LIBERO-DYNAMIC: BENCHMARKING ROBOTIC MANIPULATION IN DYNAMIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most simulation benchmarks for robotic manipulation keep target objects static and treat policy inference as instantaneous. These assumptions simplify repeatable evaluation, but they hide how a moving target evolves during computation and how stale an observation may be when its predicted action is executed. We introduce LIBERO-Dynamic, a controlled dynamic extension of all 40 LIBERO tasks. It drives objects through contact friction along three trajectory families and three speed tiers, while continuously advancing the physical world according to calibrated, model-specific inference latency. The same motion mechanism also supports training: we replay verified static demonstrations and compensate robot actions online using the target's displacement relative to the reference, producing 9,000 quality-filtered dynamic trajectories. We evaluate six pretrained policies on a shared set of 1,800 dynamic conditions per configuration, compare execution strategies and action horizons with the checkpoint fixed, and fine-tune three architectures using dynamic and mixed data. The experiments show that faster motion degrades every evaluated policy, but not uniformly; execution timing and horizon can change the behavior of the same policy; and, in the measured sweep, asynchronous replanning benefits from frequent feedback only after the action queue can be sustained without underflow. Dynamic demonstrations improve each evaluated architecture under at least one training recipe, while the magnitude of improvement varies with temporal conditioning, data composition, and training budget. In a small-scale physical study using policies trained on real-robot dynamic demonstrations, we observe execution-strategy differences consistent with those found in simulation, with task-dependent effects, providing complementary evidence for the corresponding benchmark findings. By making object motion, inference latency, and action execution explicit and controllable within a broad task suite, LIBERO-Dynamic provides a reproducible platform for evaluating and improving robot policies for moving-target manipulation.

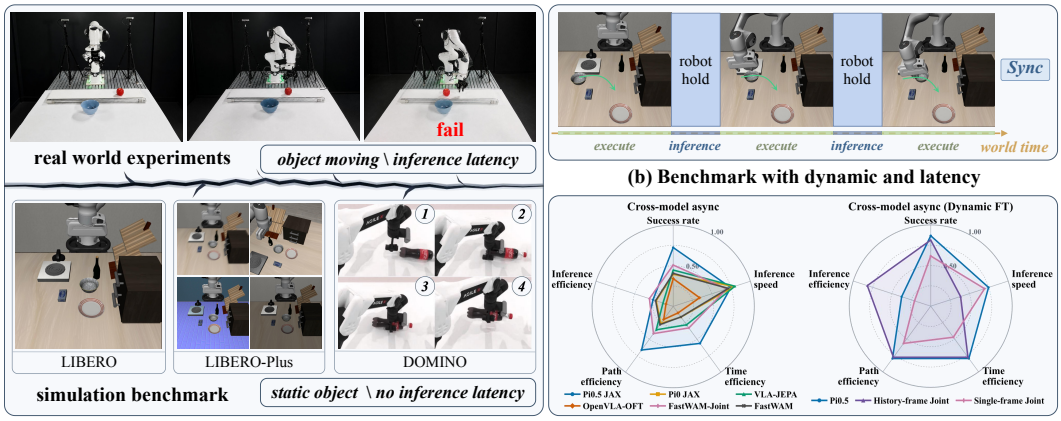
1 INTRODUCTION

General-purpose robot policies—including vision-language-action (VLA) models and policies built on video or world models—have advanced rapidly with growing data scale, architectural innovation, and improved post-training techniques (Brohan et al., 2023; Octo Model Team et al., 2024; Kim et al., 2024; Black et al., 2024; Yuan et al., 2026). Reliable comparison increasingly depends on benchmarks that evaluate not only task completion, but also behavior under deployment-relevant conditions.

Real-robot platforms such as RoboArena (Atreya et al., 2025), RoboChallenge (Yakefu et al., 2025), and RoboDojo (Chen et al., 2026a) provide direct evidence under physical deployment, whereas simulation benchmarks such as LIBERO (Liu et al., 2023), RLBench (James et al., 2020), and RoboTwin (Mu et al., 2024) enable repeatable evaluation across broad task collections. Recent extensions including LIBERO-Plus (Fei et al., 2025), LIBERO-PRO (Zhou et al., 2025), and LIBERO-X (Wang et al., 2026) further examine robustness and generalization under controlled variations.

Dynamic manipulation adds a critical temporal dimension. In common step-based simulation evaluations, policy inference is typically treated as instantaneous, or the physical world does not continue to advance during inference, leaving observations and actions ideally aligned in time. In

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



(a) Real world and simulation benchmark gap

(b) Benchmark with dynamic and latency

(c) Cross-Paradigm Assessment

Figure 1: Overview of LIBERO-Dynamic: **(a)** dynamic real-world manipulation and conventional simulation evaluation; **(b)** latency-aware evaluation; and **(c)** comparisons across execution and training settings.

a static scene, waiting for inference does not change the state of the target object; when the object continues to move, however, the observation underlying a returned action may already be stale. DOMINO (Fang et al., 2026) and DOM (Xie et al., 2026) have begun to study interactions with moving objects, while ReflexBench (Chen et al., 2026b) further exposes the effect of inference latency. Systematically evaluating general manipulation competence and dynamic responsiveness across a broad task collection under a common protocol nevertheless remains underexplored.

To address this gap, we introduce LIBERO-Dynamic, a dynamic manipulation benchmark that combines controlled object motion with model-specific inference latency in a unified simulation loop; its overall framework is shown in Figure 2. Across LIBERO's 40 tasks, LIBERO-Dynamic introduces three trajectory families and three motion-speed tiers, using a friction-driven invisible contact platform to preserve contact interactions among the robot, target, and surrounding objects. The physical world continues to evolve during policy inference, and the benchmark advances simulation according to pre-calibrated model latency so that observation–action staleness can be evaluated explicitly. With the checkpoint fixed, we conduct controlled comparisons of synchronous execution, the LeRobot asynchronous queue (Cadene et al., 2026), and Real-Time Chunking (RTC) (Black et al., 2025a), while treating VLASH (Tang et al., 2025), which uses a separate checkpoint, as a future-state-aware system reference.

Under a unified protocol, we evaluate diverse robot policies and compare dynamic manipulation performance across execution mechanisms and action horizons. We further conduct controlled fine-tuning experiments on three model families using dynamic demonstrations and mixed static–dynamic data, examining how gains from dynamic data vary with the form of temporal conditioning. A compact real-robot study, using policies trained on physical dynamic demonstrations in a DROID-style setup (Khazatsky et al., 2024), provides supplementary evidence for the execution-timing observations under physical sensing and actuation.

In summary, this work makes the following contributions:

1. We introduce LIBERO-Dynamic, a dynamic robotic manipulation benchmark that combines controlled object motion with model-specific inference latency. It defines a controlled spatiotemporal evaluation space through motion pattern, speed, trajectory phase, and perception–action delay for comparing dynamic responsiveness across robot policies.

2. We introduce a method for converting successful static demonstrations into dynamic ones: invisible conveyors drive objects through contact friction, while static manipulation references are adapted to changing target states through online motion compensation. Using this method, we construct a dataset of 9,000 dynamic trajectories spanning multiple motion patterns, speeds, spatial trajectories, and motion phases.

3. We evaluate representative models and execution strategies under a unified protocol and use controlled fine-tuning to analyze the relationship among dynamic data, temporal conditioning, and execution. Dynamic data improves several evaluated policies, while the gains vary with the form of temporal conditioning, providing empirical evidence for model training and execution design in dynamic manipulation.

2 Related Work

Robot Policies and Temporal Modeling. General-purpose robot policies span autoregressive action-token prediction, continuous action generation, and video- or world-model-based control, represented by RT-2 and OpenVLA, OpenVLA-OFT and the π family, and VLA-JEPA and Fast-WAM, respectively (Brohan et al., 2023; Kim et al., 2024; 2025; Black et al., 2024; Physical Intelligence et al., 2025; Sun et al., 2026; Yuan et al., 2026). Recent methods for dynamic settings further incorporate observation history, future-state prediction, or continuous inference–execution coordination into policy design (Fang et al., 2026; Chen et al., 2026b; Xie et al., 2026). Because these systems also differ in training data and execution protocol, direct comparison across reported results is difficult. We evaluate representative policy families under a common task distribution and timing protocol, and study how temporal conditioning interacts with dynamic-data fine-tuning.

Benchmarks and Data for Dynamic Manipulation. LIBERO, RLBench, and RoboTwin support reproducible multi-task manipulation evaluation, while LIBERO-Plus, LIBERO-PRO, LIBERO-X, VLA-Arena, and LIBERO-Safety extend this setting toward robustness, generalization, or safety under controlled variations (Liu et al., 2023; James et al., 2020; Mu et al., 2024; Fei et al., 2025; Zhou et al., 2025; Wang et al., 2026; Zhang et al., 2026; Cui et al., 2026). Dynamic manipulation benchmarks place greater emphasis on time-varying interactions: DOMINO studies dynamic data and temporal generalization, DynamicVLA introduces simulation and real-world data for moving-object manipulation, and ReflexBench decouples simulation stepping from control to expose inference latency. These works respectively cover dynamic data, moving-object interaction, and inference latency; LIBERO-Dynamic focuses on controlled motion of task-relevant objects across LIBERO's 40-task skill set, together with calibrated model latency and dynamic demonstrations for training studies.

Latency-Aware Execution of Action-Chunking Policies. Action chunking trades feedback frequency for temporal consistency: long execution horizons reduce policy queries, whereas short horizons refresh observations more often but can exhaust the action queue under non-negligible inference latency. LeRobot implements a producer–consumer asynchronous queue that overlaps action prediction and execution (Cadene et al., 2026). RTC conditions generation on committed actions through inference-time inpainting, while training-time RTC learns prefix-conditioned continuation during training to avoid the additional guidance pass (Black et al., 2025a;b). A2C2 adds a lightweight per-step correction head driven by the latest observation, and VLASH aligns prediction with execution by rolling the robot state forward over the expected delay (Sendai et al., 2025; Tang et al., 2025). FASTER reduces time to first action through horizon-aware flow-sampling schedules (Lu et al., 2026). Together, these works address latency and action-chunk continuity at the levels of execution, training, online correction, and sampling.

3 The LIBERO-Dynamic Benchmark

LIBERO-Dynamic studies manipulation in which task-relevant objects move continuously while policy inference advances the physical world according to calibrated model latency. The action returned by a policy may therefore respond to an earlier target state. We instantiate this setting on all 40 LIBERO tasks, with 1,800 fixed episodes per model–execution configuration (Figure 2).

3.1 Tasks, Motion Families, and Evaluation Scale

Each of the four ten-task LIBERO suites contributes three motion families and five valid trajectory variants per family, yielding $40 \times 3 \times 5 = 600$ dynamic configurations. The families capture constant-speed linear reciprocation, a smooth closed Catmull–Rom path (Catmull & Rom, 1974), and

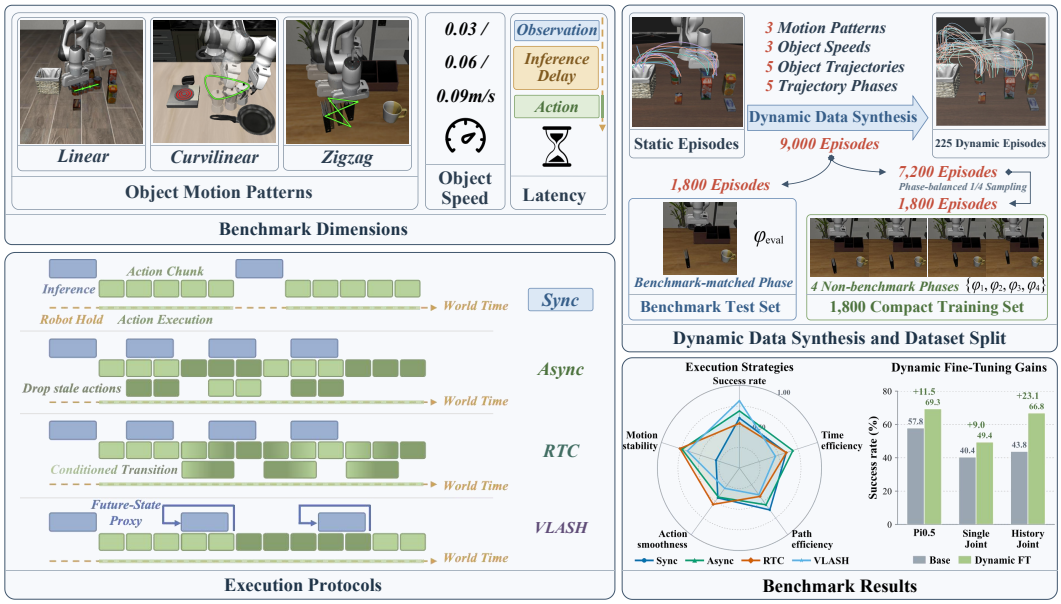


Figure 2: Benchmark dimensions, dynamic-data construction, execution protocols, and representative results. Blue denotes inference; alternating light and dark green denote actions from consecutive predictions.

a piecewise-linear multi-waypoint zigzag. They differ in direction continuity and turning structure; their labels do not prescribe a difficulty order. Additional target and trajectory statistics are in Section A.1.

Every configuration is evaluated at 0.03, 0.06, and 0.09 m/s, producing $600 \times 3 = 1{,}800$ episodes per deployment configuration. All methods share the same initial states, paths, motion phases, and success conditions.

3.2 Contact-Driven Motion and Continuous Physics

Instead of overwriting object poses, LIBERO-Dynamic places an invisible rigid platform beneath each target and transfers its commanded motion through MuJoCo contact friction (Todorov et al., 2012). The target remains force-driven, can separate from the platform when grasped, and interacts with the robot and scene through the standard solver.

Control runs at $f_c = 20$ Hz. Measured inference latency τ is quantized to $D = \lceil \tau f_c \rceil$ control steps, and simulation advances by D/f_c before the returned action becomes available. During this interval, physics and object motion continue while the robot either holds or executes queued actions according to the selected protocol. All strategies use the same latency quantization and physics advancement rule; contact and calibration details are reported in Sections A.1 and A.7.

3.3 Static-to-Dynamic Demonstration Conversion

Reference construction. For each task, we replay candidates from the official static teleoperation data with object motion disabled. The primary template is the replay-valid candidate that completes the task earliest within a deterministic scan. Replaying this template records end-effector, gripper, target, and destination trajectories; gripper state and target proximity identify grasp, release, and fixture-contact phases.

Dynamic replay and validation. At each step, the generator maps the reference target–end-effector relation into the target's current frame and adds a smoothed, bounded translational correction to the reference action. Compensation is activated around the relevant grasp, placement, or fixture-contact phase and falls back to spatial proximity when a temporal phase is unavailable. All robot commands

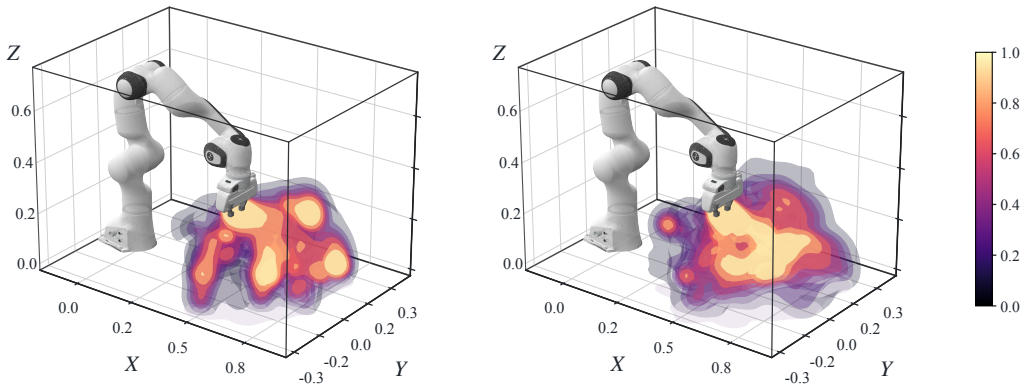


Figure 3: Normalized end-effector position density in the robot base frame for original LIBERO demonstrations (left) and generated LIBERO-Dynamic demonstrations (right).

pass through `env.step(action)`, and target motion remains contact-driven. An episode is retained only after task success, measured dynamic motion, and physical diagnostics all pass. A failed primary rollout enters a finite, deterministic fallback sequence that may change the verified reference, controller route, or grasp/place margins; every attempt is recorded, and failure of the full sequence excludes the condition. Exact equations, thresholds, and retry logic are in Section A.3.

For every one of the 1,800 task–trajectory–speed conditions, generation covers five initial motion phases, producing 9,000 successful dynamic demonstrations. Formal evaluation fixes one phase by advancing the scene for 40 control steps before the first observation. The remaining four phases form 7,200 training candidates, from which the compact split selects 1,800 trajectories. Training and evaluation have no episode-ID overlap but share tasks, trajectory families, paths, and speeds; the study therefore measures phase-separated, condition-matched adaptation rather than generalization to unseen tasks or trajectories. Figure 3 visualizes the resulting end-effector coverage.

3.4 Latency-Aware Execution Protocols

A policy predicts H actions, while the effective horizon $K \leq H$ limits how many may be adopted before replanning. We evaluate four protocols. **Synchronous hold** waits after the current chunk is exhausted. **Asynchronous queue** overlaps inference and action execution and blends overlapping predictions. **RTC** conditions continuation on committed actions and expected delay. **VLASH** rolls the robot state forward over the delay and hands off a returned chunk without weighted blending. Trigger and queue-update details are in Section A.7.

Latency is the end-to-end wall-clock time from observation submission to chunk receipt. Each deployment path is measured after warm-up on the target GPU and its median is frozen. Protocols share all benchmark conditions and physics parameters, while protocol-specific inputs and latency remain part of each complete configuration; comparisons therefore do not isolate a single mechanism causally.

3.5 Evaluation Metrics

An episode succeeds when the original LIBERO BDDL goal is satisfied within 600 control steps. The primary metric is task success rate (SR), reported overall and at each speed. We use four descriptive quantities. Tables report frozen latency τ. For successful condition (s, i), let c_T be the shortest successful step count in the declared comparison set divided by the current count; time efficiency is $E_T = \sqrt{p\,\bar{c}_T}$, where p is SR. Replacing steps with end-effector path length gives path efficiency $E_P = \sqrt{p\,\bar{c}_P}$. Query efficiency is $E_I = 1000 N_{\text{succ}}/N_{\text{query}}$.

E_T and E_P combine success coverage with efficiency conditional on success and are comparable only within the same declared reference set; synchronous waiting time is excluded from the control-step count. Radar mappings and display ranges are in Section A.2. We also report **steady underflow**:

action-queue-empty steps while inference is pending, after at least one predicted action has been executed, per 1,000 executed steps. It is used only to diagnose asynchronous queue continuity.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate six released LIBERO policies: flow-matching $\pi_{0.5}$ and π_0 (Physical Intelligence et al., 2025; Black et al., 2024), latent-world-model VLA-JEPA (Sun et al., 2026), OpenVLA-OFT (Kim et al., 2025), and video-model-based FastWAM and FastWAM-Joint (Yuan et al., 2026). Cross-model evaluation uses native checkpoints and interfaces without dynamic training; the training study additionally includes a five-frame history variant of FastWAM-Joint. Every configuration uses the same 1,800 benchmark episodes and success criteria. Deployment-path latency is calibrated and frozen on an RTX 5090.

4.2 CROSS-MODEL EVALUATION UNDER DYNAMICS

All six policies use asynchronous execution. For each model, K^* is the highest-SR point among measured $K{=}1$, even horizons within its native action length, and the native odd endpoint when applicable. It is an oracle best-measured point on the fixed evaluation population, rather than a validation-selected setting. Table 1 reports results at K^*; complete sweeps and motion-family results are in Figure 9 and Table 3.

Table 1: Six pretrained policies under asynchronous execution at their best measured K^* on the fixed 1,800 episodes. Ret. is dynamic All SR divided by the speed $= 0$ result.

	τ		Static baseline		Dynamic SR (%)						
Model	(ms)	D	speed $= 0$	Ret. (%)	0.03	0.06	0.09	All	E_T	E_P	E_I
$\pi_{0.5}$ (JAX)	60	2	96.9	**59.6**	64.8	57.5	51.2	**57.8**	**.715**	**.727**	3.14
FastWAM-Joint	64	2	98.5	41.2	49.3	39.5	33.0	40.6	.593	.606	**3.70**
VLA-JEPA	56	2	97.2	36.7	41.0	37.5	28.7	35.7	.566	.577	1.59
FastWAM	63	2	97.6	32.9	34.2	32.5	29.5	32.1	.515	.537	2.68
π_0 (JAX)	57	2	94.2	33.4	36.2	32.7	25.7	31.5	.507	.530	2.65
OpenVLA-OFT	93	2	97.1	28.1	34.3	26.8	20.7	27.3	.479	.504	1.12

Overall dynamic SR ranges from 27.3% to 57.8%. Normalization by speed $= 0$ largely preserves the ordering but swaps FastWAM and π_0, so static ability alone does not determine the dynamic ranking. Raising object speed from 0.03 to 0.09 m/s lowers every model by 4.7–16.3 points, with model-dependent declines.

Finding 1: Faster target motion reduces success for every evaluated model, but speed sensitivity differs across models.

All six models decline as motion accelerates, but by different amounts and with changes in relative ordering.

4.3 EXECUTION STRATEGY AND ACTION HORIZON

We fix the $\pi_{0.5}$ (PyTorch) checkpoint and compare Sync, Async, and RTC (fixed $= 10$). Sync and Async use the 147 ms standard path; RTC is separately calibrated at 170 ms. VLASH uses its own future-state-aware checkpoint and native interface (Tang et al., 2025), so it is reported as a system reference rather than a same-checkpoint ablation. Figure 4 shows the four sweeps; extended RTC settings and exact best-point metrics are in Section A.8 and Table 4.

With synchronous hold, every replanning event introduces an approximately 147 ms pause. The zero-latency gap therefore grows from 2.4 points at $K{=}16$ to 25.3 points at $K{=}2$: more frequent replanning also creates more frequent stalls. At their best measured horizons, Async, Sync, and RTC obtain 48.4%, 43.1%, and 37.8% SR with the same checkpoint. Async also uses the fewest

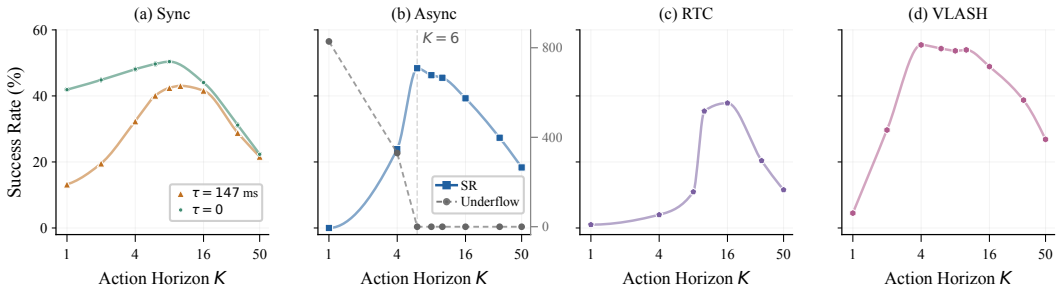


Figure 4: $\pi_{0.5}$ (PyTorch) SR versus action horizon under **(a)** Sync, **(b)** Async, **(c)** RTC, and **(d)** VLASH. Panel (a) includes a zero-latency diagnostic; panel (b) overlays steady underflow (right axis).

completion steps on common successes, whereas RTC produces the lowest action variation and end-effector acceleration (Table 5). RTC's extra conditioning increases latency to 170 ms and changes queue updates, so its lower SR cannot be attributed to latency alone. VLASH reaches 55.4% as a separately trained complete system.

The Async sweep separates two regimes. At $K{=}1$ and 4, steady underflow is 828.3 and 329.5 per 1,000 steps and SR is 0.0% and 23.9%; at $K{=}6$, underflow reaches zero and SR peaks at 48.4%. For measured $K{\geq}6$, underflow remains zero while SR declines to 39.3% at $K{=}16$. Queue continuity therefore sets a configuration-dependent lower horizon; beyond it, shorter open-loop execution is associated with higher success.

Finding 2: Execution timing changes dynamic-task performance even when model weights are fixed.

Waiting, queuing, and plan handoff alter the world state to which the same checkpoint ultimately responds.

Finding 3: Asynchronous execution has a minimum viable horizon set by queue continuity; beyond it, shorter horizons correspond to higher success.

In the measured underflow-free regime, increasing K reduces feedback frequency and coincides with lower SR.

4.4 Training with Dynamic Data

We compare $\pi_{0.5}$ (JAX), single-frame FastWAM-Joint, and FastWAM-Joint with five video frames and sixteen prior actions. Each uses static training, dynamic-only fine-tuning, and mixed dynamic–static fine-tuning. The Joint recipes run for 3,000 steps; $\pi_{0.5}$ dynamic-only and mixed recipes use 1,500 and 3,000 steps, so their difference does not isolate data composition. Figure 5 reports measured even-K sweeps under frozen latency; exact configurations and best-point metrics are in Tables 6 and 7.

At their best measured horizons, $\pi_{0.5}$ reaches 57.8% (static), 59.6% (dynamic only), and 69.3% (mixed); single-frame Joint reaches 40.6%, 49.4%, and 43.8%; and history-frame Joint reaches 45.4%, 67.1%, and 66.8%. Under the matched 3,000-step dynamic-only recipe, adding history changes the gain from 8.8 to 21.7 points and shifts the best measured horizon from $K{=}4$ to $K{=}8$. This associates temporal conditioning with greater use of the dynamic data in these variants, without establishing superiority for every horizon or architecture.

Finding 4: Dynamic fine-tuning gains vary with model design and training recipe, motivating architectures tailored to dynamic manipulation.

Each model improves under at least one dynamic-data recipe, but the magnitude depends on model design and training configuration.

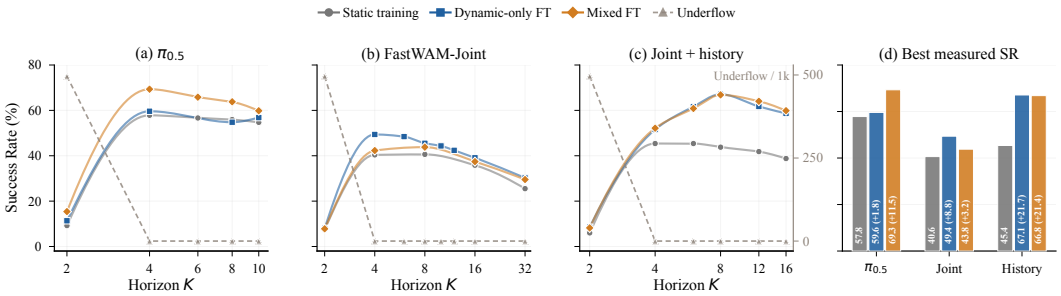
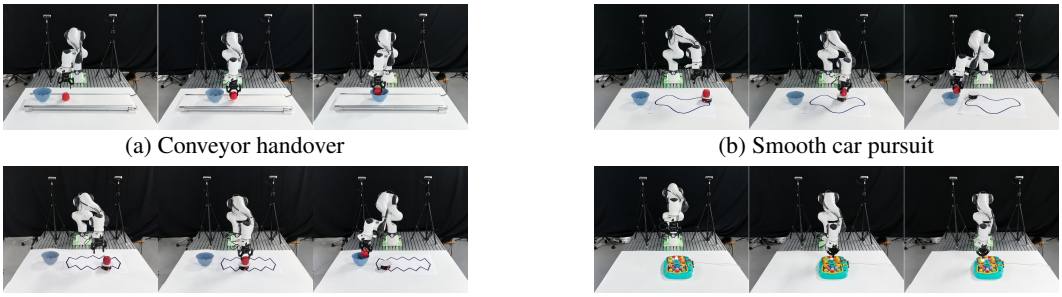


Figure 5: Dynamic-data fine-tuning under asynchronous execution. **(a–c)** Measured K sweeps for three models and steady underflow (right axis). **(d)** Best-measured SR; parentheses give percentage-point changes from the corresponding static checkpoint.

4.5 Real-Robot Validation

We deploy $\pi_{0.5}$, FastWAM-Joint, and history-frame FastWAM-Joint on a Franka Research 3 at 15 Hz using two external cameras and one wrist camera. Policies are trained on physical dynamic demonstrations; within each model, Async and RTC share a checkpoint. Hardware, sensing, and training details are in Section A.14.

Four conditions cover conveyor handover, whack-a-mole, and car pursuit on smooth-polynomial and zigzag paths. Each manipulation condition uses 50 trials; whack-a-mole uses 50 sessions of 30 s. All six model–execution configurations are evaluated at $K \in \{4, 8\}$. Figure 6 shows the task sequences; exact values and complete sequences are in Table 9 and Section A.14.



(a) Conveyor handover

(b) Smooth car pursuit

(c) Zigzag car pursuit

(d) Whack-a-mole

Figure 6: Real-robot task sequences. Each panel shows three keyframes from one successful rollout; complete sequences are in the appendix.

The four tasks probe different temporal demands: conveyor handover and smooth pursuit require continuous interception, zigzag pursuit adds abrupt direction changes, and whack-a-mole requires repeated time-critical responses. We keep the model–execution pairs fixed and report both horizons for every task. At $K{=}4$, Async exceeds RTC on all three manipulation tasks for each model. Moving to $K{=}8$ lowers the aggregate for both $\pi_{0.5}$ settings and single-frame Joint Async, leaves single-frame Joint RTC unchanged, and improves both history-frame settings; whack-a-mole decreases for all six settings. Figure 7 summarizes these results, showing model- and task-dependent horizon effects and providing a complementary physical check that execution timing can affect dynamic behavior.

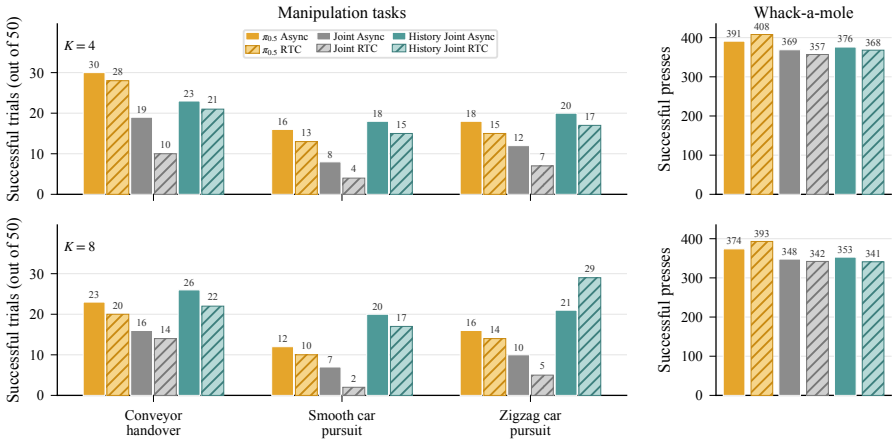


Figure 7: Real-robot results at $K \in \{4, 8\}$. The manipulation tasks report successful trials out of 50; whack-a-mole reports successful presses over 50×30 s sessions.

5 Limitations

The reported K^* values are best measured points on the fixed evaluation population rather than validation-tuned settings. The main text compares these best measured points, while the appendix additionally reports a fixed-$K{=}8$ comparison across training recipes. The benchmark freezes median latency and prescribed trajectories, omitting latency jitter and less structured motion. Only strategies sharing a checkpoint and deployment path form controlled comparisons; VLASH is an external reference. The compact real-robot study uses physical dynamic training data and does not establish broad sim-to-real generalization.

6 Conclusion

We introduced LIBERO-Dynamic, extending all 40 LIBERO tasks with contact-driven motion, calibrated inference latency, and 9,000 dynamic demonstrations converted from static references. Across six pretrained policies, faster motion reduces success with model-dependent sensitivity. Same-checkpoint sweeps show that short asynchronous horizons can exhaust the action queue; beyond this boundary, less frequent feedback is associated with lower success. Dynamic-data gains vary with model design and training recipe, motivating policies tailored to dynamic manipulation. A compact physical study gives complementary evidence that execution timing remains relevant outside simulation. LIBERO-Dynamic provides a reproducible basis for moving-target manipulation while leaving latency jitter, less structured motion, and broader physical evaluation to future work.

Reproducibility Statement

The paper and supplement provide protocols, configurations, code, exact results, audits, and videos needed to reproduce the reported results with public checkpoints.

Ethics Statement

This work uses no human-subject or personal data. Robot experiments were supervised with safety controls; released artifacts contain no personal identifiers.

AI Use Statement

Generative AI assisted language editing, translation, and literature discovery and summarization. The authors verified all AI-assisted text and citations and take responsibility for the final paper. No experimental data were AI-generated.

REFERENCES

Pranav Atreya, Karl Pertsch, Tony Lee, Moo Jin Kim, Arhan Jain, Artur Kuramshin, Clemens Eppner, et al. RoboArena: Distributed real-world evaluation of generalist robot policies. *arXiv preprint arXiv:2506.18123*, 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, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025a. URL https://arxiv.org/abs/2506.07339.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025b. URL https://arxiv.org/abs/2512.05964.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, Mustafa Shukor, Jess Moss, Alexander Soare, Dana Aubakirova, Quentin Lhoest, Quentin Gallouédec, and Thomas Wolf. LeRobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026. URL https://arxiv.org/abs/2602.22818.

Edwin Catmull and Raphael Rom. A class of local interpolating splines. In Robert E. Barnhill and Richard F. Riesenfeld (eds.), *Computer Aided Geometric Design*, pp. 317–326. Academic Press, 1974. doi: 10.1016/B978-0-12-079050-0.50020-5.

Tianxing Chen et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026a.

Yuxuan Chen, Wanruo Zhang, and Xiao Li. Reflex: Enabling fast and predictive vision-language-action models for reaction-critical manipulation. *arXiv preprint arXiv:2608.14379*, 2026b.

Rongxu Cui, Zongzheng Zhang, Jingrui Pang, Haohan Chi, Jinbang Guo, Saining Zhang, Shaoxuan Xie, Xin Jin, Yao Mu, Jiaolong Yang, Guocai Yao, Xianyuan Zhan, Ya-Qin Zhang, and Hao Zhao. LIBERO-Safety: A comprehensive benchmark for physical and semantic safety in vision-language-action models. In *European Conference on Computer Vision*, 2026. arXiv:2606.23686.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. *arXiv preprint arXiv:2603.15620*, 2026.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, et al. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Mayol-Cuevas. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2): 3019–3026, 2020.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2502.19645.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. FASTER: Rethinking real-time flow VLAs. *arXiv preprint arXiv:2603.19199*, 2026. URL https://arxiv.org/abs/2603.19199.

Yao Mu et al. RoboTwin: Dual-arm robot benchmark with generative digital twins. *arXiv preprint arXiv:2409.02920*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Physical Intelligence et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

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

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.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. VLASH: Real-time VLAs via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025. URL https://arxiv.org/abs/2512.01031.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Guodong Wang, Chenkai Zhang, Qingjie Liu, Jinjin Zhang, Jiancheng Cai, Junjie Liu, and Xinmin Liu. LIBERO-X: Robustness litmus for vision-language-action models. *arXiv preprint arXiv:2602.06556*, 2026.

Haozhe Xie et al. DynamicVLA: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026.

Adina Yakefu et al. RoboChallenge: Large-scale real-robot evaluation of embodied policies. *arXiv preprint arXiv:2510.17950*, 2025.

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.

Borong Zhang, Jiahao Li, Jiachen Shen, Yuhao Zhang, Yishuai Cai, Lu Liu, Hailu Ji, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. VLA-Arena: An open-source framework for benchmarking vision-language-action models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

A Appendix

A.1 Additional Benchmark Construction Details

Among the 40 tasks, 25 contain two or three independently moving movable objects, five center on moving fixtures such as cabinet doors or stoves, and ten use one movable object; together they cover 23 target-object types. The linear family uses constant-speed piecewise-linear reciprocation through three waypoints. The Catmull–Rom family interpolates 3–8 waypoints into a C^1-continuous closed curve traversed at constant arc-length speed. The zigzag family follows 5–8 waypoints with unsmoothed intermediate turns.

All retained platform–target configurations remain in the static-friction regime at the evaluated speeds, including zigzag turns, and selective contact masks prevent the platform from interacting with non-target objects. Dynamic-demo generation begins object motion at control step 30 and caps episodes at 600 steps. The evaluation phase advances the scene by 40 steps before the first observation; the compact training split draws 450 episodes from each of the other four phase ranks.

A.2 Metric Display Details

Each table defines its own reference set for E_T and E_P, whereas the two radar panels use a joint nine-model reference set. Radar display ranges are fixed to SR 0–80%, latency 40–120 ms (reversed), E_T/E_P 0.40–0.85, and E_I 0–12 successes per 1,000 requests. Tables retain values before display mapping. Because the efficiency quantities also depend on successful outcomes, radar area is not aggregated into a scalar score.

A.3 Dynamic-Demonstration Generation Details

Static-reference selection and phase annotation. For each task, the generator scans the official LIBERO HDF5 demonstrations in a fixed order and replays each candidate with dynamic motion disabled. A candidate is valid only if the original BDDL goal succeeds. The primary reference minimizes successful completion step, then demonstration length and scan order, among the successful candidates found by the deterministic scan; the selected key is cached per task. A task-specific fallback may instead request another explicitly identified demonstration, which is replay-validated before use. Thus, reference changes occur only through recorded fallback specifications rather than random resampling.

The selected reference is executed once more to record the end-effector position, gripper command, and poses of task-relevant objects and goal receptacles at every control step. A gripper-close event near an object marks grasp onset, and the first later reopening marks release. For articulated fixtures, the closest end-effector approach anchors the interaction phase. These annotations determine when target tracking should affect the action and prevent compensation from being applied throughout unrelated parts of the demonstration.

Before formal generation and evaluation, each trajectory configuration is preflighted with the robot stationary. Configurations are excluded if a target leaves the safe workspace or camera view, or collides with non-target scene objects.

Online target-relative compensation. Let the reference end-effector position and target pose at step t be p_t^e and (p_t^o, R_t^o), and let the measured target pose during dynamic replay be $(\tilde{p}_t^o, \tilde{R}_t^o)$. The generator preserves the reference target–end-effector relation in the current target frame:

$$\hat{p}_t^e = \tilde{p}_t^o + \tilde{R}_t^o (R_t^o)^{-1} (p_t^e - p_t^o), \tag{1}$$

and uses $\hat{p}_t^e - p_t^e$ as a candidate displacement correction. Around grasp, the weight ramps up before contact, remains active briefly after grasp, and then fades; for placement, compensation is handed from the picked object to its destination; fixture compensation remains active through the recorded interaction. If no temporal annotation applies, candidates are weighted by spatial proximity. For multiple candidates, temporally active targets take precedence. The aggregate correction is clipped, exponentially smoothed, and converted into a bounded translational update to the reference action.

The primary route uses template replay with online compensation. Recorded task-specific fallbacks may use target-relative pick-and-place servoing, retain a static prefix or suffix, or adjust

bounded grasp/place parameters. Every route advances the robot and physics only through standard `env.step(action)` calls. Object motion is induced through contact friction between the carrier and target; the generator never teleports the target or writes its pose to complete a demonstration.

Strict failure detection. The task predicate alone is insufficient for acceptance. The strict validator also requires a clean rollout status, zero assisted steps, no forbidden state edit, and positive motion evidence. For platform-carried targets, recorded platform displacement must reach 5 mm; for other dynamic fixtures, either translation must reach 5 mm or rotation must reach 0.03 rad. The physical checks in Table 2 are evaluated before robot contact so that intended grasp-induced separation does not appear as a conveyor-following failure. Packaging additionally rejects missing episode arrays or inconsistent trajectory lengths.

Table 2: Strict physical checks used for generated demonstrations. Crossing any rejection threshold prevents the episode from entering the released dataset.

Check	Rejection condition
Task and rollout	non-successful BDDL goal, runtime error, assisted step, or forbidden state edit
Motion evidence	platform/fixture motion below the thresholds above
Carrier path	maximum platform path error > 0.01 m
Target following	pre-contact object–carrier relative error > 0.08 m
Object stability	relative tilt $> 35^\circ$ or downward displacement > 0.04 m
Visibility	target outside the diagnostic bird-view projection for more than 25% of sampled pre-contact steps

Bounded retry and provenance. If the primary rollout fails strict validation, the generator traverses a finite, deterministic list of task-specific fallback specifications. A fallback may select another replay-validated static reference, switch between compensated-template and target-relative servo control, enable a grasp guard, or adjust fixed grasp/place margins. Scripted grasp routes may also try a finite center-and-neighborhood offset pattern; they do not keep sampling until success. Validation is rerun after every attempt and the cascade stops at the first strict success. If no candidate passes, the item remains failed, the process exits with an incomplete audit, and the episode is excluded from the packed HDF5 release. The manifest preserves the item identifier, source demonstration, motion condition and realized phase, controller route, per-attempt failure reasons, physical diagnostics, output paths, and random seed. The phase-partition auditor separately fails on count, uniqueness, coverage, or train/test-overlap violations.

A.4 Representative Dynamic Demonstrations

Figure 8 selects one representative task from each LIBERO subset: placing the black bowl between the plate and ramekin onto the plate (Spatial), placing tomato sauce in the basket (Object), placing the bowl on the stove (Goal), and placing both moka pots on the stove (LIBERO-10). Each task is shown under linear ping-pong, curvilinear spline, and multi-waypoint zigzag motion at 0.09 m/s. In every cell, the first three frames span the pre-contact approach and the fourth is the final successful state. The green curve is the episode-specific object path projected into AgentView using the stored MuJoCo camera, with arrows indicating motion direction. Selection is independent of the success or failure of any evaluated model. The electronic supplement provides the complete dual-view videos and eight-frame sequences for all 12 conditions.

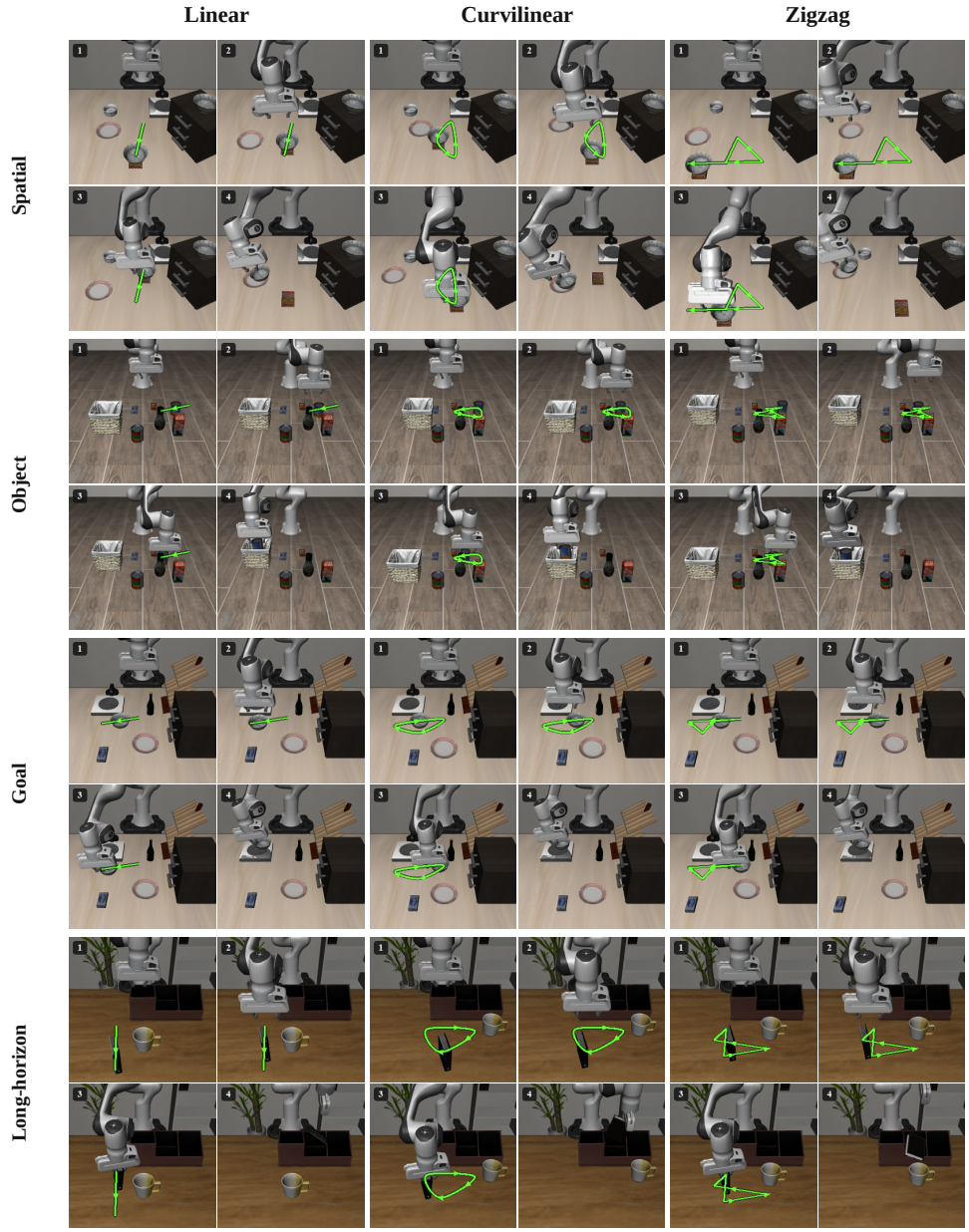


Figure 8: Dynamic demonstrations for four representative tasks under the three motion families. In each cell, frames 1–3 span the measured approach from episode start to first robot–target contact, while frame 4 is the final successful state. The target position is read from every stored MuJoCo state and projected directly into AgentView; green arrows indicate direction. The narrow foreground trajectory remains legible after the figure is reduced to page width. Rows identify task subsets and columns identify motion families.

A.5 Motion-Family Breakdown

Table 3 decomposes the cross-model result by trajectory family at the same asynchronous K^* used in Table 1. Each family contains 600 episodes (40 tasks $\times$ 5 trajectories $\times$ 3 speeds). We do not retune K separately by family, so the rows preserve the deployment configuration used for the overall comparison. The breakdown is descriptive: the three families vary path geometry and do not define an ordered difficulty scale.

Table 3: Asynchronous success rate by motion family at each model's overall best measured eligible horizon K^*. Each family contains 600 episodes; All contains 1,800.

Model	K^*	linear ping-pong	Catmull-Rom spline	multi-waypoint zigzag	All
$\pi_{0.5}$ (JAX)	4	62.7	51.2	59.7	57.8
FastWAM-Joint	8	45.2	36.3	40.3	40.6
VLA-JEPA	4	39.3	33.3	34.5	35.7
FastWAM	8	34.8	29.8	31.5	32.1
π_0 (JAX)	8	37.5	27.5	29.5	31.5
OpenVLA-OFT	4	30.8	25.0	26.0	27.3

A.6 Complete Cross-Model Action-Horizon Sweeps

Figure 9 reports the complete synchronous and asynchronous sweeps underlying the best-measured operating points in Table 1, together with the per-speed asynchronous results at those operating points.

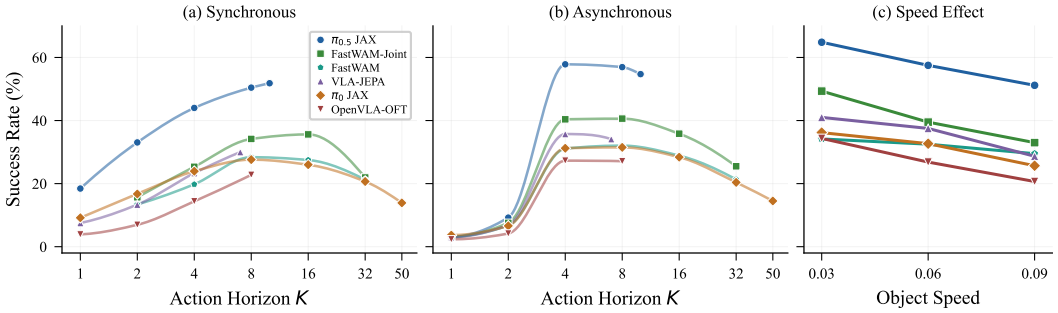


Figure 9: Cross-model action-horizon analysis on LIBERO-Dynamic: **(a)** synchronous and **(b)** asynchronous K sweeps; **(c)** SR by object speed at each model's best measured asynchronous K.

A.7 Execution-Protocol Implementation Details

Each policy produces a native action chunk of length H, and evaluation retains the prefix specified by the effective action horizon K. Measured latency τ is quantized as $D = \lceil \tau f_c \rceil$ control steps at $f_c = 20$ Hz; thus 60 ms produces $D = 2$ and a realized 100 ms simulation delay. Synchronous hold submits a new observation after these K actions are consumed and locks the robot joints for the subsequent D delay steps; each cycle therefore contains K action steps and D waiting steps.

The asynchronous implementation maintains a first-in-first-out queue. When the remaining action count falls below $\lceil 0.5K \rceil$, a background thread submits the next inference request while the robot continues consuming queued actions. When the new chunk arrives, overlapping positions are blended with weights 0.3 for the old action and 0.7 for the new action, and the non-overlapping suffix is appended. The implementation uses the producer–consumer structure of the LeRobot asynchronous queue (Cadene et al., 2026); the trigger ratio and blending weights are evaluation settings used in this work.

RTC uses the same prefetch trigger but additionally conditions the policy on unexecuted actions and the expected delay d. The first d returned actions, corresponding to elapsed time, are removed, and

the remainder replaces the current queue without weighted blending (Black et al., 2025a). VLASH submits a request when the queue length is at most $\min(K, d)$, using the current image and a causal expected-execution-time robot state predicted from committed actions; the predictor has no access to future scene state. The returned chunk replaces the queue intact after committed actions are exhausted, with neither prefix dropping nor weighted blending (Tang et al., 2025).

Inference latency includes observation preprocessing, data transfer when applicable, model forward computation, and action postprocessing, ending when the complete action chunk reaches the controller. Each distinct model deployment path is calibrated using single-request inference on representative LIBERO observations; RTC calibration includes processing of its conditioning inputs. Warm-up requests are discarded. When multiple independent processes are used, their retained samples are pooled, and the median is frozen and converted to $d = \lceil \tau f_c \rceil$ control steps.

The $\pi_{0.5}$ JAX, FastWAM, $\pi_{0.5}$ PyTorch Sync/Async, and RTC calibrations each use one run with 8 warm-up and 88 retained requests. FastWAM-Joint and history-frame Joint each use one run with 20 warm-up and 100 retained requests. π_0, OpenVLA-OFT, and VLASH each use two independent processes with 8 warm-up and 88 retained requests per process; VLA-JEPA uses two processes with 30 warm-up and 300 retained requests per process. Fine-tuned checkpoints that preserve the deployment path reuse the corresponding base calibration. The electronic supplement's `tables/latency_calibrations.csv` records every calibration identifier, exact frozen value, sample count, and evidence granularity.

Underflow accounting separates startup and steady state. An empty queue before the first queued prediction has been executed is startup underflow; an empty queue afterward is steady underflow. Figures report $1000 N_{\text{steady}}/N_{\text{exec}}$, where N_{exec} is the total number of executed control steps in the episode.

A.8 RTC Settings and Complete Strategy Sweeps

Both RTC settings use a calibrated latency of 170 ms. The fixed=10 setting peaks at 37.83% and θ=0.5 at 32.94%, both at K=16. We retain both settings here to make the main-text choice explicit; fixed=10 does not outperform θ=0.5 at every measured K.

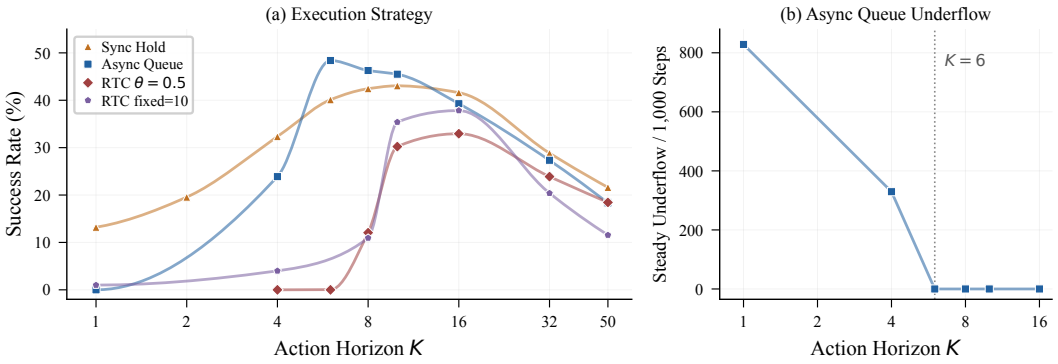


Figure 10: Extended $\pi_{0.5}$ (PyTorch) strategy sweeps, including two RTC settings and asynchronous underflow. Markers are measured results; smooth SR curves are visual guides.

A.9 Best-Measured Execution-Strategy Results

Table 4 reports the per-speed success rates and efficiency metrics for the execution settings summarized by the main-text strategy sweep and radar plot. Each row uses that setting's best measured eligible action horizon.

A.10 Paired Execution-Strategy Behavior

Table 5 supplements the main five-axis metrics for the three calibrated $\pi_{0.5}$ configurations. Means are computed on 343 common successful (s, i) pairs; all 1,029 matched traces are valid, with no

Table 4: Execution-strategy results on LIBERO-Dynamic at the best measured eligible horizon for each setting (1,800 episodes per row). Sync, Async, and RTC use $\pi_{0.5}$ (PyTorch); VLASH uses its official checkpoint. Zero-latency Sync is an idealized timing diagnostic.

	τ		SR (%)						
Strategy	(ms)	D	0.03	0.06	0.09	All	E_T	E_P	E_I
VLASH	104	3	**65.7**	**54.7**	**46.0**	**55.4**	**.671**	**.689**	5.61
Sync Hold (zero-lat.)	0	0	57.7	49.8	43.7	50.4	.658	.664	**10.11**
Async Queue	147	3	55.7	45.2	44.3	48.4	.634	.648	3.56
Sync Hold	147	3	48.7	42.5	38.0	43.1	.593	.618	9.85
RTC (fixed=10)	170	4	44.7	38.3	30.5	37.8	.556	.565	5.02

exclusions. Extra successes contribute to the main SR and efficiency metrics but do not enter this matched subset. Control steps exclude synchronous inference waits, and paths connect recorded EEF positions only. Δa averages the recorded per-step action-change norm within each episode. A_{EEF} estimates acceleration using actual simulation timestamps, followed by equal episode weighting; uniformly spaced samples are not assumed. Unrecorded motion between synchronous samples remains a limitation. Smaller action changes or acceleration do not necessarily imply better dynamic tracking; these are behavioral descriptions, not a scalar ranking.

Table 5: Behavioral metrics on 343 tasks successful under all compared strategies. Δa measures action change, and A_{EEF} is a control-sampled acceleration estimate.

Strategy	K	Steps	Path (m)	Δa	A_{EEF} (m/s^2)
Async Queue	6	167.1	1.159	0.142	0.379
Sync Hold	10	183.5	1.170	0.133	0.605
RTC (fixed=10)	16	174.3	1.236	0.114	0.364

A.11 Dynamic Fine-Tuning Configuration

All fine-tuning runs start from the corresponding static checkpoint: the official $\pi_{0.5}$ LIBERO checkpoint, the 60K-step single-frame Joint checkpoint, or the 60K-step history-frame Joint checkpoint. The dynamic set contains 1,800 demonstrations (284,502 frames at 20 Hz) drawn from training phases whose episode IDs do not appear in evaluation; tasks, paths, and speeds remain condition-matched. Mixed fine-tuning uses the union of these demonstrations and 1,693 repaired static LIBERO demonstrations (273,465 frames); dynamic episodes therefore constitute 51.5% of the union by episode count. For the two Joint models, the dynamic gripper signal is mapped to the same $[0, 1]$ convention as the static data. For $\pi_{0.5}$, the static data are converted to the model's native gripper convention and LeRobot v2.1 format. Normalization statistics are recomputed for each fine-tuning dataset.

Table 6: Dynamic fine-tuning configurations. Batch is the effective global batch. The mixed set is the union of the dynamic and repaired static demonstrations described above.

Model	FT data	Steps	Batch	Peak LR	Warm-up
$\pi_{0.5}$	Dynamic only	1,500	256	5×10^{-5}	75
	Dynamic + static	3,000	256	5×10^{-5}	150
Joint	Dynamic only	3,000	128	1×10^{-4}	150
	Dynamic + static	3,000	128	1×10^{-4}	150
History Joint	Dynamic only	3,000	128	1×10^{-4}	150
	Dynamic + static	3,000	128	1×10^{-4}	150

The two Joint variants use AdamW with $\beta = (0.9, 0.95)$, weight decay 10^{-2}, gradient-norm clipping at 1.0, bf16 precision, and a cosine learning-rate schedule. Single-frame Joint receives one current

observation and predicts 32 actions. History-frame Joint receives five consecutive observations and the preceding 16 executed actions; it predicts a 32-step sequence whose first 16 actions form the clean conditioning prefix, and the loss is applied to the 16 future actions. Dynamic-only Joint uses ZeRO-1 on eight GPUs; history-frame dynamic-only and mixed fine-tuning use DDP on eight and four GPUs, respectively. $\pi_{0.5}$ uses FSDP on eight GPUs and follows the OpenPI fine-tuning implementation with gradient clipping at 1.0, no EMA, and cosine decay to zero. Because the $\pi_{0.5}$ dynamic-only and mixed recipes differ in both data composition and training steps, their comparison is reported descriptively rather than as an isolated data-mixture ablation.

A.12 Best-Measured Training Results

The following table retains the complete per-speed success rates and efficiency metrics underlying Figure 5(d). Each row uses the best result among its measured action horizons; the fixed-K=8 controlled comparison follows in the next section.

Table 7: Results for three training recipes under asynchronous execution. Each row uses its best measured horizon K^* and 1,800 episodes.

		τ			SR (%)						
Model	Training data	(ms)	D	K^*	0.03	0.06	0.09	All	E_T	E_P	E_I
$\pi_{0.5}$ (JAX)	Static	60	2	4	64.8	57.5	51.2	57.8	.715	.727	3.14
	Dynamic only			4	70.5	63.3	45.0	59.6	.726	.719	3.38
	Dynamic + static			4	81.5	70.8	55.7	69.3	.788	.779	4.50
FastWAM-Joint	Static	64	2	8	49.3	39.5	33.0	40.6	.593	.606	3.70
	Dynamic only			4	64.7	48.5	35.0	49.4	.651	.671	2.51
	Dynamic + static			8	52.8	43.5	35.1	43.8	.603	.624	.451
FastWAM-Joint + history	Static	89	2	4	56.8	43.0	36.3	45.4	.610	.622	2.15
	Dynamic only			8	76.2	70.3	54.8	67.1	.784	.773	8.40
	Dynamic + static			8	77.8	70.8	51.8	66.8	.786	.777	8.45

A.13 Dynamic Fine-Tuning at a Fixed Action Horizon

The fixed-K=8 comparison separates training changes from action-horizon tuning. Table 8 additionally compares zero-latency Sync and calibrated-latency Async for one selected fine-tuning recipe per architecture; the main text reports all available recipes under Async. Zero-latency synchronous results are diagnostic controls under ideal feedback and are excluded from the main results. Their difference from calibrated asynchronous execution changes both timing and execution strategy, so it cannot isolate the effect of latency alone.

Table 8: Fixed-K comparison before and after dynamic-data fine-tuning on LIBERO-Dynamic (K=8; 1,800 episodes per configuration). Sync is zero-latency, and Async uses calibrated latency.

Model	Training	Sync (%)	Async (%)
Single-frame Joint	Static base	43.0	40.6
	Dynamic-only FT	**52.0**	**45.5**
History Joint	Static base	43.5	43.8
	Dynamic-only FT	**62.4**	**67.1**
$\pi_{0.5}$	Official (static)	59.3	55.9
	Mixed FT	**65.4**	**63.8**

A.14 Real-Robot Task Details and Full Rollouts

The real-robot study uses four dynamic evaluation conditions under the platform and observation setup described in Section 4.5: conveyor handover, whack-a-mole, and car pursuit on smooth-polynomial and zigzag paths. The two car trajectories are evaluated and presented separately because one changes direction continuously whereas the other contains abrupt turns. The main-text zigzag panel uses frames 2, 4, and 6; the figures below retain all seven selected frames for every condition.

Hardware and sensing. The high-level policy runs at 15 Hz on an RTX 5090 workstation, while an independent NUC executes the Franka controller at 1 kHz. The two external cameras and the wrist camera stream at 30 Hz into shared-memory buffers. At every 15 Hz policy tick, frames are selected by nearest timestamp without image interpolation. The single-frame Joint input resizes each view to 224×128 and stacks the three views vertically into a 384×224 image; the history model uses the same spatial layout at the most recent five policy ticks. The robot state and action are both 8D: seven joint coordinates or native joint commands plus one absolute gripper command. Joint states and actions are z-score normalized, and the history model additionally conditions on the most recent 16 commands actually sent to the controller.

Training and deployment. The real-robot policies are trained on demonstrations from seven tasks collected at 15 Hz. The $\pi_{0.5}$ and single-frame Joint checkpoints are trained for 20K steps. The history-frame checkpoint is initialized from the 20K-step Joint weights and trained for 30K additional steps with a five-frame visual history and a 16-action clean prefix. Within each architecture, Async and RTC use the same checkpoint, so their comparison changes the inference and execution protocol rather than the learned weights. Tables 9 and 10 report the exact outcomes and corresponding deployment settings.

Table 9: Real-robot results for six model–execution configurations at $K\in\{4,8\}$. Conveyor and the two car-pursuit conditions report successful trials out of 50; whack-a-mole reports total successful presses across 50×30 s sessions.

Model	Exec.	K	Conv. /50	Smooth /50	Zigzag /50	Whack presses
$\pi_{0.5}$	Async	4	30	16	18	391
		8	23	12	16	374
	RTC	4	28	13	15	408
		8	20	10	14	393
FastWAM-Joint	Async	4	19	8	12	369
		8	16	7	10	348
	RTC	4	10	4	7	357
		8	14	2	5	342
FastWAM-Joint + history	Async	4	23	18	20	376
		8	26	20	21	353
	RTC	4	21	15	17	368
		8	22	17	29	341

Table 10: Deployment configuration for the real-robot results in Table 9. All six settings run at 15 Hz and are evaluated at $K \in \{4, 8\}$. "History" denotes the five-frame input and 16-action clean prefix.

Model	Execution	Input	H	Denoising	Evaluated K
$\pi_{0.5}$	Async	Current	16	10	4, 8
	RTC	Current + RTC	16	10	4, 8
Joint	Async	Current	32	1	4, 8
	RTC	Current + RTC	32	10	4, 8
History Joint	Async	5 frames + 16 actions	16[†]	1	4, 8
	RTC	History + RTC	16[†]	10	4, 8

[†]The history model returns 32 actions in total; the first 16 are the clean conditioning prefix and only the future slice is eligible for execution.

Async inference runs in the background while the current plan is executed. RTC conditions denoising on the unexecuted action prefix and the expected inference delay, and the controller selects the newest time-aligned plan. For all methods, late or expired commands are skipped rather than replayed in a burst. If the valid queue is exhausted, the controller sends zero joint velocity and holds the last gripper command. Native joint commands are clipped by the same deployment safety envelope before conversion to 15 Hz joint-velocity commands; the gripper remains an absolute command with 0 denoting closed and 1 denoting open.

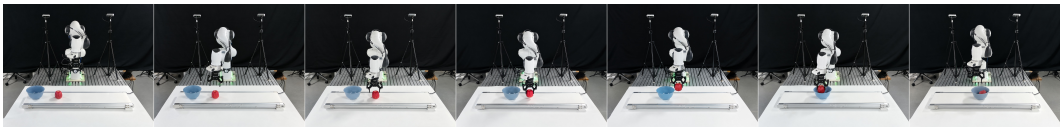


Figure 11: Conveyor handover: seven keyframes from one successful episode.

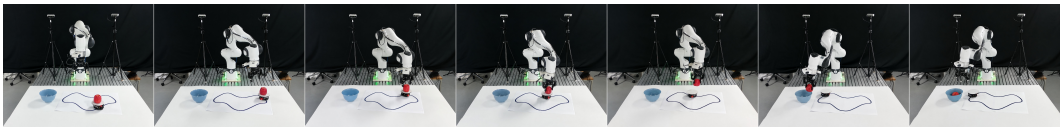


Figure 12: Car pursuit with a smooth-polynomial trajectory: seven keyframes from one successful episode.

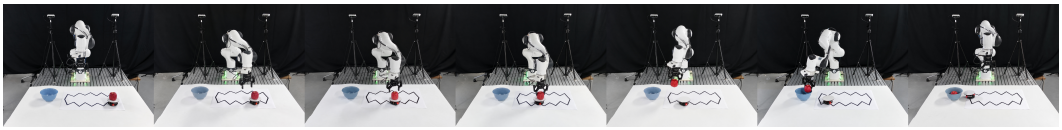


Figure 13: Car pursuit with a zigzag trajectory: seven keyframes from one successful episode.

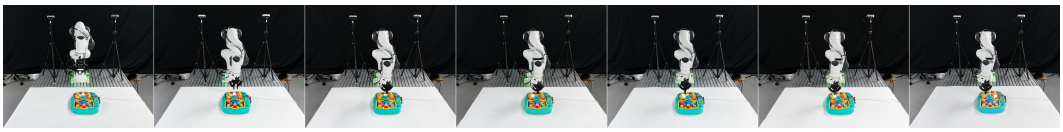


Figure 14: Whack-a-mole: seven keyframes from one successful episode.