PA-WAM: Progress-Aligned World–Action Model for Long-Horizon Robot Manipulation

Anonymous authors
Paper under double-blind review

ABSTRACT

World–Action Models (WAMs) provide dynamics supervision to robot policies by predicting future states, but in long-horizon manipulation, a smaller future-representation error does not necessarily imply correct task progress. Existing future-reconstruction objectives implicitly contain stage changes from demonstrations, yet do not explicitly distinguish prediction errors by their impact on task advancement. We propose the Progress-Aligned World–Action Model (PA-WAM), which uses a history-conditioned stage transition both as a condition for latent-future generation and as a consistency target. PA-WAM first aligns successful demonstrations for the same task to construct a continuous progress coordinate, then predicts a nominal transition over a fixed horizon from current manipulation evidence and multi-scale history. Bidirectional Progress–Dynamics Coupling (BPDC) uses this transition to modulate latent action queries and constrains the stage change of the generated future through a frozen progress readout. Target–readback consistency is jointly optimized with future reconstruction and action imitation, while the readback branch is used only during training. Compared with LaWAM, PA-WAM increases the mean success rate on LIBERO from 98.6% to 99.2%, with the largest gain on Long, from 97.0% to 98.2%; on MimicGen-D1, it improves the mean from 81.8% to 84.3%. Across three RM65 real-robot tasks, the mean success rate increases from 82.2% to 87.8%. These results show that treating task progress as an internal constraint on future generation improves long-horizon execution while preserving local dynamics modeling.

1 INTRODUCTION

Vision–Language–Action models (VLAs) map visual observations, language instructions, and robot states to executable actions and have become an important interface for language-conditioned manipulation (Zitkovich et al., 2023; Kim et al., 2024; Ghosh et al., 2024; Black et al., 2025; Physical Intelligence et al., 2025). World–Action Models (WAMs) additionally predict action-relevant future states, allowing a policy to generate actions from expected scene changes (Bi et al., 2026; Ye et al., 2026a; Su et al., 2026; Chen et al., 2026). As future prediction moves from pixels to compact latent spaces, explicit future conditioning becomes better suited to closed-loop control. In long-horizon manipulation, however, the usefulness of a future also depends on its relation to the current task stage: the same prediction horizon can correspond to different amounts of progress in different execution contexts.

This issue is particularly clear in multi-stage tasks. Approaching an object, making fine contact adjustments, and switching stages after a placement may take similar action-block durations while serving different task roles. Similar current observations can also occur after different execution histories, requiring the policy either to continue the current operation or to switch to another subgoal. Future-representation reconstruction targets the demonstrated successor state and therefore contains these differences implicitly, but its error does not separately measure progress. For a capacity-limited and imperfect world model, reducing overall representation error need not correct the stage bias that matters most for subsequent execution.

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.

Our key idea is to use the temporal structure of successful demonstrations to construct a *history-conditioned progress target* for a fixed-horizon future. After aligning demonstrations of the same task to a shared stage coordinate, the model estimates the relative transition typically produced by successful demonstrations over the same horizon, using current manipulation evidence and executed history. History determines the target transition, while progress supervision constrains the generated future to change consistently in that coordinate. The transition serves both as a condition for future generation and as a training target for the generated result: the world model receives the target transition, and a readout in the same coordinate system evaluates the predicted future, with their difference forming a task-relevant auxiliary loss. Because the target is derived from the current context, the constraint permits a small but appropriate advance rather than requiring the stage coordinate to increase maximally in every action block.

Based on this observation, we propose PA-WAM. Stage I uses Soft-DTW to construct an aligned progress coordinate and trains a Progress Transition Model (PTM) that predicts the current stage and nominal fixed-horizon transition from multi-scale history. Stage II applies BPDC: the target transition modulates latent action queries, a frozen progress readout evaluates the generated future, and a target–readback consistency loss updates the world model. Future reconstruction and action supervision are retained, and the action expert generates actions from the modulated latent future.

Our contributions are:

- We formulate fixed-horizon future learning for latent WAMs as a **history-conditioned progress-alignment** problem, separating generic representation-reconstruction error from stage-progression bias and defining a nominal transition in a shared coordinate learned from successful demonstrations.
- We propose **PA-WAM**, which predicts a nominal transition from aligned demonstrations and multi-scale history, and uses **BPDC** to apply the same target to latent-query modulation and future readback supervision, jointly optimizing target–readback consistency, future reconstruction, and action imitation.
- We evaluate PA-WAM on LIBERO, MimicGen-D1, CALVIN, and three RM65 real-robot tasks, and use component ablations to examine history conditioning, query modulation, and progress readback.

2 Related Work

2.1 World–Action Models

World models support planning and control by predicting environmental changes. Dreamer and TD-MPC2 learn policies through latent imagination and local trajectory optimization, respectively (Hafner et al., 2023; Hansen et al., 2024), while visual and hierarchical foresight organize actions through predicted images or visual subgoals (Ebert et al., 2018; Nair and Finn, 2020). Recent WAMs combine future modeling with general robot policies: Motus jointly models latent actions and world states, DreamZero uses video priors to learn policies, and Ctrl-World emphasizes action-controllable future generation (Bi et al., 2026; Ye et al., 2026a; Guo et al., 2026). World Guidance, LaWAM, and LiLa-WAM further explore future supervision in condition and visual-latent spaces; ST-WAM focuses on semantic–temporal modeling under visual distribution shifts; and Fast-WAM and GigaWorld-Policy emphasize action-centered structure and inference efficiency (Su et al., 2026; Chen et al., 2026; Yang et al., 2026a; Wang et al., 2026; Yuan et al., 2026; Ye et al., 2026b). PA-WAM studies consistency between progress conditions and future representations: a history-estimated stage transition guides future generation and checks whether the generated future expresses the corresponding advance.

2.2 Long-Horizon Manipulation and Memory

Long-horizon benchmarks such as CALVIN (Mees et al., 2022) require a policy to preserve state and stage information across consecutive subtasks. SayCan, finite-state task machines, and subgoal-transition models organize multi-step execution through skill selection or explicit intermediate states (Ahn et al., 2022; Liang and Boularias, 2021; Jain et al., 2025). MemoryVLA, ReMem-VLA, MEM, and EventVLA preserve cross-stage information using memory, recurrent queries, multi-scale states, or event selection (Shi et al., 2025; Li et al., 2026; Torne et al., 2026; Yang et al.,

2026b); structured state-space models also provide a basis for fixed-capacity representations of long histories (Gu et al., 2022). DiM-WAM combines historical event memory with auxiliary task-progress supervision so that memory encodes both completed events and the current stage (Wang et al., 2026). In contrast to these memory methods, PA-WAM uses history to predict a fixed-horizon progress transition and directly constrains the progress readout of the generated future.

2.3 Task Progress and Future Guidance

Time-Contrastive Networks, Temporal Cycle-Consistency, and Soft-DTW learn temporal structure through multi-view correspondence, cycle consistency, or temporal alignment (Sermanet et al., 2018; Dwibedi et al., 2019; Cuturi and Blondel, 2017). For robot policies, ProgressVLA feeds predicted futures to a progress estimator, jointly fine-tunes the world model and estimator through future-progress regression, and guides action diffusion with progress gradients at inference (Yan et al., 2026). StageWAM uses a joint embedding of the next stage to guide future modeling (Liu et al., 2026). PA-WAM focuses on continuous stage changes over a fixed physical horizon: its history-conditioned nominal transition modulates latent queries and supervises future readback. The target is to match demonstration-supported transitions rather than directly maximize progress or predict a next-stage embedding. This consistency constraint is applied only during training.

3 Problem Formulation

At decision time t, the beginning of an action block, the policy receives a visual observation o_t, proprioceptive state r_t, and language instruction g, and predicts an action block $u_t = (a_{t,0}, \dots, a_{t,K_a-1})$. The true execution history available at this decision is denoted by $\mathcal{H}_t = (o_{\le t}, r_{\le t}, u_{<t}, g)$. Here, H is the specified physical prediction horizon and $\mathcal{D}_g$ is the set of successful demonstrations for task g. Let δ_c be the control period and n_H the number of control steps covered by this horizon, so that $H = n_H\delta_c$; K_a is the number of actions in an action block, and H is an independent future-supervision horizon without assuming $H = K_a\delta_c$. Let $\kappa(t, H)$ return the control index after horizon H from decision time t, and define $t + H \triangleq \kappa(t, H)$. H is converted to a future-supervision index using the control period and kept fixed within each comparison. For a demonstration of length T_i, we use only samples satisfying $\kappa(t, H) \le T_i$. Action-block length, executed prefix, and replanning frequency are kept fixed within each benchmark comparison. Future readback is computed only during training; deployment produces actions from the current observation, execution history, and action expert.

The base WAM encodes the current context, predicts a future representation through latent action queries q_t, and passes that future to the action expert:

$$h_t = E(o_{\le t}, r_{\le t}, g), \qquad \hat{z}_{t+H} = F_\theta(h_t, q_t), \qquad \hat{u}_t \sim \pi_\phi(\cdot \mid h_t, \hat{z}_{t+H}). \tag{1}$$

where F_θ is the latent world model, π_ϕ is the action expert, and z_{t+H} is the ground-truth future representation. The base training objective is

$$\mathcal{L}_{\mathrm{WAM}} = \lambda_z d_z(\hat{z}_{t+H}, z_{t+H}) + \lambda_a \mathcal{L}_{\mathrm{act}}(\pi_\phi; u_t), \tag{2}$$

PA-WAM retains future prediction and action supervision and adds a progress-alignment objective.

Let $p_t \in [0, 1]$ denote the position of the current state in the shared temporal structure of successful demonstrations for the same task, which we call the *demonstration-aligned stage coordinate*. We assume that demonstrations for a task have an approximately consistent and approximately linear stage order that can be described by a one-dimensional monotonic coordinate. The method therefore targets long-horizon tasks with broadly consistent stage order; multi-branch, parallel, or rollback structures require a corresponding richer stage representation. This coordinate represents relative temporal position rather than the completion ratio of equally sized subtasks. The demonstration transition over fixed horizon H is

$$\Delta_t^\star = p_{\kappa(t,H)} - p_t, \qquad 0 \le \Delta_t^\star \le 1 - p_t, \tag{3}$$

$\Delta_t^\star$ measures the relative stage transition of a successful demonstration over the fixed horizon H.

PA-WAM predicts a nominal target $\widehat{\Delta}_t$ from execution history, namely an estimate of the successful-demonstration transition under the current context. A progress readout P_ψ evaluates the generated

future as

$$\Delta_t^e = P_\psi(\widehat{z}_{t+H}; \mathcal{H}_t) - \hat{p}_t, \qquad \Delta_t^e \approx \widehat{\Delta}_t. \tag{4}$$

where $\hat{p}_t$ is the current progress estimate and Δ_t^e is the readout transition of the predicted future. By constraining Δ_t^e to agree with $\widehat{\Delta}_t$, PA-WAM makes task progress directly participate in latent-future learning.

4 METHOD

PA-WAM connects demonstration supervision and future generation through a shared progress target. Demonstration alignment provides a stage coordinate shared across trajectories, a history-conditioned transition estimates the advance associated with the current state over a given horizon, and BPDC uses this target both for query modulation and future-readback constraints. Stage I learns the coordinate and transitions; Stage II fixes this coordinate system and trains the world–action model so that the generation condition and training target have the same stage semantics.

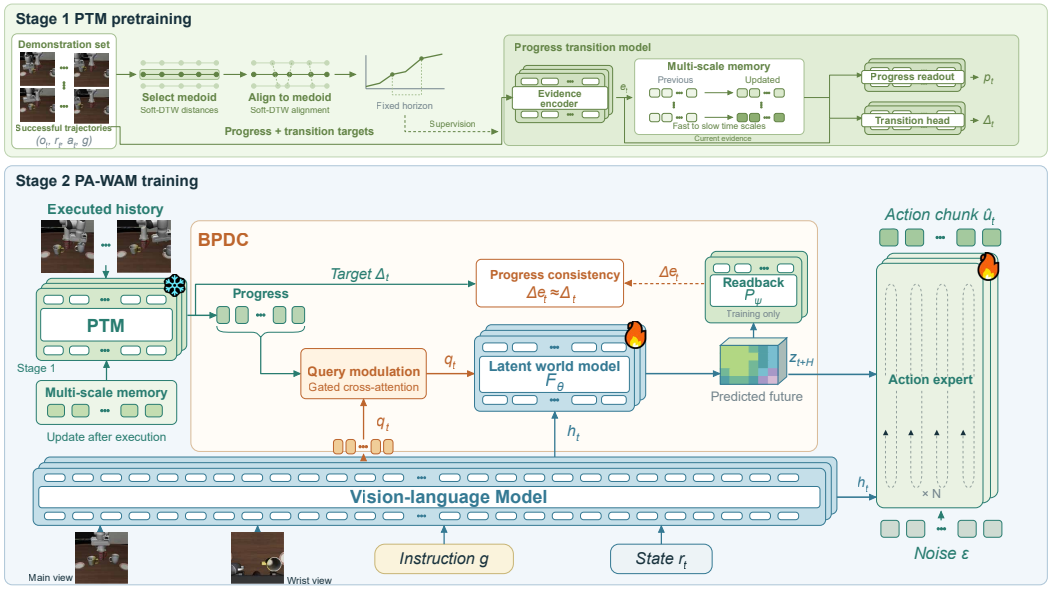

Figure 1: Overview of PA-WAM. Stage I uses Soft-DTW to align successful demonstrations for the same task to a shared time axis, constructing a continuous stage coordinate and a fixed-horizon progress-transition target. An evidence encoder, multi-scale memory, progress readout, and transition head jointly train PTM to predict the current stage p_t and nominal transition $\widehat{\Delta}_t$ from current manipulation evidence and executed history. In Stage II, BPDC injects the current stage and target transition into latent action queries through gated cross-attention, and the latent world model generates a future representation $\widehat{z}_{t+H}$. During training, a frozen progress readout evaluates the predicted future and computes the consistency loss $\Delta_t^e \approx \widehat{\Delta}_t$, while future reconstruction and action imitation are optimized simultaneously. At deployment, the readback branch is removed, leaving progress conditioning, latent-future generation, and action-block prediction.

4.1 DEMONSTRATION-ALIGNED CONTINUOUS PROGRESS SUPERVISION

Demonstrations can differ in execution speed and local dwell times, so directly using each trajectory's normalized time would mix these differences into the stage labels. We construct state sequences from frozen visual features and normalized proprioception, choose as the center demonstration the one minimizing the sum of pairwise Soft-DTW (Cuturi and Blondel, 2017) distances within a task, and softly align every trajectory to its reference time axis. Alignment uses training demonstrations only. Specifically, outputs of the frozen visual encoder are ℓ_2-normalized, proprioception is standardized per dimension using training-set means and standard deviations, and the squared Euclidean distances of the two modalities are divided by their feature dimensions and added with equal weight to form

the local cost matrix D. Soft-DTW uses a fixed smoothing temperature $\gamma > 0$; the gradient with respect to D_{ts} gives soft path-occupancy weights A_{ts}, normalized as $\bar{A}_{ts} = A_{ts} / \sum_{s'} A_{ts'}$. Center selection and soft alignment use the same cost and temperature. The center is selected using the length-normalized sum of pairwise costs so that trajectory length does not dominate the reference choice. Let $\bar{A}^{(i)}_{ts}$ be the normalized weight aligning frame t of demonstration i to frame s of the reference demonstration. The raw position and alignment concentration are

$$\tilde{p}^{(i)}_t = \sum_{s=1}^{T_{i^\star}} \bar{A}^{(i)}_{ts} \frac{s-1}{T_{i^\star}-1}, \tag{5}$$

$$c^{(i)}_t = 1 - \frac{\mathrm{Ent}(\bar{A}^{(i)}_{t:})}{\log T_{i^\star}}. \tag{6}$$

Here $\mathrm{Ent}(\cdot)$ denotes entropy and c_t measures alignment concentration rather than the probability that a label is correct. To avoid non-unique projections caused by zero weights, the implementation uses $\max(c_t, \varepsilon_c)$ as the projection weight, where $\varepsilon_c > 0$ is a fixed small constant. Loss weighting still uses the original c_t, so uncertain alignments receive weaker supervision. We then obtain the stage coordinate through weighted monotonic projection and construct fixed-horizon transition supervision:

$$p^{(i)}_{1:T_i} = \arg \min_{0=s_1 \le \cdots \le s_{T_i}=1} \sum_{t=1}^{T_i} c^{(i)}_t (s_t - \tilde{p}^{(i)}_t)^2. \tag{7}$$

$$\Delta^\star_t = p^{(i)}_{\kappa(t,H)} - p^{(i)}_t, \qquad c_{t,\Delta} = \min(c^{(i)}_t, c^{(i)}_{\kappa(t,H)}). \tag{8}$$

The weighted monotonic projection is solved with the pool-adjacent-violators algorithm, with the first and last coordinates fixed to 0 and 1. The reference demonstration must contain at least two frames; ties in center-selection cost are broken by demonstration index. If a row of the soft alignment underflows numerically, its corresponding position is recovered from hard DTW using the same cost matrix and its supervision weight is set to zero. Monotonic projection removes local regressions from the labels, while $\Delta^\star_t$ preserves context-dependent stage changes over the same horizon. This construction reduces speed misalignment across demonstrations but retains the reference demonstration's temporal allocation as the coordinate scale.

4.2 Multi-Scale Progress-Transition Modeling

PTM jointly models local manipulation evidence and history across action blocks. Let x_t be the fused representation of frozen visual features and normalized proprioception. A short-window encoder produces

$$e_t = E_s(x_{t-w+1:t}, a_{t-w:t-1}, g). \tag{9}$$

Multi-scale recurrent states accumulate this evidence with different decay rates:

$$m^{(j)}_t = \rho_j m^{(j)}_{t-1} + (1-\rho_j) B_j e_t, \quad 0 \le \rho_j < 1. \tag{10}$$

Different ρ_j values correspond to history summaries at different time scales. The recurrent states are updated at every decision and reset at the beginning of an episode. During training, history is reconstructed in true trajectory order using only executed actions. If a short window contains fewer than w steps, the first frame is used for padding with a validity mask; missing actions are zero-filled and masked. For randomly sampled training segments, the true prefix before the segment first restores the recurrent state, after which losses are computed within the segment. Stage II uses cached or replayed true history summaries from the frozen modules and never updates history with predicted futures. Let $m_t = \{m^{(j)}_t\}_{j=1}^J$. The current and future representations z come from the same frozen target encoder, keeping the representation space consistent across both stages; x_t additionally fuses proprioception to extract manipulation evidence. Under the same history condition (e_t, m_t, g), the progress readout evaluates the current and ground-truth future representations. The current coordinate is $\hat{p}_t = P_\psi(z_t; e_t, m_t, g)$ and the ground-truth future coordinate is $\hat{p}_{t+H|t} = P_\psi(z_{t+H}; e_t, m_t, g)$. The readout P_ψ outputs a stage coordinate in $[0,1]$, and the target-transition head predicts

$$\widehat{\Delta}_t = (1 - \mathrm{sg}(\hat{p}_t))\, \sigma(T_\psi(e_t, m_t, g)), \tag{11}$$

This parameterization constrains the nominal transition to the remaining stage-coordinate range. We stop the gradient through $\hat{p}_t$ when computing the transition loss so that the multiplicative factor does not change the optimization target of current-coordinate regression. Coordinate supervision is

$$\mathcal{L}_{\text{coord}}(t) = \tfrac{1}{2}\big[c_t^{(i)}\,\text{Huber}(\hat{p}_t - p_t^{(i)}) + c_{\kappa(t,H)}^{(i)}\,\text{Huber}(\hat{p}_{t+H|t} - p_{\kappa(t,H)}^{(i)})\big]. \tag{12}$$

The transition head uses fixed-horizon transition supervision from demonstrations. The total Stage-I loss is

$$\mathcal{L}_{\text{PTM}} = \mathbb{E}_t\Big[\mathcal{L}_{\text{coord}}(t) + \lambda_\Delta c_{t,\Delta}\,\text{Huber}(\widehat{\Delta}_t - \Delta_t^\star)\Big]. \tag{13}$$

After Stage I, E_s, the recurrent modules, P_ψ, and T_ψ are frozen; Stage II only replaces the ground-truth future input with the generated future.

4.3 Bidirectional Progress–Dynamics Coupling

BPDC uses the nominal target both for future generation and result readback. The bidirectional coupling consists of two paths: the progress condition enters future generation, and the generated future returns to the progress readout. We first project current progress, target transition, and history summary into condition tokens. The semicolons denote concatenation along the feature dimension, so C_t^p is a condition vector; after projection, it is replicated to the query-sequence length as attention keys and values, where $C_t^p = [E_p(\hat{p}_t); E_\Delta(\widehat{\Delta}_t); E_m(m_t)]$. A gated residual-attention block then modulates the latent action queries, where α_t is a per-query-channel gate with the same feature dimension as q_t:

$$\widetilde{q}_t = q_t + \alpha_t\,\text{Attn}(q_t, C_t^p, C_t^p), \quad |\alpha_t| \le \alpha_{\max}. \tag{14}$$

The gate network predicts α_t from $(e_t, m_t, \hat{p}_t, \widehat{\Delta}_t)$, bounds it in $[-\alpha_{\max}, \alpha_{\max}]$ using $\tanh$, and uses zero initialization for its output layer.

The world model then generates the future representation:

$$\widehat{z}_{t+H} = F_\theta(h_t, \widetilde{q}_t). \tag{15}$$

The frozen readout evaluates this future under the same pre-execution history condition, yielding the readout transition:

$$\Delta_t^e = P_\psi(\widehat{z}_{t+H}; e_t, m_t, g) - \hat{p}_t. \tag{16}$$

The generated future is not written back into the history state. The target–readback consistency loss is

$$\mathcal{L}_{\text{PA}} = c_{t,\Delta}\,\text{Huber}\Big(\Delta_t^e - \text{sg}(\widehat{\Delta}_t)\Big), \tag{17}$$

Here sg denotes stop-gradient. P_ψ remains frozen in Stage II but is differentiable with respect to its input, so $\mathcal{L}_{\text{PA}}$ updates only the generated future and its upstream parameters.

The final training objective is

$$\mathcal{L}_{\text{PA-WAM}} = \mathcal{L}_{\text{WAM}} + \lambda_{\text{PA}}\mathcal{L}_{\text{PA}}. \tag{18}$$

Stage II updates the condition projections, gate, latent world model, and action expert, while the progress modules and target representation encoder remain frozen. The context encoder E uses the same trainable-layer range in all internal comparisons. During readback, only the parameters of P_ψ are frozen; gradients to its input are retained, allowing the progress loss to update the world model and query-modulation path, while the action expert is updated by the action loss. The action expert remains conditioned on the base context and latent future and does not receive an explicit progress variable directly.

At deployment, only current-progress readout, nominal-transition prediction, query modulation, one latent-future generation, and one action-block prediction are retained; the readback branch after future generation is used only for training.

5 Experiments

We evaluate whether PA-WAM improves long-horizon execution of a latent WAM through progress alignment. The experiments address four questions: (Q1) can PA-WAM consistently improve the

base WAM's task success rate on standard LIBERO? (Q2) does the gain extend to multi-stage MimicGen-D1 tasks and continuous CALVIN chains? (Q3) how do query modulation, BPDC readback, history-conditioned progress transitions, and multi-scale memory affect success? (Q4) can PA-WAM be deployed directly on a real robot and improve long-horizon execution without test-time search or iterative optimization? The main experiments measure complete-task success and completed chain length, while ablations test the roles of key components. Training and evaluation settings are given in Appendices A.3, B, and D.

5.1 Comparison with Baselines

Table 1 compares OpenVLA (Kim et al., 2024), $\pi_{0.5}$ (Physical Intelligence et al., 2025), LaWAM (Chen et al., 2026), Motus (Bi et al., 2026), ProgressVLA (Yan et al., 2026), and PA-WAM on LIBERO (Liu et al., 2023) and MimicGen-D1 (Mandlekar et al., 2023). LIBERO uses the Spatial, Object, Goal, and Long suites; MimicGen-D1 uses four multi-stage tasks covering continuous manipulation, assembly, and stacking. Methods within each benchmark share the same evaluation conditions. Simulation details are given in Appendix B.

Table 1: Task success rates (%) on LIBERO and MimicGen-D1

Method	LIBERO					MimicGen-D1				
	Spatial	Object	Goal	Long	Avg.	Coffee Prep.	Kitchen	3-Piece Assembly	Stack Three	Avg.
OpenVLA	84.7	88.4	79.2	53.7	76.5	48.0	55.0	52.0	78.0	58.3
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9	70.0	82.0	79.0	94.0	81.3
LaWAM	99.4	99.6	98.4	97.0	98.6	72.0	81.0	80.0	94.0	81.8
Motus	96.8	**99.8**	96.6	97.6	97.7	74.0	**84.0**	82.0	95.0	83.8
ProgressVLA	88.2	96.4	87.2	66.2	84.5	73.0	82.0	81.0	94.0	82.5
PA-WAM	**99.6**	**99.8**	**99.2**	**98.2**	**99.2**	**75.0**	83.0	**83.0**	**96.0**	**84.3**

On LIBERO, PA-WAM achieves the highest mean success rate of 99.2%, improving over LaWAM by 0.2, 0.2, 0.8, and 1.2 percentage points on Spatial, Object, Goal, and Long, respectively, with larger gains on Goal and Long. Compared with ProgressVLA, PA-WAM improves the mean by 14.7 points and reaches 98.2% on Long versus 66.2%. On MimicGen-D1, PA-WAM achieves 84.3%, exceeding LaWAM and ProgressVLA by 2.5 and 1.8 points, respectively, with particularly clear gains on Coffee Preparation and 3-Piece Assembly. Although Motus slightly outperforms PA-WAM on Kitchen (84.0% versus 83.0%), PA-WAM obtains a higher average across the four tasks than Motus (84.3% versus 83.8%), indicating that history-conditioned fixed-horizon progress transitions are especially useful for preserving state continuity across multi-stage manipulation.

To evaluate execution on longer task chains, we further compare continuous-subtask completion on CALVIN (Mees et al., 2022). We use the official $ABC \rightarrow D$ protocol, training in environments A, B, and C and testing in environment D. We report the success rates (%) for completing at least 1–5 consecutive subtasks and the mean completed length. Results are shown in Table 2; training settings, chain-execution rules, and metric definitions are given in Appendix B.3.

On CALVIN $ABC \rightarrow D$, PA-WAM achieves a 67.3% success rate on five-step chains, 3.8 percentage points above LaWAM, and increases the mean completed length from 3.85 to 4.01. Compared with ProgressVLA, it improves success at every chain length and raises the mean length by 0.28 subtasks, reaching 67.3% versus 52.0% on five-step chains. The increasing advantage with chain length indicates that history-conditioned progress transitions mainly improve state and stage continuity across subtasks, reducing stage confusion and error accumulation beyond local action prediction.

To separate first-subtask success from subsequent retention, Appendix C.2 reports chain retention after the first subtask. Conditioned on sequences that successfully complete the first subtask, PA-WAM completes all five tasks in 70.0% of sequences versus 67.1% for LaWAM; the corresponding three-task rates are 81.7% and 78.4%. Thus, the advantage of PA-WAM comes not only from a higher first-subtask success rate but also from more reliable continuation after entering the chain.

Table 2: Continuous-task evaluation on CALVIN $ABC \rightarrow D$

Method	1-step	2-step	3-step	4-step	5-step	Mean chain length
MCIL	30.4	1.3	0.2	0.0	0.0	0.32
RT-1	53.3	22.2	9.4	3.8	1.3	0.90
HULC	41.8	16.5	5.7	1.9	1.1	0.67
RoboFlamingo	82.4	61.9	46.6	33.1	23.5	2.48
GR-1	85.4	71.2	59.6	49.7	40.1	3.06
ProgressVLA	95.2	84.8	73.6	67.2	52.0	3.73
LaWAM	94.6	84.4	74.2	68.3	63.5	3.85
PA-WAM	**96.2**	**86.8**	**78.6**	**72.0**	**67.3**	**4.01**

5.2 Ablation Study

We perform component ablations on the four MimicGen-D1 tasks to test history information, target-conditioned generation, progress-readback supervision, and future reconstruction. All variants use the same data split, base-model initialization, and training budget, changing only the specified factor. Results in Table 3 are reported as mean $\pm$ standard error. Training and evaluation settings are given in Appendix B.2, and hyperparameter ablations are reported in Appendix C.1.

Table 3: Component ablations on MimicGen-D1 (%)

Variant	Coffee Prep.	Kitchen	3-Piece Assembly	Stack Three	Avg.
LaWAM	71.8 ± 0.8	81.2 ± 0.9	79.6 ± 1.2	94.6 ± 0.5	81.8 ± 0.3
LaWAM + History Only	72.6 ± 0.9	81.4 ± 0.8	80.2 ± 0.9	94.2 ± 1.1	82.1 ± 0.4
LaWAM + Linear Progress	72.2 ± 1.2	82.4 ± 0.8	81.8 ± 1.1	92.8 ± 1.3	82.3 ± 0.6
w/o Query Modulation	73.8 ± 1.1	82.2 ± 1.0	82.4 ± 0.8	94.8 ± 0.6	83.3 ± 0.4
w/o BPDC Readback	71.2 ± 1.4	79.2 ± 0.9	79.8 ± 0.8	94.2 ± 1.2	81.1 ± 0.8
w/o Context Transition	72.8 ± 0.9	81.2 ± 0.8	80.8 ± 1.2	95.2 ± 0.7	82.5 ± 0.4
w/o Multi-Scale Memory	74.2 ± 1.1	82.8 ± 0.8	82.2 ± 0.9	94.8 ± 0.7	83.5 ± 0.4
w/o Future Reconstruction	71.4 ± 1.2	79.8 ± 1.3	79.4 ± 0.8	93.8 ± 0.9	81.1 ± 0.6
PA-WAM	$\mathbf{74.8 \pm 0.8}$	$\mathbf{83.4 \pm 0.7}$	$\mathbf{83.2 \pm 1.0}$	$\mathbf{95.8 \pm 0.7}$	$\mathbf{84.3 \pm 0.3}$

History Only reaches 82.1%, only 0.3 points above LaWAM and 2.2 points below the complete model, while Linear Progress reaches 82.3%, showing that history alone and normalized time do not capture execution-stage differences. Replacing the context-conditioned transition with a task-average transition further reduces the mean to 82.5%, with larger drops on Coffee Preparation, Kitchen, and 3-Piece Assembly, indicating that fixed-horizon progress depends on the current observation and executed history. A single-scale memory achieves 83.5%, confirming that different temporal scales are useful for representing short-term contacts and longer execution history. Removing query modulation lowers the mean to 83.3%, whereas removing BPDC readback reduces it to 81.1%, showing that both progress-conditioned generation and frozen-readout consistency are necessary for the generated future to express the expected task advance.

5.3 Real-Robot Experiments

We use a RealMan RM65 six-degree-of-freedom arm with an EG2-4C2 electric parallel gripper. A fixed RealSense D435i and a wrist-mounted D405 provide two RGB views, while proprioceptive inputs include end-effector pose and gripper state. The three tasks are Sequential Button Press, Ordered Pick-and-Place, and Drawer Retrieve-and-Place. The first two require multiple targets to be completed in a specified order; the third requires opening a drawer before placing an object from inside it at a target location.

For each task, we collect 50 successful demonstrations and split complete episodes and scene layouts into 35 training and 15 validation demonstrations. We compare $\pi_{0.5}$, MemoryVLA, LaWAM, and

PA-WAM; each method executes 30 trials per task using the same recorded object initializations and instructions. The execution budgets for the button, pick-and-place, and drawer tasks are 400, 800, and 600 policy-control steps, respectively. A step is counted whenever an action is sent and executed; a trial succeeds only after the specified order and final goal are completed, and is unsuccessful if the step limit is reached first. Table 4 reports complete-task success rates, with Avg. denoting the unweighted mean across the three tasks.

Table 4: Real-robot task success rates (%)

Method	Sequential Button Press	Ordered Pick-and-Place	Drawer Retrieve-and-Place	Avg.
$\pi_{0.5}$	23.3	40.0	80.0	47.8
MemoryVLA	56.7	63.3	73.3	64.4
LaWAM	76.7	86.7	83.3	82.2
PA-WAM	**83.3**	**93.3**	**86.7**	**87.8**

PA-WAM completes 79 of 90 trials, compared with 74 for LaWAM. The per-task counts are PA-WAM $(25, 28, 26)/30$ and LaWAM $(23, 26, 25)/30$, giving a 5.6-point mean success-rate gap. Evaluation is based on completing the full operation sequence, requiring the policy to carry scene states created by earlier operations into later targets; the success rule is given in Appendix D.

At deployment, the policy sends actions at 10 Hz, predicts 12 actions per round, and executes the first four, giving a nominal replanning rate of 2.5 Hz; one replanning event is not a control step. History is retained within an instruction and cleared at the beginning of each episode. Future readback is used only during training, while execution uses one latent-future generation and action-block prediction. The real-robot comparison therefore measures the trained policies in closed-loop execution. Control adaptation and execution records are provided in Appendix D.

6 Conclusion

We presented the Progress-Aligned World–Action Model, PA-WAM, which introduces history-conditioned task progress into latent-future learning. The method aligns successful demonstrations to construct a continuous stage coordinate, uses PTM to estimate a nominal fixed-horizon transition from current manipulation evidence and multi-scale history, and uses BPDC to apply the same transition to latent-query modulation and future-readback supervision. Progress consistency is optimized jointly with future reconstruction and action imitation, giving the world model a task-stage-aware constraint while it learns future state changes. The readback branch is used only during training; deployment still uses a single latent-future generation and action-block prediction without test-time search or iterative optimization.

Experiments on LIBERO and MimicGen-D1, together with RM65 real-robot experiments, show gains over the base model on continuous manipulation, assembly, stacking, and ordered execution. The ablations further show that the gain cannot be explained by adding history alone: the context-conditioned transition provides a target adapted to the current execution state, readback supervision turns that target into a constraint on the generated result, and base future reconstruction remains important for execution performance. These results support supplementing general future-representation learning with task-progress constraints, using the generation condition and training target together to improve support for long-horizon execution.

The method assumes that successful demonstrations can be aligned to a one-dimensional monotonic stage coordinate, making it suitable for long-horizon tasks with a consistent stage order. Its applicability is limited when tasks contain multiple valid orders, parallel subtasks, stage regressions, or non-monotonic recovery, where a single coordinate may not represent the underlying progress structure; richer representations such as branching stage graphs or multidimensional coordinates are needed in such cases.

Reproducibility Statement

Sections 3 and 4 specify the progress supervision, PTM, BPDC, and two-stage objectives. LaWAM and PA-WAM use the same data splits, model initialization, action-block length, and training budget, and the internal ablations follow these settings. Section 5 reports the experimental results; Appendix A.3 summarizes training parameters, while Appendices B and D provide the simulation and real-robot evaluation protocols.

AI Use Statement

This work used generative AI tools to assist with idea organization, literature-search leads, experimental-design wording, drafting, language editing, structural organization, and format checking. The authors manually verified, selected, and revised all AI-assisted content and take full responsibility for the research design, technical judgments, experimental execution, analysis, conclusions, and final manuscript.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Chelsea David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *CoRL*, 2022.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *CVPR*, 2026.

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. In *RSS*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026.

Marco Cuturi and Mathieu Blondel. Soft-DTW: A differentiable loss function for time-series. In *ICML*, 2017.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *CVPR*, 2019.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *RSS*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *ICLR*, 2022.

Yanjiao Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. In *ICLR*, 2026.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4), 11205–11212, 2022.

Shivam Jain, Sachit Sachdeva, and Rohan Paul. Enabling long(er) horizon imitation for manipulation tasks by modeling subgoal transitions. In *CoRL*, 2025.

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. In *CoRL*, 2024.

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.

Junchi Liang and Abdeslam Boularias. Self-supervised learning of long-horizon manipulation tasks with finite-state task machines. In *L4DC*, 2021.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS Datasets and Benchmarks Track*, 2023.

Xiao Liu, Yuguang Yang, Xi Wang, Kai Jiang, Cheng Chi, Yong Xu, Wenchao Ding, Yilun Chen, and Yan Wang. StageWAM: Joint-Embedding Stage Prediction for World-Action Models in Robot Manipulation. *arXiv preprint arXiv:2608.10780*, 2026.

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

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *CoRL*, 2023.

Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *ICLR*, 2020.

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

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators. In *ICLR*, 2024.

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 *ICLR*, 2024.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018.

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. *arXiv preprint arXiv:2508.19236*, 2025.

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

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haoxiang Wang, Jiajun Tang, Kyle Stachowicz, et al. MEM: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Peng Su, Zhaowen Li, Yan Huang, and Liang Wang. DiM-WAM: World action modeling with diverse historical event memory. *arXiv preprint arXiv:2606.27677*, 2026.

Mingxin Wang, Bin Hu, Bin Qian, Kaitao Jiang, Haoning Wu, Feng Yan, Bowen Jing, Ruiyang Hao, Enyi Wang, Kangning Niu, Yandan Yang, Mu Xu, Yan Wang, Houde Liu, and Tianlun Li. ST-WAM: Semantic-temporal world action model for robust manipulation under visual distribution shifts. *arXiv preprint arXiv:2607.28993*, 2026.

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

Fan Yang, Yuting Su, Xiaobo Wang, Yuncheng You, Fugui Fan, Yuting Wu, Minghui Wu, Chenxu Zhao, Jiahong Ning, and Peiguang Jing. LiLa-WAM: Lightweight latent reasoning world-action model for robotic manipulation. *arXiv preprint arXiv:2608.03701*, 2026.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026.

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.

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

A Implementation Details

A.1 Model Configuration

PA-WAM organizes stage estimation, target transitions, and future readback in the same progress coordinate system. The local-evidence encoder is a two-layer causal Transformer with hidden width 256, four attention heads, and feed-forward width 1,024. The total history width is 384, formed by three 128-dimensional memory states that retain short-term contact information, the current manipulation stage, and longer execution history, respectively. The progress readout and transition head use a $256 \to 128 \to 1$ MLP; the current observation and predicted future share the readout. Each of the three condition projections has dimension 128, and their concatenation is mapped to the query width. The gate output layer is zero-initialized.

Target encoder and memory projections. The current representation z_t and future representation z_{t+H} are both extracted by a DINOv3 ViT-B/16 target vision encoder. This encoder remains frozen in both Stage I and Stage II and is used only to provide a consistent latent space for current and future representations; the inverse-dynamics teacher is also frozen. The local-evidence encoder outputs $e_t \in \mathbb{R}^{256}$, and each memory projection $B_j \in \mathbb{R}^{128\times256}$ corresponds independently to one time scale. Thus $m_t^{(j)} \in \mathbb{R}^{128}$, and with $J = 3$ the concatenated multi-scale memory has total dimension $3 \times 128 = 384$. The projections B_j are not shared across scales, while the recurrent update form is shared with different decay coefficients ρ_j.

A.2 Progress and Sample Construction

All visual observations are resized to 256×256, while proprioception and actions are normalized using statistics from each training set. During training, images use random crops with area ratios of 0.9 and 1.0 and color augmentation with magnitude 0.1; the same augmentation is shared within a temporal segment. Evaluation uses fixed image preprocessing. To construct samples, we first sample tasks uniformly and then sample a trajectory segment from the selected task. Each segment contains 32 consecutive decision times, and the true executed prefix before the segment is replayed to restore recurrent memory. For every time t in the segment, we pair it with the supervision endpoint H seconds later. Samples whose endpoint lies outside the trajectory are excluded from future supervision, and padding positions in short windows are excluded from losses by a validity mask.

The future-supervision horizon H is the physical time between the current observation and the target future, w is the number of recent decision times read by the local-evidence encoder, and J is the number of parallel history states. The temporal parameters for each benchmark are as follows.

Table 5: Temporal parameters for each benchmark

Benchmark	H (s)	w	J	Time constants τ_j (s)
LIBERO	1.2	4	3	$0.5/2/8$
MimicGen-D1	1.2	4	3	$0.5/3/12$
CALVIN	1.2	6	3	$0.5/2/8$
RM65	1.2	3	3	$1/4/16$

Memory decay coefficients are computed from the actual replanning interval δ_c and time constants τ_j:

$$\rho_j = \exp\left(-\frac{\delta_c}{\tau_j}\right). \tag{19}$$

Window length and the number of memory scales are implementation settings; sensitivity to key hyperparameters is analyzed in Appendix C.1.

Default loss and coupling parameters. The following default parameters are used for training.

Table 6: Default loss and coupling parameters

Parameter	**Value**
Alignment temperature γ	0.10
Target–readback consistency weight λ_{PA}	0.10
Gate amplitude limit $\alpha_{\max}$	0.10
Future reconstruction weight λ_z	0.10
Action imitation weight λ_a	1.0
Progress-transition weight λ_Δ	1.0
Monotonic-projection stabilizer ε_c	10^{-6}

A.3 Training Settings

Stage I updates only the local-evidence encoder E_s, multi-scale memory projections $\{B_j\}_{j=1}^{J}$, progress readout P_ψ, and nominal-transition head T_ψ. The base policy, action expert, latent world model, target vision encoder, and inverse-dynamics teacher remain frozen in Stage I. The Stage-I training parameters are as follows.

Table 7: PA-WAM Stage-I training parameters

Parameter	LIBERO	MimicGen-D1	CALVIN	RM65
Training steps	5,000	10,000	10,000	3,000
Batch size	256	256	256	128
Optimizer	AdamW	AdamW	AdamW	AdamW
β_1	0.9	0.9	0.9	0.9
β_2	0.95	0.95	0.95	0.95
ϵ	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Weight decay	0.01	0.01	0.01	0.01
LR scheduler	cosine decay	cosine decay	cosine decay	cosine decay
Warmup ratio	5%	5%	5%	5%
Minimum LR ratio	10%	10%	10%	10%
Gradient clipping threshold	1.0	1.0	1.0	1.0
Mixed precision	BF16	BF16	BF16	BF16
Progress-module learning rate	3×10^{-4}	2×10^{-4}	2×10^{-4}	3×10^{-4}

Stage-II training parameters are given in Table 8. Training steps count completed gradient updates, and batch size is the number of samples used in each update. Training steps, batch sizes, and module learning rates are set separately for each benchmark to match its task sequences and training data. All benchmarks use AdamW, cosine learning-rate decay, 5% linear warmup, a minimum learning rate equal to 10% of the peak rate, gradient clipping with norm limit 1.0, and BF16 mixed precision.

Table 8: PA-WAM Stage-II training parameters

Parameter	LIBERO	MimicGen-D1	CALVIN	RM65
Training steps	25,000	30,000	50,000	8,000
Batch size	256	128	128	64
Optimizer	AdamW	AdamW	AdamW	AdamW
β_1	0.9	0.9	0.9	0.9
β_2	0.95	0.95	0.95	0.95
ϵ	10^{-8}	10^{-8}	10^{-8}	10^{-8}
Weight decay	0.01	0.01	0.01	0.01
LR scheduler	cosine decay	cosine decay	cosine decay	cosine decay
Warmup ratio	5%	5%	5%	5%
Minimum LR ratio	10%	10%	10%	10%
Gradient clipping threshold	1.0	1.0	1.0	1.0
Mixed precision	BF16	BF16	BF16	BF16
Action expert (Flow) learning rate	1×10^{-4}	8×10^{-5}	5×10^{-5}	5×10^{-5}
BPDC conditioner learning rate	1×10^{-4}	8×10^{-5}	5×10^{-5}	5×10^{-5}
Action-query vector learning rate	1×10^{-4}	8×10^{-5}	5×10^{-5}	5×10^{-5}
Flow action-query vector learning rate	1×10^{-4}	8×10^{-5}	5×10^{-5}	5×10^{-5}
VLM–LAM bridge learning rate	3×10^{-5}	2×10^{-5}	1×10^{-5}	1.5×10^{-5}
LAM decoder learning rate	3×10^{-5}	2×10^{-5}	1×10^{-5}	1.5×10^{-5}

CALVIN uses a 50,000-step budget to cover longer continuous task chains, whereas RM65 uses a smaller batch and shorter budget to match the scale of real-robot data. Module learning rates are also separated by function: the action expert, BPDC conditioner, and two query-vector groups use higher rates so action generation and progress conditioning can adapt to each task; the VLM–LAM bridge and LAM decoder use lower rates to limit changes to pretrained representations.

B Simulation Protocol

B.1 LIBERO

Data and evaluation. The Spatial, Object, Goal, and Long suites contain 40 tasks in total, with 50 demonstrations per task split into 45 training and 5 validation trajectories, for 1,800/200 trajectories overall. Splits are made by complete trajectory, and alignment references are constructed within the training set for each task. Each task is evaluated for 100 episodes with the same initial states across methods; the execution limits are 220, 280, 300, and 520 control steps for the four suites, respectively. The policy uses a main view, wrist RGB, and language instruction at 20 Hz, predicts 24 actions at a time, executes the first 8, and replans. Success is determined by the environment's complete-task condition.

B.2 MimicGen-D1

Data and evaluation. We use four tasks: Coffee Preparation, Kitchen, 3-Piece Assembly, and Stack Three. Each has 500 successful demonstrations split into 450 training and 50 validation trajectories. Each task is evaluated for 100 episodes, with execution limits of 1,000, 1,000, 800, and 600 steps, respectively. The control frequency is 20 Hz; the policy predicts 24 actions and executes 8 before replanning. An episode terminates when the complete task succeeds or the step limit is reached. Training samples tasks uniformly, and history is retained across stage transitions within an instruction.

B.3 CALVIN

Data and task chains. We use the $ABC \rightarrow D$ protocol, with language-labeled segments from environments A, B, and C split by source trajectory into 15,200 training and 800 validation segments;

environment D is used for evaluation. Alignment references are constructed independently within each language subtask, and future supervision does not cross instruction boundaries. Evaluation uses a fixed set of 1,000 five-subtask sequences, with a limit of 360 control steps per subtask. The control frequency is 30 Hz; the policy predicts 36 actions and executes the first 6.

After a subtask succeeds, the physical scene is preserved, the instruction is switched, instruction-conditioned memory is reset, and the old action block is discarded before replanning from the current observation. A timeout on any subtask terminates the whole chain. Let L_n be the number of completed subtasks in sequence n. The cumulative success rate and mean chain length are

$$S_k = \frac{100}{N}\sum_{n=1}^{N}\mathbf{1}[L_n \geq k], \qquad \bar{L} = \frac{1}{N}\sum_{n=1}^{N} L_n = \sum_{k=1}^{5}\frac{S_k}{100}, \qquad N = 1000. \tag{20}$$

C Supplementary Results

C.1 Hyperparameter Sensitivity

We test key hyperparameters on MimicGen-D1: the consistency weight λ_{PA}, alignment temperature γ, future-supervision horizon H, number of memory scales J, and gate-amplitude limit $\alpha_{\max}$. Each experiment changes only one factor; data splits, model structure, action-block length, executed prefix, training budget, and evaluation protocol are unchanged. Table 9 reports success rates and means for the four tasks under each setting.

When varying H, we use the common sample set satisfying the maximum supervision horizon of 2.4 seconds. When varying J, total history-memory width is fixed at 384, with one-, three-, and six-scale memories. Temperature, loss weights, supervision horizon, memory scales, and gate amplitude are retrained and evaluated according to the settings in the table.

Table 9: MimicGen-D1 hyperparameter sensitivity (%)

Factor	Setting	Coffee Prep.	Kitchen	3-Piece Assembly	Stack Three	Avg.
Consistency weight	$\lambda_{\mathrm{PA}} = 0$	71.0 ± 1.5	79.0 ± 0.9	80.0 ± 0.8	94.0 ± 1.4	81.0 ± 1.0
	$\lambda_{\mathrm{PA}} = 0.05$	72.0 ± 1.3	80.0 ± 0.9	81.0 ± 1.1	94.0 ± 0.8	81.8 ± 0.5
	$\boldsymbol{\lambda_{\mathrm{PA}} = 0.10}$	**75.0 ± 0.9**	**83.0 ± 0.6**	**83.0 ± 1.2**	**96.0 ± 0.8**	**84.3 ± 0.3**
	$\lambda_{\mathrm{PA}} = 0.20$	69.0 ± 0.9	81.0 ± 1.6	79.0 ± 1.0	93.0 ± 1.3	80.5 ± 0.3
Alignment temperature	$\gamma = 0.05$	73.0 ± 0.9	80.0 ± 1.4	84.0 ± 0.4	93.0 ± 0.8	82.5 ± 0.7
	$\boldsymbol{\gamma = 0.10}$	**75.0 ± 0.9**	**83.0 ± 0.6**	**83.0 ± 1.2**	**96.0 ± 0.8**	**84.3 ± 0.3**
	$\gamma = 0.20$	69.0 ± 1.1	77.0 ± 0.8	78.0 ± 0.5	94.0 ± 1.0	79.5 ± 0.4
Prediction horizon	$H = 0.6\,\mathrm{s}$	69.0 ± 0.6	77.0 ± 1.3	81.0 ± 1.4	92.0 ± 0.9	79.8 ± 0.5
	$\boldsymbol{H = 1.2\,\mathrm{s}}$	**75.0 ± 0.9**	**83.0 ± 0.6**	**83.0 ± 1.2**	**96.0 ± 0.8**	**84.3 ± 0.3**
	$H = 2.4\,\mathrm{s}$	67.0 ± 0.9	79.0 ± 0.7	76.0 ± 1.0	91.0 ± 1.6	78.3 ± 0.6
Memory scales	$J = 1$	74.0 ± 1.2	83.0 ± 0.9	82.0 ± 1.0	95.0 ± 0.8	83.5 ± 0.5
	$\boldsymbol{J = 3}$	**75.0 ± 0.9**	**83.0 ± 0.6**	**83.0 ± 1.2**	**96.0 ± 0.8**	**84.3 ± 0.3**
	$J = 6$	75.0 ± 0.6	82.0 ± 1.3	77.0 ± 0.8	93.0 ± 0.9	81.8 ± 0.3
Gate amplitude	$\alpha_{\max} = 0.05$	73.0 ± 1.3	79.0 ± 0.7	82.0 ± 1.1	94.0 ± 1.3	82.0 ± 0.7
	$\boldsymbol{\alpha_{\max} = 0.10}$	**75.0 ± 0.9**	**83.0 ± 0.6**	**83.0 ± 1.2**	**96.0 ± 0.8**	**84.3 ± 0.3**
	$\alpha_{\max} = 0.20$	68.0 ± 0.7	80.0 ± 1.3	78.0 ± 0.9	93.0 ± 1.0	79.8 ± 0.5

C.2 CALVIN Chain Retention

Figure 2 reports chain retention after the first subtask. For the overall success rate S_k of completing at least k consecutive subtasks, define

$$R_k = 100\frac{S_k}{S_1}, \tag{21}$$

where S_1 is the success rate for the first subtask. Thus R_1 is fixed at 100%, and R_k is the fraction of sequences that complete at least k subtasks among those that complete the first.

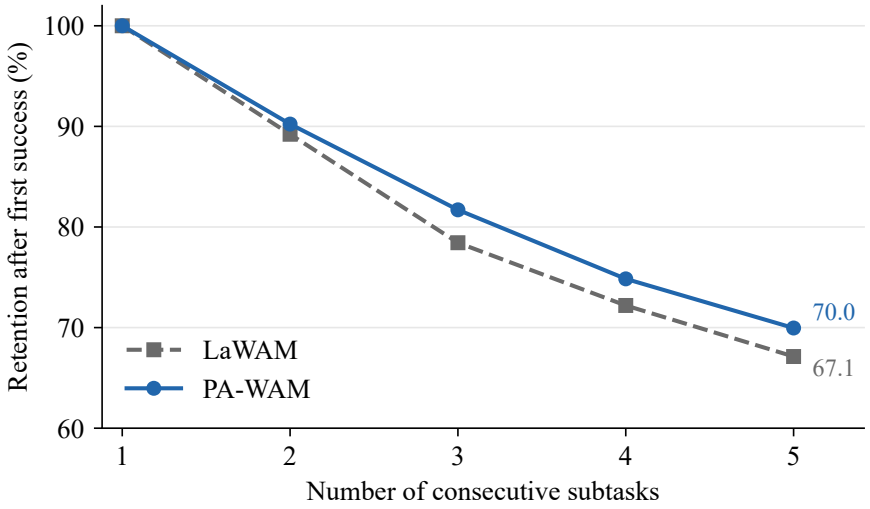

Figure 2: CALVIN chain retention. R_k is the fraction of sequences that complete at least k subtasks among sequences that complete the first subtask.

C.3 Execution Examples

This section presents representative execution sequences in simulation to visualize stage transitions and action continuity in long-horizon tasks. Figure 3 shows a complete sequence from LIBERO Long, and Figure 4 shows representative executions for the four multi-stage MimicGen-D1 tasks.

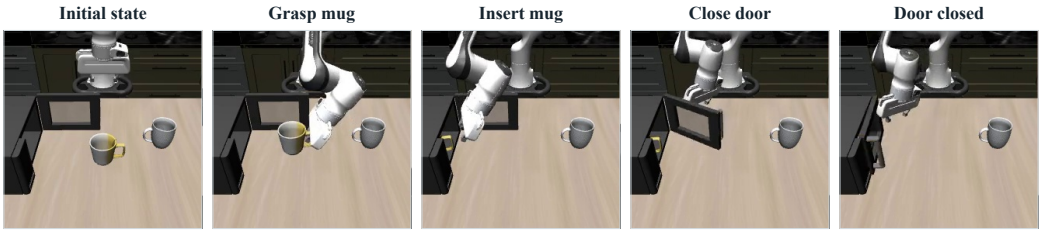

Figure 3: LIBERO Long execution example. The sequence proceeds through the initial state, grasping a cup, placing the cup in the microwave, closing the door, and task completion, illustrating continuous execution from object manipulation to subsequent environment interaction.

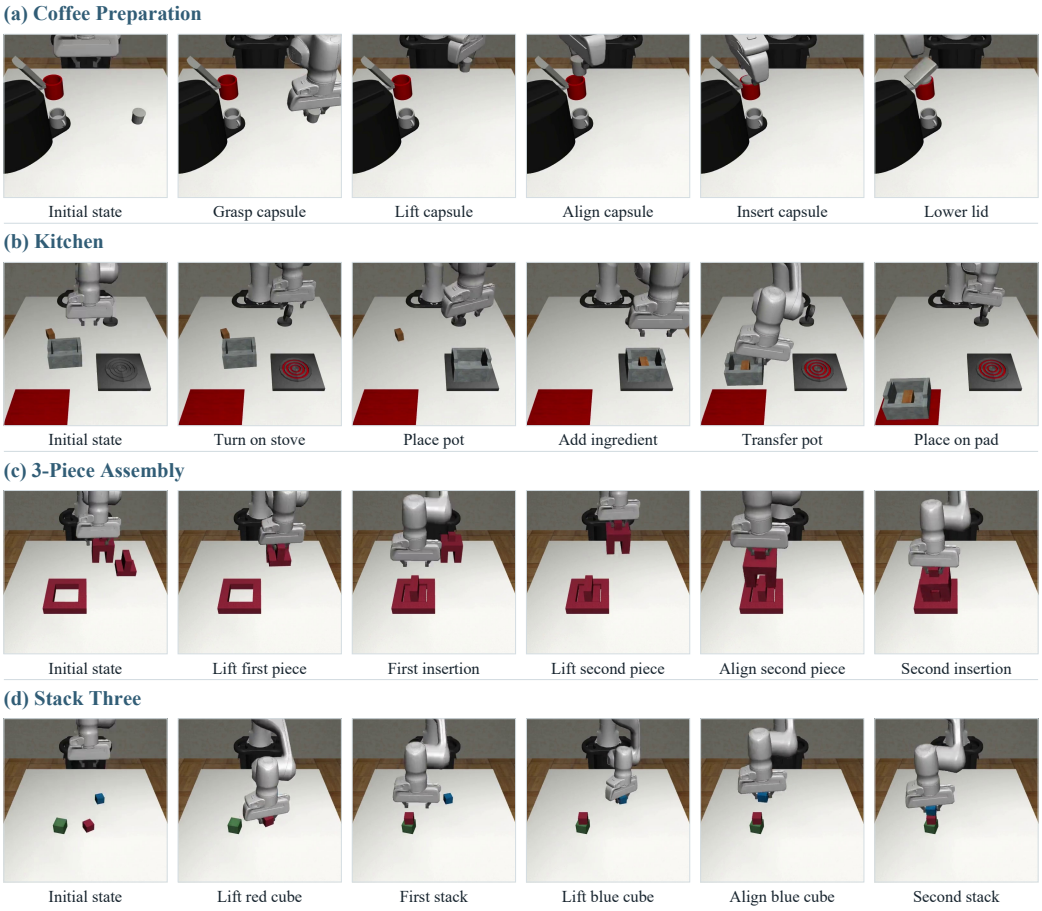

Figure 4: Execution examples for the four MimicGen-D1 tasks. The four rows correspond to Coffee Preparation, Kitchen, 3-Piece Assembly, and Stack Three; each row shows the initial state, key intermediate operations, and final placement or assembly in task order.

D Real-World Experiments

D.1 Robot Platform and Control Deployment

The platform uses a RealMan RM65 six-degree-of-freedom arm with an EG2-4C2 electric parallel gripper. The vision system consists of a fixed RealSense D435i and a wrist-mounted RealSense D405, used to observe the overall workspace and close-range contact regions, respectively. Proprioception includes end-effector pose and gripper state. Demonstration splits, trial counts, and success rules for the three tasks are given in Section 5.3.

The policy sends actions at 10 Hz, predicts 12 actions per round, and executes the first 4. Each control step corresponds to the actual execution of one end-effector-pose and gripper target; model calls, unexecuted predictions, and episode resets are not counted. The control-step limits for Sequential Button Press, Ordered Pick-and-Place, and Drawer Retrieve-and-Place are 400, 800, and 600, respectively. An episode terminates immediately upon task completion and is marked unsuccessful if its limit is reached first.

At each replanning step, the policy reads the two RGB views and proprioception and updates progress memory from the true execution history; history is cleared at the start of an episode. Actions use the same end-effector frame, pose-increment scale, and gripper-control convention as the training data. Future readback is used only during training. At deployment, one latent future and one action block are predicted, the first 4 actions are executed, and the policy replans.

D.2 Execution Records

Execution records are indexed by method, task, initial configuration, and trial number, and store the success flag, control-step count, action-block start positions, and two-view videos. To complement the task-level success rates in Table 4, Figures 5–7 show keyframe sequences from one successful trial for each real-robot task. Each figure retains eight stage states in actual execution order; the numbers indicate temporal order and the stage titles identify the current manipulation target.

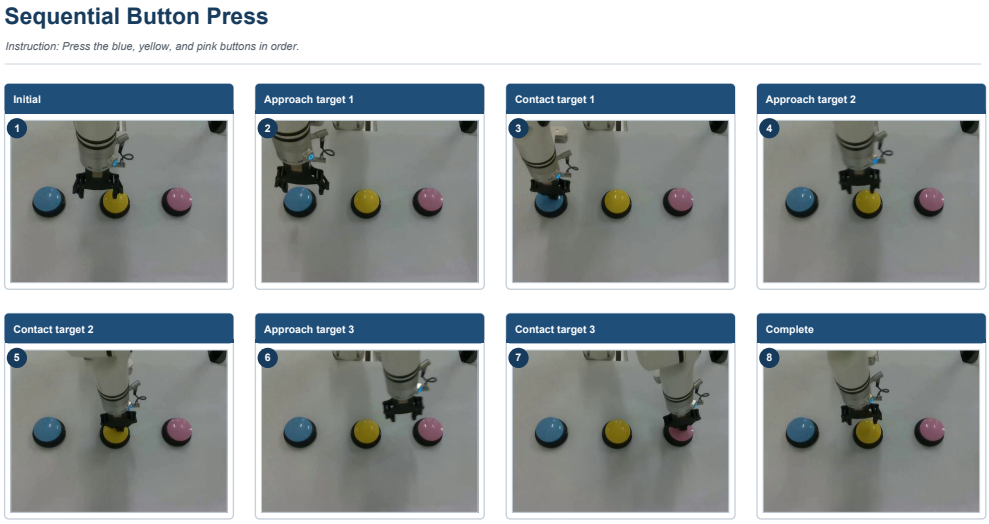

Figure 5: Sequential Button Press execution record. The keyframes show the robot approaching and pressing the blue, yellow, and pink buttons in the specified order.

The keyframes for Sequential Button Press cover the initial state, approaching and contacting the three buttons, and final completion. After each button, the robot relocates to the next one, showing that success requires both individual contacts and ordered transitions between multiple targets.

Ordered Pick-and-Place contains three consecutive object transfers: a green cube, an orange cube, and an orange semicircle. The first group of keyframes shows the green cube being grasped and placed in the tray; the next groups process the orange cube and then the orange semicircle while preserving the previous placements. The final state contains all three objects in their target region. This sequence shows that the scene state created by each earlier stage is retained for later target switches.

The Drawer Retrieve-and-Place chain consists of opening the drawer, approaching and grasping the object inside, transferring it, and completing the placement. The workspace changes from a closed to an open drawer, after which the gripper carries the object to the tray on the right. The sequence therefore shows continuity across a container-state change, retrieval of an occluded object, and subsequent placement.

Together with the success-rate statistics, the three execution records show that real-robot evaluation concerns the complete task chain rather than an isolated contact or placement. The representative sequences also make the model's continuous closed-loop execution after target switches and intermediate state changes explicit.

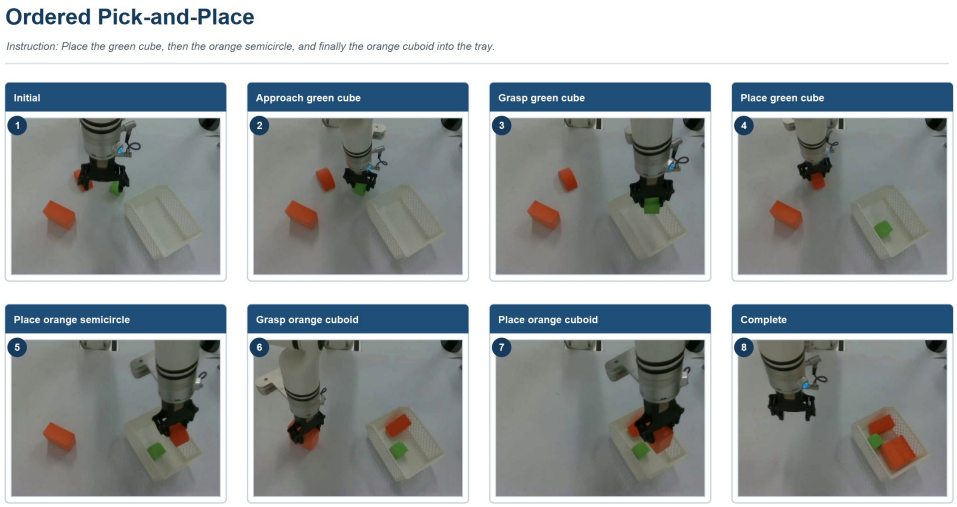

Figure 6: Ordered Pick-and-Place execution record. The keyframes show the robot sequentially grasping and placing the green cube, orange cube, and orange semicircle.

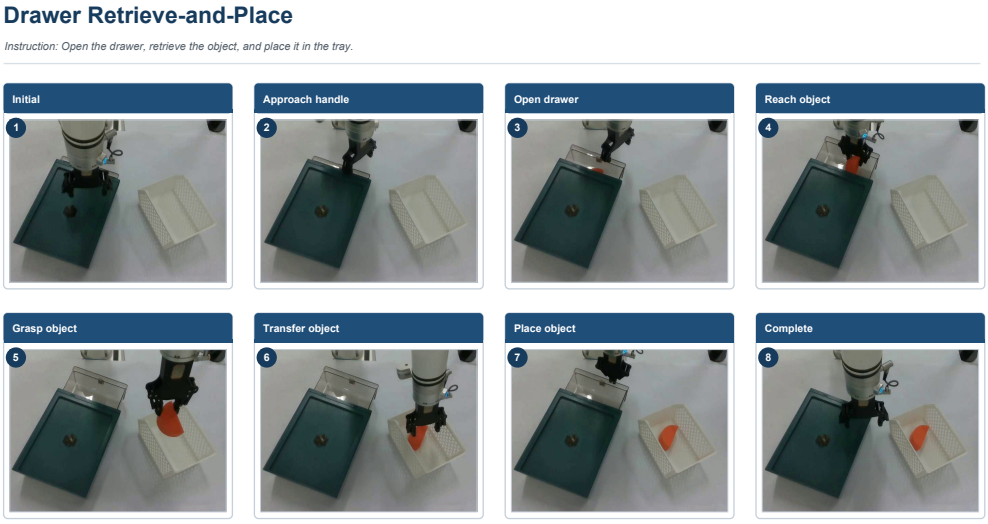

Figure 7: Drawer Retrieve-and-Place execution record. The keyframes show the robot opening the drawer, retrieving the object inside, and placing it in the tray.