PHYSICAL TIME AS A MISSING DIMENSION OF ROBOT POLICY MEMORY DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing robot policy memories typically evolve with observation sequences, implicitly treating each update as the same amount of physical time. Yet identical observation histories can span different durations and require different actions, exposing physical time as a missing dimension of policy memory. We introduce Temporal Flow Policy (TFP), which learns memory dynamics to retain past events and evolve the memory state over measured time intervals. Observations and prior memory determine the content and time scales of each transition; elapsed time determines how far the state advances. TFP conditions diffusion and flow-matching policies through this memory state and is trained with their native imitation objectives. We evaluate TFP with Diffusion Policy and $\pi_{0.5}$ through controlled comparisons of memory designs. These experiments show that successful timing depends on both measured intervals and retained history. TFP achieves higher average success than the recurrent and state-space memories we equip with the same intervals and than a variant that feeds the interval to a learned gate. Memory readouts recover the time since an event and motion information after occlusion. Changing the supplied intervals systematically delays or advances actions, linking time inputs to behavior. Real-robot tasks requiring timing or memory, together with LIBERO evaluations, demonstrate benefits in physical and general manipulation. These results suggest viewing robot memory not as stored observation history, but as a state that retains past events and evolves with physical time.

1 INTRODUCTION

Consider a robot preparing tea. After it lowers an infuser into a cup, the scene may remain nearly unchanged, yet the correct action changes from waiting to lifting once the tea has steeped long enough. The robot needs to remember when steeping began and track how much time has passed. A robot picking up a cup from a moving conveyor faces a related challenge. It must estimate the cup's velocity from earlier observations and retain that information while the cup is briefly hidden from view. The same observed displacement over different time intervals implies different speeds, requiring the robot to match the cup's speed at contact. Policy memory must therefore use elapsed time both to respond to remembered events and to interpret motion from past observations (Figure 1).

Memory-augmented robot policies, including vision-language-action (VLA) models, retain earlier cues and track completed subtasks, allowing actions to depend on information beyond the current observation (Koo et al., 2026; Shi et al., 2026; Li et al., 2026). Yet memories driven only by observation sequences treat each observation as one step: the same sequence can span different physical durations and require different actions. This mismatch becomes apparent when observation intervals vary with observation rate or inference latency. Timestamped histories make recording times available to the policy (Yin et al., 2026). Continuous-time recurrent models provide a mechanism for evolving state over measured intervals (Hasani et al., 2021; 2022). We therefore study physical time as a missing dimension of robot policy memory: how remembered events and measured intervals jointly shape memory evolution and guide action.

We introduce **Temporal Flow Policy (TFP)**, which retains past events in a persistent memory state that evolves with physical time. At each query, separate heads use current observation features and previous memory to predict a candidate state and channel-wise time constants. A closed-form exponential update advances memory toward the candidate over the measured interval, so elapsed time

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

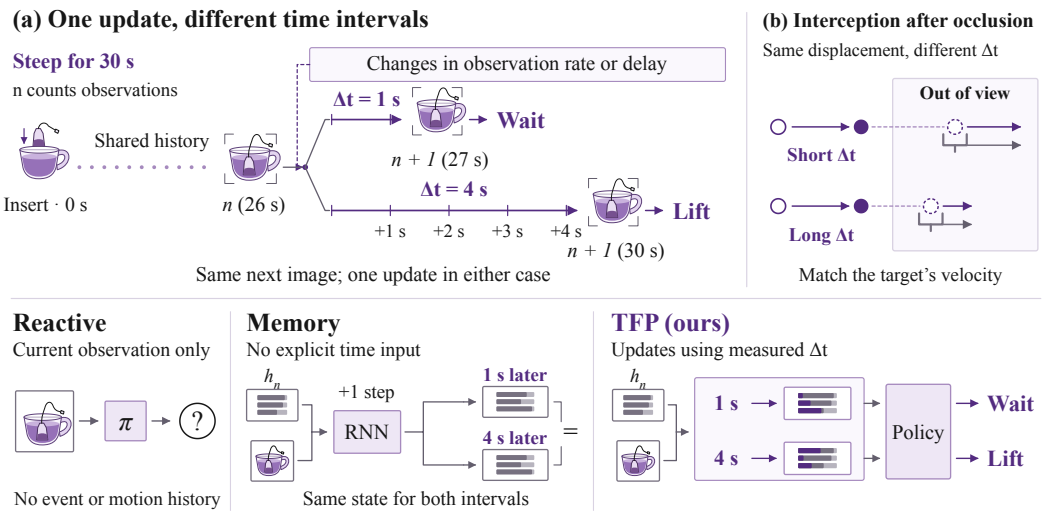


Figure 1: **Physical time gives remembered observations meaning for action.** (a) The same tea-steeping history and next image require waiting or lifting depending on elapsed time. (b) The same displacement over different intervals implies different velocities, which must be remembered for interception after occlusion. TFP uses measured intervals to evolve memory and guide action. Dashed paths indicate hidden motion; timings, trajectories, and memory states are schematic.

directly governs state evolution. Prior memory shapes both predictions after visual cues disappear, while adaptive timescales support rapid updates around events and slower evolution during waiting. The updated memory conditions diffusion and flow-matching policies (Chi et al., 2023; Physical Intelligence et al., 2025) through adaptive normalization, alongside their native observation and task context. We jointly learn memory dynamics and conditioning from causal demonstration histories using each backbone's native imitation objective, without event-time labels.

Controlled comparisons with Diffusion Policy and $\pi_{0.5}$ show higher average success than recurrent and state-space memories supplied with the same intervals and a learned-gate alternative. Memory readouts recover event age and motion information, while scaling the supplied intervals delays or advances actions without changing the physical execution schedule. Real-robot timing and sequencing tasks and LIBERO demonstrate benefits in physical and general manipulation.

Our contributions are:

- We identify the ambiguity between observation order and elapsed physical time as a representation problem for robot policy memory.
- We introduce TFP, which uses learned event-dependent memory dynamics over measured intervals to condition diffusion and flow-matching policies.
- We connect temporal memory to behavior through time-aware controls, memory readouts, and clock interventions, and evaluate its utility in real-robot and benchmark tasks.

2 Related Work

Memory in robot policies. Robot policies use history to recover information absent from the current observation, including earlier cues and completed subtasks. HAMLET aggregates compact moment tokens into memory features (Koo et al., 2026), while ReMem-VLA carries context through frame- and chunk-level recurrent queries (Li et al., 2026). MemoryVLA retrieves perceptual and cognitive representations from a consolidated memory bank (Shi et al., 2026), and MEM combines short-term visual history with long-term language memory (Torne et al., 2026). MemoryVLA uses episode timesteps to locate memory entries, and MEM evaluates duration-sensitive cooking behavior. These systems encode temporal context or evaluate timing behavior, but do not specify how elapsed physical time should shape memory evolution.

Temporal information for robot control. The role of time depends on how it enters the policy. SimpleMemVLA timestamps visual history relative to the current decision (Yin et al., 2026). DAM-VLA maintains modality-specific latent buffers refreshed at different sensor rates, decoupling action generation from individual encoding schedules (Vanjani et al., 2026). Time as a Control Dimension conditions policies on remaining time and a time ratio that regulates the progression of an internal clock (Jia & Chen, 2026). These approaches make time available through history, sensing schedules, or control conditions, but leave open how elapsed time should enter memory updates: as an input feature or an interval of state evolution.

Continuous-time dynamics in policy memory. Continuous-time recurrent models provide tools for modeling state evolution over elapsed time. LTC learns adaptive time constants (Hasani et al., 2021), while CfC provides explicit time-dependent updates without numerical ODE solvers (Hasani et al., 2022). S4D discretizes continuous-time linear state-space dynamics (Gu et al., 2022b), while Mamba learns input-dependent discretization steps for selection (Gu & Dao, 2023). M-CfC augments CfC with a gated memory pathway to support long-horizon object-goal navigation (Chen et al., 2025). TFP combines measured physical intervals with event-dependent memory dynamics to condition diffusion and flow-matching action generators.

3 Method

3.1 Memory for Control in Physical Time

TFP combines past-event evidence and measured intervals in a persistent memory state that conditions the action generator after events leave view (Figure 2).

At query n, the backbone encodes observation o_n and task specification g into compact features $\mathbf{x}_n$ and its native action context C_n. The memory summarizes the timed history, while C_n provides direct access to the current scene and task:

$$\begin{aligned} \mathbf{h}_n &= \mathcal{U}_\theta(\mathbf{h}_{n-1}, \mathbf{x}_n, \Delta_n), \\ \mathbf{A}_n &\sim \pi_{\psi,b}(\cdot \mid C_n, \mathbf{h}_n). \end{aligned} \tag{1}$$

Here $\mathbf{h}_n \in \mathbb{R}^{d_h}$ is a persistent state, $\mathbf{A}_n$ is an action chunk, and b denotes the policy backbone. For observations captured at timestamps t_n, $\Delta_n = t_n - t_{n-1}$ for $n \geq 1$. The memory state must infer how much time has passed since an event from the preceding observations and intervals. Memory is reset at episode boundaries; the first-query convention is specified in Appendix A.2.

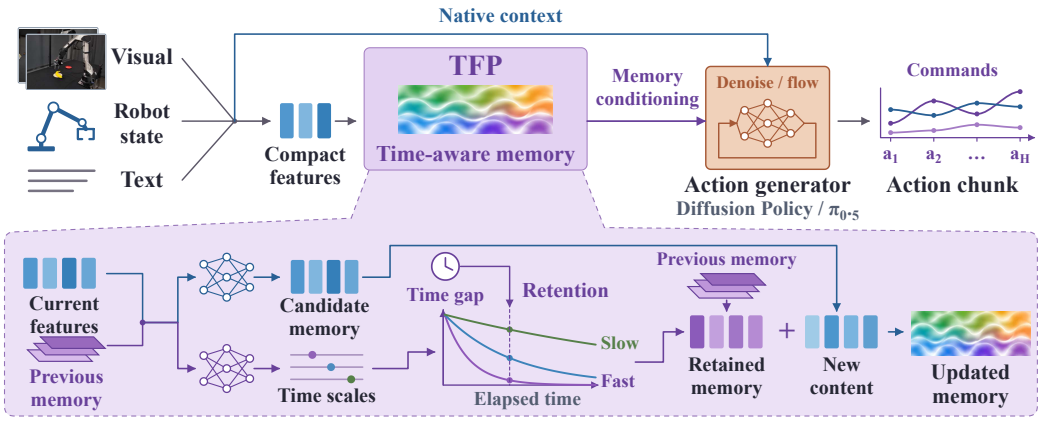


Figure 2: **Memory evolution and action generation in TFP.** Observation features and previous memory determine candidate content and adaptive time scales. The measured interval controls how far memory moves toward the candidate. The updated state conditions the backbone's action generator alongside its current context. Repeated previous-memory symbols denote the same state. Retention curves and command traces are schematic.

3.2 Memory Dynamics in Physical Time

Remembering past events. Following continuous-time recurrent models (Hasani et al., 2021; 2022), TFP predicts a candidate state and positive, channel-wise time constants from the current features and previous memory:

$$\begin{aligned} \widehat{\mathbf{h}}_n &= \tanh\big(f_\theta^h([\mathbf{x}_n; \mathbf{h}_{n-1}])\big)\,, \\ \tau_n &= f_\theta^\tau([\mathbf{x}_n; \mathbf{h}_{n-1}]) > \mathbf{0}. \end{aligned} \tag{2}$$

TFP shares a local relaxation form with canonical LTC and parameterizes candidate content and time scales through separate heads. Canonical LTC couples these quantities through a shared drive (Appendix A.3). The separate heads let the model learn what memory should represent and how quickly it changes, with past events shaping both predictions through the prior state.

Evolving with elapsed time. Elapsed time sets how far memory moves toward the candidate:

$$\begin{aligned} \mathbf{k}_n &= \exp(-\Delta_n/\tau_n), \\ \mathbf{h}_n &= \mathbf{k}_n \odot \mathbf{h}_{n-1} + (1 - \mathbf{k}_n) \odot \widehat{\mathbf{h}}_n. \end{aligned} \tag{3}$$

Operations are elementwise; τ_n and Δ_n share physical units.

Unlike memories that receive elapsed time only as an additional feature, TFP uses it as the integration interval of memory evolution. At fixed observation and prior memory, Δ_n sets this interval without changing the predicted candidate or time constants.

Local physical-time dynamics. The transition is the exact solution of a local relaxation process,

$$\frac{d\mathbf{y}(u)}{du} = \frac{\widehat{\mathbf{h}}_n - \mathbf{y}(u)}{\tau_n}, \qquad \mathbf{y}(0) = \mathbf{h}_{n-1}. \tag{4}$$

With the predicted coefficients held fixed, the transition evaluated at $u = \Delta_n$ preserves the state at zero duration, approaches the candidate monotonically, and composes across intervals (Appendix B.1). A generic learned gate need not satisfy these constraints. The coefficients are recomputed at each query.

Under these fixed coefficients, a channel displaced from equilibrium provides a monotonic temporal trace, with sensitivity decreasing toward saturation. Different time constants provide complementary temporal responses, allowing different memory channels to retain task-relevant events over different physical timescales. Together, these channels form a multi-timescale temporal basis for the memory state, echoing exponential traces studied in neuroscientific models of temporal representation (Shankar & Howard, 2012). TFP's adaptive rates change with events and interaction phases, supporting rapid updates at event onset and slower evolution during waiting.

The exponential update specifies how a given memory transition depends on elapsed time, including intervals absent from training. With the candidate and time constants held fixed, changing the interval changes the extent of relaxation through the same analytic rule. This provides a temporal inductive bias for transferring from a single training cadence to different observation schedules, which we evaluate through the controlled comparisons in Section 4.1.

3.3 Acting on Memory

Memory conditions action generation through adaptive normalization (Peebles & Xie, 2023; Perez et al., 2018). A learned projection M_b augments the native decoder condition:

$$\mathbf{c}_{n,s} = \mathbf{c}_{n,s}^{\text{native}} + M_b(\mathbf{h}_n), \tag{5}$$

Here s is diffusion or flow solver time; the condition modulates normalized decoder activations.

For Diffusion Policy (Chi et al., 2023), memory summarizes visual and robot-state features and augments the condition in the denoising network's adaptive layer-normalization blocks. For $\pi_{0.5}$ (Physical Intelligence et al., 2025), it summarizes visual, language, and robot-state features in the controlled timing tasks; the LIBERO instantiation summarizes visual and robot-state features, with

language retained in the backbone context. Both augment the action expert's adaptive RMS-normalization condition. Memory updates once per policy query and remains fixed throughout that query's solver steps: physical time governs memory evolution, while solver time governs action generation.

3.4 Learning Memory Dynamics from Demonstrations

The backbone's native imitation objective jointly trains memory dynamics and the conditioning projection from demonstrated actions. Each action loss uses the recurrent state computed from the causal observation history and its measured intervals. Controlled timing models reconstruct memory from causal prefixes; LIBERO uses consecutive episode segments with gradients truncated between segments. Appendices A.1 and D.1 detail sequence training and task protocols.

4 Experiments

We test control across observation schedules, connect temporal memory to action timing, and evaluate real-robot and general manipulation (Figure 3).

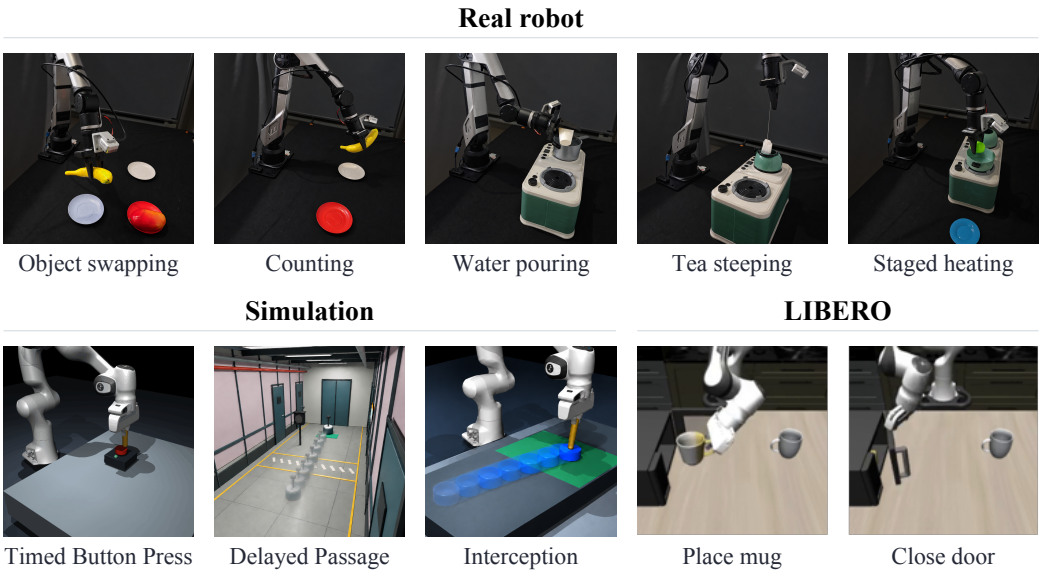


Figure 3: **Task overview.** Real-robot tasks, simulation tasks, and LIBERO tasks (Including LIBERO-Spatial, LIBERO-Goal, LIBERO-Object and LIBERO-10).

4.1 Does Policy Memory Need Physical Time?

Separating duration from observation count. Both Diffusion Policy and $\pi_{0.5}$ models are trained with fixed 0.1 s observation intervals, without mixed-cadence sampling augmentation. Time-aware models receive the measured observation interval: it is 0.1 s during training and follows the actual observation schedule at evaluation. Each trained model is evaluated at 10 Hz, 5 Hz, and under jitter and dropout, with actions continuing between queries. Task durations and target speeds remain fixed, so equal physical durations can span different observation counts.

Timed Button Press requires continuous contact for 3.0–3.5 seconds before release and retraction. **Delayed Passage** presents a brief cue at a randomized onset and requires crossing 6–9 seconds later. **Interception** requires matching a previously observed target's velocity after occlusion; the robot moves only after the target disappears. These tasks test action duration, event age, and motion estimation. Time since contact, cue timestamps, and target velocity are not policy inputs.

Evaluation. We evaluate Diffusion Policy (Chi et al., 2023) and $\pi_{0.5}$ (Physical Intelligence et al., 2025) with reactive and short-history controls, and GRU (Cho et al., 2014) and S4D (Gu et al., 2022b) both with and without measured intervals. We also evaluate LTC (Hasani et al., 2021) with measured intervals for both backbones. The $+\Delta t$ suffix denotes measured intervals. We report mean success across three training seeds. Per seed and schedule, we evaluate 50 trials each for Button Press and Passage and 300 direction–speed cases for Interception, paired across methods within each backbone. Appendices D.1 and E.1 give protocols and task-average sample SDs.

Table 1: Temporal control across observation schedules. Mean success over three training seeds (%, ↑). Fixed-Δt uses 0.1 s at inference. Bold: column maxima by backbone, including ties.

	Button Press				**Passage**				**Interception**			
Memory	10 Hz	5 Hz	Jit.	Drop.	10 Hz	5 Hz	Jit.	Drop.	10 Hz	5 Hz	Jit.	Drop.
Diffusion Policy												
Reactive	0.0	0.0	0.0	0.0	0.0	0.7	5.3	3.3	8.7	10.3	11.7	11.7
History	0.0	0.0	0.0	0.0	1.3	2.7	2.7	2.0	30.0	34.3	29.0	23.0
GRU	42.7	4.7	28.0	3.3	92.0	4.0	50.0	5.3	40.3	10.3	32.3	18.3
GRU+Δt	44.0	5.3	30.0	4.7	90.0	5.3	50.0	6.0	41.3	11.0	33.0	19.0
S4D	52.7	16.0	24.0	10.0	90.0	12.0	44.0	16.0	36.3	16.3	26.3	20.3
LTC+Δt	18.7	13.3	20.7	11.3	24.7	18.0	22.7	16.0	24.3	21.1	25.4	20.0
S4D+Δt	56.7	46.0	50.0	40.0	96.7	94.7	96.7	92.7	40.3	46.3	44.3	40.3
TFP fixed Δt	**88.0**	0.7	12.7	0.7	**97.3**	0.7	32.7	0.7	**55.3**	15.9	39.1	17.4
TFP *(ours)*	**88.0**	**94.7**	**76.0**	**66.0**	**97.3**	**96.7**	**98.7**	**96.7**	**55.3**	**52.6**	**56.3**	**51.3**
$\pi_{0.5}$												
Reactive	0.0	0.0	0.0	0.0	6.7	16.0	8.7	6.0	14.0	18.4	18.1	22.6
History	0.0	14.0	4.7	3.3	2.7	4.7	0.7	3.3	35.6	**35.1**	35.1	34.7
GRU	6.7	1.3	28.0	6.0	**90.0**	4.7	46.0	6.7	31.3	18.3	30.3	28.3
GRU+Δt	7.3	2.7	30.0	6.7	88.0	5.3	48.0	8.0	32.3	19.0	31.0	29.0
S4D	90.0	10.0	20.0	8.0	52.7	10.0	24.0	12.0	28.3	10.3	20.3	16.3
LTC+Δt	23.3	18.7	22.7	16.0	26.7	22.7	28.7	18.7	29.3	25.3	28.3	23.3
S4D+Δt	92.0	**98.7**	93.3	88.0	52.0	84.0	78.0	**84.0**	29.1	32.4	34.0	35.6
TFP fixed Δt	**94.0**	0.7	14.0	0.7	**90.0**	0.7	46.0	6.0	**47.3**	20.6	34.3	26.4
TFP *(ours)*	**94.0**	95.3	**94.7**	**90.0**	**90.0**	**84.7**	**88.0**	**84.0**	**47.3**	33.2	**37.1**	**41.0**

The $+\Delta t$ suffix denotes measured intervals; plain GRU and S4D are independently trained without them for both backbones. Jit./Drop.: jitter/dropout. Task-average sample SDs: Appendix E.1.

Memory improves control. Reactive policies fail Button Press under every schedule with both backbones (Table 1). Short history improves Interception but achieves at most 4.7% on Passage, where recurrent memory must retain the cue throughout the waiting interval.

Averaged equally across tasks and schedules, TFP achieves the highest reported success: 77.5% with Diffusion Policy versus 62.1% for the stronger time-aware control, S4D+Δt; with $\pi_{0.5}$, the corresponding rates are 73.3% and 66.8%. Passage performance remains high across schedules, supporting responses to remembered cues despite changes in observation cadence.

4.2 How Memory Controls Behavior

Frozen-weight interventions test how the supplied clock and stored memory affect behavior; separately trained ablations assess architectural choices. Linear probes characterize information recoverable from memory.

Action timing follows the supplied clock. We scale the intervals supplied to trained policies while keeping weights and physical execution schedules fixed. Under jitter, halving the intervals delays button release and passage entry, whereas doubling them advances these actions (Figure 4). For $\pi_{0.5}$ on Button Press and both backbones on Passage, each directional shift occurs in all 50 paired trials. Both scalings eliminate success on these tasks with either backbone (Appendix D.5).

Replacing measured intervals with 0.1 s reduces success, and clearing memory at every query eliminates success on all three tasks at 10 Hz (Appendix Table 8). Timing thus depends on both measured intervals and accumulated history.

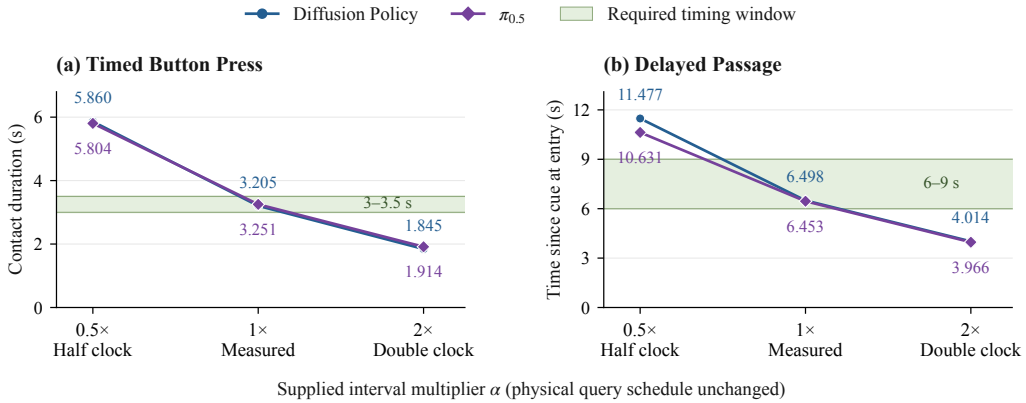


Figure 4: **Scaling the supplied clock shifts action timing.** Median physical durations under jitter with input intervals $\alpha\Delta_n$; weights and physical query schedules stay fixed. Shading marks the required timing windows. Button measures contact until release; Passage measures first attempted entry since cue onset. DP Button medians use 20, 38, and 36 observed releases out of 50 paired trials per condition for $\alpha = 0.5, 1, 2$, respectively; all other points use 50 observed events.

Elapsed event time is recoverable from memory. Linear probes of frozen Diffusion Policy recover event duration and motion information from TFP memory. For Passage, predicting time since the cue from memory reduces mean absolute error from about 0.83 s using episode time alone to 0.04 s, supporting event-relative timing beyond episode progress. After occlusion, memory also retains speed information beyond single-frame cues. Appendix D.6 gives the full comparisons and protocols.

Memory conditioning supports action generation. Without memory conditioning throughout Diffusion Policy training and evaluation, success is 0.5% on Button Press and 3.7% on Passage, compared with the Full TFP means of 81.2% and 97.3%, respectively (Appendix D.3).

Three design choices for physical-time memory. Separately trained ablations remove history from the candidate, use static time constants, or replace interval-based evolution with a learned Δt gate (Table 2). Each lowers mean success relative to Full TFP, supporting event-dependent content, adaptive timescales, and the update rule in Section 3.2. On Diffusion Policy Button and Passage, the learned gate achieves 14.5% and 15.0%, compared with 81.2% and 97.3% for TFP's exponential update. Appendix D.2 gives additional ablations.

Table 2: Which memory components matter? Mean success over three training seeds (%, ↑), averaged equally over four schedules. Parentheses show changes from Full TFP (pp).

	Diffusion Policy			$\pi_{0.5}$		
Variant	Button	Passage	Intercept.	Button	Passage	Intercept.
Full TFP *(ours)*	**81.2**	**97.3**	**53.9**	**93.5**	**86.7**	**39.7**
Candidate w/o history	42.7 (−38.5)	8.0 (−89.3)	44.2 (−9.7)	12.0 (−81.5)	65.7 (−21.0)	31.7 (−8.0)
Static time constants	58.5 (−22.7)	82.7 (−14.6)	44.8 (−9.1)	64.3 (−29.2)	13.3 (−73.4)	31.4 (−8.3)
Learned Δt gate	14.5 (−66.7)	15.0 (−82.3)	30.5 (−23.4)	27.0 (−66.5)	37.0 (−49.7)	30.1 (−9.6)

Task-average sample standard deviations across the three seeds appear in Appendix E.1.

Figure 5 illustrates these dynamics around the passage cue. Different channels exhibit distinct temporal responses, with some rapidly adapting around events and others maintaining slower traces dur-

ing waiting. Write gains, $\mathbf{g}_n = \mathbf{1} - \mathbf{k}_n$, show the resulting variation across channels. Appendix D.7 shows corresponding event-aligned visualizations for Button Press and Interception.

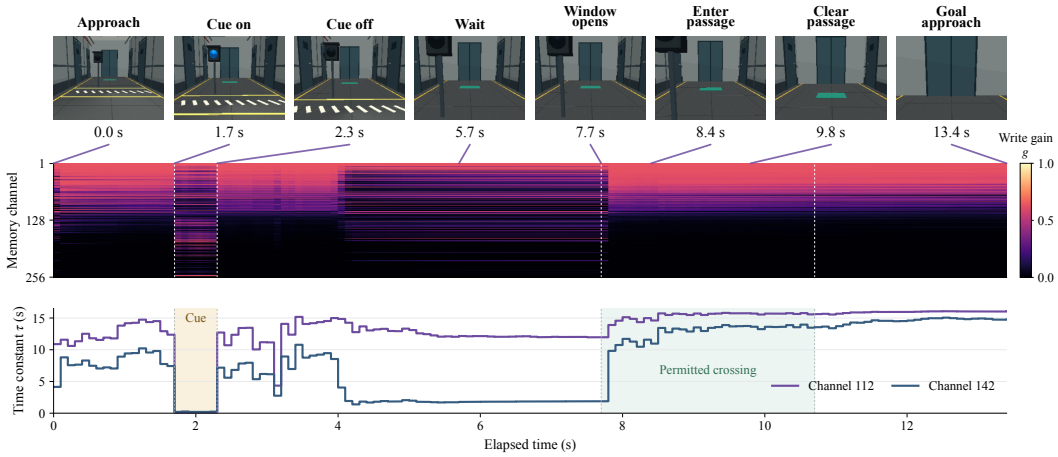


Figure 5: **Memory dynamics during Delayed Passage.** Observations, channel-wise write gains, and two channel time constants share a physical time axis. Heatmap cells and step traces start at their query timestamps. Shading marks the cue and permitted crossing window. A demonstration is replayed through frozen $\pi_{0.5}$ with TFP; Appendix D.6 details channel and trajectory selection.

4.3 Does TFP Improve Real-Robot Control?

On a Galaxea A1, we compare TFP with each backbone's standard reactive policy. Object swapping, counting pick-and-place, and water pouring require tracking action sequences. Tea steeping requires 3.0 ± 0.5 s from the infuser entering the water to its removal. Staged heating requires two such intervals: heater activation to food contact, then contact to heater deactivation. Success requires completing each sequence in order within its timing constraints. Each task uses 50 demonstrations and 20 evaluation trials per method. Event timestamps are annotated in evaluation videos with the same timing criteria for all methods.

Table 3: Real-robot manipulation on Galaxea A1. Success (%, ↑); Avg. weights all five tasks equally. Reactive denotes the standard backbone policy without persistent memory. Gains are TFP minus Reactive, in percentage points (pp).

Method	Object swapping	Counting pick-place	Water pouring	Tea steeping	Staged heating	**Avg.**
Diffusion Policy						
Reactive	10.0	30.0	20.0	15.0	10.0	17.0
TFP *(ours)*	**60.0**	**75.0**	**60.0**	**65.0**	**60.0**	**64.0**
Gain (pp)	+50.0	+45.0	+40.0	+50.0	+50.0	+47.0
$\pi_{0.5}$						
Reactive	15.0	40.0	30.0	25.0	20.0	26.0
TFP *(ours)*	**75.0**	**90.0**	**70.0**	**75.0**	**70.0**	**76.0**
Gain (pp)	+60.0	+50.0	+40.0	+50.0	+50.0	+50.0

20 trials per task and method. Bold compares Reactive and TFP within each backbone.

TFP improves all five tasks, with mean gains of 47 and 50 percentage points for Diffusion Policy and $\pi_{0.5}$, respectively (Table 3). Both backbones gain 50 points on tea steeping and staged heating, extending the timing results to physical duration constraints. See Appendix D.10 for details.

4.4 MEMORY BEYOND EXPLICIT TIMING TASKS

We evaluate $\pi_{0.5}$ with TFP on LIBERO (Liu et al., 2023) and visually perturbed LIBERO-plus (Fei et al., 2025) (Table 4). Standard LIBERO Reactive scores are taken from the official OpenPI evaluation. Our reproduced memory variants share demonstrations, execution settings, and evaluation protocols, with three training seeds and 50 rollouts per task and seed on standard LIBERO.

Table 4: General manipulation and visual robustness. Success (%, ↑); reproduced variants report means over three training seeds. Bold marks column maxima across listed methods, including ties. Gains compare TFP with the corresponding reactive $\pi_{0.5}$ baseline (pp).

Policy	LIBERO					LIBERO-plus			
	Spatial	**Object**	**Goal**	**Long-10**	**Avg.**	**Noise**	**Light**	**Backgr.**	**Avg.**
Published results									
Reactive $\pi_{0.5}$ (OpenPI)	98.8	98.2	98.0	92.4	96.9	–	–	–	–
HAMLET (3B)	99.0	**100.0**	99.2	92.2	97.6	–	–	–	–
AVA-VLA (Xiao et al., 2026) (7.3B)	97.4	99.4	97.4	**97.6**	98.0	78.0	95.5	88.9	87.5
MemoryVLA (7.3B)	98.4	98.4	96.4	93.4	96.7	–	–	–	–
Reproduced $\pi_{0.5}$ variants (3.3B)									
Reactive (LIBERO-plus)	–	–	–	–	–	85.2	93.9	95.1	91.4
GRU	99.0	98.4	95.8	89.1	95.6	85.9	91.5	96.1	91.1
S4D	98.5	98.4	97.0	92.5	96.6	82.6	95.0	92.3	90.0
TFP *(ours)*	**99.5**	99.0	**99.3**	96.9	**98.7**	**88.5**	**96.1**	**96.6**	**93.7**
Gain vs. Reactive (pp)	+0.7	+0.8	+1.3	+4.5	+1.9	+3.3	+2.1	+1.5	+2.3

Standard LIBERO Reactive: official OpenPI results. Our variants use 50 rollouts per task and training seed. Gains use unrounded averages. Seed SDs and TFP rollout CIs: Appendix E.2. Published policies use their reported settings; – denotes an unreported result.

TFP raises mean LIBERO success from 96.9% to 98.7%. The largest gain is on Long-10, from 92.4% to 96.9% (+4.5 points), compared with gains of 0.7–1.3 points on the other suites. On LIBERO-plus, success rises from 91.4% to 93.7%, leading reproduced variants in every condition. Appendices D.8 and E give variant settings and rollout uncertainty.

5 CONCLUSION

We identify physical time as a missing dimension of robot policy memory driven by observation sequences. TFP retains past events in a memory state that evolves over measured time intervals. Observations and history determine the content and timescales of the transition, while measured intervals determine how far the memory state advances. This state conditions existing diffusion and flow-matching action generators.

Comparisons with recurrent and state-space memories supplied with the same measured intervals support using elapsed time to govern memory evolution. Memory readouts recover elapsed event time beyond episode progress and motion information after occlusion. Clock-scaling interventions systematically advance or delay button release and passage entry, while memory resets and conditioning ablations support the role of accumulated history in control. Real-robot tea steeping and staged heating extend the evidence to physical duration constraints; memory-dependent tasks and LIBERO show broader benefits in manipulation. When actions depend on how long ago an event occurred, memory dynamics should follow physical time.

AI USE STATEMENT

Large language model tools assisted with language editing and revision of the manuscript, as well as code development. The authors reviewed all revisions and take full responsibility for the methods, experiments, analyses, and claims.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A describe the memory dynamics, backbone integration, parameterization, and training procedure. Appendix D documents the experimental settings, including training configurations, observation schedules, evaluation protocols, success criteria, and the procedures for interventions, ablations, and memory probes. The real-robot setup and timing criteria are described in Appendix D.10.

REFERENCES

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1):6085, 2018. doi: 10.1038/s41598-018-24271-9. URL https://www.nature.com/articles/s41598-018-24271-9.

Zifan Chen, Jiahong Zhang, Sijun Shen, Wenlong Fu, and Lihong Cao. Memory-augmented continuous-time neural policy for vision-guided embodied navigation. In *Proceedings of the 7th ACM International Conference on Multimedia in Asia*, pp. 1–8. Association for Computing Machinery, 2025. doi: 10.1145/3769748.3773343. URL https://doi.org/10.1145/3769748.3773343.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022a.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, 2022b. URL https://arxiv.org/abs/2206.11893.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7657–7666, 2021. doi: 10.1609/aaai.v35i9.16936. URL https://ojs.aaai.org/index.php/AAAI/article/view/16936.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4(11):992–1003, 2022. doi: 10.1038/s42256-022-00556-7.

Yinsen Jia and Boyuan Chen. Time as a control dimension in robot learning, 2026. URL https://arxiv.org/abs/2511.07654v2. Version 2; first posted in 2025.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.00695.

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. ReMem-VLA: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026. doi: 10.48550/arXiv.2603.12942. URL https://arxiv.org/abs/2603.12942.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023. doi: 10.48550/arXiv.2306.03310. URL https://arxiv.org/abs/2306.03310.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, October 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ODEs for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1907.03907.

Karthik H. Shankar and Marc W. Howard. A scale-invariant internal representation of time. *Neural Computation*, 24(1):134–193, 2012. doi: 10.1162/NECO_a_00212. URL https://doi.org/10.1162/NECO_a_00212.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.19236.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026. doi: 10.48550/arXiv.2603.03596. URL https://arxiv.org/abs/2603.03596.

Pankhuri Vanjani, Zhuoyue Li, Jakub Suliga, Moritz Reuss, Gianluca Geraci, Xinkai Jiang, and Rudolf Lioutikov. DAM-VLA: Decoupled asynchronous multimodal vision language action model, 2026. URL https://arxiv.org/abs/2606.12105.

Lei Xiao, Jifeng Li, Juntao Gao, Feiyang Ye, Yan Jin, Jingjing Qian, Jing Zhang, Yong Wu, and Xiaoyuan Yu. AVA-VLA: Improving vision-language-action models with active visual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13453–13463, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Xiao_AVA-VLA_Improving_Vision-Language-Action_models_with_Active_Visual_Attention_CVPR_2026_paper.html.

Cheng Yin, Wang Xu, Junpeng Yang, Sikyuen Tam, Hanyu Liu, Yuan Yao, Xiangrui Zeng, Junbo Cui, Yequan Wang, Zhouping Yin, and Yankai Lin. SimpleMemVLA: A simple but effective native-video memory for vision-language-action models, 2026. URL https://arxiv.org/abs/2609.05533.

A Additional Method Details

A.1 Sequence Training

The controlled timing experiments reconstruct memory from causal prefixes. For $\pi_{0.5}$, the recurrence starts from zero and gradients flow through the full prefix to each of four supervised queries. Padded observations leave memory unchanged and padded actions are excluded from the loss. LIBERO instead uses consecutive episode segments with memory carried forward and gradients truncated between segments, as detailed below.

For the LIBERO $\pi_{0.5}$ instantiation, each episode e has a hidden state $\mathbf{h}_n^{(e)} = \mathcal{U}_\theta(\mathbf{h}_{n-1}^{(e)}, \mathbf{x}_n^{(e)}, \Delta_n^{(e)})$, initialized to $\mathbf{h}_{-1}^{(e)} = \mathbf{0}$ and reset only at episode boundaries. At optimization step r, we sample B active episodes and unroll each for K consecutive queries. For episode e_i, let n_i be the segment's first query index and $k = 0, \ldots, K-1$ index queries within the segment, so that $\mathbf{x}_{i,k} = \mathbf{x}_{n_i+k}^{(e_i)}$ and $\Delta_{i,k} = \Delta_{n_i+k}^{(e_i)}$. Writing $\tilde{\mathbf{h}}$ for the stored state, we retrieve $\mathbf{h}_{i,-1} = \tilde{\mathbf{h}}_{n_i-1}^{(e_i)}$ and update

$$\mathbf{h}_{i,k} = \mathcal{U}_\theta(\mathbf{h}_{i,k-1}, \mathbf{x}_{i,k}, \Delta_{i,k}), \qquad k = 0, \ldots, K-1. \tag{6}$$

The optimization objective is

$$\mathcal{L}_r = \frac{1}{BK} \sum_{i=1}^{B} \sum_{k=0}^{K-1} \ell_{\text{flow}}^{(i,k)}. \tag{7}$$

After the parameter update, the final hidden state is written back with gradient truncation,

$$\tilde{\mathbf{h}}_{n_i+K-1}^{(e_i)} \leftarrow \text{stopgrad}(\mathbf{h}_{i,K-1}). \tag{8}$$

This preserves forward memory across training segments while bounding the backpropagation length.

A.2 Memory Initialization and Parameterization

Using zero-based query indices, memory is initialized as $\mathbf{h}_{-1} = \mathbf{0}$ at each episode boundary. The controlled timing tasks use a bootstrap interval $\Delta_0 = 0.1\,\text{s}$, whereas the released LIBERO implementation uses $\Delta_0 = 0$; subsequent updates use timestamp differences. The released LIBERO client uses wall-clock intervals between inference requests. In experimental labels, Δt denotes this elapsed-time input.

Diffusion Policy. The memory has 128 channels. Separate heads predict a $\tanh$-bounded candidate and positive time constants using softplus with a 10^{-6} floor. For the controlled Button and Passage recipes, time-constant head weights start at zero with the default linear-layer bias; Interception retains the default linear-layer initialization.

$\pi_{0.5}$. The memory has 256 channels and a $\tanh$-bounded candidate. For the controlled timing tasks, a sigmoid maps time constants to $[\tau_{\min}, \tau_{\max}]$, set by the median query interval and the longest episode duration in the training data, respectively. The time-constant head has zero initial weights; its biases initialize the channel time constants geometrically from $1.01\tau_{\min}$ to $\tau_{\max}/1.01$. The memory-to-action projection starts with its default weights scaled by 0.02 and zero bias. The released LIBERO implementation instead uses the positive softplus parameterization described above. These TFP implementations use the exponential update in Equation 3.

A.3 Relation to Canonical LTC Dynamics

Canonical LTC dynamics (Hasani et al., 2021) take the form

$$\frac{d\mathbf{h}}{du} = -\tau_0^{-1} \odot \mathbf{h} + \mathbf{f}_\theta(\mathbf{h}, \mathbf{x}) \odot (\mathbf{a} - \mathbf{h}), \tag{9}$$

with positive baseline time constants and a nonnegative drive. Writing

$$\lambda = \tau_0^{-1} + \mathbf{f}_\theta, \qquad \mu = \frac{\mathbf{f}_\theta \odot \mathbf{a}}{\lambda}, \tag{10}$$

gives the relaxation form

$$\frac{d\mathbf{h}}{du} = \lambda \odot (\mu - \mathbf{h}). \tag{11}$$

For frozen coefficients its exact update is

$$\mathbf{h}^{+} = e^{-\delta\lambda} \odot \mathbf{h}^{-} + (1 - e^{-\delta\lambda}) \odot \mu. \tag{12}$$

The interpolation has the same form as TFP when $\tau_n = \lambda^{-1}$ and $\widehat{\mathbf{h}}_n = \mu$. However, canonical LTC couples these quantities through the same drive, whereas TFP predicts them independently. TFP therefore shares the local relaxation form while learning candidate content and time constants through separate heads.

As in CfC (Hasani et al., 2022), content and temporal modulation depend on inputs and prior state; TFP uses this dependence to build persistent robot policy memory that retains task-relevant events, evolves over measured physical intervals, and conditions diffusion and flow-matching action generators.

B Properties of the Memory Update

The memory dynamics in Equation 3 have a local relaxation interpretation. The following analysis holds the predicted candidate and time constants fixed over an interval, as in the update evaluated at a single policy query.

B.1 Local Relaxation and Interval Composition

For a fixed candidate μ and positive rate λ, the solution of $d\mathbf{y}/du = \lambda \odot (\mu - \mathbf{y})$ with initial state $\mathbf{h}$ is

$$\Phi_\delta(\mathbf{h}) = \mu + e^{-\delta\lambda} \odot (\mathbf{h} - \mu). \tag{13}$$

Setting $\mu = \widehat{\mathbf{h}}_n$ and $\lambda = \tau_n^{-1}$ recovers TFP's update. At zero duration, $\Phi_0(\mathbf{h}) = \mathbf{h}$. For $\delta \geq 0$, each coordinate interpolates between the initial state and the candidate, with distance

$$|[\Phi_\delta(\mathbf{h})]_j - \mu_j| = e^{-\delta\lambda_j} |h_j - \mu_j|. \tag{14}$$

Since $\lambda_j > 0$, this distance decreases monotonically with duration unless $h_j = \mu_j$, in which case the coordinate remains unchanged. Direct substitution also gives

$$\Phi_{\delta_2}(\Phi_{\delta_1}(\mathbf{h})) = \Phi_{\delta_1+\delta_2}(\mathbf{h}). \tag{15}$$

Thus, splitting an interval leaves the result unchanged when both parts use the same candidate and rate.

At the level of a scalar retention function, the exponential form follows from this composition condition. If a continuous function $k : [0, \infty) \to (0, 1]$ satisfies $k(0) = 1$ and $k(\delta_1 + \delta_2) = k(\delta_1)k(\delta_2)$, then $\log k$ is continuous and additive, so $\log k(\delta) = -\lambda\delta$ for some $\lambda \geq 0$. Consequently, $k(\delta) = e^{-\lambda\delta}$.

TFP recomputes the candidate and rate from the observation and memory at each query, then holds them fixed over the local integration interval.

Temporal inductive bias and unseen intervals. Identity at zero duration, monotonic approach to a candidate, and interval composition constrain how elapsed time affects the transition. A generic learned gate that receives elapsed time as a feature does not structurally enforce these properties and must learn its temporal response from data. For fixed candidate and positive time constants, TFP's transition remains defined for every nonnegative interval, including intervals outside the training range. Each coordinate stays between its initial value and its candidate, and converges to the candidate as the interval tends to infinity. This specifies the transition's behavior beyond observed intervals without requiring the model to learn those constraints from examples at every duration. Table 2 compares this exponential update with a learned temporal gate.

B.2 Duration Sensitivity

Differentiating Equation 13 with respect to the interval gives

$$\frac{\partial \Phi_\delta(\mathbf{h})}{\partial \delta} = \lambda \odot e^{-\delta\lambda} \odot (\mu - \mathbf{h}). \tag{16}$$

The state changes with duration when it differs from the candidate, with the rate controlling how quickly that difference decays. At equilibrium, $\mathbf{h} = \mu$, duration sensitivity vanishes.

C Relation to Alternative Temporal Models

C.1 Recurrent and Continuous-Time Models

A GRU (Cho et al., 2014) or LSTM supplied with Δ_n can learn duration-dependent transitions from temporal inputs. Without that input, identical observation sequences produce the same recurrent trajectory even if they span different durations. TFP assigns elapsed duration a specific role in the transition: it sets the integration interval of the learned relaxation dynamics.

Time-aware recurrent models also provide mechanisms for handling irregular observations. GRU-D uses elapsed time in learned input and hidden-state decays (Che et al., 2018), while ODE-RNNs and Latent ODEs integrate hidden dynamics over irregular observation intervals (Rubanova et al., 2019). These models establish that elapsed time can shape hidden-state evolution beyond serving as an additional input.

A continuous-time relaxation baseline can also use a constant learned rate,

$$\frac{d\mathbf{h}}{du} = \lambda_0 \odot (\mathbf{q}_\theta(\mathbf{h}, \mathbf{x}) - \mathbf{h}), \qquad \lambda_0 > \mathbf{0}. \tag{17}$$

Comparing this baseline with TFP tests the value of adapting the update rate to state and input while preserving history-dependent candidate content.

C.2 State-Space Models and Computational Cost

Continuous-time state-space models (Gu et al., 2022a) can likewise represent transitions that depend on elapsed time, and input-dependent state-space models provide additional adaptive mechanisms (Gu & Dao, 2023). Our manipulation comparison includes S4D alongside recurrent variants within the same action-conditioning setup (Appendix D.8). Mamba's learned discretization step controls selection and is not itself a measured physical interval.

Evaluating the candidate and time-constant heads costs $\mathcal{O}(d_h(d_x + d_h))$ per query, followed by an $\mathcal{O}(d_h)$ elementwise update. Projecting the memory costs $\mathcal{O}(d_h d_c)$, with additional adapter work depending on the decoder. Here d_x and d_c denote the observation-feature and decoder-conditioning widths, respectively. Each active episode requires $\mathcal{O}(d_h)$ persistent state, independent of episode length. These costs are incurred at the existing policy queries.

D Additional Experimental Details

D.1 Controlled Timing Protocols

Model configurations. Table 5 summarizes the controlled-task models, trained separately for each task. Diffusion Policy uses a compact visual architecture (Chi et al., 2023), trained from scratch: convolutional stages of 16, 32, and 64 channels, 3×3 pooling, and a 128-dimensional projection. Its denoiser has three Transformer blocks of width 128, four attention heads, and feed-forward width 512. The $\pi_{0.5}$ models freeze the visual-language backbone and train the action expert, memory, and conditioning projections; visual features are cached during training. They use one RGB camera and a continuous state token. Both backbones retain current-observation conditioning and keep memory fixed throughout each query's solver steps.

Table 5: TFP architecture and sampling settings for the controlled timing tasks.

Setting	Diffusion Policy	$\pi_{0.5}$
RGB resolution	96×96	224×224
Observation summary d_x	2×128 (vision, state)	3×256 (vision, state, text)
Memory width d_h	128	256
Input to each memory head	$384 = 256 + 128$	$1024 = 768 + 256$
Candidate head	Linear + $\tanh$	Linear + RMS norm + $\tanh$
Time-constant head	Linear + softplus + 10^{-6}	Linear + bounded sigmoid
Memory projection	$128 \to 128$, AdaLN	$256 \to 1024$, AdaRMS
Action horizon	16	10
Evaluation sampler	50-step DDIM, $\eta = 0$	20-step Euler flow solver

For $\pi_{0.5}$, each modality projection is RMS-normalized before concatenation. Time-constant ranges and initialization are given in Appendix A.2. DP uses 100 training diffusion timesteps with a cosine noise schedule and predicts clean actions (x_0). The short-history baseline uses two frames.

Baseline memory implementations. Recurrent alternatives replace the memory core while retaining each backbone's observation projections and action-conditioning pathway. Their memory outputs have width 128 for Diffusion Policy and 256 for $\pi_{0.5}$.

For **GRU+Δt**, Diffusion Policy adds a learned affine projection of the interval in seconds to the observation features before the GRU update. The $\pi_{0.5}$ implementation instead concatenates $\log(1 + \Delta t/(0.1\,\mathrm{s}))$ with the observation and recurrent features used by the reset/update gates and candidate. **Plain GRU** removes this explicit time contribution throughout training and evaluation and is trained separately.

S4D+Δt uses a diagonal complex-valued state-space recurrence with zero-order-hold discretization. Measured intervals scale each channel's learned nominal step by $\Delta t/(0.1\,\mathrm{s})$. A learned readout, input skip connection, and gated output projection produce the memory condition. For both Diffusion Policy and $\pi_{0.5}$, **plain S4D** is trained independently with a constant 0.1-second input interval throughout training and evaluation.

LTC+Δt receives elapsed seconds directly in its state updates. Diffusion Policy uses the `ncps` 1.0.1 cell; $\pi_{0.5}$ uses the corresponding JAX implementation. LTC uses dense sensory and recurrent conductances with positive parameter constraints and six semi-implicit integration steps per observation interval. We compare LTC and TFP under the same backbone and action-conditioning interface. TFP uses the exponential update in Equation 3.

Temporal inputs. TFP starts from zero memory with a common 0.1-second first interval. For both backbones, fixed-Δt evaluation replaces each input interval with 0.1 seconds while keeping the trained weights fixed. At 10 Hz, fixed and measured intervals coincide.

Training configurations. Table 6 summarizes the optimization settings. Both Diffusion Policy and $\pi_{0.5}$ use fixed 0.1-second observation intervals (10 Hz) during training, without mixed-cadence sampling augmentation; the other observation schedules are used at evaluation. The main controlled timing comparison uses three training seeds. For DP, Button uses 400/100 training/validation demonstrations; Passage uses 400/100 original plus 128/32 recovery demonstrations. Interception uses 60/12 motion profiles, formed from 10/two scenes crossed with two directions and three speeds.

Observation schedules and execution. Reported capture intervals are 0.1 seconds (10 Hz), 0.2 seconds (5 Hz), uniform draws from $\{0.05, 0.10, 0.15, 0.20\}$ seconds (jitter), or draws from $\{0.10, 0.20, 0.40\}$ seconds with probabilities $\{0.6, 0.2, 0.2\}$ (dropout). A captured observation replaces the action queue; missing captures create no extra memory updates. Timeouts are 12, 22, and 6 seconds for button, passage, and interception. The simulations keep controller timing separate from GPU inference wall time. The DP Passage evaluator instead uses jitter intervals $\{0.10, 0.20, 0.30, 0.40\}$ seconds and dropout intervals $\{0.20, 0.40, 0.80\}$ seconds with probabilities $\{0.7, 0.2, 0.1\}$; these release defaults differ from the reported-result protocol above.

Physical success criteria. Button success requires a single continuous press lasting 3.0–3.5 seconds followed by retraction above 0.30 meters. Passage presents a 0.6-second cue at a randomized

Table 6: Training configurations for the controlled timing tasks.

Backbone	Task	Batch	Learning rate	Updates	EMA
DP	Button	4	3×10^{-4}	8,000	0.995
DP	Passage	4	$3 \times 10^{-4} \rightarrow 10^{-4}$	$8{,}000 + 8{,}000$	0.99
DP	Interception	8	3×10^{-4}	8,000	0.995
$\pi_{0.5}$	Button	32	10^{-5}	2,000	—
$\pi_{0.5}$	Passage	32	10^{-5}	7,000	—
$\pi_{0.5}$	Interception	32	$10^{-5} \rightarrow 3 \times 10^{-6}$	10,000	—

All use AdamW with global gradient-norm clipping at 1; weight decay is 10^{-4} for DP and 0.01 for $\pi_{0.5}$ on decayed parameter groups. For $\pi_{0.5}$, batch denotes supervised queries per update, accumulated over microbatches (eight groups of four for Button); rates are base peak rates, with a $10\times$ update multiplier for newly initialized parameters in Passage and Interception. The initial warmup is 500 updates; later Interception stages use 100, and its last 4,000 updates use the lower peak rate. DP Passage changes rate after 8,000 updates. A dash denotes no EMA.

onset and permits crossing during $[t_{\text{cue}} + 6, t_{\text{cue}} + 9]$; the robot must clear the window and reach the goal without collision. Reported Interception evaluations use target speeds of 0.08, 0.12, and 0.16 m/s in two directions under each observation schedule. The target disappears at 0.55 seconds for the remainder of the six-second episode, and the robot holds until 0.6 seconds. Success requires legal first contact with pre-collision planar velocity mismatch at most 0.04 m/s, sustained for at least 0.1 seconds.

Statistics. Cross-schedule means weight the four schedules equally and are computed from unrounded success counts. The ablations in Tables 2 and 7 report three-seed means.

D.2 Memory-Structure Ablations

Table 7: Additional Diffusion Policy ablations. Success (%, ↑), reported as mean ± sample SD over three training seeds. Each seed's result averages the four observation schedules equally.

Variant	Button	Passage
Full TFP	$\mathbf{81.2 \pm 1.2}$	$\mathbf{97.3 \pm 1.0}$
τ: observation	68.3 ± 1.5	88.5 ± 1.0
τ: memory	49.3 ± 1.5	94.0 ± 1.0
τ: fixed	58.5 ± 1.3	82.7 ± 1.3
No memory condition	0.5 ± 0.5	3.7 ± 0.8
Reset each observation	0.3 ± 0.6	6.2 ± 0.8
Shared scalar τ	60.7 ± 1.3	93.0 ± 1.0
Rational retention	50.7 ± 1.3	93.0 ± 1.0

In Table 2, static time constants retain only a learned bias, and the learned Δt gate uses a sigmoid head on $[\mathbf{x}_n; \mathbf{h}_{n-1}; \Delta_n]$. Observation features, memory width, and action-conditioning pathways are held fixed within each comparison.

The Diffusion Policy candidate and static-time-scale comparisons share task data, training budgets, and common-parameter initialization with the time-scale-input variants in Table 7. The latter restrict the time-constant head to observation or memory inputs, or freeze a static time constant. These comparisons use measured intervals.

D.3 Accumulated Memory and Action Conditioning

Table 7 compares full TFP with two separately trained ablations. *No memory conditioning* removes the recurrent contribution to the decoder condition. *Reset memory each observation* retains the memory update and measured interval but starts every update from zero, eliminating information carried across observations. Both preserve current-observation and diffusion-timestep conditioning.

The ablations apply throughout training and evaluation under the task-specific training protocols and budgets in Appendix D.1. Shape-compatible parameters share initialization.

Results average four schedules equally, with 50 paired test scenes per task and schedule for each of three training seeds.

D.4 Update Rule

Table 7 compares two update variants under the same task-specific training protocols. Results are averaged over four schedules with 50 scenes per task and schedule for each of three training seeds. *Shared scalar* τ predicts one time constant for all memory channels. *Rational retention* replaces $\exp(-\Delta_n/\tau_{n,j})$ by $(1 + \Delta_n/\tau_{n,j})^{-1}$ in each channel j, retaining measured time and the same candidate update.

D.5 Frozen-Weight Interventions

Table 8: Time-input and memory-reset interventions. Success (%, ↑). Each intervention uses the same trained weights as its corresponding unmodified policy. Fixed-input results average four query schedules; clock scaling uses jitter; memory resets use 10 Hz.

		Button Press		**Passage**		**Interception**	
Backbone	Memory	Normal	Fixed	Normal	Fixed	Normal	Fixed
(a) Fixed time input: $\Delta t \to 0.1\,s$							
DP	**TFP**	**81.2**	25.5	**97.3**	32.8	**53.9**	31.9
$\pi_{0.5}$	**TFP**	**93.5**	27.3	**86.7**	35.7	**39.7**	32.2
(b) Scaled time input: $\Delta t \to \alpha\Delta t$							
Backbone	Memory	1.0	$0.5\,/\,2.0$	1.0	$0.5\,/\,2.0$	1.0	$0.5\,/\,2.0$
DP	**TFP**	**76.0**	0.0 / 0.0	**98.7**	0.0 / 0.0	**56.3**	12.7 / 33.0
$\pi_{0.5}$	**TFP**	**94.7**	0.0 / 0.0	**88.0**	0.0 / 0.0	**37.1**	18.0 / 4.0
(c) TFP: clear memory at every query							
Backbone		Normal	Reset	Normal	Reset	Normal	Reset
DP		**88.0**	0.0	**97.3**	0.0	**55.3**	0.0
$\pi_{0.5}$		**94.0**	0.0	**90.0**	0.0	**47.3**	0.0
(d) TFP: clear memory once							
DP		**88.0**	18.0	**97.3**	40.0	**55.3**	2.3
$\pi_{0.5}$		**94.0**	2.0	**90.0**	26.0	**47.3**	5.0

Single resets occur at the first query at or after 3.0 s (button), 6.0 s (passage), or 0.6 s (interception), measured from episode start.

Each closed-loop intervention keeps the trained weights and physical task schedule fixed and changes either the supplied interval or stored memory. Time-input changes act at policy queries; memory resets occur at the times stated in the table.

Figure 4 complements the success rates in Table 8(b) with median Button and Passage event times under the same clock-scaling conditions. With $\pi_{0.5}$, halving the supplied intervals lengthens button contact in all 50 paired trials, whereas doubling them shortens it in all 50. Passage entry similarly shifts later with halved intervals and earlier with doubled intervals in all 50 paired trials for each backbone. For Diffusion Policy on Button, the 20 observed releases under halved intervals all occur after the permitted hold window; the 36 releases under doubled intervals all occur before it. Trials without a release do not contribute to the reported contact-duration medians.

D.6 Temporal and Motion Information in Memory

For the Diffusion Policy duration readouts in Tables 9 and 10(a), we fit linear ridge probes to frozen full-TFP representations extracted from demonstration histories at a nominal 10 Hz cadence. Targets are continuous hold time during button contact and elapsed time since the passage cue after it disappears. These labels supervise only the probe. The features are the current encoded observation, recurrent memory, their concatenation, or elapsed time since episode start.

Table 9: Event time and motion are accessible in memory. Linear-probe MAE (↓) from frozen Diffusion Policy.

(a) Event duration (s)

Features	Button	Passage
Current observation	0.3336	1.3073
Episode clock	0.6178	0.8334
TFP memory	**0.0140**	**0.0373**

(b) Motion after occlusion (m/s)

Features	Velocity	Speed
Current observation	0.1200	0.0267
First visible frame	0.0279	0.0267
Last visible frame	0.0158	0.0194
TFP memory	**0.0101**	**0.0087**

Button: contact duration; Passage: time since cue. Motion is read at episode time 0.6 s, after occlusion and before robot movement; targets are signed velocity or speed magnitude.

Table 10: Additional linear-readout comparisons using frozen policy representations, supplementing Table 9. MAE and RMSE are reported separately; direction is classification accuracy.

(a) DP: duration MAE (s) ↓

Features	Button	Passage
Observation + memory	**0.0121**	**0.0355**
Episode clock	0.6178	0.8334

(b) $\pi_{0.5}$: time since cue

Features	RMSE (s) ↓
TFP + clock	**0.0685**
GRU + clock	0.0993
History + clock	1.5366
Clock only	1.5653

(c) DP Interception: after occlusion, before movement (0.6 s)

Features	Velocity MAE (m/s) ↓	Speed MAE (m/s) ↓	Direction (%) ↑
Current observation	0.1200	0.0267	50
First visible frame	0.0279	0.0267	**100**
Last visible frame	0.0158	0.0194	**100**
Memory	**0.0101**	**0.0087**	**100**

Each probe uses four causal queries per episode: 128 episodes (512 queries) for probe fitting and 32 episodes (128 queries) for validation. Features are standardized using training statistics, and the ridge penalty is fixed at 1.

Table 10(b) also compares readouts of time since the Passage cue from $\pi_{0.5}$ memory variants, each augmented with elapsed episode time, against the episode clock alone.

Motion information after occlusion. For Interception, we extract representations from frozen Diffusion Policy with TFP at 0.6 seconds, after the target disappears and before the robot moves. Ridge probes fit 210 prefixes from 35 scenes and are tested on 90 prefixes from 15 other scenes; all six direction–speed variants of a scene remain in the same split. Standardization and penalty selection use only fitting scenes, with five-fold grouped cross-validation. Table 10(c) reports signed-velocity MAE, independently fitted speed-magnitude MAE, and direction accuracy. Single-frame controls use encoded images and robot state.

We also predict speed magnitude to distinguish motion information from the directional cue provided by the target's starting side.

Navigation memory visualization. The demonstration in Figure 5 has a cue at 1.7 seconds. Its layout has the readout error for time since the cue closest to the median over the 32 analyzed layouts. Channels 112 and 142 are drawn from the three largest readout contributors on the probe-fitting layouts. Heatmap rows are ordered once by their median time constant over the demonstration.

D.7 Event-Aligned Memory in the Other Simulation Tasks

Figures 6 and 7 complement the Passage visualization in Figure 5. Both use frozen Diffusion Policy with TFP at 10 Hz. Heatmaps show write gains $\mathbf{g}_n = \mathbf{1} - \mathbf{k}_n$ for all 128 channels, ordered once by their median time constant over the displayed sequence, with a fixed $[0, 1]$ color scale. The lower

panels show selected channels of the raw memory state $\mathbf{h}_n$ without normalization or smoothing; the Passage panel shows channel time constants.

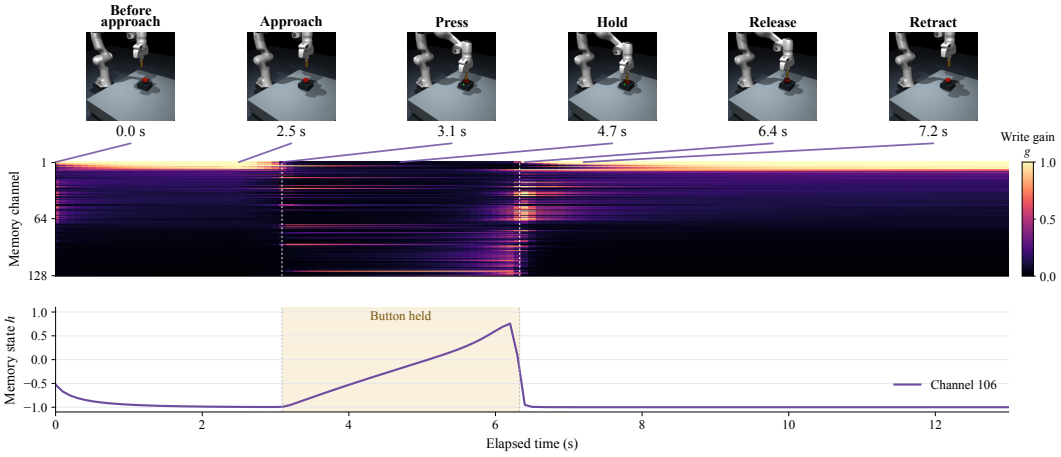


Figure 6: **Memory dynamics during Timed Button Press.** Simulation frames, write gains, and memory channel 106 are aligned in physical time for a successful sequence. Shading marks contact from 3.09 to 6.33 s. The displayed channel rises during the hold and drops around release.

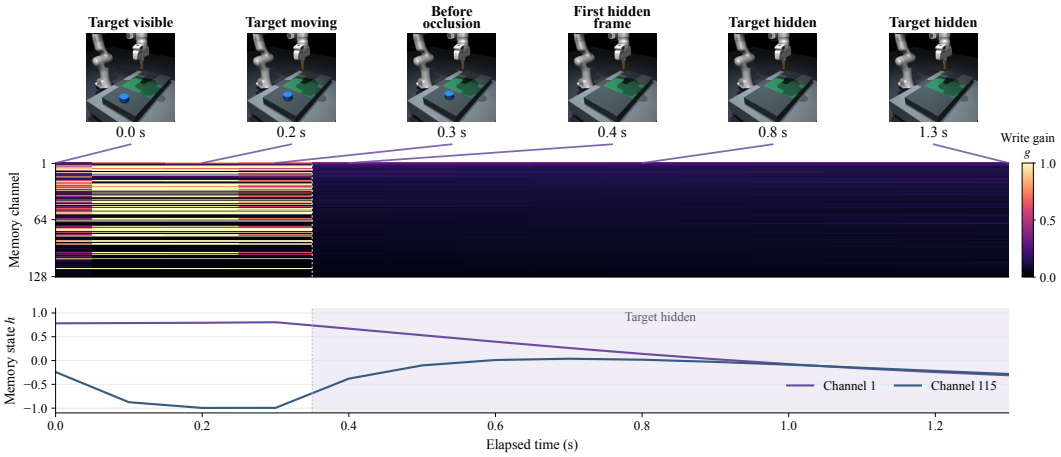


Figure 7: **Memory evolution under occlusion with a stationary robot.** The target moves at 0.16 m/s and becomes invisible to the policy at 0.35 s; shading marks the hidden interval. The robot remains stationary throughout this 1.3 s observation sequence. Channels 1 and 115 evolve after occlusion while the visible observation remains unchanged. The Interception task-evaluation protocol is specified in Appendix D.1.

D.8 Manipulation Memory Ablations

The reproduced $\pi_{0.5}$ manipulation variants in Table 4 share demonstrations, action horizon, optimizer, execution settings, and evaluation protocol.

Training cost. The largest $\pi_{0.5}$ runs used eight-chunk unrolls, batch size 128, over 100 GB GPU memory, and approximately 80 hours on four H200 GPUs. Memory storage scales with the number of active episodes; sequence processing costs depend on the training unroll length and any episode-prefix reconstruction used during evaluation.

D.9 Auxiliary Memory Diagnostics

We summarize memory write-gain changes around manipulation events and action variation under memory replacement.

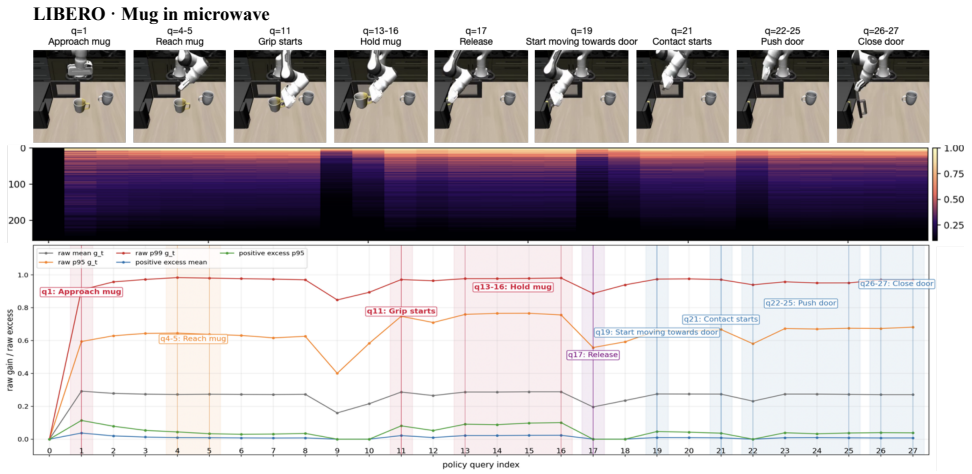


Figure 8: **Memory updates during LIBERO manipulation.** Observations (top), channel-wise write gains (middle), and gain summaries (bottom) show changes around grasping, release, and door interaction in a successful rollout. The horizontal axis indexes policy queries (denoted t in the plot labels). Gains depend on both measured intervals and predicted time constants.

Event-aligned write gains. For the write gains in Figure 8, we aggregate 224 events from 30 successful mug-in-microwave rollouts. Event alignment uses simulator-side gripper states and object poses. With $\mathbf{g}_n = \mathbf{1} - \mathbf{k}_n$, define $\Delta\mathbf{g}_n = \mathbf{g}_n - \mathbf{g}_{n-1}$ and $\bar{g}_n = d_h^{-1} \sum_{j=1}^{d_h} g_{n,j}$. We compare the maximum of $D_n^g = d_h^{-1} \sum_{j=1}^{d_h} |g_{n,j} - g_{n-1,j}|$ within ± 6 queries of each event with a non-event baseline excluding ± 8-query neighborhoods (Table 11).

Table 11: Event-aligned memory write-gain changes in 30 successful rollouts. Best ± 6 selects a local maximum around each event.

Metric	Center/base	Best ± 6/base	Lag
Mean dim. $\lvert\Delta g_{n,j}\rvert$	$0.42\times$	$5.97\times$	-1.64
Top-10% dim. $\lvert\Delta g_{n,j}\rvert$	$0.48\times$	$5.90\times$	-2.13
$\lvert\Delta \bar{g}_n\rvert$	$0.34\times$	$6.24\times$	-1.78
$\lVert\Delta \mathbf{g}_n\rVert_2/\sqrt{d_h}$	$0.35\times$	$6.14\times$	-1.88

Fixed-observation memory replacement. At one LIBERO query, we fix the observation, robot state, and instruction, and substitute memory states from 25 rollout timesteps. Each condition produces a 10-step action chunk. For chunks $\mathbf{A}_i, \mathbf{A}_j \in \mathbb{R}^{H \times d_a}$, where H is the action horizon and d_a the action dimension, we measure $D(i,j) = H^{-1} \sum_{q=1}^{H} \lVert\mathbf{A}_{i,q} - \mathbf{A}_{j,q}\rVert_2$. Across the 300 pairs formed from these 25 conditions, the mean distance is 0.0442 and the maximum is 0.1303 (Table 12). Figure 9 shows the pairwise distances between the predicted action chunks.

The distances summarize the combined variation from memory substitution and action sampling.

Table 12: Action-chunk distances for 25 memory substitutions at a fixed observation.

Metric	Value
Memory states	25
Pairwise comparisons	300
Mean pairwise dist.	0.0442
Median pairwise dist.	0.0363
Max pairwise dist.	0.1303

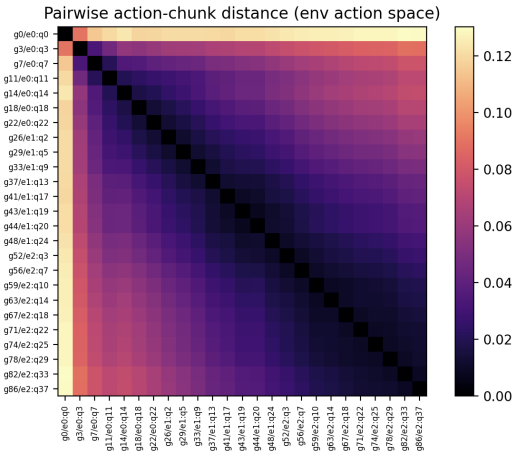


Figure 9: Pairwise distances between predicted action chunks under a fixed observation and different memory states. Rows and columns index the 25 substituted states; color encodes $D(i, j)$.

D.10 Real-World Setup and Evaluation Protocol

We use a single-arm Galaxea A1 with a parallel gripper, a wrist-mounted fisheye camera, and a static Intel RealSense D435 camera.

The real-robot Diffusion Policy experiments use our custom compact visual diffusion-policy implementation described in Appendix D.1, with the same backbone architecture, TFP memory update, and action-conditioning interface as in the controlled simulations.

The baseline for each backbone is its standard reactive policy without persistent policy memory. Each task uses 50 demonstrations for training. Both backbones use fixed 0.1-second query intervals (10 Hz) during training, as in the controlled timing experiments. Each method is evaluated in 20 trials per task under nominal conditions.

Tea-steeping success requires an immersion duration of 3.0 ± 0.5 s, measured from the infuser entering the water to its removal from the water, followed by successful task completion. For staged heating, the preheating interval runs from heater activation to the food contacting the heating surface; the cooking interval runs from that contact to heater deactivation. Both intervals must be 3.0 ± 0.5 s, and the robot must complete the operations in the prescribed order. Event timestamps are annotated in evaluation videos, using the same timing criteria for all methods.

E Statistical Reliability of Rollout Results

E.1 Controlled Timing: Variation Across Training Seeds

Tables 1 and 2 report mean success across three independent training seeds. Additional Diffusion Policy ablations in Table 7 use the same seed-level aggregation and report mean $\pm$ sample SD. Table 13 summarizes task-average success as mean $\pm$ sample standard deviation. For each seed, we first average success equally across 10 Hz, 5 Hz, jitter, and dropout, then compute the mean and standard deviation of the three seed-level averages. Overall success in the main text additionally weights the three tasks equally, irrespective of their different rollout counts.

LTC+Δt has overall success of $19.7 \pm 2.3\%$ for Diffusion Policy and $23.6 \pm 2.4\%$ for $\pi_{0.5}$, averaging tasks equally within each seed. The lower panel of Table 13 gives its results for each observation schedule as mean $\pm$ sample SD and rollout 95% confidence intervals.

Table 13: Variation across training seeds. Task-average success (%), reported as mean ± sample SD over three independent training seeds. The lower panel gives LTC+Δt results by schedule with rollout 95% confidence intervals.

	Diffusion Policy			$\pi_{0.5}$		
Memory	Button	Passage	Intercept.	Button	Passage	Intercept.
Reactive	0.0 ± 0.0	2.3 ± 0.3	10.6 ± 0.1	0.0 ± 0.0	9.3 ± 0.6	18.3 ± 0.3
History	0.0 ± 0.0	2.2 ± 0.3	29.1 ± 0.5	5.5 ± 0.0	2.8 ± 0.6	35.1 ± 0.2
GRU	19.7 ± 1.3	37.8 ± 0.3	25.3 ± 1.5	10.5 ± 1.5	36.8 ± 1.5	27.1 ± 1.5
GRU+Δt	21.0 ± 0.0	37.8 ± 0.3	26.1 ± 0.6	11.7 ± 0.3	37.3 ± 0.3	27.8 ± 1.1
S4D	25.7 ± 2.1	40.5 ± 1.8	24.8 ± 1.5	32.0 ± 2.0	24.7 ± 2.3	18.8 ± 1.5
LTC+Δt	16.0 ± 3.0	20.3 ± 2.5	22.7 ± 1.3	20.2 ± 2.8	24.2 ± 3.1	26.6 ± 1.5
S4D+Δt	48.2 ± 2.1	95.2 ± 0.8	42.8 ± 1.5	93.0 ± 0.9	74.5 ± 0.9	32.8 ± 0.8
TFP fixed Δt	25.5 ± 1.8	32.8 ± 1.0	31.9 ± 1.8	27.3 ± 1.5	35.7 ± 0.3	32.2 ± 1.5
TFP *(ours)*	$\mathbf{81.2 \pm 1.2}$	$\mathbf{97.3 \pm 1.0}$	$\mathbf{53.9 \pm 1.2}$	$\mathbf{93.5 \pm 0.9}$	$\mathbf{86.7 \pm 0.8}$	$\mathbf{39.7 \pm 0.7}$
Memory-structure ablations						
Candidate w/o history	42.7 ± 1.3	8.0 ± 1.0	44.2 ± 0.8	12.0 ± 2.0	65.7 ± 1.5	31.7 ± 0.8
Static time constants	58.5 ± 1.3	82.7 ± 1.3	44.8 ± 0.7	64.3 ± 1.5	13.3 ± 8.3	31.4 ± 0.5
Learned Δt gate	14.5 ± 1.0	15.0 ± 1.0	30.5 ± 0.9	27.0 ± 1.0	37.0 ± 1.0	30.1 ± 0.7

LTC+Δt: results by observation schedule

		Diffusion Policy		$\pi_{0.5}$	
Task	Schedule	Mean ± SD	95% CI	Mean ± SD	95% CI
Button	10 Hz	18.7 ± 3.1	$[13.2, 25.7]$	23.3 ± 3.1	$[17.3, 30.7]$
	5 Hz	13.3 ± 3.1	$[8.8, 19.7]$	18.7 ± 3.1	$[13.2, 25.7]$
	Jitter	20.7 ± 3.1	$[15.0, 27.8]$	22.7 ± 3.1	$[16.7, 30.0]$
	Dropout	11.3 ± 3.1	$[7.2, 17.4]$	16.0 ± 2.0	$[11.0, 22.7]$
Passage	10 Hz	24.7 ± 3.1	$[18.5, 32.1]$	26.7 ± 3.1	$[20.2, 34.3]$
	5 Hz	18.0 ± 2.0	$[12.7, 24.9]$	22.7 ± 3.1	$[16.7, 30.0]$
	Jitter	22.7 ± 3.1	$[16.7, 30.0]$	28.7 ± 3.1	$[22.0, 36.4]$
	Dropout	16.0 ± 2.0	$[11.0, 22.7]$	18.7 ± 3.1	$[13.2, 25.7]$
Interception	10 Hz	24.3 ± 1.5	$[21.6, 27.2]$	29.3 ± 1.5	$[26.5, 32.4]$
	5 Hz	21.1 ± 1.2	$[18.6, 23.9]$	25.3 ± 1.5	$[22.6, 28.3]$
	Jitter	25.4 ± 1.7	$[22.7, 28.4]$	28.3 ± 1.5	$[25.5, 31.4]$
	Dropout	20.0 ± 1.0	$[17.5, 22.7]$	23.3 ± 1.5	$[20.7, 26.2]$

E.2 LIBERO Seed Variability and Rollout Uncertainty

The reproduced LIBERO and LIBERO-plus results use three training seeds. Table 14 reports mean $\pm$ sample SD. For each seed, Avg. weights the four LIBERO suites or three LIBERO-plus perturbation conditions equally; its SD is computed from the three seed-level averages. Statistics are rounded only after aggregation.

For our evaluated variants, each standard LIBERO suite contains 10 tasks with 50 rollouts per task and seed, giving 500 rollouts per seed and 1,500 across three seeds. Table 15 reports suite-level rollout 95% confidence intervals for TFP. These quantify finite-rollout uncertainty in each evaluated policy's success rate. Reproduced variants share the evaluation protocol, while published policies use their reported settings.

Table 14: LIBERO and LIBERO-plus success (%): mean $\pm$ sample SD across three training seeds. Avg. weights suites or perturbation conditions equally within each seed.

LIBERO					
Policy	Spatial	Object	Goal	Long-10	Avg.
GRU	99.0 ± 0.2	98.4 ± 0.2	95.8 ± 0.3	89.1 ± 0.4	95.6 ± 0.1
S4D	98.5 ± 0.1	98.4 ± 0.2	97.0 ± 0.2	92.5 ± 0.3	96.6 ± 0.0
TFP *(ours)*	99.5 ± 0.1	99.0 ± 0.2	99.3 ± 0.1	96.9 ± 0.3	98.7 ± 0.2

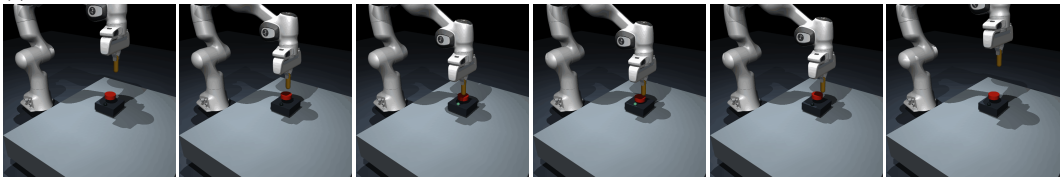
LIBERO-plus				
Policy	Noise	Light	Background	Avg.
Reactive	85.2 ± 0.4	93.9 ± 0.3	95.1 ± 0.2	91.4 ± 0.1
GRU	85.9 ± 0.4	91.5 ± 0.3	96.1 ± 0.3	91.1 ± 0.1
S4D	82.6 ± 0.4	95.0 ± 0.2	92.3 ± 0.3	90.0 ± 0.2
TFP *(ours)*	88.5 ± 0.5	96.1 ± 0.3	96.6 ± 0.2	93.7 ± 0.3

Table 15: TFP success on standard LIBERO (%) with rollout 95% confidence intervals across the three-seed evaluation.

Suite	TFP
Spatial	99.5 $[99.0, 99.8]$
Object	99.0 $[98.4, 99.4]$
Goal	99.3 $[98.8, 99.6]$
Long-10	96.9 $[95.9, 97.7]$

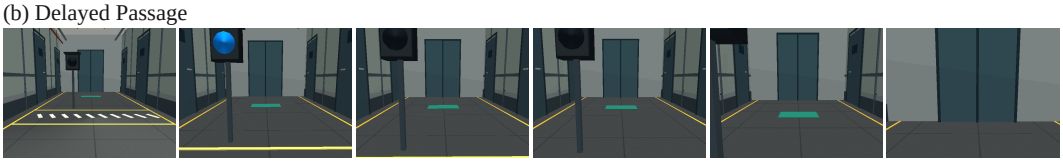
F TASK SEQUENCES

(a) Timed Button Press



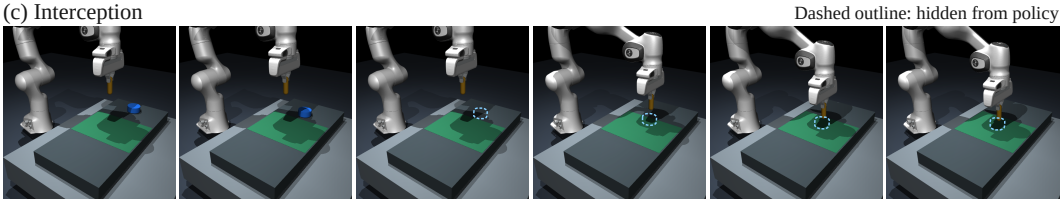
0.00 s 3.80 s 4.25 s 6.00 s 7.50 s 12.00 s

(b) Delayed Passage



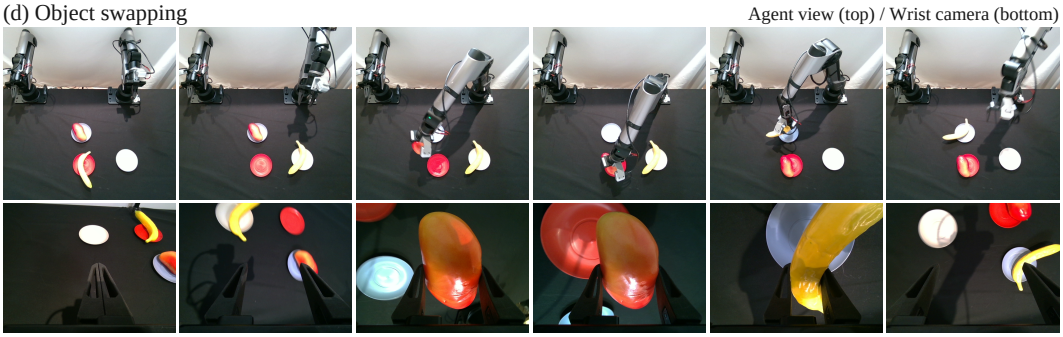
0.00 s 3.45 s 3.95 s 9.05 s 10.45 s 14.87 s

(c) Interception — Dashed outline: hidden from policy



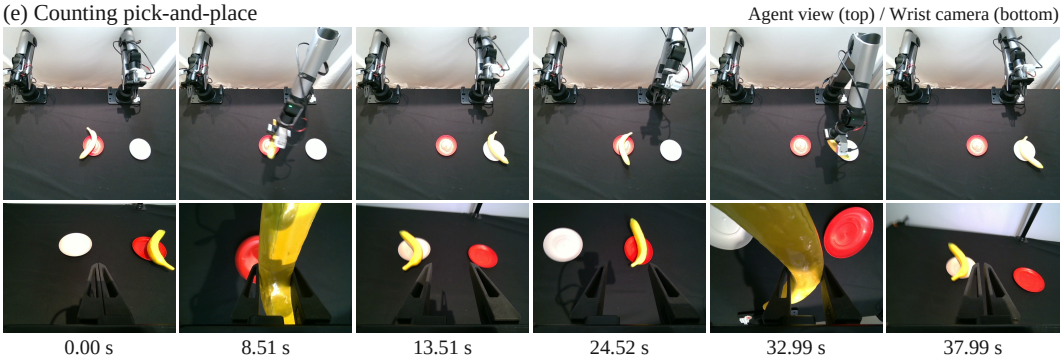
0.00 s 0.50 s 0.60 s 1.30 s 1.65 s 1.772 s

(d) Object swapping — Agent view (top) / Wrist camera (bottom)



0.00 s 15.51 s 24.51 s 26.01 s 40.48 s 42.98 s

(e) Counting pick-and-place — Agent view (top) / Wrist camera (bottom)



0.00 s 8.51 s 13.51 s 24.52 s 32.99 s 37.99 s

Figure 10: Task sequences in simulation (a–c) and real-robot demonstrations (d–h, continued on the next page). Frames progress from left to right; timestamps give elapsed seconds with unequal intervals. In the Interception expert demonstration (c), the target becomes invisible to the policy at 0.55 s and the robot holds until 0.60 s. Dashed outlines show the hidden target's true position for visualization only; they are absent from policy inputs. For real-robot tasks, each column pairs Agent view (top) and Wrist camera (bottom), preserving the model's camera input framing.

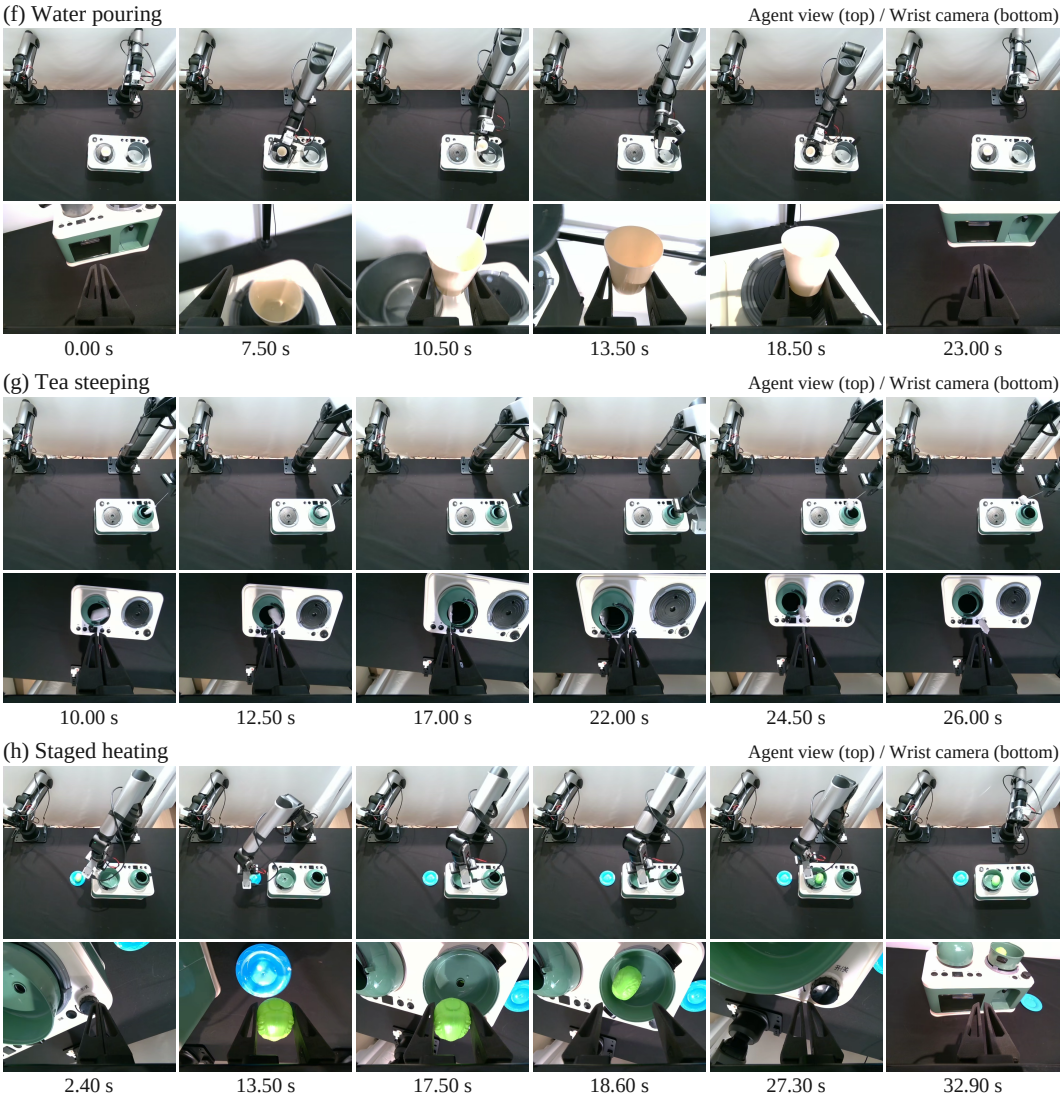


Figure 10: Task sequences (continued). Paired Agent view (top) and Wrist camera (bottom) for Water pouring, Tea steeping, and Staged heating. Columns follow key actions and waiting stages in each demonstration.