \delta t \sum_{j=0}^{\infty} A^j \Delta_\infty = \delta t (I - A)^{-1} \Delta_\infty. \tag{34}$$

Since $I - A = \delta t L$, this becomes

$$\lim_{t\to\infty} e_t = L^{-1} \Delta_\infty. \tag{35}$$

Therefore,

$$\lim_{t\to\infty} \|e_t\| = \|L^{-1}\Delta_\infty\| \le \frac{\gamma}{\lambda_{\min}(L)}. \tag{36}$$

Equivalently,

$$\lim_{t\to\infty} \|z_t - \bar{z}_t\| \le \frac{\gamma}{\lambda_{\min}(L)}. \tag{37}$$

In particular, when $L = lI$ and $0 < \delta t\, l < 2$, the stability condition $\rho(I - \delta t L) < 1$ holds. If additionally $0 < \delta t\, l \le 1$, then

$$\|e_{t+1}\| \le (1 - \delta t\, l)\|e_t\| + \delta t\, \|\Delta_t\|. \tag{38}$$

Since $\Delta_t \to \Delta_\infty$, the exact scalar-gain error dynamics converges to

$$\lim_{t\to\infty} e_t = \frac{1}{l}\Delta_\infty, \tag{39}$$

and hence

$$\lim_{t\to\infty} \|z_t - \bar{z}_t\| = \frac{1}{l}\|\Delta_\infty\| \le \frac{\gamma}{l}. \tag{40}$$

Finally, we relate this observer convergence result to the corrected one-step prediction used for policy guidance. For the executed action $A_t^{(0)}$, the corrected prediction is

$$z_{t+1}^{\mathrm{fb}}(A_t^{(0)}) = z_t + \delta t\big(v_t(A_t^{(0)}) + Le_t\big). \tag{41}$$

Using Eq. (17) and Eq. (19), its prediction error satisfies

$$\begin{aligned} z_{t+1}^{\mathrm{fb}}(A_t^{(0)}) - z_{t+1} &= z_t + \delta t\big(v_t(A_t^{(0)}) + Le_t\big) - z_t - \delta t\big(v_t(A_t^{(0)}) + \Delta_t\big) \\ &= \delta t\big(Le_t - \Delta_t\big). \end{aligned} \tag{42}$$

Since $e_t \to L^{-1}\Delta_\infty$ and $\Delta_t \to \Delta_\infty$, we have

$$\lim_{t\to\infty} (Le_t - \Delta_t) = LL^{-1}\Delta_\infty - \Delta_\infty = 0. \tag{43}$$

Therefore,

$$\lim_{t\to\infty} \left\| z_{t+1}^{\mathrm{fb}}(A_t^{(0)}) - z_{t+1} \right\| = 0. \tag{44}$$

This result shows that, under a bounded and asymptotically convergent latent dynamics residual, the auxiliary feedback state converges to a bounded neighborhood of the observed latent state, with radius controlled by the residual level and the feedback gain. Moreover, the corrected one-step prediction used for guidance becomes asymptotically unbiased for the executed action. Hence, the feedback world model prevents deployment-time prediction bias from growing unboundedly, and the remaining observer bias is controlled by the learned latent dynamics residual and the feedback gain.

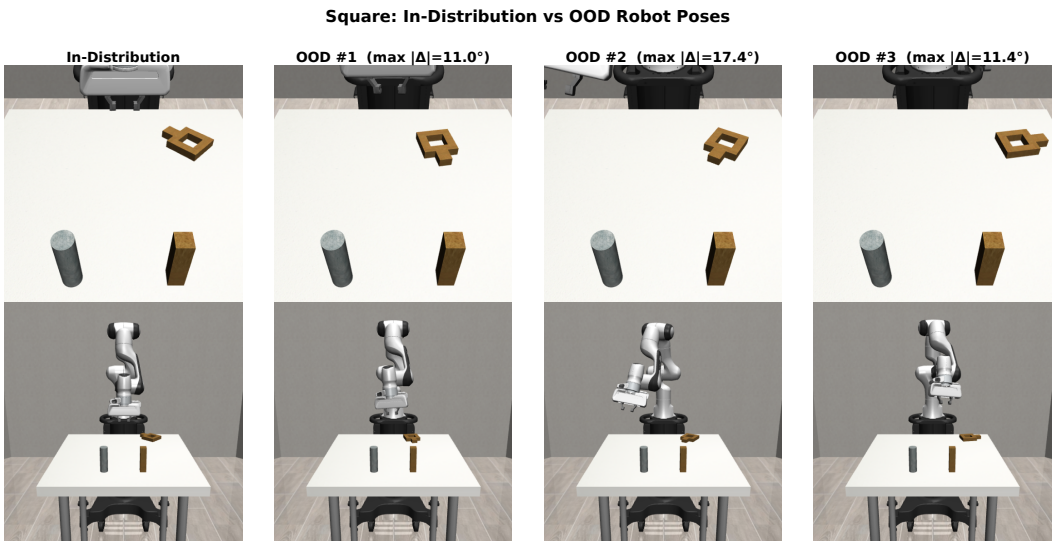


Figure 7: **Square OOD initial states.** The leftmost column shows the in-distribution demonstration start. The remaining columns show OOD starts generated by perturbing only the robot arm joints. Top row: agentview camera; bottom row: front-view camera.

B Robomimic OOD perturbation settings

Standard Robomimic evaluation initializes test episodes from the same distribution as the training demonstrations. As a result, both the policy and the world model are usually evaluated on familiar robot configurations. To test robustness under a realistic deployment shift, we construct an OOD protocol that perturbs the robot's initial joint configuration.

For each test episode, we start from the initial simulator state of Robomimic demonstrations and perturb seven arm joint angles:

$$\tilde{\mathbf{q}}^{(i)} = \mathrm{clip}\left(\mathbf{q}^{(i)} + \boldsymbol{\epsilon}^{(i)}, \mathbf{q}_{\min}, \mathbf{q}_{\max}\right), \qquad \boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_7), \tag{45}$$

where $\mathbf{q}^{(i)}$ is the demonstration initial joint configuration and $[\mathbf{q}_{\min}, \mathbf{q}_{\max}]$ are the Panda joint limits. For the dual-arm Transport task, the same perturbation procedure is applied independently to each arm. All other simulator state variables are left unchanged.

We use $\sigma = 0.1\,\mathrm{rad}$ in the main experiments. This perturbation is large enough to move the robot away from the demonstration manifold, while remaining physically feasible for task execution. The same perturbed initial states are used for all baselines and our method to ensure a fair comparison. Figs. 7, 8, and 9 visualize the resulting OOD initial states for Square, Transport, and ToolHang. In each figure, the leftmost column shows the in-distribution demonstration start, while the remaining columns show independently sampled OOD starts.

C LIBERO-Plus single-task settings

We evaluate on four single-task settings from the LIBERO-Plus benchmark, covering representative manipulation tasks from the *Kitchen* and *Living Room* scenes of LIBERO-10. For robot-initial-state evaluation, the task instruction, scene layout, camera configuration, lighting, and object categories match the corresponding LIBERO-10 task. The distribution shift is introduced through perturbed initial robot configurations following the official LIBERO-Plus *Robot Initial States* protocol. This setting allows us to evaluate whether the policy and world model remain robust when the robot starts from configurations outside the demonstration distribution, while the task semantics remain unchanged.

Tasks. The four tasks used in our experiments are summarized in Table 4. They include sequential stove manipulation, microwave-door interaction, long-horizon dual-object placement, and

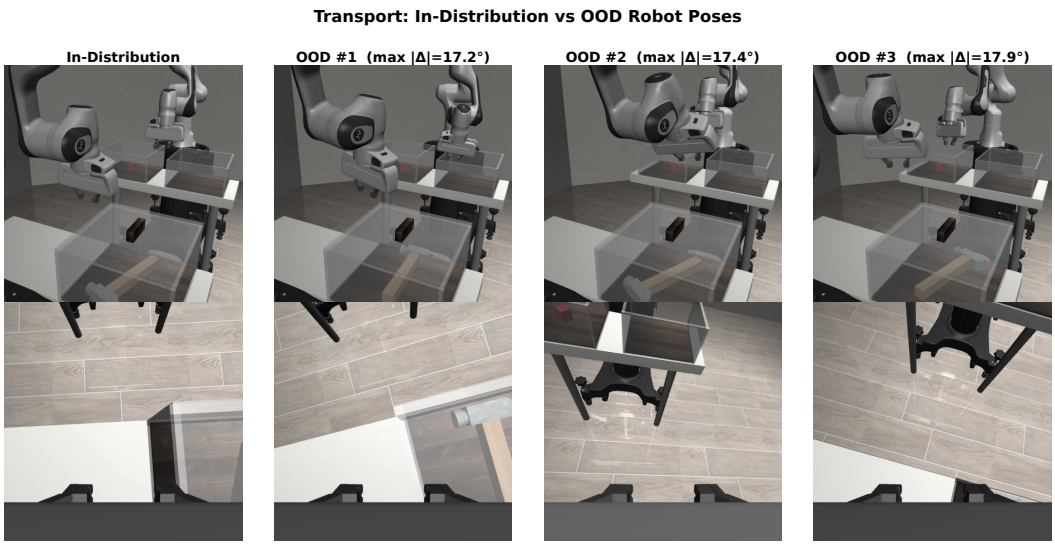


Figure 8: **Transport OOD initial states.** The layout follows Fig. 7. For this dual-arm task, joint perturbations are applied independently to the two arms. Top row: `shouldercamera0`; bottom row: robot-0 wrist camera.

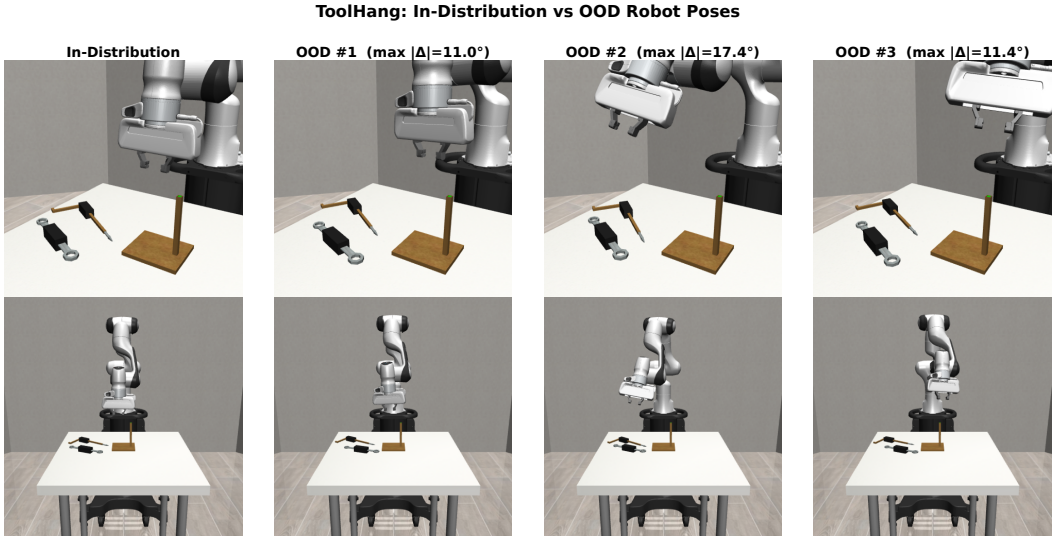


Figure 9: **ToolHang OOD initial states.** The layout follows Fig. 7. Top row: side view; bottom row: front view.

relational tabletop placement. These tasks cover both single- and multi-stage manipulation, as well as rigid-object and articulated-object interactions. We select these tasks because they cover diverse manipulation challenges while keeping the distribution-shift factor controlled: Kitchen-S3 and Kitchen-S6 involve articulated-object interaction and sequential execution, Kitchen-S8 requires long-horizon placement of multiple objects, and LR-S6 tests relational object placement in a different scene layout. Since the task instruction and scene semantics remain unchanged, these settings isolate robustness to robot initial-state shifts from changes in task specification or visual domain.

Demonstrations and evaluation protocol. For each task, we use the 50 expert demonstrations from the corresponding LIBERO-10 task as the only supervision. No LIBERO-Plus demonstrations are used during training. At evaluation time, we use the perturbed initial states released by LIBERO-Plus under the `libero_10` suite and the `Robot Initial States` category. We evaluate 44 variants for Kitchen-S3, 42 for Kitchen-S6, 34 for Kitchen-S8, and 42 for LR-S6, following the official protocol

from LIBERO-Plus benchmark. We report the average success rate over all evaluated initial-state variants.

Table 4: LIBERO-Plus single-task settings used in our experiments. All tasks follow the official `Robot Initial States` protocol.

Alias	Scene	Language instruction	# variants
Kitchen-S3	KITCHEN_SCENE3	turn on the stove and put the moka pot on it	44
Kitchen-S6	KITCHEN_SCENE6	put the yellow and white mug in the microwave and close it	42
Kitchen-S8	KITCHEN_SCENE8	put both moka pots on the stove	34
LR-S6	LIVING_ROOM_SCENE6	put the white mug on the plate and put the chocolate pudding to the right of the plate	42

D Rollout data collection

We include rollout-data baselines to test whether the gains of feedback world model can be matched by simply collecting additional policy executions and retraining. To avoid information leakage, all rollout trajectories are collected in the original in-distribution Robomimic or LIBERO environments. None of the rollout datasets contains trajectories from the final robot-initial-state OOD evaluation setting.

Robomimic. For each Robomimic task, we first train the base diffusion policy using the same low-data setting as in the main experiments, i.e., 20% of the official expert demonstrations. We then deploy intermediate policy checkpoints in the original Robomimic environment and record closed-loop trajectories. We collect 270 rollout episodes for Square and 150 rollout episodes for Transport and Tool-Hang. The rollout checkpoints are sampled across training to include both partially trained and stronger policies, producing a mixture of successful and failed behaviors.

For data-augmentation baselines, we construct two behavior-cloning datasets: *Mixed BC*, which combines the 20% expert subset with all collected rollout trajectories, and *Filtered BC*, which combines the same expert subset with only successful rollout trajectories. For the rollout-enhanced world model baseline, we train the world model on all official expert demonstrations plus all collected rollout trajectories, matching the setting used to evaluate whether additional offline data can improve prediction quality.

LIBERO-Plus single-task setting. For LIBERO-Plus, rollout data are collected from the four selected LIBERO-10 single tasks: Kitchen-S3, Kitchen-S6, Kitchen-S8, and LR-S6. For each task, rollout trajectories are generated by the corresponding single-task policy using the original LIBERO-10 task specification and language instruction. Episodes are initialized from the training demonstration initial states in a round-robin manner to keep the rollout distribution aligned with the expert-data distribution. We collect rollouts from multiple policy checkpoints with a maximum horizon of 500 environment steps. The resulting rollout sets contain 150 episodes for Kitchen-S3 and Kitchen-S6 respectively, 100 episodes for Kitchen-S8 and LR-S6 respectively.

As in Robomimic, we build two rollout-augmented policy-training datasets. *Mixed BC* uses the original expert demonstrations together with all collected rollout episodes, regardless of success. *Filtered BC* uses the same expert demonstrations but retains only rollout episodes that achieve task success. The rollout-enhanced world model is trained on the corresponding mixed rollout-augmented dataset. These baselines therefore represent a direct offline-data-augmentation alternative to our method, whereas the feedback world model uses no additional rollout data and instead corrects the world model state online from observations produced during the current inference episode.

E Complete inference pipeline

Algorithm 1 summarizes the complete inference pipeline of our method. The procedure consists of an offline preprocessing stage and an online deployment stage. The offline stage constructs the

expert latent memory and estimates the action-aware controllability weights, while the online stage performs feedback world model guidance during diffusion-policy sampling.

Offline preprocessing. Given the expert demonstration set $\mathcal{D}_{\text{expert}}$, we first encode all expert observations into the latent space of the observation encoder ψ, yielding an expert latent memory $\mathcal{Z}^E = \{\psi(O) \mid O \in \mathcal{D}_{\text{expert}}\}$. This memory is later used as a task-progress prior during guidance: predicted future latents are encouraged to approach nearby expert latents. In addition, we compute the action-aware weights $\{w_j^{(\beta)}\}_{j=1}^D$ from the demonstrations via Eqs. (14)–(15). These weights estimate the degree to which each latent dimension is controllable by the robot action, and are fixed throughout deployment. As a result, the guidance objective focuses more on action-relevant latent changes and suppresses the influence of latent components that are visually salient but weakly controllable.

Online deployment. At the beginning of an episode, the robot observes o_0 and obtains its latent state $z_0 = \psi(o_0)$. We initialise the internal feedback state as $\bar{z}_0 = z_0$. At each environment timestep t, the diffusion policy samples an initial noisy action sequence $A_t^{(T)}$ and iteratively denoises it from $\tau = T$ to $\tau = 1$. During high-noise denoising steps, we use the original policy score $s_\phi(A_t^{(\tau)}, O_t, \ell, \tau)$ without additional guidance. Guidance is only applied in the final τ_g low-noise denoising steps, where the action sequence already contains meaningful task structure and the world model prediction is more informative.

For each guided denoising step, the feedback world model predicts the next latent state under the current candidate action sequence. It evaluates the base predictor at the observed latent z_t and adds the correction $\delta t L(z_t - \bar{z}_t)$. The feedback state $\bar{z}_t$ carries information from previous interactions, allowing observed prediction residuals to refine the candidate future prediction. The predicted latent is then matched to its nearest expert latent $z_{i^\star}^E$ in $\mathcal{Z}^E$, and the action-aware guidance energy $E_{\text{ctrl}}(A_t^{(\tau)})$ is computed using the controllability-weighted latent distance via Eq. (16). The resulting gradient is added to the diffusion-policy score, producing the guided score $\tilde{s}$ used for the next denoising update. This forms the inner guidance loop of our method.

After denoising, the clean action sequence $A_t = A_t^{(0)}$ is obtained and the first T_a actions are executed in the environment. The feedback state is then propagated using the corrected latent dynamics induced by the executed action:

$$\bar{z}_{t+1} = \bar{z}_t + \delta t \cdot \hat{v}_t(A_t^{(0)}).$$

The next observation o_{t+1} is encoded as $z_{t+1} = \psi(o_{t+1})$, and the residual

$$e_{t+1} = z_{t+1} - \bar{z}_{t+1}$$

is formed for next-step guidance. This residual measures the mismatch between the feedback world model's internal belief after executing $A_t^{(0)}$ and the actual latent state reached by the environment. It is then held fixed during the next denoising process and used to correct all candidate world model predictions at timestep $t + 1$.

Role of the two key components. The feedback world model and action-aware guidance address two complementary limitations of standard world model guided diffusion policies. The feedback world model reduces prediction drift during inference by closing the loop between predicted latent dynamics and actual observed transitions. This is particularly important under robot initial-state-induced distribution shifts, where open-loop predictions can quickly deviate from the true trajectory. In contrast, the action-aware guidance improves how the corrected predictions are used for policy guidance. Rather than treating every latent dimension equally, it emphasizes dimensions that are more sensitive to robot actions and therefore more useful for controlling task progress. Together, these two components yield a closed-loop inference procedure in which the world model provides more accurate future-state estimates, and the diffusion policy receives guidance concentrated on action-controllable task-relevant changes.

F Implementation details.

Lightweight latent world model architecture. The seven-task and real-world experiments use a lightweight, task-specific world model. The separate pretrained Fast-WAM experiment is described

Algorithm 1 Inference Pipeline

Require: Diffusion policy score s_ϕ, encoder ψ, world model f_θ, expert demos $\mathcal{D}_{\text{expert}}$, feedback gain L, controllability strength β, guidance window $\tau_g \leq T$.

Offline preprocessing:

$\mathcal{Z}^E \leftarrow \{\, \psi(O) \mid O \in \mathcal{D}_{\text{expert}} \,\}$; compute action-aware weights $\{w_j^{(\beta)}\}_{j=1}^{D}$ via Eqs. (14)–(15).

Online deployment:

Observe o_0, encode $z_0 = \psi(o_0)$, initialise feedback state $\bar{z}_0 \leftarrow z_0$.

for each timestep t **do**

  Sample $A_t^{(T)} \sim \mathcal{N}(0, \mathbf{I})$.

  for denoising step $\tau = T, \ldots, 1$ **do**

    if $\tau \leq \tau_g$ **then** ▷ *guidance applied only at the last τ_g low-noise steps*

      Predict next latent with feedback-corrected dynamics: $z_{t+1}^{\text{fb}}(A_t^{(\tau)})$ via Eq. (10).

      Retrieve nearest expert latent $z_{i^\star}^E$ (Eq. (5)) and form $E_{\text{ctrl}}(A_t^{(\tau)})$ (Eq. (16)).

      $\tilde{s} \leftarrow s_\phi(A_t^{(\tau)}, O_t, \ell, \tau) - \lambda \nabla_{A_t^{(\tau)}} E_{\text{ctrl}}(A_t^{(\tau)})$. ▷ *Eq. (7)*

    else

      $\tilde{s} \leftarrow s_\phi(A_t^{(\tau)}, O_t, \ell, \tau)$.

    end if

    Denoise the action sequence from $A_t^{(\tau)}$ to $A_t^{(\tau-1)}$ using $\tilde{s}$.

  end for

  Execute first T_a actions of $A_t = A_t^{(0)}$ in $\mathcal{E}$.

  Advance feedback state $\bar{z}_{t+1} \leftarrow \bar{z}_t + \delta t \cdot \hat{v}(z_t, A_t^{(0)})$ via Eq. (11).

  Receive o_{t+1}, encode $z_{t+1} = \psi(o_{t+1})$. ▷ *form $e_{t+1} = z_{t+1} - \bar{z}_{t+1}$ via Eq. (12) for next-step guidance*

end for

in Appendix J. For the lightweight models, we use a compact latent dynamics model trained from the same task demonstrations as the policy. The model predicts the next latent state from the current observation latent, proprioceptive state, and the executed action chunk. Each RGB view is encoded by the visual encoder associated with the deployed diffusion policy, and the resulting visual features are fused with a small proprioception encoder and an action encoder. A Transformer predictor then maps this fused representation to the next-step latent state. The model is trained with a one-step latent prediction objective and does not reconstruct pixels or generate future videos.

This design keeps the world model small and data-efficient: it is trained per task from the available demonstrations only, without large-scale pretraining or additional real-world rollout data. During deployment, the world model is frozen. Our feedback mechanism only maintains and corrects an online latent state estimate using the latest real observation; it does not update the world model parameters at test time.

F.1 Implementation details of simulated tasks.

Robomimic tasks. We evaluate three Robomimic tasks: Square, Transport, and Tool-Hang. For each task, we train an image-conditioned diffusion policy under a low-data setting using 20% of the official proficient-human demonstrations. The policy uses RGB observations and proprioception as input. Square and Tool-Hang are single-arm tasks, while Transport is a dual-arm task. The visual observations follow the standard task camera setup, using dual-view inputs such as the agent/shoulder view and wrist-mounted camera view. The policies are trained with a UNet-based diffusion policy, AdamW optimization, a cosine learning-rate schedule, EMA, and DDPM denoising. Square uses a shorter action horizon, while Transport and Tool-Hang use a longer action horizon to match their longer execution sequences.

For each Robomimic task, we train a separate lightweight latent world model using the same task demonstrations. The world model uses the same observation modalities as the deployed policy. It predicts the next latent state after the executed action chunk and is queried during diffusion-policy inference to provide world model guidance. In the OOD evaluation, we perturb only the robot initial joint configuration while keeping the task goal, object arrangement, camera placement,

and environment semantics unchanged. This produces a robot-initial-state OOD setting without introducing visual domain randomization.

LIBERO-Plus single-task tasks. We evaluate four LIBERO-10 single tasks under the Robot Initial States category of LIBERO-Plus: `KITCHEN_SCENE3`, `KITCHEN_SCENE6`, `KITCHEN_SCENE8`, and `LIVING_ROOM_SCENE6`. For each task, the policy and world model are trained on demonstrations from the corresponding original LIBERO-10 task and evaluated directly on the LIBERO-Plus robot-initial-state perturbation variants.

The LIBERO policies use dual-view RGB observations from the agent-view camera and the eye-in-hand camera, together with proprioceptive state and the task language embedding. We train a dedicated diffusion policy for each single task. The corresponding lightweight world model is trained on the same task demonstrations and uses the same visual, proprioceptive, action, and language inputs as the policy. During evaluation, the world model predicts future latent states under candidate action chunks, and the feedback mechanism corrects its latent state estimate after each real environment observation.

Baselines and method configuration. We compare against the base diffusion policy, vanilla world model guidance, and rollout-augmentation baselines when available. The base policy is executed without world model guidance. Vanilla world model guidance uses the frozen latent world model during diffusion denoising but does not use feedback correction. Rollout-augmentation baselines retrain the diffusion policy with additional policy rollouts: Mixed BC uses all collected rollouts, while Filtered BC uses only successful rollouts. These rollouts are collected from the original validation environments rather than from the final OOD evaluation setting.

For our method, we use the same trained policy and the same trained base world model as the vanilla guidance baseline. The only change is at inference time: after each executed action chunk, the new observation is encoded and used to correct the world model latent state before the next guidance step. When action-aware weighting is enabled, we compute per-dimension controllability weights from demonstration observations by measuring counterfactual action variance in latent space offline.

F.2 Implementation details of real-world tasks.

Robot platform. We conduct all real-world experiments on an R1 Lite robot. Although the platform has two arms, all real-world experiments in this paper use only the right arm and the right gripper; the left arm is kept idle throughout data collection and evaluation. The policy outputs a right-arm control command and a right-gripper command. The proprioceptive input contains the right-arm joint state and the right-gripper state. To construct a real-world robot-initial-state OOD setting, we reset the right arm to abnormal initial poses outside the training distribution before evaluation, while keeping the task goal and workspace layout unchanged.

Peach P&P. Peach P&P is a pick-and-place task in which the robot must grasp a peach from the table and place it into a target container. We collect 50 expert demonstrations on the R1 Lite platform using the right arm only. The final deployed policy and world model use two camera views: an agent-view RGB camera and the right-wrist RGB camera. The agent view provides global context of the tabletop scene and the target container, while the right-wrist camera provides close-range visual feedback during grasping and placement.

We train a task-specific image-conditioned diffusion policy from scratch on the 50 demonstrations. The policy uses two observation frames and predicts an action chunk for right-arm execution. RGB observations are resized and cropped before being passed to the visual encoder. The lightweight world model is trained from the same 50 demonstrations and uses the same two visual views, right-arm proprioception, right-gripper state, and right-arm action input as the policy. Thus, both the policy and world model are trained only from the expert demonstration set, without additional real-world rollout data.

Drawer-Open. Drawer-Open is a more contact-rich manipulation task. The robot must locate the drawer handle, align the gripper, establish contact, and pull the drawer open with a suitable direction and force. The handle is made of deformable rubber, so the task is sensitive to wrist pose, contact

timing, and local visual feedback near the handle. We collect 65 expert demonstrations using the right arm of R1 Lite.

For Drawer-Open, the final deployed policy and world model use only the right-wrist RGB camera view, together with right-arm and right-gripper proprioception. We use this wrist-only observation setting because the task depends primarily on local handle geometry, gripper-handle alignment, and contact evolution after the wrist approaches the drawer. We train a task-specific diffusion policy from scratch on the 65 demonstrations. The corresponding lightweight world model is trained from the same demonstrations and uses the same right-wrist visual observation and proprioceptive/action inputs as the policy.

Real-world deployment protocol. At deployment time, the base diffusion policy generates action chunks from the current observation history. For world-model-guided methods, the frozen task-specific world model evaluates candidate denoising trajectories in latent space. After each executed action chunk, the latest real observation from the robot is encoded and used as feedback to correct the world model latent state before the next guidance step. This online correction is applied only at inference time and does not update the policy or world model parameters. All real-world methods are evaluated under the same reset protocol and task success criteria. Peach P&P is counted as successful when the peach is grasped and placed into the target container. Drawer-Open is counted as successful when the drawer is pulled open to the required extent.

F.3 Compute resources.

For the task-specific policy and world-model experiments under robot initial-state shifts, training and evaluation run on an internal GPU workstation/cluster with one RTX A6000 GPU (48 GB memory), 8–16 CPU cores, and 64–128 GB host memory per job. These experiments require approximately 400–700 single-GPU hours, including policy training, world-model training, hyperparameter sweeps, and evaluation. Preliminary and exploratory runs used additional compute, roughly 1.5–$2\times$ this budget. Inference cost measured on an RTX PRO 6000 is reported in Appendix N.

F.4 Existing assets and licenses

We use public robotic benchmarks and assets only for research purposes. Robomimic is cited in the main paper and used following its public release and license. LIBERO and LIBERO-Plus are cited in the main paper and used following their official benchmark instructions. At the time of writing, the official LIBERO-Plus repository does not provide an explicit license file. We do not redistribute or repackage these assets in this work. All real-world demonstrations are collected on our own robot platform.

G Evaluation statistics

Prediction error. For Fig. 3, we evaluate latent MSE one executed action chunk ahead. Simulation MSE is averaged over all valid prediction steps, with standard errors estimated by resampling complete episodes 10,000 times. Real-world MSE uses the mean of episode means and its SEM. These error bars quantify variability across evaluation episodes for fixed trained models. Table 5 lists the values and episode counts.

Table 5: Latent prediction MSE in Fig. 3, scaled by 10^3. Simulation values are step-pooled mean $\pm$ episode-cluster bootstrap SE; real-world values are episode-equal mean $\pm$ SEM. n is the number of episodes per available method. Real-world rollout models were not evaluated.

Task	n	Base WM	Rollout-Enhanced WM	Feedback WM
Square	50	16.246 ± 3.904	15.472 ± 1.092	4.333 ± 0.528
Tool-Hang	50	8.399 ± 1.006	10.295 ± 1.647	5.373 ± 0.545
Transport	50	16.248 ± 2.617	13.160 ± 1.832	10.432 ± 0.495
Kitchen-S3	44	15.676 ± 0.820	17.510 ± 0.740	13.400 ± 0.633
Kitchen-S6	42	11.999 ± 0.920	11.698 ± 0.829	8.959 ± 0.484
Kitchen-S8	34	10.189 ± 1.073	9.514 ± 0.908	8.553 ± 0.680
LivingRoom-S6	42	20.212 ± 0.907	18.044 ± 0.582	15.831 ± 0.726
Peach-P&P	20	2.177 ± 0.169	n/a	0.754 ± 0.107
Drawer-Open	20	1.309 ± 0.099	n/a	0.309 ± 0.080

H Evaluation under LIBERO-Plus perturbations

We evaluate four LIBERO-Plus tasks under five perturbation categories: robot initial states, lighting, background textures, object layout, and sensor noise. Each category is evaluated on the same four tasks, yielding 20 task–perturbation settings. Table 6 reports success rates for each setting, and Table 3 summarizes the results pooled within each category.

Table 6: Success rates for all five LIBERO-Plus perturbation categories. K-S3, K-S6, K-S8, and LR-S6 denote the four tasks in Appendix C.

Perturbation	Method	K-S3	K-S6	K-S8	LR-S6
Robot initial states	Base	0.55	0.45	0.59	0.26
	Ours	0.68	0.55	0.65	0.36
Lighting	Base	0.50	0.21	0.42	0.58
	Ours	0.40	0.24	0.58	0.74
Background textures	Base	0.97	0.50	0.55	0.38
	Ours	0.94	0.68	0.58	0.44
Object layout	Base	0.81	0.74	0.48	0.63
	Ours	0.85	0.85	0.64	0.66
Sensor noise	Base	0.98	0.88	0.80	0.59
	Ours	0.98	0.90	0.87	0.70

Our method matches or exceeds the base policy in 18 of the 20 task–perturbation settings (Table 6). Pooled success increases in all five categories, by 4.5–9.9 percentage points, alongside reductions in latent MSE (Table 3). For example, success on Kitchen-S6 under background changes rises from 50% to 68%, while Kitchen-S8 under object-layout changes improves from 48% to 64%. These results demonstrate improvements in both aggregate prediction accuracy and policy performance across the evaluated distribution shifts.

I Action dependence and physical-state validation

I.1 Action-dependent prediction

Effective world-model guidance requires predictions that respond to the chosen action. Table 7 shows that using correctly paired actions yields substantially lower latent MSE than persistence, a constant

training-mean prediction, or shuffled actions. Shuffling the actions while keeping the predictor fixed increases MSE by approximately $20.9\times$ on Square and $36.6\times$ on Transport, indicating that action information contributes substantially to predictive accuracy. The advantage over persistence and the training mean further supports the model's ability to capture action-conditioned transitions beyond state continuity and average latent structure.

Table 7: Action-conditioning controls. Values are latent MSE.

Predictor	Square	Transport
Persistence	0.0190	0.0324
Constant training mean	0.0382	0.0446
WM with shuffled actions	0.0335	0.0914
WM with correct actions	0.0016	0.0025

Controlled interventions further connect this action dependence to physical motion. We vary the action from identical OOD simulator states and use a frozen probe to decode the predicted end-effector effects. Predicted and realized effects have overall correlations of 0.81 on Square and 0.84 on Transport (Fig. 10), with per-axis correlations of 0.71–0.93 and 0.84–0.87, respectively. This agreement suggests that the model captures how action changes affect robot motion. Together, the two comparisons support using its predictions to distinguish the physical consequences of candidate actions for policy guidance.

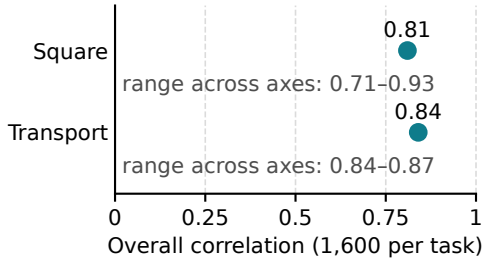


Figure 10: **Controlled action effects.** Overall correlation between predicted and realized end-effector effects over 1,600 interventions per task, from 200 saved states with eight action branches each. Text labels give the range of per-axis correlations.

I.2 Physical information in the latent representation

Physical-state probes show that the learned latent representation retains information about robot configuration and object position. We fit episode-disjoint ridge probes to the frozen task-trained latent, with ImageNet-pretrained ResNet-18 features from the same observations as a reference (Table 8). The recovered end-effector positions and orientations, together with object and payload positions, connect the latent representation to physical quantities relevant to manipulation.

Table 8: Physical-state probe RMSE. Slash-separated entries refer to Transport arms 0/1.

Task	Target	Task-trained	ImageNet control
Square	EEF position (mm)	34	29
	EEF orientation (degrees)	10.5	10.4
	Object position (mm)	30	30
Transport	EEF position (mm)	87/57	77/61
	EEF orientation (degrees)	5.8/4.0	6.4/5.3
	Payload position (mm)	84	88

The predicted latents preserve this physical information. Using the frozen probes to decode world-model predictions gives position errors comparable to the probes' held-out errors on observed latents (Table 9). For example, the decoded Square end-effector and object positions have errors of 25.8 mm

and 21.9 mm, respectively, against probe errors of 29.6 mm and 25.3 mm. These results indicate that the predictions capture physical state at a precision consistent with the probe's decoding accuracy, complementing latent MSE with an interpretable measure of robot and object position.

Table 9: Physical error decoded from predicted latents and probe resolution on observed latents. All values are in millimeters.

Target	Prediction error	Probe resolution
Square EEF position	25.8	29.6
Square object position	21.9	25.3
Transport arm 0 EEF position	55.9	64.6

I.3 Qualitative latent trajectories

To examine how feedback correction evolves during execution, Fig. 11 compares the base and feedback world-model predictions with the observed latent trajectories in a shared PCA projection. The time axis reveals when prediction errors emerge and how subsequent feedback updates affect trajectory tracking.

In cases (a) and (b), the base model develops noticeable departures from the observed trajectory during the middle and later stages of execution. The feedback-corrected predictions follow the observed bends and changes in direction more closely, reducing these persistent offsets. Case (c) highlights the correction of early-stage deviations, followed by close tracking as execution proceeds. These examples show how online observation feedback helps the predictor follow the evolving state across different phases of an episode. They complement the aggregate MSE results in Fig. 3 by illustrating the temporal behavior of the correction: prediction discrepancies observed during execution are used to improve alignment in subsequent predictions, without updating model parameters.

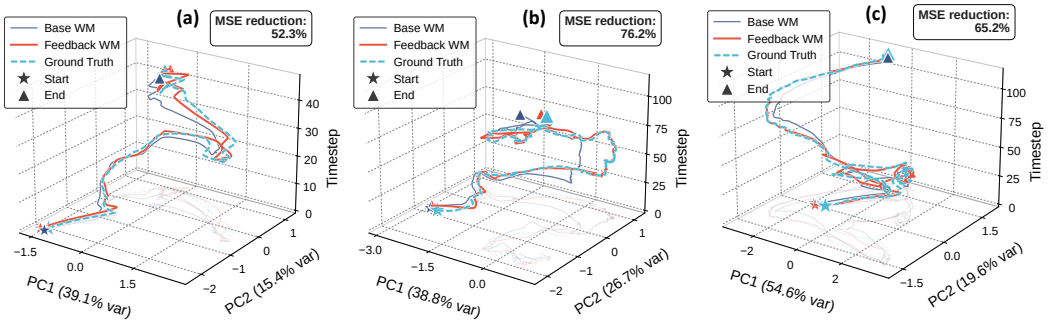


Figure 11: **Online latent-trajectory tracking with feedback.** Base WM, Feedback WM, and observed (ground-truth) latent trajectories for three example episodes, shown in a shared PCA projection with time as the third axis. Feedback correction brings predictions closer to the observed trajectories through changes in direction and across execution stages. Stars and triangles mark the start and end of each trajectory.

J Transfer to Fast-WAM-Joint

Model and feedback implementation. We implement feedback on Fast-WAM-Joint [30], which uses a pretrained Wan2.2-5B backbone and two-camera inputs across LIBERO-10. The architecture and learned parameters remain frozen. Feedback operates in the video branch's native future-latent space. For a predicted future latent F and its observed counterpart G, encoded by the same VAE, we maintain an exponential moving average $b \leftarrow 0.7b + 0.3(G - F)$ and correct subsequent predictions as $F_{\mathrm{fb}} = F + \kappa b$. The residual is incorporated only after the corresponding observations become available, allowing past prediction errors to inform subsequent predictions with the model parameters fixed.

Prediction evaluation. Each prediction covers four future frames at offsets of 8, 16, 24 and 32 environment steps. We compare base and corrected predictions on the same recorded rollouts, without changing executed actions, and average errors over paired prediction calls. The ID set contains 1,203 calls from 100 episodes; the robot-initial-state OOD set contains 7,553 calls from 393 episodes covering all 393 LIBERO-Plus variants across LIBERO-10. Latent MSE is the mean squared difference in the native future-latent coordinates. For pixel metrics, both predictions and the VAE-encoded observations are decoded with the same VAE, including the initial frame as causal context. We compute MSE and PSNR on the four future frames at pixel range 0–255; the reference is the VAE reconstruction of the observations.

Feedback improves latent prediction accuracy in both ID and OOD settings at all three tested gains, 0.2, 0.3 and 0.5 (Fig. 6). Within this range, increasing the gain progressively reduces error. At gain 0.5, OOD latent MSE falls from 0.3015 to 0.2464, an 18.3% reduction, compared with a 7.8% reduction under ID conditions. The larger relative improvement under distribution shift suggests that observation feedback is particularly useful when the pretrained predictor encounters a mismatch with deployment conditions. These gains show that a large pretrained model can still benefit from correcting its predictions using observations from the current execution.

The improvement also carries through to decoded visual predictions. At gain 0.5, OOD pixel MSE decreases by 18.0% and mean PSNR increases by 0.86 dB, with positive PSNR gains at all four prediction offsets (0.76–0.94 dB). The agreement between latent-space and image-space metrics indicates that the correction improves the fidelity of the predicted visual outcomes across the evaluated horizons. Because the backbone remains frozen, these results demonstrate that feedback can complement pretrained dynamics through inference-time correction alone.

Policy evaluation. To assess the utility of feedback for control, we evaluate policy success on the LIBERO-Plus Robot Initial States OOD variants across LIBERO-10. Our method improves Fast-WAM-Joint success from 77.1% to 81.4%, a gain of 4.3 percentage points. Together with the prediction results, this improvement supports the plug-and-play applicability of feedback to a pretrained world-action model: the same model benefits in both prediction accuracy and task performance while retaining its learned parameters.

K Comparison with filtering baselines

Observation feedback is also central to classical filtering and learned belief updates. We therefore compare these approaches to examine whether our lightweight residual correction can improve OOD prediction while reusing a frozen predictor. The comparison uses the same frozen encoder and 50 expert demonstrations per task. An estimated-gain Kalman filter (Est. KF) provides a classical filtering baseline, while a PlaNet-style RSSM represents learned state estimation. RSSM Open predicts through its learned prior; RSSM Filter incorporates observations through its posterior.

Table 10 shows that our method achieves the lowest OOD latent MSE on all three tasks, reducing error by 31.2–57.1% relative to Base WM and by 8.4–21.6% relative to Est. KF. The Kalman baseline already improves on Base WM, demonstrating the value of observation-based correction. The further reduction achieved by our method shows that its residual-feedback mechanism provides an effective way to correct model mismatch beyond this filtering baseline.

Within the RSSM, posterior conditioning reduces error relative to the prior, confirming that observations help refine learned state estimates. Our method achieves lower error by correcting the existing predictor directly. In this shared-encoder, limited-data setting, the results favor reusing learned dynamics with online residual correction: the method exploits deployment observations without learning a posterior network or updating model parameters. This offers a practical route to improved OOD prediction when a trained world model is already available.

L Empirical behavior of the dynamics residual

Feedback correction uses observed prediction errors to refine subsequent predictions. To understand the model mismatch that this mechanism must track, we examine how the base world model's residual evolves throughout execution. The analysis asks how its magnitude changes across deployment conditions and across stages of a task, complementing the aggregate prediction errors in Fig. 3.

Table 10: OOD latent MSE, mean ± SEM across stored trajectories. The unit of replication is one trajectory, with $n = 150$, 150, and 95 for the three columns.

Method	Kitchen-S3	Kitchen-S6	LivingRoom-S6
Base WM	0.04565 ± 0.00040	0.04250 ± 0.00031	0.04535 ± 0.00044
Est. KF	0.03433 ± 0.00026	0.02325 ± 0.00013	0.03021 ± 0.00032
RSSM Open	0.05895 ± 0.00075	0.05377 ± 0.00036	0.05630 ± 0.00059
RSSM Filter	0.05698 ± 0.00074	0.04938 ± 0.00027	0.05313 ± 0.00054
Ours	$\mathbf{0.03143} \pm 0.00036$	$\mathbf{0.01823} \pm 0.00015$	$\mathbf{0.02671} \pm 0.00038$

Evaluation. We evaluate four LIBERO tasks under robot-initial-state and camera-viewpoint shifts, keeping the task-specific policy, world model, and feedback gain fixed across the two conditions. The residual magnitude is

$$\|\Delta_t\|_2 = \frac{\|z_{t+1} - f_\theta(z_t, A_t)\|_2}{\delta t}, \tag{46}$$

where z_{t+1} is the observed next state in the weighted latent representation used for feedback, $f_\theta(z_t, A_t)$ is the base model's prediction, and $\delta t = 0.75\,\text{s}$ is the interval between updates. Fig. 12 shows the residual histories aligned by episode progress. Table 11 summarizes their level and relative variability during the final third of each episode.

Temporal patterns and implications. Fig. 12 shows that residuals continue to vary throughout execution. In Kitchen-S3, both conditions exhibit a lower residual level around the middle of the episode followed by a later rise. In Kitchen-S8, camera-viewpoint shifts produce higher residuals through most of the episode, while the camera-shift residual in LivingRoom-S6 increases gradually with progress. These patterns reveal both temporal structure within an execution and differences between deployment conditions.

The late-episode statistics in Table 11 reinforce this observation: camera-viewpoint shifts yield higher residual levels in all four tasks, including an increase from 3.56 to 4.89 on Kitchen-S8. The coefficient of variation ranges from 0.11 to 0.21 across conditions, indicating continued fluctuations around these levels. Together, the curves and summaries motivate maintaining a feedback state throughout execution: updating it with each new observation allows the correction to respond to the current mismatch as the task progresses. This provides an empirical rationale for the continuous observation–prediction loop used by our method.

Connection to the theoretical analysis. The persistent residual levels and temporal variation in Fig. 12 motivate two complementary roles of feedback: compensating persistent model mismatch and updating the correction as execution progresses. Under the residual assumptions in Eq. (19) and a stable gain, the observer correction satisfies $Le_t \to \Delta_\infty$, cancelling the limiting prediction bias (Eq. (43)). This bias-compensation mechanism provides a theoretical interpretation of the prediction improvements in Fig. 3. Incorporating successive observations keeps the correction responsive to the evolving mismatch, connecting the observer formulation to the online adaptation that makes feedback useful under distribution shift.

Table 11: **Late-episode residual statistics.** n is the number of episodes. Late residual is the median across episodes of each episode's median residual in its final third. CoV is the median across episodes of the within-window standard deviation divided by the mean. State and Camera denote robot-initial-state and camera-viewpoint OOD, respectively.

Scene	Condition	n	Late residual	CoV
Kitchen-S3	State	44	5.17	0.18
	Camera	47	5.72	0.16
Kitchen-S6	State	42	4.17	0.15
	Camera	40	4.59	0.17
Kitchen-S8	State	34	3.56	0.21
	Camera	43	4.89	0.14
LivingRoom-S6	State	42	6.16	0.12
	Camera	46	6.98	0.11

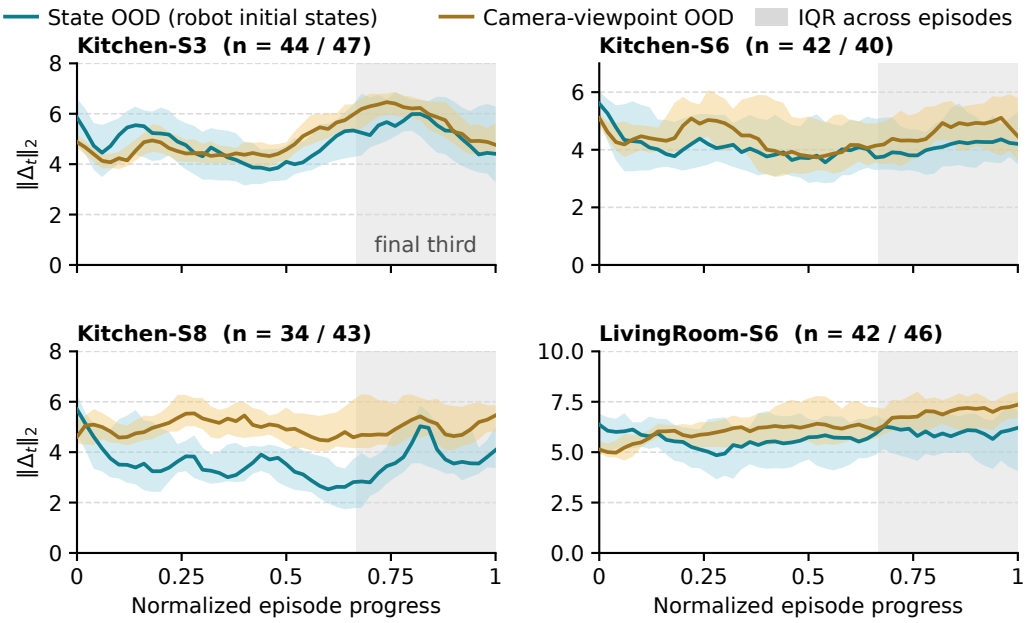


Figure 12: **Temporal structure of the base world model's residual.** Episodes are aligned from start to finish on a normalized progress axis. Curves show the across-episode median; shaded bands span the 25th–75th percentiles. Teal and brown denote robot-initial-state and camera-viewpoint OOD, respectively. Gray marks the final third used in Table 11.

M Feedback gain and action-weight interpolation

We study sensitivity to two complementary choices: the feedback gain l, which controls prediction correction, and the interpolation coefficient β, which controls the emphasis on action-responsive latent dimensions. The experiments assess the range of settings that yield useful correction and guidance, and inform parameter selection for deployment.

Feedback gain. For $L = lI$, the effective correction strength per update is $\delta t l$; the LIBERO experiments use $\delta t = 0.75$ s. Table 12 evaluates how changing l affects latent prediction accuracy on four tasks. Every tested gain from 0.02 to 0.5 reduces MSE relative to Base WM, with reductions of 7.6–24.9%. The best tested gains lie between 0.1 and 0.3, showing that substantial improvements are available across several moderate settings. The results at $l = 1.0$ further favor moderate correction strengths in this evaluation. Together, these observations support selecting the gain using held-out prediction error within a practical operating range.

Table 12: Latent MSE reduction relative to Base WM under gain variation. Negative values denote degradation.

l	Kitchen-S3	Kitchen-S6	Kitchen-S8	LR-S6
0.02	7.6%	11.0%	10.8%	10.7%
0.1	13.5%	23.0%	21.3%	19.9%
0.2	16.0%	24.9%	18.1%	22.3%
0.3	17.6%	24.3%	20.5%	22.7%
0.5	9.5%	19.9%	12.8%	18.8%
1.0	−20.0%	−9.4%	−12.5%	−8.6%

The task gains used in our evaluations are $0.5/0.1/0.5$ for Square/Tool-Hang/Transport and $0.05/0.5/0.05/0.1$ for Kitchen-S3/Kitchen-S6/Kitchen-S8/LR-S6, selected using held-out latent MSE.

Action-weight interpolation. Table 13 examines how the balance between uniform and action-aware weighting affects policy success, with other settings fixed. Intermediate interpolation gives the highest tested success on both tasks: Kitchen-S8 reaches 0.667 at $\beta = 0.3$, and Kitchen-S3 reaches 0.697 at $\beta = 0.7$. These improve on full weighting ($\beta = 1$) by 4.0 and 3.8 percentage points, respectively. This pattern supports the interpolation in Eq. (15): retaining a uniform component while emphasizing action-responsive dimensions can improve guidance. The different maximizing values also indicate that the appropriate emphasis can be adapted to the task. Across the tested coefficients, success varies by only 4.0 and 3.8 percentage points on the two tasks, respectively.

Table 13: Success rate under action-weight interpolation β in Eq. (15), with other settings fixed.

Task	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.7$	$\beta = 1.0$
Kitchen-S8	0.656	0.667	0.637	0.627
Kitchen-S3	0.667	0.682	0.697	0.659

N Inference cost

We evaluate how much inference overhead feedback correction and action-aware weighting introduce when integrated into world-model-guided diffusion inference. To quantify their contribution, we measure the time spent in each component (Fig. 13). Per 15-action chunk, the base diffusion policy takes approximately 484 ms, and generic world-model guidance adds about 525 ms. Feedback correction takes only about 15 ms, while action-aware weighting adds less than 1 ms. Together, our two modules account for less than 2% of the approximately 1.02 s total inference time. Thus, the additional computation is dominated by the underlying guidance procedure. Feedback correction and action-aware weighting enhance this procedure with minimal marginal inference overhead.

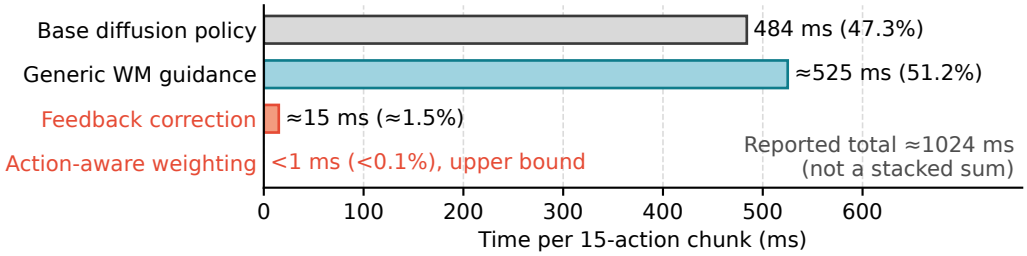


Figure 13: **Inference cost by component.** Approximate time per 15-action chunk on one RTX PRO 6000, batch size 1, using DDPM-100 with guidance in the final 15 denoising steps. Feedback correction and action-aware weighting together account for less than 2% of the total.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction 1 clearly state the paper's main contributions and scope: a feedback world model for online correction of latent predictions, an action-aware guidance strategy for diffusion-policy inference, and evaluation under robot-initial-state and visual perturbations. These claims are supported by the convergence analysis in Appendix A and by experimental results (in section 5.2) on LIBERO-Plus, Robomimic, and real-world manipulation tasks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 6 discusses guidance overhead and future directions, with costs quantified in Appendix N. Sec. 4.2 and Appendix A state the assumptions underlying the theoretical guarantees.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper states the main convergence result for the feedback world model in Sec. 4.2 and provides the complete proof in Appendix A. The proof explicitly states the bounded latent-dynamics residual assumption, the positive-definite feedback gain, and the discrete-time stability condition, and derives the resulting bound on the feedback estimation error and corrected one-step prediction error.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the main information needed to reproduce the experimental results in Sec. 5 and Appendices B,C,D,E,F, including the benchmark tasks, training data, OOD evaluation protocols, rollout-data baselines, complete inference pipeline, world model architecture, and implementation details for both simulated and real-world tasks. The paper also specifies the evaluation metrics and reports the main results for prediction error and policy success rates in Fig. 3, Fig. 4, and Table 1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide an anonymized code package in the supplemental material, together with instructions for environment setup, data preparation, training, feedback-guided inference, and evaluation. The package includes scripts for reproducing the main simulated experimental results and the corresponding baselines, while the public benchmark data can be obtained from the official Robomimic and LIBERO-Plus releases.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper describes the experimental settings in Sec. 5 and provides full details in Appendices B,C,D,E,F, including benchmark/task selection, data splits, OOD test protocols, rollout-data baselines, inference pipeline, implementation details, optimizer type, and real-world deployment protocol. These details are sufficient to understand the reported prediction-error and policy-success results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Fig. 3 reports episode-cluster bootstrap standard errors for simulation MSE and standard errors of episode means for real-world MSE, quantifying variability across evaluation episodes for fixed trained models. Appendix G describes the estimators and sample counts. Fig. 4 reports 95% Wilson confidence intervals for success rates from 20 trials per method, task, and condition. Table 10 reports mean $\pm$ SEM across evaluation trajectories.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides compute-resource information in Appendix F.3, including the GPU type, single-GPU-per-job setup, CPU and host-memory requirements, estimated total GPU hours for reproducing the reported experiments, and additional compute used for preliminary and failed exploratory runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms to the NeurIPS Code of Ethics. It uses public robotic benchmarks and controlled real-world robot experiments, does not involve human subjects or sensitive personal data, and provides transparent descriptions of the method, experimental setup, limitations, and reproducibility-relevant details.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: The paper does not explicitly discuss broader societal impacts. The work is primarily a foundational robotics-learning study evaluated in controlled manipulation benchmarks and robot experiments; while it may contribute positively to more reliable robotic systems, it is not directly connected to human-facing decisions, sensitive personal data, surveillance, or other high-risk societal applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models or datasets that require special safeguards, such as pretrained language models, image generators, or scraped Internet datasets. The work focuses on a robotic policy-guidance algorithm evaluated on public simulation benchmarks and controlled real-world robot experiments.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper credits the existing benchmarks and assets used in the experiments, including Robomimic, LIBERO, and LIBERO-Plus, through citations and uses them following their official benchmark instructions. However, the current version cannot explicitly list a license for LIBERO-Plus because no license file was found in its official repository at the time of writing.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release new datasets, model checkpoints, or benchmark assets. The real-world demonstrations collected in this work are used only for training and evaluating the proposed method and are not released as standalone assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects. The real-world experiments are conducted on a robot platform and use robot manipulation demonstrations and deployment rollouts, rather than participant studies or crowdworker annotations.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval or equivalent review is not applicable. The real-world experiments are conducted on a robot platform and consist of robot manipulation demonstrations and deployment evaluations rather than participant studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this work does not involve LLMs as an important, original, or non-standard component. The proposed method is based on feedback correction for latent world models and action-aware guidance for diffusion policies; any writing, editing, or formatting assistance, if used, does not affect the methodology, scientific rigor, or originality of the research.

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.