PREM: PREDICTION-REGULATED MEMORY FOR ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A robot's next action can depend on recent motion or earlier visual evidence absent from its current observation. A recurrent visual-memory architecture can retain this history in a visual state, but its usefulness depends on what the state represents and how it is updated with new observations. We introduce *Prediction-Regulated Memory (PREM)*, a framework that uses action-conditioned future prediction to guide both memory learning and updating in vision–language–action policies. During training, predicting future visual features provides temporal supervision for memory learning. During execution, PREM compares current visual features with the prediction made at the preceding decision and uses the spatially normalized prediction discrepancy to regulate updates to the visual state at each spatial location. The updated visual state supplies temporal context for action generation alongside current observations. Built on $\pi_{0.5}$, PREM improves average success rate from 9.63% to 20.74% on DOMINO dynamic manipulation tasks and from 14.4% to 83.8% on the RMBench M(1) memory-dependent setting. Further evaluations on LIBERO and LIBERO-Pro show improvements in general manipulation and robustness to spatial perturbations. PREM incurs a mean inference overhead of 7.7 ms on a single NVIDIA H800 GPU.

1 INTRODUCTION

Robot manipulation often takes place under partial observability, where the current observation may not contain all information needed for action selection. Visual observations, language instructions, and proprioceptive readings can be insufficient when relevant motion or task state must be inferred from earlier interaction. The appropriate action can therefore depend on what the robot observed and did previously. This motivates integrating current observations with task-relevant interaction history.

For example, intercepting a moving object requires information about its recent motion, even when the object's current position is visible. Similarly, restoring an earlier arrangement requires remembering where objects were previously located, even when the current scene looks similar to another state. In both cases, similar current observations may require different actions because they correspond to different visual histories. A manipulation policy therefore needs to combine current observations with task-relevant visual history. Figure 1 illustrates these complementary requirements.

Existing approaches provide access to visual history in different ways. Multi-frame models process a window of recent observations (Li et al., 2024; 2026), while memory-based policies store or summarize earlier evidence (Fang et al., 2025; Shi et al., 2026b; Huang et al., 2026). We focus on a recurrent visual-memory architecture that carries a fixed-size visual state across action decisions. However, retaining history alone is not sufficient. The visual state must preserve information relevant to future action generation, while incorporating new observations without discarding useful history. These requirements concern both what the visual state represents and how it is updated during interaction.

Future prediction has been used to learn visual representations through self-supervised objectives (Han et al., 2019; 2020). Recent VLA methods combine future prediction with action generation and memory (Sun et al., 2026; Shi et al., 2026a; Yuan et al., 2026). Related video-understanding work also studies recognizing, localizing, and anticipating unintentional actions (Epstein et al., 2020). Future prediction provides a natural way to address both requirements. During training, predicting subsequent visual features encourages the visual state to retain information useful for anticipating

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.

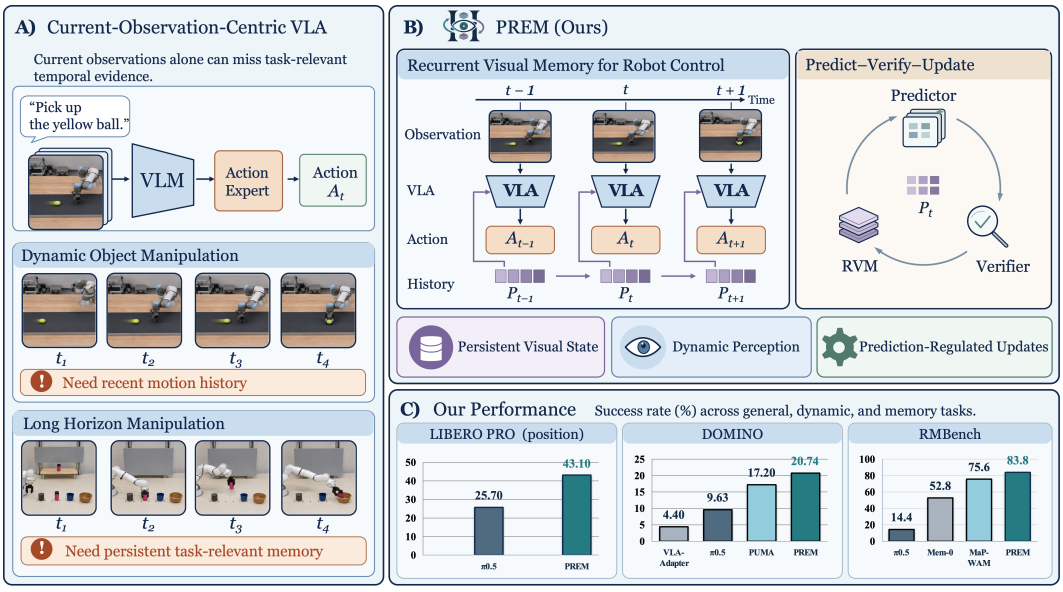


Figure 1: **Motivation and overview of PREM. (A)** Current observations alone can miss recent motion and earlier task-relevant visual evidence. **(B)** PREM uses future prediction to supervise memory learning and prediction discrepancy to regulate visual-state updating. **(C)** PREM improves general manipulation, dynamic manipulation, and memory-dependent manipulation.

interaction dynamics. The prediction targets come directly from future observations, providing temporal supervision without additional annotations. During execution, the discrepancy between predicted and observed visual features can reveal where the current visual state is insufficient. This motivates using action-conditioned future prediction not only to supervise memory learning, but also to guide visual-state updating.

This leads to our central question: *how can action-conditioned future prediction guide both memory learning and visual-state updating for robot manipulation?*

We introduce *Prediction-Regulated Memory (PREM)*, a framework that uses action-conditioned future prediction to guide both memory learning and visual-state updating in VLA policies. PREM couples a recurrent visual-memory (RVM) architecture with an action-conditioned predictor. The RVM architecture maintains a fixed-size visual state across action decisions. The policy, RVM architecture, and predictor are jointly trained with action and prediction objectives. The prediction objective provides temporal supervision for memory learning, while the discrepancy between prediction and observation provides feedback for visual-state updating.

At each replanning decision, PREM compares the current visual features with the prediction made at the preceding decision. The spatially normalized prediction discrepancy regulates how much of the candidate visual-state update is incorporated at each spatial location. The updated visual state is then used as temporal context for action generation together with the current observation. The policy's latent action representation is also provided to the predictor, which predicts the visual features of the next replanning observation.

During training, the prediction objective supervises memory learning through the predictor's visual-state input. During execution, the prediction is cached until the corresponding observation arrives and then provides the reference for the next visual-state update. In this way, PREM uses the same future-prediction mechanism to learn the visual state and to regulate how it changes over time.

Our evaluation follows the two temporal requirements in Figure 1. We test dynamic manipulation on DOMINO and memory-dependent manipulation on RMBench M(1). Built on $\pi_{0.5}$, PREM improves average success rate from 9.63% to 20.74% on DOMINO and from 14.4% to 83.8% on RMBench M(1). Component ablations on DOMINO show performance gains from both predictive supervision for memory learning and prediction-regulated visual-state updating. Additional evaluations on

LIBERO and LIBERO-Pro show improvements in general manipulation and robustness to spatial perturbations. PREM incurs a mean inference overhead of 7.7 ms on a single NVIDIA H800 GPU.

2 METHOD

PREM couples an action-conditioned predictor with a recurrent visual-memory (RVM) architecture. Future prediction serves two complementary roles: its training objective supervises memory learning, while its discrepancy from subsequent observations provides feedback for visual-state updating. At each replanning decision, PREM receives the current observation, verifies the prediction cached at the preceding decision, updates the visual state, generates actions, and predicts the next visual observation. Figure 2 summarizes this prediction–verification–update cycle.

2.1 PROBLEM FORMULATION

We consider language-conditioned robot manipulation under partial observability. At each replanning decision t, the robot receives visual observations O_t, robot proprioception s_t, and a language instruction ℓ. The current observation may not contain recent motion or earlier task-relevant visual evidence. The base VLA policy predicts an action chunk A_t, a sequence of robot control commands:

$$A_t \sim \pi_{\text{base}}(\cdot \mid O_t, s_t, \ell). \tag{1}$$

The robot executes a selected prefix of A_t, then obtains a new observation and replans at $t+1$. Thus, t indexes replanning decisions rather than individual control steps. PREM retains the relevant history in a fixed-size visual state. At decision t, it (1) extracts current visual features, (2) compares them with the prediction cached at the preceding decision, (3) updates the visual state, (4) uses the updated state for action generation, and (5) predicts the visual features at the next decision.

Temporal context P_t derived from the updated visual state augments the original policy inputs:

$$A_t \sim \pi_\theta(\cdot \mid O_t, s_t, \ell, P_t). \tag{2}$$

The following sections describe the prediction discrepancy, the visual-state update, the action-conditioned predictor, and the joint training objective.

2.2 PREDICTION DISCREPANCY

Prediction discrepancy supplies spatial feedback for visual-state updating. PREM compares each new observation with the prediction cached at the preceding decision. The comparison operates on visual features:

$$X_t = \mathcal{F}_{\text{vis}}(O_t) \in \mathbb{R}^{N \times d}, \tag{3}$$

where $\mathcal{F}_{\text{vis}}$ comprises the policy's image encoder and a learned projection. N is the number of spatial patches, and d is the feature dimension.

Let $\hat{X}_t$ denote the cached prediction of X_t. Both have the same spatial layout, so PREM compares them at each spatial location:

$$e_{t,i} = \delta(\hat{X}_{t,i}, X_{t,i}), \qquad i = 1, \ldots, N. \tag{4}$$

The function δ combines cosine distance and a mean elementwise Smooth-L1 penalty after token-wise feature normalization. Its complete definition is given in Appendix B. No external motion estimator is required.

We use relative spatial discrepancy to regulate updates at different locations. The spatial median is $\mu_t = \text{median}_j\, e_{t,j}$, and the median absolute deviation is $\rho_t = \text{median}_j\, |e_{t,j} - \mu_t|$. The normalized prediction discrepancy is

$$\xi_{t,i} = \frac{e_{t,i} - \mu_t}{\max(\rho_t, \epsilon_m)}, \tag{5}$$

where $\epsilon_m > 0$ lower-bounds the denominator. Larger values identify locations with greater discrepancy relative to the rest of the observation. This is a spatial update signal, not an absolute measure of frame-level unpredictability. A uniform additive shift in the discrepancies leaves it unchanged.

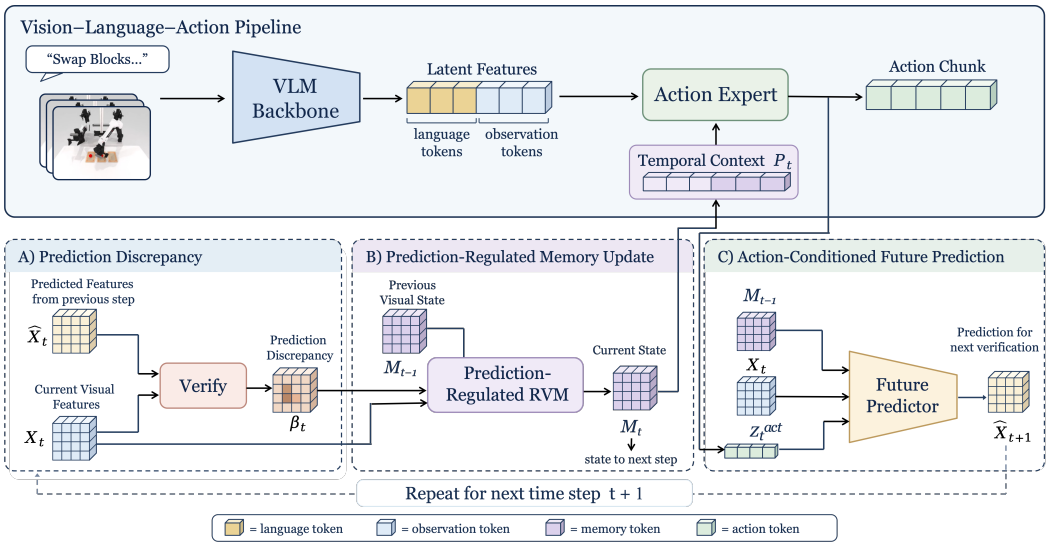


Figure 2: **PREM architecture.** **(A)** PREM compares the cached prediction with observed visual features at each spatial location to compute prediction discrepancy. **(B)** The spatially normalized discrepancy regulates how much of the candidate visual-state update is incorporated. **(C)** An action-conditioned predictor predicts the next visual features from the incoming visual state, observed visual features, and latent action representation. The prediction objective supervises memory learning. The cached prediction provides a reference for subsequent visual-state updating.

2.3 Prediction-Regulated Visual-State Update

Visual-state updating separates the candidate content from how much of that candidate is incorporated. The RVM architecture proposes the candidate visual state, while prediction discrepancy regulates the amount of the proposed change.

Let $M_{t-1} \in \mathbb{R}^{N \times d}$ denote the incoming visual state. The RVM architecture is adapted from recurrent video representations (Zoran et al., 2026). Its gated network R_ϕ combines M_{t-1} with X_t to produce a candidate visual state $\widetilde{M}_t = R_\phi(X_t, M_{t-1})$.

Visual changes poorly predicted from the available history and action information may mark key interaction transitions. This motivates greater incorporation of the corresponding visual evidence. PREM implements this intuition with an update coefficient at each spatial location:

$$\beta_{t,i} = \beta_{\min} + (1 - \beta_{\min})\sigma\left(\frac{\xi_{t,i} - \tau}{\gamma}\right), \tag{6}$$

where σ is the sigmoid function. The threshold τ and transition scale $\gamma > 0$ control the response to relative discrepancy. The lower bound $0 < \beta_{\min} < 1$ preserves a nonzero amount of the candidate update. The resulting visual state is

$$M_t = M_{t-1} + \beta_t \odot (\widetilde{M}_t - M_{t-1}), \tag{7}$$

where $\odot$ denotes elementwise multiplication and β_t is broadcast across the feature dimension.

Larger relative discrepancies incorporate more of the candidate update; smaller ones yield more conservative updates. The coefficient controls the amount of the proposed update, not the absolute change in the visual state, which also depends on $\widetilde{M}_t - M_{t-1}$. This modulation supplements the RVM architecture's learned gates and requires no keyframe annotations. At the first valid observation after a reset, we initialize $M_0 = X_0$ and skip discrepancy-based regulation because no preceding prediction is available.

2.4 ACTION-CONDITIONED FUTURE PREDICTION

The predictor connects memory learning with visual-state updating. It predicts visual features at the next replanning decision from historical, visual, and action information. The action information comes from the policy's own action-generation computation.

Action conditioning. The updated visual state first supplies temporal context for action generation. Let $\mathcal{P}_{\mathrm{mem}}$ denote the projection used to form temporal context at the policy token width:

$$P_t = \mathcal{P}_{\mathrm{mem}}(M_t). \tag{8}$$

The policy uses P_t through Eq. (2). Historical evidence supplements the original observation pathway rather than replacing it. PREM reuses the final-layer action-token features from the action expert, the policy's action-generation module. Temporal encoding and pooling yield the latent action representation z_t^{act}, without another action-expert evaluation.

Future prediction. The incoming visual state M_{t-1} provides historical information, X_t provides observed visual features, and z_t^{act} provides action information. The predictor combines these inputs:

$$\hat{X}_{t+1} = D_\psi(M_{t-1}, X_t, z_t^{\mathrm{act}}). \tag{9}$$

Here, D_ψ is a query-based Transformer decoder. Its input streams are projected to a common width and concatenated along the token dimension. Learned spatial queries aggregate these inputs through cross-attention and exchange information through self-attention. The output shares the patch layout of X_t. Action tokens are read through attention, not matched one-to-one with visual patches.

The predictor directly receives the incoming state M_{t-1} rather than the updated state M_t. This keeps the prediction target future-looking instead of using a state that already incorporates the current observation. Its action input is computed after M_t conditions the policy through P_t, so all inputs are available at decision t. During training, supervision from X_{t+1} propagates through the predictor into the RVM architecture. This encourages the visual state to retain information useful for predicting subsequent visual features. During execution, PREM caches $\hat{X}_{t+1}$ until O_{t+1} arrives and then uses the prediction in the comparison of Eq. (4). The prediction thus supports memory learning and subsequent update feedback; it is not supplied to the action expert as imagined future context.

2.5 TRAINING AND CONFIGURATION

We initialize from $\pi_{0.5}$ (Physical Intelligence et al., 2025) and jointly fine-tune the policy, RVM architecture, and predictor. The action objective supervises action generation. The prediction objective supervises memory learning through the predictor's incoming-visual-state branch.

The target X_{t+1} comes from the next replanning observation via Eq. (3). Let v_t indicate a valid prediction–target pair, and sample $q_{t,i} \sim \mathrm{Bernoulli}(0.75)$ to select supervised patches. The prediction loss is

$$\mathcal{L}_{\mathrm{pred}} = \frac{\sum_{t,i} v_t q_{t,i}\, \delta\Big(\hat{X}_{t+1,i}, \mathrm{sg}(X_{t+1,i})\Big)}{\sum_{t,i} v_t q_{t,i}}, \tag{10}$$

where sg denotes stop-gradient. Unlike update regulation, this objective uses δ before spatial normalization. The mask selects target patches for loss computation, not predictor inputs. Targets are detached after projection; no separate or EMA target encoder is used. The complete objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{act}} + \lambda_{\mathrm{pred}} \mathcal{L}_{\mathrm{pred}}, \tag{11}$$

where $\mathcal{L}_{\mathrm{act}}$ is the base policy's action objective and $\lambda_{\mathrm{pred}} = 0.1$.

For update regulation, $e_{t,i}$ is detached before spatial normalization. The update coefficients therefore provide no gradient path through the prediction discrepancy. Prediction learning remains differentiable through the predictor's incoming visual state. Training causally unrolls the recurrent pathway over observation clips. Gradients propagate throughout each clip, and recurrent computation resets between clips without a burn-in prefix.

The update parameters are fixed at $\beta_{\min} = 0.25$, $\tau = 2.0$, and $\gamma = 0.5$. During training, the effective coefficient starts at one and linearly transitions to $\beta_{t,i}$ in Eq. (6). Inference uses the full prediction-regulated coefficient. Appendix B provides architectural specifications, action-feature extraction, discrepancy constants, and benchmark-specific ramp, unroll, stride, and reset details.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Benchmarks and baselines. We evaluate PREM on dynamic manipulation, memory-dependent manipulation, and general manipulation. We also assess robustness to spatial perturbations. LIBERO (Liu et al., 2023) comprises four suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10,covering spatial, object-centric, goal-conditioned, and compositional manipulation. LIBERO-Pro (Zhou et al., 2025) evaluates robustness under position-perturbed layouts and initial states. Baselines include OpenVLA (Kim et al., 2024), SpatialVLA (Qu et al., 2025), π_0 (Black et al., 2024), π_0-FAST (Pertsch et al., 2025), GR00T-N1 (Bjorck et al., 2025), X-VLA (Zheng et al., 2026), PAM (Hu et al., 2026), MemoryVLA (Shi et al., 2026b), MemoryVLA++, CronusVLA (Li et al., 2026), and ChainVLA (Huang et al., 2026).

Table 1: Success rates on standard LIBERO (%).

Policy	Spatial	Object	Goal	LIBERO-10	Avg.
OpenVLA	84.7	88.4	79.2	53.7	76.5
SpatialVLA	88.2	89.9	78.6	55.5	78.1
π_0-FAST	96.4	96.8	88.6	60.2	85.5
GR00T-N1	94.4	97.6	93.0	90.6	93.9
π_0	96.8	98.8	95.8	85.2	94.2
PAM	86.1	97.0	94.5	84.7	90.6
MemoryVLA	98.4	98.4	96.4	93.4	96.7
MemoryVLA++	**99.8**	**100.0**	98.2	96.0	98.5
CronusVLA	97.3	<u>99.6</u>	96.9	94.0	97.0
X-VLA	98.2	98.6	97.8	<u>97.6</u>	98.1
ChainVLA	98.4	99.4	<u>99.2</u>	**98.2**	**98.8**
$\pi_{0.5}$ (Baseline)	98.8	98.2	98.0	92.4	96.9
PREM (Ours)	<u>99.2</u> (+0.4)	99.0 (+0.8)	**99.4** (+1.4)	97.2 (+4.8)	<u>98.7</u> (+1.8)

DOMINO (Fang et al., 2026) contains 35 dynamic-interception and dynamic-tracking tasks. Object and interaction states evolve continuously in these tasks. We compare with π_0, OpenVLA-OFT (Kim et al., 2025), $\pi_{0.5}$, and PUMA. RMBench (Chen et al., 2026) evaluates memory-dependent manipulation across long temporal gaps. Baselines include DP (Chi et al., 2025), ACT (Zhao et al., 2023), X-VLA, MemER (Sridhar et al., 2026), MemoryVLA, Mem-0, ChainVLA, and MaP-WAM (Zhao et al., 2026). We report episode success rate. Bold and underlined values denote the best and second-best results. Green values indicate PREM's percentage-point gains over $\pi_{0.5}$.

Training and implementation protocol. We initialize PREM from the official pretrained $\pi_{0.5}$ checkpoint. We fine-tune the complete policy end to end using $8\times$ NVIDIA H800 80GB GPUs. The VLM backbone and original action-generation pathway are retained. At episode boundaries, we reset the visual state and clear cached predictions.

Training configurations are benchmark-specific. LIBERO uses $1{,}693$ demonstrations, a global batch size of 128, and 30k steps. The five RMBench tasks are trained independently. Each task uses 50 demonstrations, a global batch size of 32, and 20k steps. DOMINO uses 1,750 episodes across 35 tasks, a global batch size of 64, and 100k steps. For all benchmarks, the learning rate is cosine-annealed from 2×10^{-5} to 1×10^{-5}.

For latency evaluation, we measure all policies on a single NVIDIA H800 80GB GPU with batch size 1. All policies receive the same LIBERO input: base- and wrist-camera images, a language instruction, and 8-dimensional robot proprioception. We perform 10 warm-up runs and report the mean over 100 consecutive runs. Timing starts when the observation is received and ends when the complete action chunk is available.

3.2 Main Results

General manipulation. Table 1 reports results on standard LIBERO. The $\pi_{0.5}$ base policy achieves an average success rate of 96.9%. PREM increases this to 98.7%. The largest gain occurs on LIBERO-10, from 92.4% to 97.2%, an improvement of 4.8 percentage points.

PREM also improves Spatial, Object, and Goal by 0.4, 0.8, and 1.4 percentage points, respectively. These results show improvements across all four suites and support the effectiveness of PREM on general manipulation.

Robustness to spatial perturbations. Figure 3 reports results under LIBERO-Pro position perturbations. PREM improves average success rate from 25.7% to 43.1%, with gains across all four suites. The largest improvement occurs on Object, from 16.2% to 67.4%, a gain of 51.2 percentage points.

These results show improved robustness to position perturbations. DOMINO and RMBench M(1) provide more targeted evaluations of dynamic manipulation and memory-dependent manipulation, respectively.

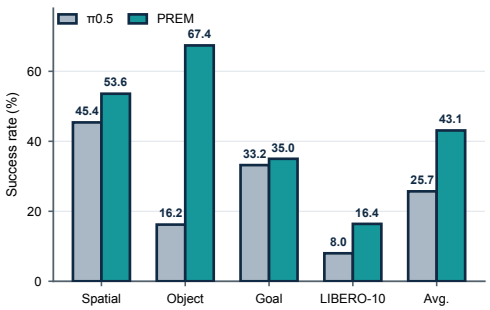


Figure 3: LIBERO-Pro success rates.

Dynamic manipulation. DOMINO evaluates manipulation under continuously evolving object and interaction states. Table 2 reports an average success rate of 20.74% for PREM. This exceeds $\pi_{0.5}$ by 11.11 percentage points and PUMA by 3.54 percentage points. PREM improves 31 of the 35 tasks over $\pi_{0.5}$. Examples include Adjust Bottle ($52\% \rightarrow 74\%$) and Dump Bin Bigbin ($0\% \rightarrow 37\%$). These results show improved performance on dynamic manipulation. Complete results for all 35 tasks are reported in the appendix.

Memory-dependent manipulation. RMBench M(1) requires earlier task-relevant visual evidence. This evidence may no longer be visible in current observations. Table 3 reports an average success rate of 83.8% for PREM. This exceeds $\pi_{0.5}$ and MaP-WAM by 69.4 and 8.2 percentage points, respectively. PREM achieves 100%, 100%, 98%, and 99% on Rearrange Blocks, Put Back Block, Swap Blocks, and Swap-T. These results support the effectiveness of the complete PREM framework on memory-dependent manipulation.

Table 2: Representative DOMINO task success rates (%).

Task	π_0	OpenVLA-OFT	$\pi_{0.5}$	PUMA	PREM
Dynamic Interception					
Adjust Bottle	19	44	52	65	**74** (+22)
Dump Bin Bigbin	0	0	0	0	**37** (+37)
Handover Block	0	0	1	17	**20** (+19)
Hanging Mug	9	0	5	9	**25** (+20)
Place Object Basket	16	6	11	13	**23** (+12)
Put Bottles Dustbin	20	**43**	17	23	27 (+10)
Dynamic Tracking					
Beat Block Hammer	10	9	10	15	**23** (+13)
Move Stapler Pad	1	0	0	1	**3** (+3)
Place Bread Basket	6	7	8	12	**23** (+15)
Place Container Plate	15	8	22	26	**44** (+22)
Place Mouse Pad	4	1	5	2	**10** (+5)
Place Shoe	20	6	25	16	**52** (+27)
35-task Average	8.17	10.86	9.63	17.20	**20.74** (+11.11)

Table 3: Success rates on the RMBench M(1) setting (%).

Policy	Obs.	Rearr.	Put B.	Swap B.	Swap T.	Avg.
DP	1	0	0	11	20	6.4
ACT	1	29	0	2	2	6.8
X-VLA	9	13	18	16	3	11.8
MemER	7	17	0	14	7	9.0
MemoryVLA	0	22	50	17	9	19.6
Mem-0	4	89	90	67	14	52.8
ChainVLA	11	93	96	74	40	62.8
MaP-WAM	19	66	**100**	97	96	75.6
$\pi_{0.5}$ (Baseline)	9	13	11	24	15	14.4
PREM (Ours)	**22** (+13)	**100** (+87)	**100** (+89)	**98** (+74)	**99** (+84)	**83.8** (+69.4)

Inference efficiency. Table 4 reports a mean end-to-end inference latency of 51.5 ms for PREM and 43.8 ms for $\pi_{0.5}$. PREM therefore incurs a mean inference overhead of 7.7 ms on a single NVIDIA H800 GPU. This measurement includes visual-state updating, prediction verification, and action-conditioned future prediction.

3.3 Ablation Studies

Component ablations. We conduct an incremental component ablation on DOMINO. Table 5 examines the RVM architecture, predictive supervision for memory learning, and prediction-regulated visual-state updating.

Future Prediction denotes training with the action-conditioned future-prediction objective. It provides temporal supervision for memory learning. The predictor uses the incoming visual state, visual features, and the action expert's latent action representation. *Regulated Update* denotes the use of normalized prediction discrepancy to regulate visual-state updating.

Adding the RVM architecture raises success rate from 9.63% to 15.43%, a gain of 5.80 percentage points. Predictive supervision then raises success rate to 18.31%, a gain of 2.88 percentage points. With this supervision retained, prediction-regulated visual-state updating raises success rate to 20.74%, a gain of 2.43 percentage points. Both predictive supervision for memory learning and prediction-regulated visual-state updating improve performance in this incremental configuration.

Table 4: End-to-end mean inference latency on one NVIDIA H800 80GB GPU with batch size 1.

Model	Mean latency (ms)
$\pi_{0.5}$	43.8
PREM (Ours)	51.5
PUMA	57.4
Mem-0	96.6
MemoryVLA	133.8

Table 5: Component ablation on DOMINO. Future Pred. denotes predictive supervision for memory. Regulated Update denotes feedback for visual-state updating.

Variant	RVM	Future Pred.	Regulated Update	DOMINO SR (%)
$\pi_{0.5}$				9.63
+ RVM	✓			15.43
+ Future Prediction	✓	✓		18.31
PREM	✓	✓	✓	**20.74**

Mechanism analysis. *Prediction discrepancy and visual-state updating.* Figure 4 visualizes a successful LIBERO rollout. The curve summarizes the temporal variation of prediction discrepancy over the interaction.. Larger values indicate moments with more substantial visual changes, such as grasping, object motion, occlusion changes, or placement, while smaller values indicate more stable interaction. The orange markers and vertical lines correspond to the six observations shown above. The spatial responses concentrate around the gripper and manipulated object near bowl grasp (**c**) and bowl placement (**f**), consistent with using prediction discrepancy to guide visual-state updating.

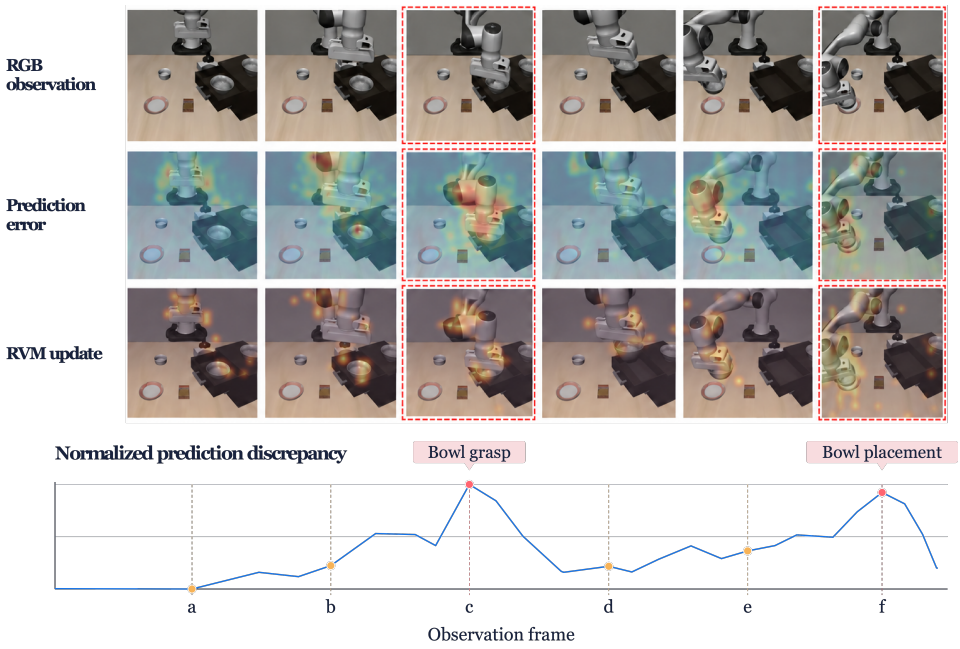


Figure 4: **Prediction discrepancy and visual-state updating.** A temporal prediction-discrepancy curve and visual-state update maps are shown for a successful LIBERO rollout. Spatial responses concentrate around the gripper and manipulated object near bowl grasp (**c**) and bowl placement (**f**).

4 Conclusion

We presented PREM, a framework that uses action-conditioned future prediction to guide both memory learning and visual-state updating for robot manipulation. Predictive supervision encourages the visual state to retain historical information useful for predicting subsequent visual features. Prediction discrepancy provides spatial feedback for visual-state updating. The updated visual state supplies temporal context for action generation alongside current observations. Built on $\pi_{0.5}$, PREM achieves average success rates of 20.74% on DOMINO and 83.8% on RMBench M(1). It also improves general manipulation on LIBERO and robustness to spatial perturbations on LIBERO-Pro. PREM incurs a mean inference overhead of $7.7\,\mathrm{ms}$ on a single NVIDIA H800 GPU. These results support the joint use of predictive supervision and prediction-discrepancy feedback for temporal modeling in VLA policies. Together, these results show that future prediction can both supervise visual-memory learning and regulate visual-state updating in recurrent VLA policies. PREM requires no keyframe or event annotations; its update signal comes directly from future observations.

AI USE STATEMENT

In this work, generative AI tools were used to edit human-written text for clarity and readability, assist with LaTeX typesetting and page layout, and adjust the placement and sizing of existing figures and tables. They were not used to generate or modify experimental data or scientific-figure content, generate synthetic data, formulate the research hypotheses or mathematical claims, develop the theoretical or conceptual framework, design the methodology or experiments, implement the proposed method, clean or reformat datasets, perform qualitative data analysis, or interpret the experimental results. The authors manually reviewed and verified all AI-assisted text, figures, numerical values, citations, and formatting against the underlying implementation and experimental records. The authors take full responsibility for the final content of the paper and all artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work follows the principles of the ICLR Code of Ethics. We aim to conduct and report our research with scientific integrity, transparency, and reproducibility. Experimental results are reported without fabrication, falsification, or intentional misrepresentation, and sufficient implementation and evaluation details are provided to facilitate verification and reproduction. We appropriately acknowledge prior work and use existing benchmarks, datasets, and associated resources in accordance with their intended research purposes and applicable licenses. This study does not involve human participants or the collection of personal information.

REPRODUCIBILITY STATEMENT

We support reproducibility by documenting the model architecture, prediction-regulated update rule, future-prediction objective, and training formulation in Section 2. Section 3 specifies the benchmarks, baselines, training-data sizes, optimization schedule, evaluation metrics, and latency protocol. The appendix provides implementation-level architectural dimensions and hyperparameters (Appendix B), benchmark-specific training and evaluation procedures (Appendices B.4– B.6), evaluation seeds where applicable, and complete per-task DOMINO results (Appendix C). Code and configuration files will be included in the supplementary material where permitted to enable independent reproduction of the reported experiments.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Zichen J Cui, Hengkai Pan, Aadhithya Iyer, Siddhant Haldar, and Lerrel Pinto. Dynamo: In-domain dynamics pretraining for visuo-motor control. *Advances in Neural Information Processing Systems*, 37:33933–33961, 2024.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Dave Epstein, Boyuan Chen, and Carl Vondrick. Oops! predicting unintentional action in video. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 916–926. IEEE, 2020.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. In *European Conference on Computer Vision*, pp. 314–333. Springer, 2026.

Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 1483–1492. IEEE, 2019.

Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *European conference on computer vision*, pp. 312–329. Springer, 2020.

Qingda Hu, Ziheng Qiu, Zijun Xu, Kaizhao Zhang, Xizhou Bu, Zuolei Sun, Bo Zhang, Jieru Zhao, Zhongxue Gan, and Wenchao Ding. Resolving state ambiguity in robot manipulation via adaptive working memory recoding. *IEEE Robotics and Automation Letters*, 2026.

Yuzhi Huang, Weijue Bu, Ziyi Xiong, Jie Wu, Fanding Huang, Jingyan Jiang, and Zhi Wang. Chainvla: Chaining vision-language-action queries through a unified execution state for long-horizon manipulation. *arXiv preprint arXiv:2608.02326*, 2026.

Tianchong Jiang, Xiangshan Tan, Samuel Wheeler, Luzhe Sun, Tewodros W Ayalew, and Matthew Walter. What are we actually benchmarking in robot manipulation? *arXiv preprint arXiv:2606.04233*, 2026.

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. *arXiv preprint arXiv:2502.19645*, 2025.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, et al. Towards efficient and robust manipulation via multi-frame vision-language-action modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18388–18396, 2026.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, volume 2024, pp. 26703–26721, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Michael S Ryoo, Keerthana Gopalakrishnan, Kumara Kahatapitiya, Ted Xiao, Kanishka Rao, Austin Stone, Yao Lu, Julian Ibarz, and Anurag Arnab. Token turing machines. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19070–19081. IEEE, 2023.

Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. Memoryvla++: Temporal modeling via memory and imagination in vision-language-action models. *arXiv preprint arXiv:2606.09827*, 2026a.

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*, volume 2026, pp. 18567–18602, 2026b.

Woo Chul Shin, Zhenyang Chen, Alfred Cueva, Nadun Ranawaka Arachchige, Yingyan Celine Lin, Benjamin Joffe, Shreyas Kousik, and Danfei Xu. Static in, dynamic out: Counterfactual action augmentation for moving object manipulation. *arXiv preprint arXiv:2607.27890*, 2026.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, volume 2026, pp. 97142–97166, 2026.

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. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13577–13587. IEEE, 2022.

Haoning Wu, Xiao Huang, Yaohui Chen, Ya Zhang, Yanfeng Wang, and Weidi Xie. Spatialscore: Towards comprehensive evaluation for spatial intelligence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31029–31041, 2026a.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pp. 10641–10662, 2024.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026b.

Haozhe Xie, Beichen Wen, Jiarui Zheng, Zhaoxi Chen, Fangzhou Hong, Haiwen Diao, and Ziwei Liu. Dynamicvla: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. Eventvla: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026a.

Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis Brown, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, et al. Cambrian-s: Towards spatial supersensing in video. In *International Conference on Learning Representations*, volume 2026, pp. 78185–78225, 2026b.

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

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.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *Advances in Neural Information Processing Systems*, 38: 24195–24228, 2026a.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. Imagewam: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026b.

Sizhe Zhao, Haozhe Xie, Weiyu Zhao, Chenchu Zhang, Huan Wang, Chenyang Wang, Qinglin Liu, and Shengping Zhang. Memory as plans: World-action modeling with memory-grounded planning. *arXiv preprint arXiv:2609.11561*, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 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.

Daniel Zoran, Nikhil Parthasarathy, Yi Yang, Drew A Hudson, João Carreira, and Andrew Zisserman. Recurrent video masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17744–17755, 2026.

A Additional Related Work

A.1 Temporal Memory for Robotic Manipulation

Generalist robot learning provides the broader setting for temporal modeling. RT-1 studies scalable robot control from diverse demonstrations, while RT-2 transfers vision–language pretraining to action generation (Brohan et al., 2022; 2023). PaLM-E integrates continuous sensory inputs with language models for embodied reasoning and planning (Driess et al., 2023). Open X-Embodiment aggregates robot data across platforms (O'Neill et al., 2024), while Octo and LingBot-VLA develop generalist policies using diverse robot datasets (Team et al., 2024; Wu et al., 2026b). These advances provide a foundation for general-purpose manipulation; our focus is how visual evidence is retained and updated during interaction.

Temporal manipulation policies differ primarily in how they retain history. Short-window methods repeatedly encode recent observations: RoboFlamingo and CronusVLA aggregate multiple frames, while DynamicVLA, PUMA, and Static In, Dynamic Out target moving-object manipulation through continuous inference, optical-flow-conditioned prediction, or future-pose estimation (Li et al., 2024; 2026; Xie et al., 2026; Fang et al., 2026; Shin et al., 2026). These approaches provide useful local motion cues, but the retained history remains bounded by the chosen context window.

Longer-range methods instead maintain explicit memory. MemoryVLA and SAM2Act+ use visual or perceptual-cognitive memory banks, MemoryVLA++ combines memory with imagined future latents, and ChainVLA propagates an execution state while retaining sparse event memory (Shi et al., 2026b; Fang et al., 2025; Shi et al., 2026a; Huang et al., 2026). EventVLA writes observations according to a predicted keyframe schedule (Yang et al., 2026a). Such designs can retain evidence over long intervals, but require explicit storage, consolidation, or retrieval policies.

PREM follows a third design: it compresses the observation stream online into a fixed-size recurrent visual state. Fixed-size memories have previously been used for long-video recognition and action prediction (Wu et al., 2022; Ryoo et al., 2023), and our recurrent core is motivated by Recurrent Video Masked Autoencoders (Zoran et al., 2026). PREM adapts this spatial carry to closed-loop manipulation and couples it with action-conditioned prediction and delayed prediction verification. Thus, its memory size does not grow with episode length, and prediction discrepancy regulates visual-state updating at each spatial location.

A.2 Predictive Representations and World Models

Future prediction is commonly used either as a training objective or as additional inference-time context. GR-1 connects video prediction with robot control by combining large-scale video generative pretraining with joint prediction of future images and robot actions (Wu et al., 2024). PUMA, Fast-WAM, DynaMo, and VLA-JEPA use future or dynamics prediction to improve learned representations (Fang et al., 2026; Yuan et al., 2026; Cui et al., 2024; Sun et al., 2026). DreamVLA, DreamZero, ImageWAM, and MemoryVLA++ instead expose imagined future content to action generation in various forms (Zhang et al., 2026a; Ye et al., 2026; Zhang et al., 2026b; Shi et al., 2026a). Related latent-prediction principles also underlie V-JEPA and V-JEPA 2 (Bardes et al., 2024; Assran et al., 2025).

PREM uses prediction differently. The predicted future is retained until the corresponding observation arrives, then used to regulate recurrent memory writing rather than supplied as forecast context to the action expert. Cambrian-S similarly uses prediction error to identify surprising events in long videos (Yang et al., 2026b); PREM instantiates this idea in a closed-loop manipulation policy through its *prediction–verification–update* cycle. Its contribution is therefore the coupling of action-conditioned future prediction, delayed verification, and persistent visual-state control.

A.3 Benchmarks for Temporal Manipulation

Beyond robotic control benchmarks, SpatialScore evaluates the spatial understanding of multimodal models through diverse question-answering tasks (Wu et al., 2026a). Such evaluations provide a complementary view of spatial perception and reasoning, but do not directly test whether a robot can retain historical evidence for closed-loop action selection.

LIBERO and CALVIN evaluate diverse or compositional manipulation (Liu et al., 2023; Mees et al., 2022), but long episodes alone do not guarantee that success depends on temporal evidence. LIBERO-Pro and subsequent benchmark audits show that standard performance can partly exploit familiar layouts, instructions, or action sequences (Zhou et al., 2025; Jiang et al., 2026). We therefore use LIBERO primarily to measure general manipulation and LIBERO-Pro to test spatial robustness.

DOMINO and RMBench provide more targeted temporal diagnostics. DOMINO evaluates dynamic interception and tracking under continuously evolving object states (Fang et al., 2026), whereas RM-Bench constructs tasks whose solutions explicitly require historical observations (Chen et al., 2026). We use them as complementary tests of dynamic perception and long-range memory, respectively; RMBench is not treated as a dynamic-environment benchmark.

B Implementation and Evaluation Details

The following details use the notation of the main text. The index t counts replanning decisions, while observation strides and future-target offsets are measured in environment steps.

B.1 Architecture and State Update

Visual representation and recurrent core. The recurrent pathway uses the current base-view image, corresponding to the head-camera view for the Aloha setups. The pretrained PaliGemma visual encoder produces a 16×16 grid of 2048-dimensional patch features. A trainable `SpatialProjector` maps these features to $X_t \in \mathbb{R}^{256\times 512}$, giving $N = 256$ spatial tokens and memory width $d = 512$. The RVM has two recurrent layers, eight attention heads, and an MLP expansion ratio of two, corresponding to a feed-forward hidden width of 1024.

Following the transformer–GRU formulation, the recurrent core uses update and reset gates to construct its candidate state:

$$\begin{aligned} u_t &= \sigma(X_t W_u + M_{t-1} U_u), \\ r_t &= \sigma(X_t W_r + M_{t-1} U_r), \\ G_t &= F_{\text{cand}}\big(X_t, r_t \odot \mathrm{LN}(M_{t-1})\big), \\ \widetilde{M}_t &= (1 - u_t) \odot M_{t-1} + u_t \odot \mathrm{LN}(G_t). \end{aligned} \tag{12}$$

Here, the learned projections act on the feature dimension of each spatial token, and G_t denotes the candidate features. The final memory M_t is obtained by applying the prediction-derived coefficient to $\widetilde{M}_t - M_{t-1}$, as in Eq. (7).

Recurrent-memory readout. The updated RVM state contributes 256 policy-prefix tokens:

$$P_t^{\text{RVM}} = \sigma(\alpha_{\text{mem}}) M_t W_{\text{mem}}, \tag{13}$$

where W_{mem} projects from width 512 to the policy token width. The scalar α_{mem} is learned and initialized to -4. This readout corresponds to $\mathcal{P}_{\text{mem}}(M_t)$ in the main text.

B.2 Action Representation and Future Predictor

Action-feature extraction. We reuse the final-layer action-token hidden states of the 18-layer `gemma_300m` Action Expert, whose hidden width is 1024. Specifically, these are the final `action_horizon` tokens of `suffix_out`, before the action output projection. No additional action-expert forward pass is performed.

During training, the hidden states come from the single forward pass at the randomly sampled flow-matching time. They are not obtained from a complete iterative denoising trajectory. During inference, they come from the final ODE/flow-denoising iteration. Let H_{act} denote the predicted action-chunk length and H_{cond} the number of leading action tokens used for future prediction. Their benchmark-specific values are given in Table 6.

Table 6: Action-token selection for future prediction. B denotes batch size. Only the first H_{cond} action tokens are used.

Benchmark	Raw hidden shape	H_{act}	H_{cond}
LIBERO	$B \times 10 \times 1024$	10	5
DOMINO	$B \times 50 \times 1024$	50	16
RMBench	$B \times 50 \times 1024$	50	50

Temporal encoding and pooling. The selected hidden states are detached and projected from 1024 to 512 dimensions. After adding sinusoidal temporal encodings, a one-layer temporal Transformer processes them using eight attention heads and a feed-forward hidden width of 1024. The sequence is then partitioned into eight ordered temporal bins, with masked mean pooling within each bin. The resulting representation has shape $z_t^{\text{act}} \in \mathbb{R}^{B\times 8\times 512}$. Empty bins are marked invalid and excluded from attention. In LIBERO, five valid action-conditioning tokens yield five valid pooled tokens among the eight output positions, giving a valid fraction of $5/8$. Action tokens are not matched one-to-one with visual patches.

Query-based spatial decoding. The predictor is a two-block Transformer decoder with 256 learned spatial queries of width 512. It receives the incoming memory M_{t-1}, the current-step context X_t defined in the main text, and z_t^{act}. The conditioning streams are mapped to a common width of 512 and concatenated along the token dimension. Each decoder block performs eight-head cross-attention from the spatial queries to this context, a $512 \to 1024 \to 512$ MLP, and self-attention among the spatial queries. The output is $\hat{X}_{t+1} \in \mathbb{R}^{B\times 256\times 512}$.

The explicit memory input is the incoming M_{t-1}, not the newly updated M_t. The action representation is obtained after action generation has incorporated the updated memory. The forecast is cached and compared with X_{t+1} at the next replanning decision.

B.3 Prediction Discrepancy and Temporal Training

Normalized feature discrepancy. For feature vectors $u, v \in \mathbb{R}^d$, let $\bar{u} = \text{Norm}_{\epsilon_c}(u)$ and $\bar{v} = \text{Norm}_{\epsilon_c}(v)$ denote the token-wise normalized features. The discrepancy used for prediction and memory modulation is

$$\delta(u, v) = 1 - \cos_{\epsilon_c}(\bar{u}, \bar{v}) + \frac{\lambda_{\text{dist}}}{d} \sum_{c=1}^{d} \text{SmoothL1}_{\kappa}(\bar{u}_c - \bar{v}_c), \tag{14}$$

where $\cos_{\epsilon_c}$ denotes cosine similarity computed with numerically stabilized L2 normalization of its operands. We use $\epsilon_c = 10^{-6}$ for token normalization and cosine L2 normalization. The Smooth-L1 weight is $\lambda_{\text{dist}} = 0.1$, and its transition parameter is $\kappa = 0.1$. The Smooth-L1 term is averaged over feature dimensions.

Spatial normalization and update coefficients. We compute $e_{t,i} = \delta(\hat{X}_{t,i}, X_{t,i})$ and detach the error before computing the memory-update coefficient. Writing $e_{t,i}^{\perp} = \text{sg}(e_{t,i})$, we obtain

$$\begin{aligned} \mu_t &= \text{median}_j \, e_{t,j}^{\perp}, \\ \rho_t &= \text{median}_j \, |e_{t,j}^{\perp} - \mu_t|, \\ \xi_{t,i} &= \frac{e_{t,i}^{\perp} - \mu_t}{\max(\rho_t, \epsilon_m)}. \end{aligned} \tag{15}$$

The MAD denominator floor is $\epsilon_m = 10^{-3}$. The coefficient is then computed as in Eq. (6), using $\beta_{\min} = 0.25$, $\tau = 2.0$, and $\gamma = 0.5$. The discrepancy and update-coefficient parameters are fixed hyperparameters, not learned parameters.

Prediction objective and supervision. The target is the base/head-camera representation at the next replanning observation, obtained from the same visual encoder and trainable $2048 \to 512$ spatial

projector. The projected target features are detached before evaluating the discrepancy. There is neither a separate target encoder nor an EMA target encoder.

Let v_t mark a valid adjacent prediction–target pair and $q_{t,i} \sim \mathrm{Bernoulli}(0.75)$ select supervised patches. The prediction objective is

$$\mathcal{L}_{\mathrm{pred}} = \frac{\sum_{t,i} v_t q_{t,i}\, \delta\Big(\hat{X}_{t+1,i}, \mathrm{sg}(X_{t+1,i})\Big)}{\sum_{t,i} v_t q_{t,i}}, \tag{16}$$

with batch indices omitted for readability. The 75% ratio refers to the random selection of target patches contributing to this loss, not to masking predictor inputs. Unselected target patches do not contribute to the prediction loss. All 256 output tokens are available at inference. The complete objective is $\mathcal{L} = \mathcal{L}_{\mathrm{act}} + 0.1\mathcal{L}_{\mathrm{pred}}$.

Warm-up and linear ramp. At optimizer step s, the modulation strength is

$$a(s) = \mathrm{clip}\left(\frac{s - S_{\mathrm{warmup}}}{S_{\mathrm{ramp}}}, 0, 1\right), \tag{17}$$

which gives the effective update coefficient

$$\beta^{\mathrm{eff}}_{t,i} = 1 - a(s) + a(s)\beta_{t,i}. \tag{18}$$

During warm-up, $a = 0$ and the full recurrent proposal is accepted. During the ramp, the coefficient interpolates linearly toward the prediction-regulated value. At inference, $a = 1$. Table 7 reports the benchmark-specific settings. These are memory-modulation schedules, not learning-rate warm-up schedules.

Causal unrolling and reset. Training uses `jax.lax.scan` to process each observation clip causally. The recurrent carry is not detached within a clip, so gradients propagate through the full unroll. Recurrent state is not carried between training clips, and no burn-in prefix is used to reconstruct pre-clip memory. The detached coefficient path described above is separate from truncation of recurrent gradients.

The RVM state and forecast cache are reset to zero/invalid states at clip and episode boundaries. The first valid observation initializes $M_0 = X_0$. Since there is no valid previous forecast, prediction-based modulation is skipped at this observation. The first forecast uses the reset incoming memory M_{-1}, which is excluded by its validity mask; the current-step context and action representation remain available.

B.4 LIBERO AND LIBERO-PRO PROTOCOL

Training configuration. We jointly train on the four LIBERO suites using 1,693 demonstrations, a global batch size of 128, and 30,000 optimization steps. The learning rate is cosine-annealed from 2×10^{-5} to 1×10^{-5}.

Temporal sampling. Each clip contains $K = 16$ replanning observations with a stride of five environment steps. For a clip beginning at environment step r_0, the observation indices are $[r_0, r_0 + 5, \ldots, r_0 + 75]$. Each future target is offset by five environment steps. Clips may start within an episode and initialize their RVM state from the first valid clip observation, without burn-in. The predicted action horizon is ten, and the first five action tokens condition the future predictor. Memory-modulation warm-up and ramp lengths are 2,000 steps each.

LIBERO evaluation horizons. We use maximum episode horizons of 220, 280, 300, and 520 environment steps for LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10, respectively. The corresponding LIBERO-Pro `_swap` suites use the same horizons. Both the standard and position-perturbed evaluations contain ten tasks per suite and 50 episodes per task, yielding 2,000 episodes per evaluation setting.

Table 7: Memory-modulation and temporal-training settings. Warm-up and ramp lengths count optimizer steps; K counts replanning observations. Stride is measured in environment steps.

Benchmark	Warm-up	Ramp	K	Stride
LIBERO	2,000	2,000	16	5
DOMINO	10,000	10,000	16	16
RMBench	1,000	2,000	task-specific	50

B.5 DOMINO PROTOCOL

Training configuration and inputs. We initialize from the official $\pi_{0.5}$ base checkpoint and perform full fine-tuning on the clean Aloha-AgileX Level-1 data from all 35 DOMINO tasks, with at least 50 demonstrations per task. Training runs for 100,000 optimizer steps using eight-way FSDP, a global batch size of 64, and no gradient accumulation. Current observations contain head, left-wrist, and right-wrist views; a missing right-wrist view is represented by a mask. All views are aspect-ratio-preserving padded to 224×224. The action horizon is 50.

Temporal sampling. Each clip contains 16 replanning observations at a stride of 16 raw environment steps, $[r_0, r_0 + 16, \ldots, r_0 + 240]$. The future target is the next replanning observation, offset by 16 steps. Clips may start within an episode, with reset recurrent memory and no burn-in. The first 16 action tokens condition future prediction. Memory-modulation warm-up and ramp lengths are 10,000 steps each.

Evaluation protocol. Evaluation covers all 35 clean Level-1 tasks at DOMINO@0.1, with 100 accepted episodes per task. Task-specific horizons range from 400 to 1,700 environment steps. We use the exact training normalization statistics, `adapt_to_pi=True`, a replanning interval of 16, and `action_speed_scale=1.0`. Policy-server and client seeds are both zero; simulator seeds begin at 100000 and are accepted sequentially without duplication until each task reaches 100 valid episodes. Each task is evaluated with a fresh policy server, with one task assigned per GPU. The audit verifies 3,500 episodes, 3,500 videos, unique accepted seeds, and the evaluation configuration.

B.6 RMBENCH PROTOCOL

Training configuration and inputs. We evaluate the five RMBench task families under the M(1) memory-dependent setting. Each task is trained independently from 50 demonstrations for 20,000 optimization steps with a global batch size of 32. Training uses eight GPUs with four-way FSDP and two-way data parallelism, with gradient accumulation set to one. The learning-rate warm-up lasts 2,000 steps; the learning rate is cosine-annealed from 2×10^{-5} to 1×10^{-5}. The memory-modulation schedule instead uses 1,000 warm-up steps and 2,000 ramp steps, becoming fully enabled at optimizer step 3,000. Current observations contain head, left-wrist, and right-wrist views.

Temporal sampling. All tasks use an action horizon, replanning interval, observation stride, future-target offset, and action-conditioning horizon of 50 environment steps. The training runner uses strict temporal anchors, with clips starting at episode step zero.

Gradient and padding handling. The recurrent carry is differentiated through the full K-step clip without an internal detach. No recurrent state is transferred between clips, and no burn-in is used. Clips shorter than the configured length are padded, with padded observation positions marked by `memory_sequence_valid=False`. For a terminal action chunk with fewer than 50 valid actions, `action_valid_mask` excludes repeated padding actions from the action loss.

Evaluation protocol. Evaluation uses 100 episodes per task. Recurrent states and pending predictions are reset at every episode boundary.

C Complete Quantitative Results

The main paper reports representative DOMINO tasks. Table 8 provides the complete 35-task comparison used to compute the aggregate result.

C.1 DOMINO Per-task Results

Table 8: Complete per-task success rates (%) on DOMINO Level-1.

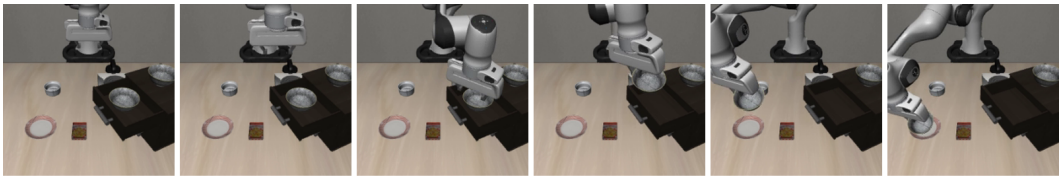
Task	$\pi_{0.5}$ (Baseline)	PUMA	PREM (Ours)
Dynamic Interception			
Adjust Bottle	52	65	**74** (+22)
Dump Bin Bigbin	0	0	**37** (+37)
Grab Roller	10	**33**	26 (+16)
Handover Block	1	17	**20** (+19)
Handover Mic	5	**35**	26 (+21)
Hanging Mug	5	9	**25** (+20)
Move Can Pot	7	22	**29** (+22)
Move Playing Card Away	**8**	6	7 (-1)
Place A2B Left	2	**13**	7 (+5)
Place A2B Right	1	**8**	4 (+3)
Place Bread Skillet	9	**19**	9 (+0)
Place Can Basket	6	**14**	13 (+7)
Place Object Basket	11	13	**23** (+12)
Put Bottles Dustbin	17	23	**27** (+10)
Put Object Cabinet	11	**34**	29 (+18)
Rotate QRcode	4	**14**	4 (+0)
Scan Object	1	5	**6** (+5)
Shake Bottle	33	55	**55** (+22)
Shake Bottle Horiz.	42	**75**	60 (+18)
Dynamic Tracking			
Beat Block Hammer	10	15	**23** (+13)
Click Alarmclock	**14**	4	6 (-8)
Click Bell	0	3	**4** (+4)
Move Pillbottle Pad	6	14	**21** (+15)
Move Stapler Pad	0	1	**3** (+3)
Place Bread Basket	8	12	**23** (+15)
Place Container Plate	22	26	**44** (+22)
Place Empty Cup	2	7	**10** (+8)
Place Fan	2	**8**	4 (+2)
Place Mouse Pad	5	2	**10** (+5)
Place Object Scale	3	4	**8** (+5)
Place Object Stand	**1**	**1**	**11** (+10)
Place Phone Stand	2	6	**10** (+8)
Place Shoe	25	16	**52** (+27)
Press Stapler	6	**10**	7 (+1)
Stamp Seal	6	**13**	9 (+3)
35-task Average	9.63	17.20	**20.74** (+11.11)

D QUALITATIVE ROLLOUTS

We provide successful rollout visualizations across LIBERO, DOMINO, and RMBench to complement the quantitative results and illustrate the distinct manipulation capabilities evaluated by each benchmark.

D.1 LIBERO

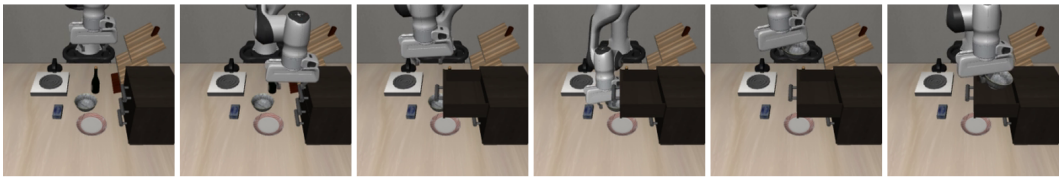
We visualize representative successful PREM rollouts from the four standard LIBERO suites. Frames progress from left to right and cover spatial transfer, object-centric placement, goal-conditioned interaction, and longer-horizon composition.



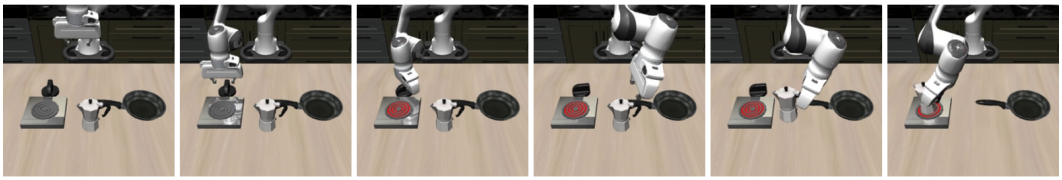
(a) LIBERO-Spatial: Bowl from Drawer to Plate



(b) LIBERO-Object: Milk into Basket



(c) LIBERO-Goal: Open Drawer and Place Bowl



(d) LIBERO-10: Turn On Stove and Place Moka Pot

Figure 5: **Representative successful rollouts on LIBERO.** The rows show LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10, respectively.

D.2 DOMINO

We additionally visualize successful PREM rollouts from the two official DOMINO task categories. Frames progress from left to right, and the green arrows indicate task-relevant object motion.

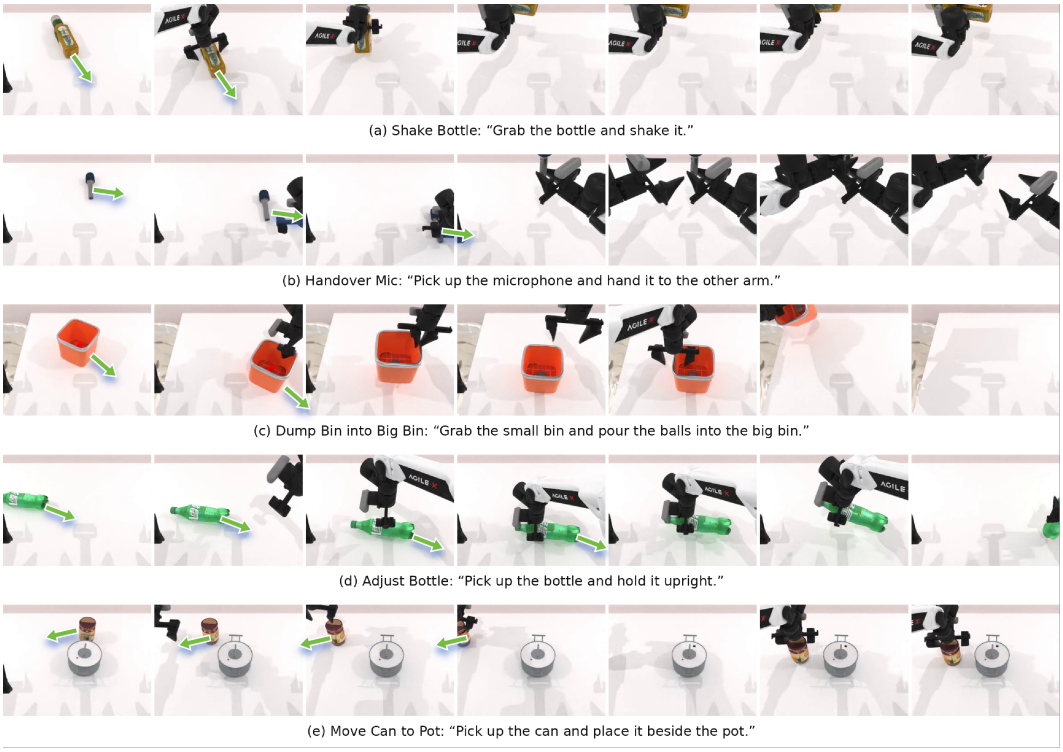


Figure 6: **Successful Dynamic Interception rollouts on DOMINO.** The five examples visualize Shake Bottle, Handover Mic, Dump Bin into Big Bin, Adjust Bottle, and Move Can to Pot. These tasks require the policy to react to objects or interaction states that evolve during execution.

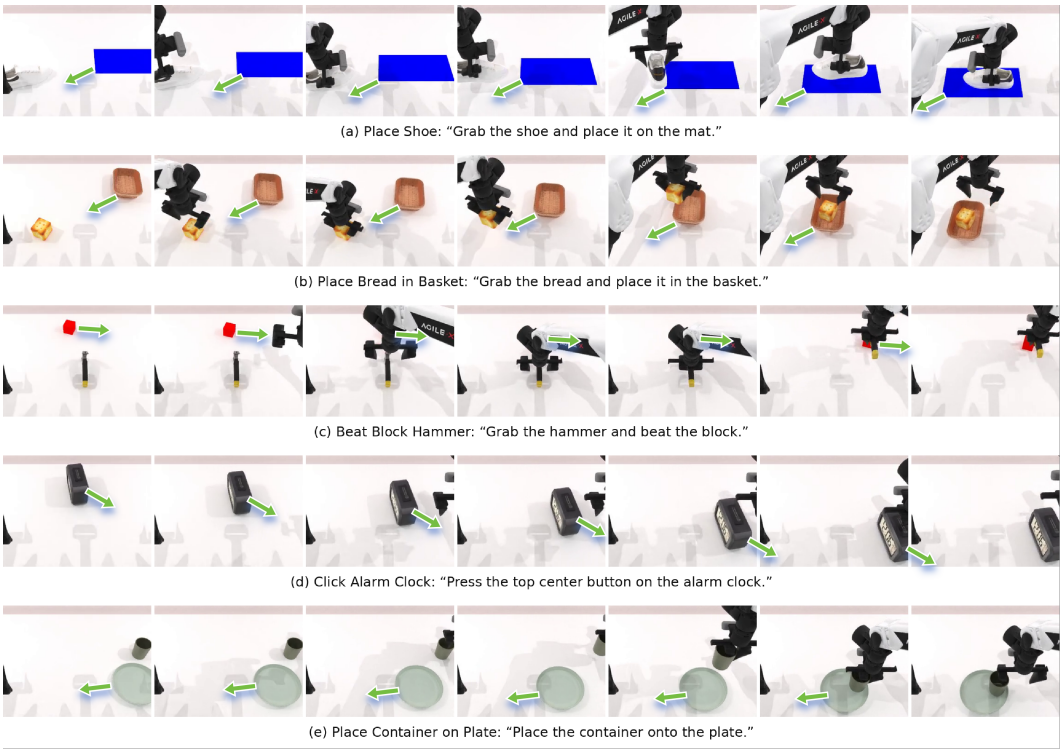


Figure 7: **Successful Dynamic Tracking rollouts on DOMINO.** The five examples visualize Place Shoe, Place Bread in Basket, Beat Block Hammer, Click Alarm Clock, and Place Container on Plate. Successful execution requires continuously tracking the task-relevant object as its spatial relationship with the robot changes.

D.3 RMBENCH

We visualize successful PREM rollouts for all five task families in the RMBench M(1) setting. Frames progress from left to right and illustrate how the policy retains task-relevant information across temporally separated observations and actions.

The corresponding PREM success rates are (22%), (100%), (100%), (98%), and (99%) for Observe and Pick Up, Rearrange Blocks, Put Back Block, Swap Blocks, and Swap-T, respectively, yielding an (83.8%) macro-average.

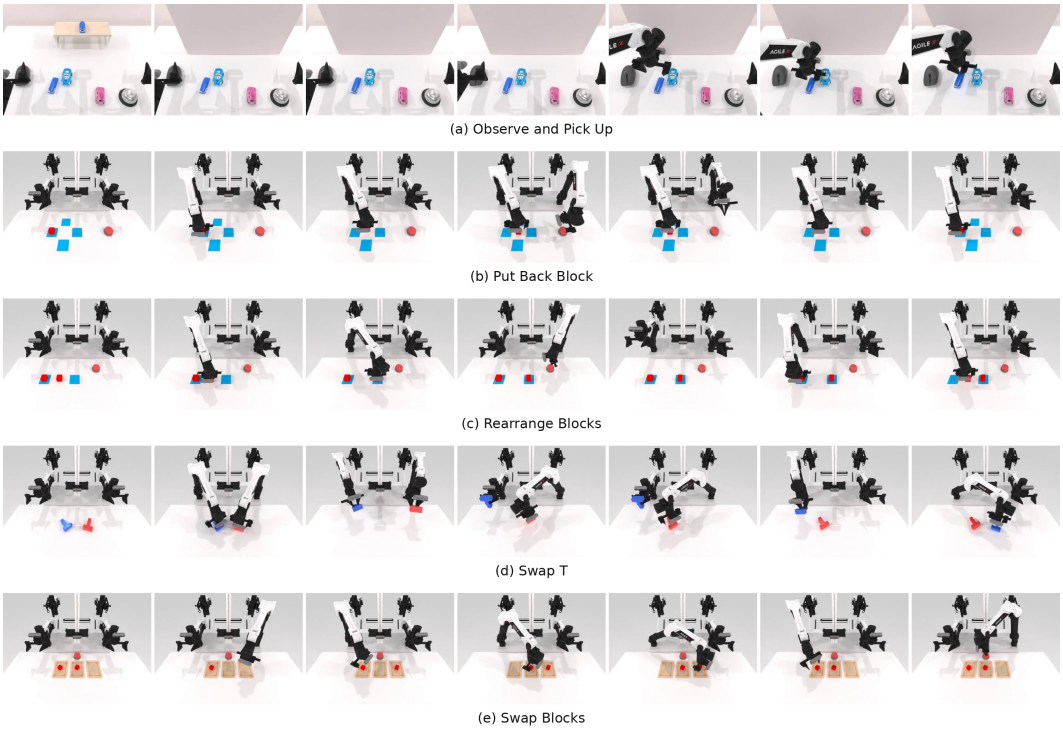


Figure 8: **Successful rollouts across the five RMBench M(1) task families.** The rows visualize Observe and Pick Up, Put Back Block, Rearrange Blocks, Swap-T, and Swap Blocks. These tasks require retaining object identities, initial spatial configurations, or earlier interaction states that cannot always be recovered from the current observation.