When to Trust Imagination: Adaptive Action Execution for World Action Models

Anonymous authors
Paper under double-blind review

Abstract

World Action Models (WAMs) jointly predict future visual observations and actions, but typically execute a fixed number of actions before replanning, regardless of whether the imagined future remains consistent with reality. We formulate adaptive WAM execution as *future–reality verification* and propose *Future Forward Dynamics Causal Attention* (FFDC), a lightweight verifier that reasons over predicted actions, predicted visual dynamics, real observations, and instruction-conditioned semantics. FFDC allows continued execution when a plan remains reliable and triggers replanning when its validity deteriorates. We combine verification with Mixture-of-Horizon Training to improve long-horizon trajectory coverage. On RoboTwin, FFDC-WAM achieves the highest average success rate among the evaluated methods in both clean and randomized settings. Under randomization, it improves success by 2.54 percentage points over Base-Motus while reducing WAM planning calls by 69.10% and completion time by 33.88%. Notably, across six real-world tasks on a Unitree G1D robot, FFDC-WAM improves average success by 9.17% over Base-Motus-LC16 while reducing calls by 74.06% and completion time by 51.31%.

1 Introduction

Humans anticipate the consequences of their actions and use sensory feedback to decide whether to continue. When a foot misses an expected stair, the unexpected lack of contact prompts an immediate correction; when events unfold as expected, we can proceed without reconsidering every movement. This comparison between prediction and observation allows efficient action while preserving the ability to respond to change.

World Action Models (WAMs) offer a computational analogue. Whereas many vision-language-action policies generate actions from observations and instructions (Shukor et al., 2025; Black et al., 2024; 2025; Lv et al., 2025), WAMs jointly predict future actions and visual observations (Pai et al., 2025; Liang et al., 2025; Li et al., 2026; Du et al., 2023; Ye et al., 2024; Guo et al., 2024). Recent WAMs have shown promising transfer across tasks, environments, and embodiments (Bi et al., 2025; Li et al., 2025; Zhu et al., 2025; Ye et al., 2026a;b; Yuan et al., 2026; BeingBeyond Team, 2026). Yet their predicted futures are seldom used to decide *when* to replan during execution. Instead, the robot typically executes a fixed number of predicted actions before calling the WAM again.

A fixed execution horizon creates an accuracy–latency tension (see Fig. 1 (c)). When the world evolves predictably, a plan may remain useful for many steps; repeatedly invoking WAM after short action chunks wastes time. During dynamic or contact-rich interaction, unexpected object motion or changes in contact can invalidate the remaining actions. Real-world actuation variability and noise increase the risk of such deviations. Short chunks provide frequent opportunities to correct the plan, but incur the cost of frequent WAM inference. The appropriate execution length therefore varies not only across tasks but also across phases of the same task. Therefore, the key challenge is not merely choosing a better chunk size, but deciding *when the WAM's imagined future should still be trusted during physical execution.*

Existing adaptive execution methods adjust chunk length using action uncertainty, entropy, or policy-side confidence (Jing et al., 2025; Wang et al., 2026a; Liang et al., 2026; Wang et al., 2026b; Wu et al., 2026). These signals reflect confidence in predicted actions, but may not reveal whether those actions have changed the world as expected. Unexpected object motion or contact can invalidate

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

even a confident plan. WAMs offer complementary evidence: their predicted visual dynamics provide an expected rollout against which new observations can be assessed without another WAM call. Yet visual mismatch alone is an unreliable trigger. Benign deviations may leave the plan executable, while a subtle change in pose or contact may invalidate the next action. A verifier must therefore consider the latest observation together with the predicted visual trajectory, remaining actions, and task instruction to decide whether to continue or replan.

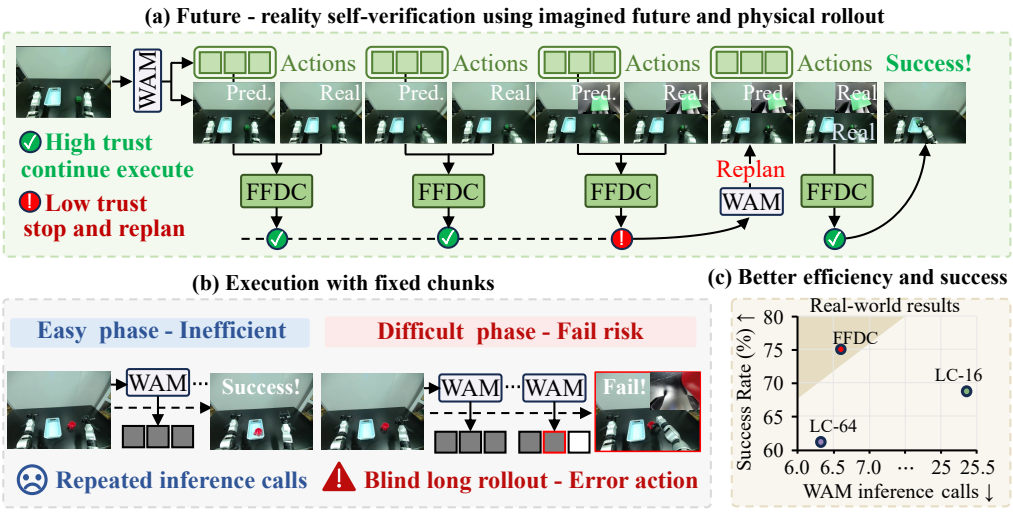

Figure 1: FFDC enables adaptive trust in WAM imagination. (a) FFDC verifies future–reality consistency to continue execution or trigger replanning. (b) Fixed execution horizons are inefficient in predictable phases and unreliable in difficult ones. (c) Success rate versus WAM calls on real-world tasks, illustrating FFDC's robustness–efficiency trade-off.

Based on this insight, we propose an adaptive WAM execution framework that explicitly compares WAM imagination with physical rollout. The core module is *Future Forward Dynamics Causal Attention* (FFDC), a lightweight verifier that estimates whether the remaining WAM-predicted action segment is still reliable. As shown in Figure 1 (a), the WAM predicts future visual dynamics and an action chunk, while FFDC verifies during execution whether the imagined future remains consistent with the real observation, planned actions, and language instruction. FFDC uses a structured attention mechanism to model the interaction between predicted vision and action, allowing it to detect task-critical mismatches and decide whether the remaining rollout can still be trusted. To equip FFDC with the ability to distinguish reliable imagined futures from deviations that require replanning, we construct a binary verification dataset using valid segments from demonstrations and successful rollouts, together with failure-prone segments from failed rollouts and synthetic action corruptions, and train it to predict the executability of the remaining action segment.

FFDC turns fixed-horizon WAM rollout into adaptive execution, continuing while new observations support the remaining actions and replanning when they do not. It thus executes longer in predictable phases and corrects sooner when the plan becomes unreliable. In real-world experiments, FFDC achieves the highest success rate among the evaluated methods with far fewer WAM calls than fixed short-chunk execution (Figure 1(c)). Notably, across six real-world manipulation tasks, FFDC improves average success by 13.33% over fixed 64-action execution with only 2.33 seconds of additional completion time. Compared with fixed 16-action execution, it reduces WAM calls by 74.06% and completion time by 51.31%, showing that selective replanning improves robustness under real-world variability without frequent WAM inference.

Our contributions are summarized as follows:

- We formulate adaptive WAM execution as a *future–reality verification* problem, where the WAM's predicted visual future is used to assess whether its remaining action rollout can still be trusted.

- We propose *Future Forward Dynamics Causal Attention* (FFDC), a lightweight verifier that models temporally aligned causal interactions between predicted actions, predicted visual dynamics, real observations, and language instructions to detect unreliable future execution.
- We show that FFDC enables adaptive trust in WAM imagination: long execution in predictable phases to reduce inference cost, and short execution in difficult phases to improve robustness. Experiments on RoboTwin and in the real world validate our method.

2 Related work

World action models. World Action Models (WAMs) extend VLA policies through joint video-action modeling, coupling action generation with future visual dynamics (Zhu et al., 2025; Li et al., 2025; Bi et al., 2025; Ye et al., 2026b; Li et al., 2026; Yuan et al., 2026). Unified formulations support forward dynamics, inverse dynamics, and conditional action or video generation, providing rich supervision for learning physical interactions (Zhu et al., 2025; Bi et al., 2025; Li et al., 2025). However, predicted futures are often used primarily for representation learning or action generation. To reduce inference cost, some methods rely on latent world features or omit explicit future rollout at test time (Yuan et al., 2026; Ye et al., 2026a). This leaves the use of predicted futures for online execution verification underexplored. Our work uses WAM predictions as expectations of physical rollout, comparing them with real observations to determine whether the remaining action sequence can still be trusted.

Adaptive action execution. Adaptive execution addresses the limitations of fixed open-loop rollout. Early approaches mitigate compounding errors through corrective supervision and recovery training (Ross et al., 2011; Laskey et al., 2017), while subsequent methods use uncertainty or risk estimates to trigger intervention or replanning (Menda et al., 2019; Hoque et al., 2021; Lee et al., 2025). Recent diffusion-policy and VLA methods adapt execution through multi-horizon prediction (Jing et al., 2025), entropy-based chunk selection (Liang et al., 2026), verification (Wang et al., 2026b), learned scheduling (Wu et al., 2026), or executable-horizon estimation (Wang et al., 2026a). Self-Guidance and Adaptive Chunking (SGAC) (So et al., 2026), our directly comparable adaptive baseline, retains or replaces the action queue based on similarity between previously planned and newly predicted actions, requiring fresh policy inference for consistency checks. FFDC instead exploits cached WAM predictions to perform lightweight *future–reality verification*, jointly reasoning over predicted visual dynamics, real observations, planned actions, and task semantics to decide whether to continue execution or replan.

3 Method

In this section, we present FFDC-WAM, a framework that combines low-frequency macro planning with high-frequency lightweight verification for efficient adaptive action execution by leveraging the joint video-action modeling capability of WAMs. We first introduce the standard action-chunking method in WAMs and adaptive action execution in Section 3.1. We then present the architecture of FFDC-WAM in Section 3.2, where a lightweight verifier performs high-frequency verification through a causal attention mechanism over visual and action modalities. Finally, in Section 3.3, we describe the training strategies for the long-horizon WAM and the verifier module.

3.1 Preliminary

World action model with action chunking. We build on Motus (Bi et al., 2025), a world action model (WAM) that jointly predicts future actions and future visual observations conditioned on the current observation and language instruction. During training, the model is optimized with rectified flow-matching losses for both action and video prediction:

$$\mathcal{L}_{\text{WAM}} = \mathcal{L}_{\text{act}} + \mathcal{L}_{\text{vid}}. \tag{1}$$

At inference time, given the current observation o_t and instruction ℓ, the WAM predicts a future action chunk and corresponding latent future visual tokens:

$$(\hat{A}_{t+1:t+H}, \hat{O}_{t+1:t+H}) = \pi_\theta(o_t, \ell), \tag{2}$$

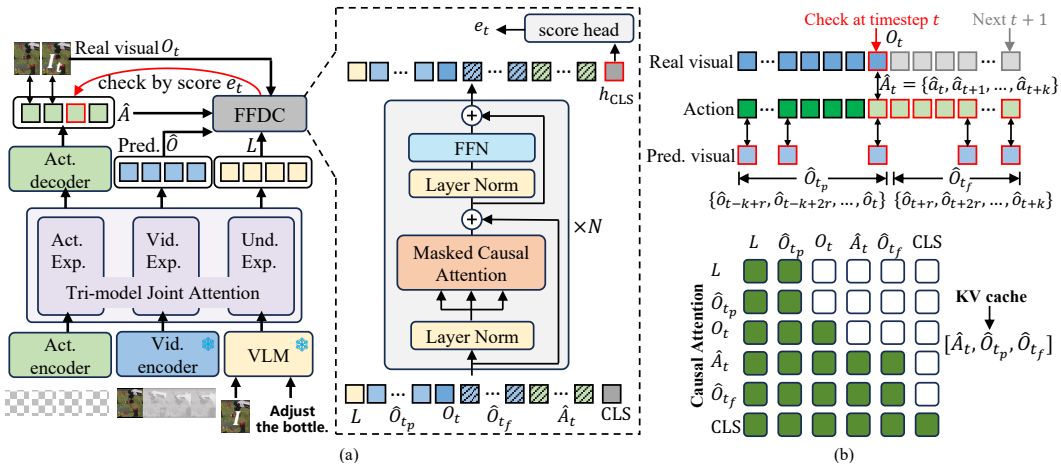

Figure 2: Overview of the proposed FFDC-WAM. (a) Given the action sequence, predicted video tokens, and semantic tokens generated by WAM, the FFDC verifier outputs an execution confidence score for the remaining plan. (b) At each check step t, FFDC performs structured causal attention, enforcing temporally aligned interaction between action and predicted visual dynamics.

where $\hat{A}_{t+1:t+H}$ denotes the predicted action chunk of length H, and $\hat{O}_{t+1:t+H}$ denotes the predicted latent visual sequence.

Adaptive action execution. Standard action chunking executes the predicted chunk in an open-loop manner and replans only after all H actions are finished. While efficient, this fixed execution scheme can accumulate errors in dynamic or contact-rich scenarios.

To enable adaptive execution, we introduce a verifier μ_ϕ that decides whether the remaining predicted rollout is still trustworthy. After executing part of the current chunk, the verifier takes the latest observation, predicted future actions, predicted future visual tokens, and instruction as input:

$$e_t = \mu_\phi\left(o_t, \hat{A}_{t:t+k}, \hat{O}_{t:t+k}, \ell\right), \tag{3}$$

where $e_t \in [0, 1]$ is a confidence score. The robot continues execution if $e_t \geq \tau$, and replans otherwise:

$$\text{execute if } e_t \geq \tau, \qquad \text{replan if } e_t < \tau, \tag{4}$$

where $\tau = 0.5$ in this paper. The objective of adaptive execution is to retain the efficiency advantage of chunked inference while restoring responsiveness to execution failures and environmental changes.

3.2 Future Forward Dynamics Causal Attention

Verifier architecture. To determine whether the remaining predicted plan is still reliable under the latest observation, we introduce a verifier based on *Future Forward Dynamics Causal Attention* (FFDC). As illustrated in Figure 2 (a), each WAM inference produces a predicted action sequence $\hat{A}$, the corresponding latent video tokens $\hat{O}$, and the semantic tokens L from the Understanding expert.

At verification step t, the verifier takes as input the current real observation tokens O_t, the semantic tokens L, the WAM-predicted historical video tokens $\hat{O}_{t_p}$, the WAM-predicted future video tokens $\hat{O}_{t_f}$, the future action segment $\hat{A}_t$, and a learnable `[CLS]` token for global aggregation. The resulting verifier input sequence is

$$X_t = [L, \hat{O}_{t_p}, O_t, \hat{O}_{t_f}, \hat{A}_t, \texttt{[CLS]}]. \tag{5}$$

Future forward dynamics causal attention. To verify whether the next predicted action segment remains executable, we consider a horizon-k candidate rollout $\hat{A}_t = [\hat{a}_t, \hat{a}_{t+1}, \ldots, \hat{a}_{t+k}]$. We also collect temporally aligned WAM-predicted visual tokens around timestep t, including a past segment $\hat{O}_{t_p} = [\hat{o}_{t-k+r}, \hat{o}_{t-k+2r}, \ldots, \hat{o}_t]$ and a future segment $\hat{O}_{t_f} = [\hat{o}_{t+r}, \hat{o}_{t+2r}, \ldots, \hat{o}_{t+k}]$, where r is the action-to-video frequency ratio. In addition, we use instruction-conditioned semantic tokens L from the understanding expert and the latest real observation token O_t.

A key design choice is that the WAM-predicted tokens, including past/future visual tokens $(\hat{O}_{t_p}, \hat{O}_{t_f})$, action tokens $\hat{A}_t$, and understanding-expert tokens L, are produced once after WAM inference and then stored as a *KV cache*. During execution, the verifier only encodes the latest real observation O_t and performs lightweight attention against these cached tokens, which makes score computation efficient without rerunning the full WAM.

FFDC is implemented as an N-layer Transformer with a structured Boolean visibility matrix M, where $M(i, j) = 1$ means token x_i can attend to token x_j. The mask enforces causal interaction between future actions and future predicted dynamics. Specifically, besides attending to L, O_t, and $\hat{O}_{t_p}$, each future visual token $\hat{O}^{(j)}_{t_f}$ attends only to $\{\hat{O}^{(\leq j)}_{t_f}, \hat{A}^{(\leq t+jr)}_t\}$, and each future action token attends only to $\{\hat{O}^{(\leq j)}_{t_f}, \hat{A}^{(\leq t+jr)}_t\}$. To further reduce computation, this attention is applied with a local window over the temporally ordered future tokens, so each token interacts only with nearby aligned action/visual tokens rather than the full future sequence. This preserves temporal causality, avoids information leakage, and keeps the verifier lightweight.

Finally, a `[CLS]` token attends to the full visible sequence and aggregates the execution state into a compact representation. Its output is passed through an MLP head g_ψ to produce

$$z_t = g_\psi(\mathrm{FFDC}(X_t)_{\texttt{[CLS]}}), \tag{6}$$

followed by

$$e_t = \sigma(z_t) \in [0, 1], \tag{7}$$

where a larger e_t indicates higher confidence that the future action segment $\hat{A}_t$ remains valid under the latest real observation.

3.3 Training strategy and dataset construction

To improve trajectory coverage for long-horizon inference, we train WAM with a mixture-of-horizon sampling strategy. For an episode of length T, we uniformly sample a conditioning timestep $s \sim \mathcal{U}\{1, \ldots, T\}$. Given horizon H, the action and video indices are defined as $\tau_i = \min(s + i, T)$, and $v_j = \min(s + jr, T)$, where $i = 0, \ldots, H - 1$ and $j = 0, \ldots, H/r - 1$, which yield action and video sequences

$$A_s = [a_{\tau_0}, \ldots, a_{\tau_{H-1}}], \qquad O_s = [o_{v_0}, \ldots, o_{v_{H/r-1}}]. \tag{8}$$

Out-of-range positions are padded by repeating the final valid action or frame. This allows late-stage states to serve as conditioning starts during training and reduces the bias toward early-episode prefixes.

For FFDC verifier training, we construct a binary dataset $\mathcal{D}_{\mathrm{ver}} = \{(X, y)\}$, where X denotes the verifier input and $y \in \{0, 1\}$ indicates whether the future action segment is executable, with $y = 1$ for valid segments and $y = 0$ for failure-inducing ones. Positive samples are collected from demonstration data and a small number of successful rollouts. Negative samples are obtained from a small number of failed rollouts and from corrupted action segments synthesized from valid demonstrations. The data augmentation methods include temporal swap, gripper flip, late-stage Gaussian noise, and tail scaling. The temporal swap operator randomly swaps two pairs of actions within a horizon; gripper flip negates the designated gripper dimensions; late-stage Gaussian noise perturbs the second half of the sequence; and tail scaling shrinks a randomly sampled suffix by a random scale factor. Using the resulting dataset, the verifier is trained as a binary classifier with the loss

$$\mathcal{L}_{\mathrm{ver}} = -\Big[y \log \sigma(z) + (1 - y) \log\big(1 - \sigma(z)\big)\Big]. \tag{9}$$

4 EXPERIMENTS

Experimental setups. We evaluate FFDC-WAM on RoboTwin (Chen et al., 2025) and on six real-world tasks. We implement all models in PyTorch with Motus (Bi et al., 2025) as the WAM backbone. Backbone training uses four NVIDIA A100 GPUs (80GB each), verifier training uses one A100, and evaluation runs on a single A100.

We evaluate on 50 RoboTwin tasks (Chen et al., 2025) under clean and randomized settings with 100 rollouts per task and setting, and on six real-world tasks using a Unitree G1D dual-arm robot with Dex-1 grippers. We report success rate (SR), average completion time (T), and WAM planning/replanning calls (Calls), excluding SGAC's additional verification inferences. We use a fixed split of five hard RoboTwin tasks and 45 easy tasks based on the Base-Motus SR, with overall SR averaged across all 50 tasks and per-task results in Appendix A.1. **Base-Motus** uses a 16-action training and execution horizon, while **LC-16/32/48/64** execute different prefixes of a MoH-trained, 64-action backbone. **FFDC-WAM** and **SGAC** (So et al., 2026) share this backbone, adapting execution through future–reality verification and action-consistency checks, respectively.

4.1 MAIN RESULTS

Table 1: RoboTwin results under (a) clean and (b) randomized conditions. Each panel reports the 45 easy tasks, five hard tasks, and task-weighted average over all 50 tasks. †Calls excludes SGAC's additional WAM inferences for verification.

(a) Clean

	Easy			Hard			All tasks		
Method	SR (%)	T (s)	Calls	SR (%)	T (s)	Calls	SR (%)	T (s)	Calls
Base-Motus	**90.98**	20.4	5.22	57.80	29.5	7.71	87.66	21.3	5.47
SGAC†	87.02	15.1	1.73	74.00	21.6	2.83	85.72	15.8	1.84
Latent cosine gate	87.24	13.0	1.68	72.40	19.3	2.69	85.76	13.6	1.78
Pixel cosine gate	87.96	30.5	1.95	75.60	35.2	2.53	86.72	31.0	2.01
LC-16	86.82	18.3	4.57	64.40	26.7	6.84	84.58	19.1	4.80
LC-32	89.38	13.4	2.57	70.60	18.3	3.56	87.50	13.8	2.67
LC-48	89.73	11.4	1.70	67.40	16.1	2.52	87.50	11.8	1.79
LC-64	88.98	10.8	1.62	73.80	14.0	2.03	87.46	11.1	1.66
FFDC-WAM	90.33	12.9	1.62	**76.00**	18.8	2.60	**88.90**	13.5	1.72

(b) Randomized

	Easy			Hard			All tasks		
Method	SR (%)	T (s)	Calls	SR (%)	T (s)	Calls	SR (%)	T (s)	Calls
Base-Motus	89.16	23.5	5.22	54.20	33.0	7.73	85.66	24.5	5.47
SGAC†	85.69	18.9	1.89	72.80	23.4	2.47	84.40	19.4	1.95
Latent cosine gate	86.91	16.1	1.70	72.20	21.2	2.37	85.44	16.6	1.77
Pixel cosine gate	86.42	30.3	1.94	73.20	36.8	2.81	85.10	31.0	2.03
LC-16	85.58	21.2	4.55	67.40	29.4	6.68	83.76	22.0	4.76
LC-32	88.64	16.0	2.57	71.60	21.2	3.55	86.94	16.5	2.67
LC-48	88.49	13.9	1.72	65.00	19.0	2.59	86.14	14.4	1.81
LC-64	88.89	13.3	1.64	74.00	16.5	2.06	87.40	13.7	1.68
FFDC-WAM	**89.51**	15.7	1.62	**76.40**	20.5	2.34	**88.20**	16.2	1.69

Results on RoboTwin. On RoboTwin (Table 1), FFDC-WAM achieves the highest average SR in both clean (88.90%) and randomized (88.20%) settings. Under randomization, it improves SR over Base-Motus by 2.54 percentage points while reducing completion time by 33.88% and WAM calls by 69.10%. Gains are most pronounced on hard tasks, where SR increases from 57.80% to 76.00% in clean conditions and from 54.20% to 76.40% under randomization; on easy tasks, it

maintains comparable SR with shorter execution time. Compared with the same-backbone LC-64, FFDC-WAM improves hard-task SR by 2.20 and 2.40 percentage points, respectively.

Compared with adaptive baselines, FFDC-WAM exceeds SGAC's SR by 3.18 and 3.80 percentage points in clean and randomized settings, respectively, while completing tasks faster. Each FFDC check takes 0.015 s on an A100, compared with 1.9 s for SGAC, which requires fresh WAM inference. Latent cosine gating achieves comparable execution time but lower SR by 3.14 and 2.76 percentage points, respectively. Pixel-level gating also yields lower SR and incurs additional decoding cost, taking 1.16 s per check and an average completion time of 31.0 s in both settings. These results support jointly reasoning over predicted dynamics, real observations, actions, and task semantics for accurate and efficient execution verification.



Figure 3: Adaptive execution on RoboTwin. (a) On *Move Can Pot*, FFDC completes the illustrated rollout with one WAM inference instead of three for fixed 16-action execution. (b) On *Hanging Mug*, it replans during the precision-critical stage and succeeds, whereas fixed 64-action execution fails. C_i denotes an executed action chunk.

Execution behavior. Figure 3 illustrates how verification changes the executed horizon. On *Move Can Pot*, high confidence allows one 43-action chunk to complete the rollout, while fixed 16-action execution uses three WAM calls. On *Hanging Mug*, FFDC executes 64 actions in the first chunk, then interrupts the second after 32 actions when confidence falls below the threshold; a final six-action chunk completes the task. As shown in the lower row, ignoring this confidence drop and continuing with fixed 64-action execution leads to failure. These examples illustrate both continued execution when a plan remains reliable and timely replanning when confidence deteriorates.

4.2 Real-world experiments

Table 2 reports results on six manipulation tasks using a Unitree G1D dual-arm robot. FFDC-WAM achieves 75.00% average SR, improving over Base-Motus-LC16 by 9.17 percentage points while reducing average completion time from 84.42 to 41.10 s (51.31%) and WAM calls from 25.44 to 6.60 (74.06%). Success improves on all six tasks by 5–15 percentage points, with lower completion time and fewer calls on every task. Compared with fixed LC-64, FFDC-WAM improves average SR by 13.33 percentage points with only 2.33 s and 0.22 additional calls. The largest gain is on Clean Table, from 55% to 75%, followed by 15-point gains on Put Cup and Stack Bowls. These results support combining efficient long-horizon execution with selective correction to improve real-world reliability.

Figure 4 illustrates this behavior on a block-manipulation task. FFDC executes a full 64-action chunk, then interrupts the second chunk after 16 actions when confidence falls. Replanning produces a final 19-action chunk that completes the task, whereas the fixed LC-64 baseline continues its

Table 2: Real-world performance on six manipulation tasks.

	Base-Motus-LC16			LC-64			FFDC-WAM		
Task	SR (%)	T (s)	Calls	SR (%)	T (s)	Calls	SR (%)	T (s)	Calls
Bowl Pouring	65	70.94	19.10	65	35.67	5.15	**75**	39.19	5.65
Clean Table	60	113.27	36.30	55	46.36	8.75	**75**	47.52	9.00
Put Block	70	66.95	17.80	70	34.10	4.90	**80**	36.55	4.95
Put Cup	70	69.24	20.00	65	35.04	4.95	**80**	37.40	5.05
Stack Blocks	55	68.38	19.70	50	34.59	4.80	**60**	37.06	4.90
Stack Bowls	75	117.72	39.75	65	46.87	9.70	**80**	48.90	10.05
Avg.	65.83	84.42	25.44	61.67	38.77	6.38	**75.00**	41.10	6.60

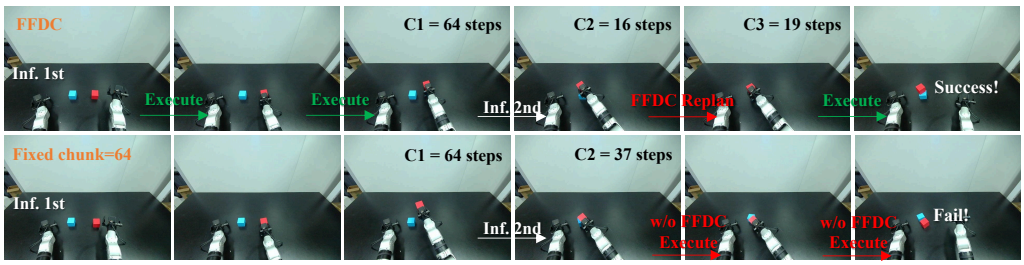

Figure 4: Real-world block manipulation on the G1D robot. FFDC executes a 64-action first chunk, interrupts the next chunk after 16 actions, and completes the illustrated rollout with a 19-action final chunk. The fixed 64-action baseline continues without this intermediate verification and fails.

existing plan and fails. This example demonstrates adaptation within a single rollout: the robot retains long execution while the plan remains trusted and replans when the remaining actions become unreliable, consistent with the success gains and modest additional planning cost observed relative to LC-64.

4.3 Ablation Study

We conduct ablations on the five hard RoboTwin tasks under the more challenging randomized setting (Table 3). In the variant names, **MoH** denotes Mixture-of-Horizon Training. The two **w/o MoH** variants use a backbone trained without MoH, with either fixed 64-action execution (**LC-64**) or adaptive verification (**FFDC**). The input ablations remove instruction-conditioned understanding tokens (**Und**), predicted visual tokens (**Pred**), the current real observation (**Real**), or predicted actions (**Action**). The decision threshold is denoted by τ, with $\tau = 0.5$ as the default. The $\tau = 0.25$ and $\tau = 0.75$ variants change only this threshold to evaluate more permissive and stricter continuation criteria, respectively. **FFDC-WAM** denotes the complete model with the default threshold. BlkRank, HangMug, PlacePad, PutCab, and ScanObj denote Blocks Ranking Size, Hanging Mug, Place Mouse Pad, Put Object in Cabinet, and Scan Object, respectively.

Verifier analysis. The complete verifier achieves 76.4% average SR. Removing Pred, Real, Action, or Und reduces SR to 71.6%, 72.4%, 73.4%, and 74.8%, respectively, while increasing completion time. Predicted visuals contribute the largest average gain, with their removal also reducing Hanging Mug SR from 44% to 32%. These results support jointly considering all four inputs. Offline evaluation on approximately 2,000 task-and-seed-matched FFDC and LC-64 episodes, labeled by final episode outcomes, yields an AUROC of 0.786 and an AUPRC of 0.761. Matched LC-64 rollouts without verification fail in 77.8% of cases associated with a replan prediction, versus 31.6% with a continue prediction, supporting the verifier's ability to identify unreliable continuation.

MoH training and decision threshold. MoH improves SR from 68.0% to 74.0% for fixed LC-64 and from 71.4% to 76.4% for FFDC-WAM; the MoH-trained LC-64 results are reported in Table 1.

Table 3: Ablations on five hard RoboTwin tasks under randomization: (a) verifier inputs; (b) MoH training and decision threshold. SR is in percent; T is in seconds. Average denotes the mean across the five tasks. Bold marks the highest SR in each row within each panel.

(a) Verifier inputs

	w/o Und			w/o Pred			w/o Real			w/o Action			FFDC-WAM		
Task	SR	T (s)	Calls	SR	T (s)	Calls	SR	T (s)	Calls	SR	T (s)	Calls	SR	T (s)	Calls
BlkRank	91.0	26.1	3.34	85.0	24.9	3.11	86.0	27.8	3.70	90.0	25.5	3.31	**93.0**	25.3	3.27
HangMug	37.0	29.2	4.16	32.0	27.8	4.03	38.0	27.3	3.92	33.0	28.9	4.24	**44.0**	27.5	3.96
PlacePad	86.0	12.5	1.23	84.0	12.4	1.24	87.0	12.3	1.20	**88.0**	13.1	1.43	**88.0**	12.4	1.27
PutCab	72.0	26.6	3.49	71.0	21.9	2.01	68.0	21.7	2.00	69.0	22.1	2.00	**73.0**	21.7	2.00
ScanObj	**88.0**	16.0	1.25	86.0	16.8	1.36	83.0	15.9	1.23	87.0	15.6	1.22	84.0	15.7	1.20
Average	74.8	22.1	2.69	71.6	20.8	2.35	72.4	21.0	2.41	73.4	21.0	2.44	**76.4**	20.5	2.34

(b) MoH training and decision threshold

	w/o MoH LC-64			w/o MoH FFDC			$\tau = 0.25$			$\tau = 0.75$			FFDC-WAM		
Task	SR	T (s)	Calls	SR	T (s)	Calls	SR	T (s)	Calls	SR	T (s)	Calls	SR	T (s)	Calls
BlkRank	77.0	23.8	3.49	90.0	26.7	3.31	91.0	25.6	3.18	91.0	25.7	3.23	**93.0**	25.3	3.27
HangMug	40.0	19.8	2.00	**44.0**	25.9	3.45	39.0	27.8	3.90	42.0	27.7	4.00	**44.0**	27.5	3.96
PlacePad	87.0	9.9	1.06	67.0	15.1	1.59	**89.0**	12.6	1.26	88.0	13.4	1.44	88.0	12.4	1.27
PutCab	48.0	21.3	2.04	**75.0**	22.8	2.03	69.0	22.6	2.01	68.0	22.2	2.04	73.0	21.7	2.00
ScanObj	**88.0**	13.1	1.13	81.0	15.3	1.19	85.0	16.9	1.28	87.0	15.6	1.14	84.0	15.7	1.20
Average	68.0	17.6	1.94	71.4	21.2	2.31	74.6	21.1	2.33	75.2	20.9	2.37	**76.4**	20.5	2.34

Verification improves SR both with and without MoH, indicating complementary contributions from long-horizon training and online correction. Setting τ to 0.25 or 0.75 yields 74.6% or 75.2% SR, versus 76.4% at the default. Across these thresholds, completion time remains within 20.5–21.1 s and calls within 2.33–2.37, showing limited sensitivity over the tested range.

5 Conclusion

Inspired by how humans compare predicted future feedback with actual observation during physical interaction, we reformulate adaptive WAM execution as a *future–reality verification* problem. Instead of using a fixed execution horizon, our method asks a more fundamental question: *whether the WAM's imagined future can still be trusted during rollout*. To this end, we propose **FFDC-WAM**, where the lightweight *Future Forward Dynamics Causal Attention* (FFDC) verifier jointly models temporally aligned interactions among predicted future actions, predicted visual dynamics, real observations, and language instructions to detect unreliable future execution.

This design enables adaptive trust in WAM imagination: the robot can continue execution when the predicted future remains reliable, and trigger replanning once future–reality consistency breaks down. In this way, FFDC-WAM moves WAM execution beyond fixed-horizon chunking toward a more principled reliability-aware control strategy, improving both efficiency and robustness across simulation and real-world settings. More broadly, our results suggest that the key to effective WAM deployment is not selecting a single execution length, but endowing the system with the ability to verify its own imagined future online. A current limitation is that FFDC is trained with binary supervision derived from successful, failed, and synthetically corrupted segments, which may not cover the full diversity of real-world execution deviations. Extending the verifier to learn from richer failure modes and more diverse real-world data is an important direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with language polishing, including improving grammar, clarity, and readability. All AI-assisted edits were reviewed and revised by the authors. We take full responsibility for the final content of this paper, including its claims, results, and conclusions.

REFERENCES

BeingBeyond Team. Being-h0.7: A latent world-action model from egocentric videos, 2026. URL https://research.beingbeyond.com/being-h07. Accessed: 2026-04-27.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 37:112386–112410, 2024.

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.

Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pp. 143–156. PMLR, 2017.

Sung-Wook Lee, Xuhui Kang, and Yen-Ling Kuo. Diff-dagger: Uncertainty estimation with diffusion policy for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4845–4852. IEEE, 2025.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5041–5048. IEEE, 2019.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Junhyuk So, Chiwoong Lee, Shinyoung Lee, Jungseul Ok, and Eunhyeok Park. Improving generative behavior cloning via self-guidance and adaptive chunking. *Advances in Neural Information Processing Systems*, 38:84432–84456, 2026.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits. *arXiv preprint arXiv:2602.21445*, 2026a.

Zihua Wang, Zhitao Lin, Ruibo Li, Yu Zhang, Xu Yang, Siya Mi, and Xiu-Shen Wei. Open-loop planning, closed-loop verification: Speculative verification for vla. *arXiv preprint arXiv:2604.02965*, 2026b.

Zhichao Wu, Junyin Ye, Zhilong Zhang, Yihao Sun, Haoxin Lin, Jiaheng Luo, Haoxiang Ren, Lei Yuan, and Yang Yu. Speedup patch: Learning a plug-and-play policy to accelerate embodied manipulation. *arXiv preprint arXiv:2603.20658*, 2026.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026a.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

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*, 2026b.

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.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

A Appendix

A.1 More results

Table 4: Results on hard tasks under random conditions.

	Base-Motus		LC-16		LC-32		LC-48		LC-64		FFDC-WAM	
Task	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)
BlkRank	62	49.1	66	43.1	78	30.0	62	26.7	86	21.5	**93**	25.3
HangMug	38	36.2	42	31.8	35	24.6	26	22.0	33	19.0	**44**	27.5
PlacePad	64	23.5	76	19.9	85	13.7	86	11.5	**90**	10.5	88	12.4
PutCab	53	32.5	67	29.7	67	21.0	66	18.7	**75**	18.0	73	21.7
ScanObj	54	23.9	86	22.6	**93**	16.7	85	16.0	86	13.8	84	15.7
Average	54.2	33.0	67.4	29.4	71.6	21.2	65.0	19.0	74.0	16.5	**76.4**	20.5

Table 5: Results on hard tasks under clean conditions.

	Base-Motus		LC-16		LC-32		LC-48		LC-64		FFDC-WAM	
Task	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)
BlkRank	78	40.3	65	40.9	78	26.2	63	21.5	81	18.5	**84**	25.0
HangMug	27	30.8	27	29.5	23	22.2	21	19.7	26	16.8	**30**	26.5
PlacePad	58	28.2	80	18.9	85	11.5	89	8.0	**95**	7.5	94	10.1
PutCab	62	29.3	60	26.1	76	18.2	73	16.2	79	16.1	**80**	19.2
ScanObj	64	18.7	90	18.1	91	13.3	91	15.1	88	11.3	**92**	13.2
Average	57.8	29.5	64.4	26.7	70.6	18.3	67.4	16.1	73.8	14.0	**76.0**	18.8

Table 6: Results on easy tasks on Clean Settings.

	Base-Motus		LC-16		LC-32		LC-48		LC-64		FFDC-WAM	
Task	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)
adjust_bottle	91	13.1	**100**	12.2	**100**	9.8	**100**	7.4	88	7.5	**100**	8.7
beat_block_hammer	**97**	13.3	94	13.6	84	10.5	85	8.5	83	8.8	90	11.6
blocks_ranking_rgb	**100**	36.3	86	40.3	98	24.3	81	22.0	94	18.2	98	21.2
click_alarmclock	**100**	7.7	**100**	7.6	**100**	5.3	**100**	5.3	99	5.4	**100**	6.1
click_bell	**100**	7.8	**100**	7.6	**100**	5.2	**100**	5.2	97	5.2	**100**	5.9
dump_bin_bigbin	**98**	18.1	97	16.8	94	12.8	96	10.6	92	10.7	**98**	12.0
grab_roller	**100**	9.2	**100**	9.1	**100**	6.4	**100**	6.5	94	6.6	**100**	7.4
handover_block	**87**	25.3	44	29.8	73	17.6	68	13.7	68	13.9	60	20.1
handover_mic	84	21.7	95	19.8	99	15.6	96	13.5	99	13.5	**100**	15.4
lift_pot	**99**	16.8	92	11.2	91	11.4	93	8.5	97	8.3	93	10.4
move_can_pot	37	25.4	**86**	13.5	82	10.6	85	8.5	85	8.4	82	9.9
move_pillbottle_pad	96	16.0	91	12.2	**98**	9.5	96	7.6	97	7.2	96	8.5
move_playingcard_away	99	12.9	99	11.2	**100**	8.6	99	6.4	97	6.4	99	7.5
move_stapler_pad	82	32.8	85	22.8	84	15.1	**90**	14.6	82	12.0	88	15.4
open_laptop	94	16.5	98	13.9	97	11.0	98	10.2	**100**	9.3	98	11.1
open_microwave	**92**	46.1	75	26.7	54	16.5	71	14.9	79	12.9	71	17.0
pick_diverse_bottles	**89**	12.4	71	12.4	67	10.2	68	8.0	69	8.0	70	9.4
pick_dual_bottles	**95**	12.2	75	12.3	74	10.0	73	7.7	73	7.7	74	9.3
place_a2b_left	88	22.7	93	13.3	**97**	10.0	96	8.1	94	7.9	96	9.1
place_a2b_right	84	24.1	94	12.7	97	10.0	**99**	8.2	92	8.0	98	9.3
place_bread_basket	90	22.1	92	20.5	95	15.5	85	14.1	91	13.0	**96**	14.9
place_bread_skillet	86	15.7	86	14.3	89	12.1	**92**	9.7	91	10.1	90	11.9
place_burger_fries	98	22.5	94	24.4	93	17.6	96	15.5	**99**	15.2	96	17.3
place_can_basket	82	34.5	57	30.0	75	21.1	**87**	18.9	78	18.9	80	22.4
place_cans_plasticbox	98	23.9	69	27.5	99	18.4	**100**	16.1	98	15.8	**100**	18.0
place_container_plate	98	12.7	97	12.2	**100**	9.8	99	7.5	98	7.5	98	8.7
place_dual_shoes	**93**	25.3	76	26.2	81	19.2	77	15.5	78	15.5	77	18.6
place_empty_cup	99	15.8	**100**	12.8	99	10.5	96	9.9	**100**	8.1	99	9.9
place_fan	94	17.2	87	14.7	94	11.8	92	9.6	93	8.7	**96**	10.6
place_object_basket	**84**	31.2	56	27.9	77	21.2	60	18.7	70	17.3	71	19.7
place_object_scale	81	16.3	78	14.0	86	11.0	**94**	8.3	91	8.3	92	9.5
place_object_stand	98	12.1	92	12.3	96	9.9	92	7.5	**99**	7.3	95	8.6
place_phone_stand	89	12.6	93	12.3	95	10.7	93	8.0	**96**	7.8	**96**	9.1
place_shoe	**100**	15.6	99	15.5	99	11.2	99	9.3	99	8.4	98	10.0
press_stapler	**96**	9.5	**96**	10.8	93	6.6	93	6.7	90	6.8	93	7.6
put_bottles_dustbin	**88**	48.6	77	45.6	78	32.6	80	26.7	75	26.4	78	36.7
rotate_qrcode	83	17.6	88	14.3	88	11.0	**92**	9.0	**92**	8.3	**92**	9.9
shake_bottle_horizontally	**100**	10.0	99	9.4	**100**	7.3	**100**	6.4	97	6.5	99	7.4
shake_bottle	**100**	9.8	**100**	9.4	**100**	7.2	**100**	6.4	**100**	6.5	**100**	7.5
stack_blocks_three	**86**	37.1	82	34.5	77	23.8	85	21.7	80	18.4	84	21.4
stack_blocks_two	**100**	24.4	99	22.4	96	15.9	**100**	14.1	97	12.6	**100**	14.8
stack_bowls_three	71	41.5	70	43.3	75	30.3	**82**	26.2	63	24.8	77	27.2
stack_bowls_two	96	26.1	95	24.2	98	17.4	98	15.5	**100**	14.8	94	17.3
stamp_seal	**98**	16.7	90	14.6	90	11.5	93	8.1	88	8.0	92	10.5
turn_switch	**74**	9.4	60	11.8	60	6.2	59	6.3	62	6.5	61	7.0
Average	**90.98**	20.4	86.82	18.3	89.38	13.4	89.73	11.4	88.98	10.8	90.33	12.9

Table 7: Results on easy tasks under random conditions.

	Base-Motus		LC-16		LC-32		LC-48		LC-64		FFDC-WAM	
Task	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)	SR (%)	T (s)
adjust_bottle	87	15.5	98	14.7	98	12.2	**99**	9.6	93	9.8	98	11.2
beat_block_hammer	90	15.4	**96**	15.2	92	12.1	95	10.5	91	10.4	94	11.7
blocks_ranking_rgb	93	44.6	91	41.3	94	28.6	80	27.5	94	21.1	**97**	24.7
click_alarmclock	99	10.3	**100**	10.1	**100**	7.8	**100**	7.6	98	7.8	**100**	8.5
click_bell	**100**	10.2	**100**	10.3	**100**	7.5	**100**	7.5	97	7.6	**100**	8.3
dump_bin_bigbin	94	22.2	**95**	20.4	92	14.3	88	12.3	92	12.7	**95**	14.3
grab_roller	**100**	12.3	**100**	11.5	**100**	8.8	**100**	8.8	99	9.0	**100**	10.0
handover_block	**74**	37.7	38	30.2	57	19.7	57	16.6	61	16.7	57	19.5
handover_mic	69	26.5	97	24.6	**99**	19.3	98	17.2	98	17.5	98	19.6
lift_pot	96	17.0	92	14.2	91	14.1	97	10.5	**98**	10.6	**98**	12.9
move_can_pot	76	23.3	74	15.6	**85**	13.0	83	10.6	80	10.7	84	12.4
move_pillbottle_pad	95	19.9	94	14.6	98	11.8	**99**	9.6	98	9.5	98	10.9
move_playingcard_away	95	15.6	98	13.8	**99**	11.5	**99**	8.7	97	8.9	**99**	10.2
move_stapler_pad	82	29.5	77	28.5	82	19.5	86	16.2	**88**	15.5	85	18.9
open_laptop	90	19.6	97	18.4	97	14.3	98	13.5	**99**	11.9	97	14.3
open_microwave	**92**	50.4	60	33.9	55	17.8	59	19.9	56	18.0	60	20.8
pick_diverse_bottles	**85**	14.7	65	14.4	63	12.3	61	9.7	69	9.8	66	11.6
pick_dual_bottles	**88**	14.8	63	14.4	59	12.2	58	9.5	64	9.6	63	11.2
place_a2b_left	84	22.9	92	15.3	95	12.6	92	10.3	**97**	10.2	93	11.8
place_a2b_right	80	25.3	94	15.9	97	12.6	97	10.7	94	10.3	**98**	11.9
place_bread_basket	93	23.7	89	23.4	91	16.8	82	15.7	91	15.1	**96**	17.0
place_bread_skillet	84	18.6	83	18.2	93	15.0	84	12.2	**94**	12.2	90	13.6
place_burger_fries	**99**	25.6	91	27.2	91	20.2	96	18.9	90	18.0	90	20.9
place_can_basket	**76**	37.7	66	32.5	71	28.3	**76**	23.6	67	25.3	72	26.5
place_cans_plasticbox	95	27.3	79	33.1	**100**	21.0	99	18.7	94	18.5	97	20.7
place_container_plate	98	15.1	**100**	14.9	98	12.0	99	9.8	98	9.8	**100**	11.3
place_dual_shoes	**90**	29.4	65	27.6	83	22.2	74	18.7	76	18.5	77	24.1
place_empty_cup	**100**	19.4	98	16.3	99	12.8	99	12.5	98	10.4	99	12.5
place_fan	80	18.1	88	17.8	91	13.6	96	11.5	**99**	11.1	94	12.6
place_object_basket	81	38.7	73	33.0	78	26.0	73	23.0	**85**	23.5	77	26.1
place_object_scale	87	20.0	73	16.3	89	14.0	**96**	11.1	95	10.4	94	11.7
place_object_stand	97	15.2	92	14.8	98	12.1	**99**	9.6	95	9.8	**99**	11.2
place_phone_stand	81	14.9	91	15.1	90	12.2	92	9.9	**93**	10.1	90	11.0
place_shoe	98	18.9	98	18.4	99	13.1	**100**	11.0	98	10.5	**100**	12.3
press_stapler	97	13.3	**98**	12.4	**98**	9.8	**98**	10.1	97	9.8	**98**	10.9
put_bottles_dustbin	**77**	52.0	70	48.5	66	32.6	74	28.1	65	24.7	73	36.7
rotate_qrcode	75	22.8	86	17.6	**92**	13.6	89	11.2	90	10.6	90	12.4
shake_bottle_horizontally	95	12.6	98	12.1	98	9.9	98	8.9	**100**	8.8	99	10.0
shake_bottle	97	13.0	96	12.1	96	9.8	**98**	8.8	96	8.8	**98**	10.1
stack_blocks_three	**94**	39.8	74	37.5	79	26.9	85	23.0	79	21.4	82	25.3
stack_blocks_two	97	27.8	99	26.3	**100**	18.7	99	16.6	**100**	15.3	97	17.9
stack_bowls_three	**85**	43.8	76	43.0	80	31.2	79	27.1	82	25.6	82	31.1
stack_bowls_two	97	30.9	97	27.9	96	20.2	98	17.2	**99**	16.5	**99**	19.6
stamp_seal	**94**	16.6	79	18.1	89	14.8	84	11.1	83	10.2	84	12.6
turn_switch	**76**	13.4	71	12.9	71	9.4	69	8.3	73	8.9	71	11.5
Average	89.16	23.5	85.58	21.21	88.64	16.0	88.49	13.9	88.89	13.3	**89.51**	15.7