PATCHVLA: PROCESS-AWARE TEST-TIME CRITIC-GUIDED SEARCH FOR LONG-HORIZON MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable action selection is essential for completing long-horizon robotic manipulation tasks. However, comparing candidates from generative vision-language-action (VLA) policies remains challenging because plausible action chunks from the same observation can imply different interaction timelines. To address this challenge, we propose PatchVLA, a process-aware framework for critic-guided action search at test time. A process-conditioned flow policy generates candidate chunks, each paired with temporally aligned forecasts of interaction phase, local progress, and subtask boundaries. A sequence critic evaluates the actions and forecasts together with the current observation. Calibrated critic scores and auxiliary penalties guide MPPI-like updates to the latent proposal distribution while learned weights remain fixed. The controller executes a short prefix of a selected candidate and replans from the next observation. Process forecasts are learned from demonstrations with expert anchors and human-reviewed VLM annotations, while recorded candidate outcomes supervise the critic. On LeHome-Fold and LIBERO-Long, PatchVLA achieves 76.25% and 96.20% success, respectively, exceeding the official $\pi_{0.5}$ baselines by 9.17 and 2.80 percentage points. In a separate fixed-model LeHome-Fold study, two-round refinement achieves 73.96% success versus 70.21% for one-round selection. Together, these results show that candidate-aligned process forecasts and critic-guided latent search improve long-horizon action selection. The anonymous project page is available at `https://patchvla-anonymous.github.io/`.

1 INTRODUCTION

An action chunk commits a robot to more than a direction of motion: it also determines when to establish contact, continue an interaction, or transition to the next subtask. Generative vision-language-action (VLA) policies can propose multiple plausible chunks from the same observation and instruction (Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). Although these candidates share the same starting scene, they can imply different interaction timelines. The control problem is therefore to judge how each candidate would advance the current interaction, and how that judgment can improve subsequent proposals before execution.

Intermediate reasoning organizes manipulation into steps, while value-guided selection compares sampled actions (Zawalski et al., 2024; Nakamoto et al., 2024). We associate each candidate with a process forecast (Figure 1). Local progress describes advancement within a subtask and can reset when the next subtask begins, so phase and boundary forecasts provide the context for interpreting each reset alongside the candidate action sequence.

We introduce PatchVLA: **P**rocess-**A**ware **T**est-Time **C**ritic-Guided Search for Long-**H**orizon Manipulation. The current context conditions a flow-based action expert, and a candidate-conditioned readout predicts phase, local progress, and boundary probability along each proposed chunk. A sequence critic reads the current observation and the candidate's action–process sequence to produce a scalar score. Calibrated scores and auxiliary penalties update the latent proposal distribution with learned weights fixed. The controller executes a short prefix of one evaluated candidate and replans from the next observation.

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.

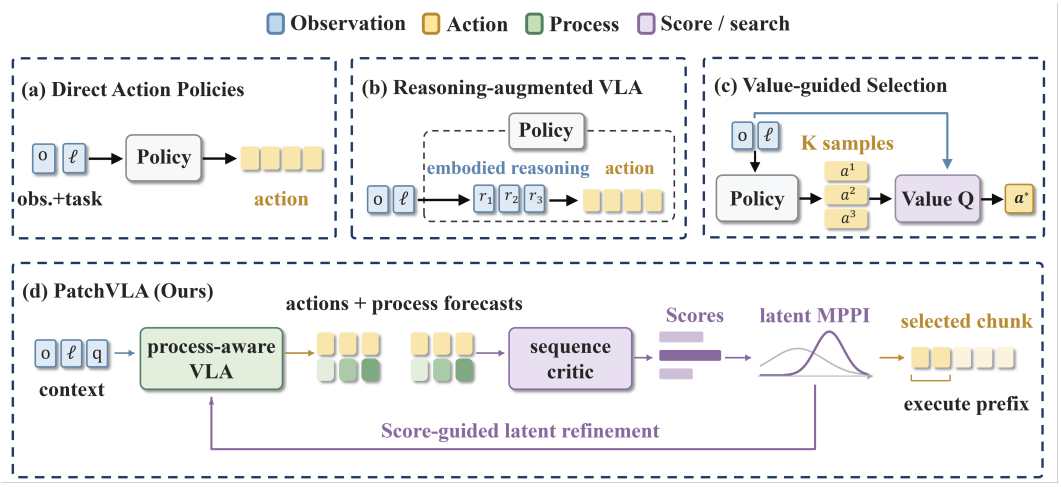


Figure 1: **Process structure connects generation and selection.** The upper panels illustrate representative design patterns: direct action generation, embodied reasoning, and value-guided selection. The lower panel pairs candidate actions with aligned process forecasts for sequence evaluation. The refinement feedback changes the latent proposal distribution while the learned policy remains fixed, and execution uses one evaluated candidate.

The shared representation is learned from temporally annotated demonstrations, using sparse expert anchors and offline VLM-assisted labeling. Recorded candidate continuations supply a different supervision signal: their outcomes train the critic to evaluate alternatives from a shared starting state (Figure 3).

We evaluate PatchVLA on LeHome-Fold for deformable contact and LIBERO-Long for sequential object interactions.

Our contributions are (i) a candidate-aligned process representation that supplies sequence evaluation with phase, local-progress, and boundary forecasts alongside proposed actions; (ii) a control procedure that turns these evaluations into latent proposal updates while executing intact candidates; and (iii) closed-loop evaluation on two manipulation benchmarks, including fixed-model candidate-budget comparisons and execution analyses.

2 Related Work

Generative action policies. Generalist robot policies connect perception and language with action learning across diverse demonstrations and shared interfaces (Han et al., 2026; Brohan et al., 2023; Open X-Embodiment Collaboration et al., 2023; Khazatsky et al., 2024; Octo Model Team et al., 2024). Action generation spans token prediction, chunked imitation, diffusion, and flow matching (Kim et al., 2024; Zhao et al., 2023; Chi et al., 2023; Black et al., 2024; Physical Intelligence et al., 2025). Lightweight models and adaptive truncation address deployment cost (Shukor et al., 2025; Zhang et al., 2026). PatchVLA builds on a learned flow policy to evaluate and improve sampled chunks before execution.

Structured representations for manipulation. Intermediate representations can organize actions through language plans, executable programs, and embodied reasoning, or describe spatial interactions through affordances (Ahn et al., 2022; Huang et al., 2022; Liang et al., 2022; Zawalski et al., 2024; Xu et al., 2025). Predictive representations further span future visual or latent states and shared geometric structure for cross-embodiment transfer (Wu et al., 2023; Yuan et al., 2026; Li et al., 2026a; Niu et al., 2026). PatchVLA describes a proposed continuation through phase, local progress, and boundary forecasts aligned with its actions. This representation gives the sequence critic a candidate-specific interaction timeline.

Candidate evaluation and latent steering. Guided trajectory sampling and policy optimization extend generative policies beyond direct imitation (Janner et al., 2022; Wang et al., 2022; Ren et al.,

2025). Candidate evaluation can score generalist-policy proposals or allocate computation through uncertainty-aware critics (Nakamoto et al., 2024; Li et al., 2026b). Latent-noise steering can also be learned through reinforcement learning with a fixed generator (Wagenmaker et al., 2025). PatchVLA uses candidate sequence evaluations to update the latent proposal distribution at test time while keeping learned weights fixed. Its proposal updates combine exponential weighting and elite moments (Williams et al., 2017; de Boer et al., 2005).

3 METHOD

At each replanning decision, PatchVLA generates candidate chunks, forecasts the process aligned with each chunk, scores the resulting action–process sequences, refines the latent proposal, and executes a prefix of one evaluated candidate. Figure 2 summarizes this loop, while the training procedure and supervision are given in Appendix A.

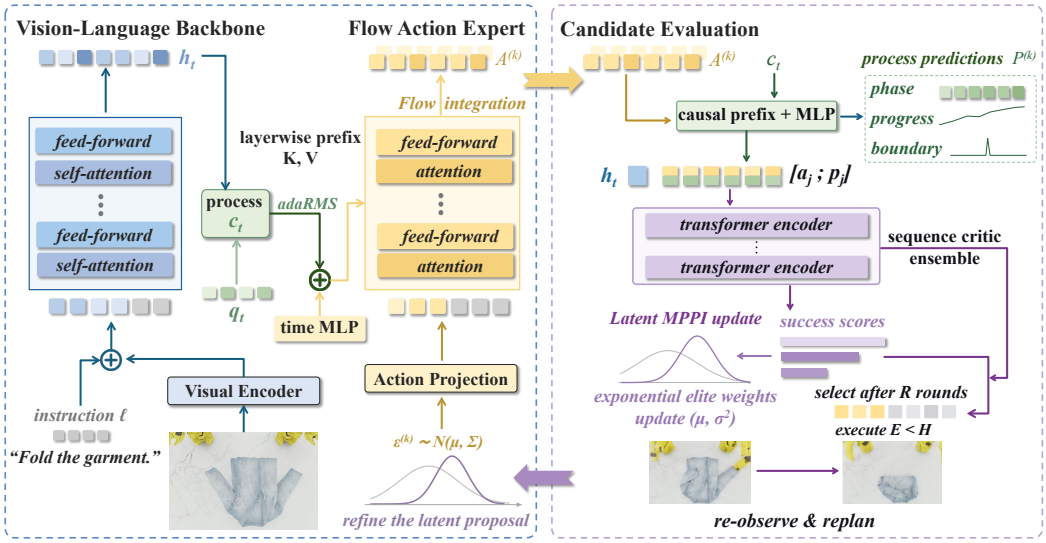


Figure 2: **PatchVLA framework.** (a) Current process context conditions the flow expert. (b) Action-aligned forecasts inform sequence evaluation and latent refinement, followed by prefix execution and replanning. Here p_j denotes $\mathbf{p}_j = (\hat{z}_j, \hat{p}_j, \hat{b}_j)$, $\Sigma = \mathrm{diag}(\sigma^2)$, and E the prefix length. The illustrated causal-prefix predictor is the LeHome-Fold implementation; LIBERO-Long uses a shared MLP under Eq. (2).

3.1 PROCESS-CONDITIONED ACTION GENERATION

Let $x_t = (o_t, \ell, q_t)$ contain current images, the instruction, and available proprioception. A $\pi_{0.5}$ backbone (Physical Intelligence et al., 2025) provides a visual–language prefix cache $\mathcal{K}_t$ and pooled features h_t. To condition action proposals on the current interaction, a process encoder supplies $c_t = E_\phi(x_t)$. Its learned projection augments the flow-time embedding used by the action expert's adaptive RMS normalization. Starting from Gaussian noise ϵ, the conditioned flow defines a candidate chunk:

$$A = G_\theta(x_t, c_t, \epsilon), \qquad \frac{dA_\tau}{d\tau} = v_\theta(A_\tau, \tau; \mathcal{K}_t, c_t), \quad A_1 = \epsilon, \quad \tau : 1 \to 0. \tag{1}$$

The chunk horizon is H, and flow time τ differs from control step t. Different noise samples yield candidates for the same observation and process context. Each chunk is decoded into executable actions using the checkpoint's fixed action transforms.

3.2 CANDIDATE-ALIGNED PROCESS FORECASTS AND EVALUATION

At decision time t, the context c_t is computed once from the current observation and shared by all candidates. The predictor associates each candidate $A^{(k)}$ with its own process forecast at the

corresponding action steps:

$$P^{(k)} = F_\phi(x_t, A^{(k)}) = \{(\hat{z}_j^{(k)}, \hat{p}_j^{(k)}, \hat{b}_j^{(k)})\}_{j=0}^{H-1}, \tag{2}$$

Here $\hat{z}$ is an interaction-phase distribution, $\hat{p}$ local progress, and $\hat{b}$ a subtask-boundary probability. The phase vocabulary is transit, pre-contact alignment, engage, manipulate, disengage, settle, and verify. Phases can recur across semantic subtasks. Local progress can reset at a subtask transition, with phase and boundary predictions placing the reset beside the corresponding proposed motion.

The sequence critic turns this description into a score for comparing candidates. It combines an observation token projected from h_t with position-encoded tokens formed from $[a_j; \hat{z}_j; \hat{p}_j; \hat{b}_j]$. A Transformer reads the complete sequence and predicts a success logit. For an ensemble of M members, the deployment score is

$$S^{(k)} = \sigma\left(\alpha \frac{1}{M} \sum_{m=1}^{M} C_{\psi_m}(\mathrm{sg}(h_t), \mathrm{sg}(A^{(k)}), \mathrm{sg}(P^{(k)})) + \beta\right). \tag{3}$$

Each C_{ψ_m} outputs a logit. S is the affine-calibrated success score used by the controller. The parameters (α, β) are fitted on a held-out calibration split. Ensemble disagreement is computed from uncalibrated member probabilities before entering the planning cost. Here sg denotes stopped gradients: critic learning reads detached observation, action, and process features.

3.3 Latent refinement and closed-loop execution

We denote the search schedule by $K \times R$, where K is the number of candidate action chunks per round and R is the number of search rounds within a single replanning decision. To improve subsequent proposals, round $r = 0, \ldots, R-1$ draws K latent samples $\epsilon_{r,k}$ from $\mathcal{N}(\mu_r, \mathrm{diag}(\sigma_r^2))$ with learned weights fixed. Each sample is decoded, paired with its forecast, and evaluated at the same observation. Episode-local state ξ retains planning history, including the preceding executed command. Calibrated success and configured penalties define the planning cost:

$$J_{r,k} = -\log \max(S_{r,k}, p_{\min}) + \lambda_u u_{r,k} + \lambda_s D_{\mathrm{smooth}}(A_{r,k}) + \lambda_c D_{\mathrm{cont}}(A_{r,k}; \xi), \tag{4}$$

Here $p_{\min}$ is a numerical probability floor and $u = \mathrm{Std}_m[\sigma(C_{\psi_m}(h, A, P))]$ measures ensemble disagreement. The optional penalties measure roughness and continuity with the command retained in ξ. Zero weights give success-only search. Lower-cost candidates then guide the next proposal. For the lowest-cost $\lceil \eta K \rceil$ candidates $\mathcal{E}_r$, with elite fraction $\eta \in (0, 1]$, exponential weights determine latent moments and their update:

$$\begin{aligned} w_{r,k} &\propto \mathbf{1}[k \in \mathcal{E}_r] \exp(-(J_{r,k} - J_{r,\min})/T), \quad \sum_k w_{r,k} = 1, \\ \bar{\epsilon}_r &= \sum_k w_{r,k} \epsilon_{r,k}, \qquad \widehat{\sigma}_r^2 = \sum_k w_{r,k} (\epsilon_{r,k} - \bar{\epsilon}_r)^{\odot 2}, \\ \mu_{r+1} &= (1-\rho)\mu_r + \rho \bar{\epsilon}_r, \\ \sigma_{r+1}^2 &= \mathrm{clip}((1-\rho)\sigma_r^2 + \rho \widehat{\sigma}_r^2, \sigma_{\min}^2, \sigma_{\max}^2). \end{aligned} \tag{5}$$

Here $J_{r,\min} = \min_{k \in \mathcal{E}_r} J_{r,k}$. T controls weight concentration and ρ sets the update rate. This gradient-free, MPPI-like procedure combines exponential cost weighting with elite moment updates (Williams et al., 2017; de Boer et al., 2005). Later samples thus depend on the action–process sequences evaluated in earlier rounds.

We update the latent proposal through weighted moments and select one complete evaluated candidate from the final round, using either the minimum-cost elite or the configured weighted medoid. The controller executes its prefix, then refreshes the observation and process context before replanning. Prefix adaptation, proposal reuse, and state updates follow the benchmark-specific settings in Appendix B.5.

3.4 Data preparation and staged learning

Demonstrations $\mathcal{D}_{\mathrm{demo}}$ supervise actions and process forecasts (Figure 3). Sparse expert anchors $\mathcal{A}_{\mathrm{exp}}$ guide offline VLM-assisted subtask labeling and human review. Semantic subtask annotations

define the temporal segments used with interaction cues to construct phase, local-progress, and boundary targets. Recorded candidate continuations supply outcome supervision for the critic.

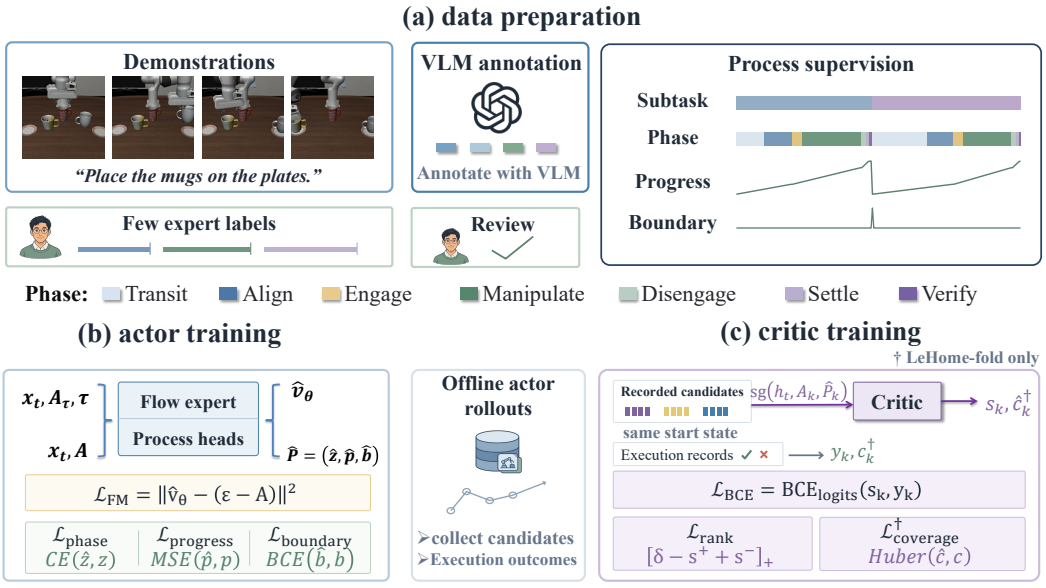


Figure 3: **Training the mappings in Figure 2.** (a) Expert anchors and VLM-assisted labeling/review yield process supervision. (b) Noisy demonstrations train flow prediction; clean actions train process forecasts. (c) Recorded outcomes supervise the critic on detached features. Coverage (†) is LeHome-Fold-only; s_k is a training logit.

Actor and process supervision. For demonstrated actions A, sample $\epsilon \sim \mathcal{N}(0, I)$ and $\tau \sim p_{\text{flow}}$, interpolate $A_\tau = (1-\tau)A + \tau\epsilon$, and regress velocity toward $\epsilon - A$ (Lipman et al., 2023). Process prediction uses the clean actions whose process labels are known, giving a separate supervision path alongside the noisy flow input:

$$\mathcal{L}_{\text{actor}} = \mathcal{L}_{\text{FM}} + \lambda_z \mathcal{L}_{\text{phase}} + \lambda_p \mathcal{L}_{\text{progress}} + \lambda_b \mathcal{L}_{\text{boundary}} + \lambda_{\text{aux}}^{\text{actor}} \mathcal{L}_{\text{aux}}^{\text{actor}}. \tag{6}$$

Core terms are masked flow MSE, phase cross-entropy, progress MSE, and boundary BCE. Phase/progress supervise current readouts and forecasts, and boundary supervises forecasts only. Ground-truth process labels supervise training. At deployment, the controller uses model-predicted process signals. Actor and process parameters form $\vartheta = (\theta, \phi)$, with trainable subset Ω specified by stage.

Detached critic learning. Groups $\mathcal{G} = \{(x_t, A^{(k)}, y^{(k)})\}_k \sim \mathcal{D}_{\text{crit}}$ contain continuations recorded from a shared initial state. Each $y^{(k)}$ is the outcome of candidate k followed by its recorded continuation. The current predictor recomputes forecasts for these recorded actions, then observation, action, and forecast features are detached before critic optimization. For member logits $s_m^{(k)}$, the objective is

$$\mathcal{L}_{\text{critic}} = \mathcal{L}_{\text{BCE}}^{\text{group}}(s, y) + \lambda_{\text{rank}} \mathcal{L}_{\text{rank}}(s, y) + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}, \tag{7}$$

Group-balanced BCE learns outcome discrimination, within-group ranking compares successful and unsuccessful continuations from the same state, and available quality targets provide auxiliary regression. Critic members are optimized on detached actor/process features. Calibration on a disjoint $\mathcal{D}_{\text{cal}}$ fits the transformation from mean logit to deployment score. The paired bundle $\mathcal{B} = (\vartheta, \Psi, \alpha, \beta)$, where $\Psi = \{\psi_m\}_{m=1}^M$, binds the learned predictor and critic to that calibration. Appendix B collects model parameterizations, action interfaces, optional readout adaptation, auxiliary targets, masks, and the flow-time law.

4 Experiments

We evaluate task completion, candidate-budget allocation, and process/execution behavior. LeHome-Fold, the garment-folding protocol on LeHome (Li et al., 2026c; LeHome Challenge Or-

ganizers, 2026), tests repeated deformable contact; LIBERO-Long, the ten-task long-horizon suite (Liu et al., 2023), tests sequential object interactions.

4.1 Experimental setup and closed-loop results

LeHome-Fold evaluates each method or variant on 240 episodes, and LIBERO-Long uses 500 episodes across ten tasks. The primary PatchVLA controller uses 4×2 latent refinement with episode action budgets of 600 and 520, respectively, while learned weights remain fixed during planning. Evaluation protocols, baseline implementations, and additional results are provided in Appendices C and F.

Task completion. PatchVLA achieves 76.25% overall success on LeHome-Fold, compared with 67.08% for the official $\pi_{0.5}$ baseline, a 9.17-point difference (Table 1). Its Seen and Unseen success rates are 77.50% and 70.00%, respectively.

Method	**SR** ↑	LT	ST	LP	SP	Seen	Unseen
ACT	50.00	61.67	31.67	25.00	81.67	55.00	25.00
Diffusion Policy	61.67	<u>65.00</u>	60.00	38.33	83.33	66.00	40.00
Official $\pi_{0.5}$	<u>67.08</u>	51.67	<u>68.33</u>	**58.33**	<u>90.00</u>	<u>68.00</u>	<u>62.50</u>
X-VLA	54.17	60.00	46.67	28.33	81.67	58.50	32.50
SmolVLA	62.92	61.67	55.00	<u>45.00</u>	<u>90.00</u>	66.50	45.00
FastWAM	7.08	8.33	6.67	5.00	8.33	8.50	0.00
LingBot-VA	35.42	5.00	11.67	36.67	88.33	37.00	27.50
PatchVLA	**76.25**	**75.00**	**78.33**	**58.33**	**93.33**	**77.50**	**70.00**

Table 1: **LeHome-Fold baseline comparison.** The shaded **SR** column is Overall success rate; LT/ST denote long-/short-sleeved tops, and LP/SP denote long/short pants. All entries are success rates (%). Bold/underline mark the best/next distinct value in each column; ties share rank.

On LIBERO-Long, PatchVLA attains 96.20% success, compared with 93.40% for official $\pi_{0.5}$ and 96.00% for X-VLA (Table 2). Its NGC is 97.70%.

Method	**Success Rate** ↑	NGC ↑	NR-SR_2 ↑	NG-AUC ↑	PG-AUC ↑	Jump ↓	Δa^2 ↓
SmolVLA	41.40	49.90	38.60	30.40	29.56	21.402	21.402
FastWAM	91.20	94.20	88.20	58.77	58.59	2.795	<u>0.803</u>
LingBot-VA	93.80	96.50	88.00	58.69	58.49	**0.548**	**0.672**
X-VLA	<u>96.00</u>	<u>97.00</u>	<u>90.00</u>	**61.00**	**60.81**	–	–
Official $\pi_{0.5}$	93.40	95.70	89.20	58.64	58.41	3.181	1.223
PatchVLA	**96.20**	**97.70**	**93.80**	<u>59.91</u>	<u>59.70</u>	<u>1.857</u>	0.959

Table 2: **LIBERO-Long baseline comparison.** Four-repeat results are reported in Appendix E.1. Success Rate (SR) is the primary metric. The first five metrics use a 0–100 scale; command roughness Δa^2 and Jump are MSE in units of 10^{-3}. X-VLA action metrics are omitted because its action representation is incompatible. Bold/underline mark best/second distinct values among the displayed methods. Definitions are in Appendix E.

Completion and execution quality. Success rate records task completion. NG-AUC accumulates newly satisfied goals, PG-AUC rewards goals that persist until termination, and NR-SR_2 counts successful episodes without a confirmed loss of an acquired goal. Command roughness and replan-boundary Jump measure adjacent-command differences over the six continuous action components; definitions and aggregation appear in Appendix E. PatchVLA has the highest NR-SR_2 in Table 2; its command roughness and Jump are 21.6% and 41.6% lower than the official $\pi_{0.5}$ values.

4.2 Component comparisons

In Tables 3–4, Process denotes the process-aware model configuration, Selection enables critic-based ranking, and MPPI enables iterative latent refinement. With Process enabled, LeHome-Fold

SR rises from 72.08% with one round to 76.25% with two rounds; without Process, the corresponding values are 68.33% and 69.58%.

	Components			**Success Rate** (%) ↑						
Variant	Process	Selection	MPPI	**SR** ↑	LT	ST	LP	SP	Seen	Unseen
(a)	×	✓	×	68.33	66.67	71.67	43.33	91.67	70.00	60.00
(b)	×	✓	✓	69.58	71.67	70.00	50.00	86.67	72.50	55.00
(c)	✓	×	×	71.25	70.00	73.33	46.67	**95.00**	73.00	62.50
(d)	✓	✓	×	72.08	66.67	73.33	55.00	93.33	73.50	65.00
PatchVLA	✓	✓	✓	**76.25**	**75.00**	**78.33**	**58.33**	93.33	**77.50**	**70.00**

Table 3: **LeHome-Fold component ablations.** The shaded **SR** column is Overall success rate; all other metric columns are also success rates (%). Check/cross indicate enabled/disabled components. Category abbreviations follow Table 1. Bold/underline mark best/second distinct percentages; ties share rank.

LeHome-Fold Unseen success is 65.00% for (d) and 70.00% for the full controller. In LIBERO-Long, the Process Actor, one-round selection without Process, and the full controller reach 95.80%, 95.20%, and 96.20% SR, respectively (Table 4).

	Components									
Variant	Process	Selection	MPPI	**SR** ↑	NGC ↑	NR-SR_2 ↑	NG-AUC ↑	PG-AUC ↑	Jump ↓	Δa^2 ↓
(a)	×	✓	×	95.20	96.90	92.00	59.24	59.03	2.411	1.065
(b)	×	✓	✓	94.20	96.10	91.60	59.37	59.19	2.004	1.033
(c)	✓	×	×	95.80	97.30	93.00	**59.91**	59.69	1.884	1.001
(d)	✓	✓	×	93.40	95.70	90.20	59.04	58.89	1.869	0.961
PatchVLA	✓	✓	✓	**96.20**	**97.70**	**93.80**	**59.91**	**59.70**	**1.857**	**0.959**

Table 4: **LIBERO-Long configuration comparison.** The first five metrics are on a 0–100 scale; command roughness Δa^2 and Jump are MSE in units of 10^{-3}. The full PatchVLA row is reused in the baseline comparison in Table 2. Bold/underline mark best/second distinct values; ties share rank.

Fixed-model candidate-budget allocation. Table 5 compares two rounds of four candidates (4×2) with one round of eight candidates (8×1). In 4×2, first-round costs update the latent proposal distribution before the second round is sampled. Both search configurations evaluate eight candidates per replanning decision. We use the same Actor, Critic, flow steps, execution prefix, and costs, and disable proposal reuse in both configurations. Each configuration has two complete 240-episode executions. Two-round refinement is higher in both repeats and reaches 73.96% aggregate SR versus 70.21% for one-round selection; the Process Actor reference is 71.88% (Table 5; Appendix D.1).

Method	Repeat 1	Repeat 2	**Overall SR**
Two-round refinement (4×2)	**74.17**	**73.75**	**73.96**
One-round selection (8×1)	72.08	68.33	70.21
Process Actor	72.50	71.25	71.88

Table 5: **Fixed-model candidate-budget allocation.** Success rates (%) from two complete executions with eight candidate evaluations per decision.

4.3 Process structure in garment folding

Figure 4 follows two sleeve folds and the torso fold. Local progress resets between subtasks, while phase and boundary forecasts place each reset within the interaction. Panel (b) shows current readouts over observations; panel (c) shows the forecasts for the candidates at one decision.

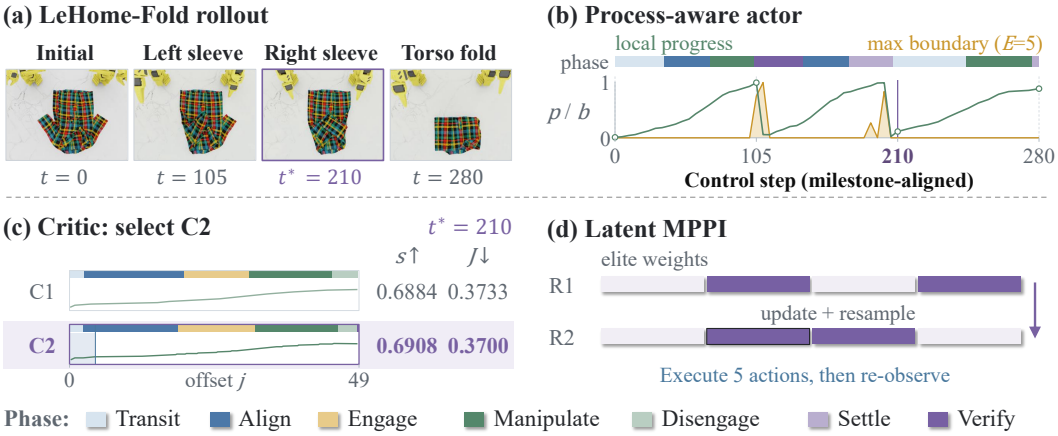


Figure 4: **Process transitions on LeHome-Fold.** (a) Folding milestones. (b) Phase, local progress, and boundary over the executed prefix. (c) Candidate forecast at $t^* = 210$, where C2 has the lower planning cost. (d) Two refinement rounds and prefix execution.

4.4 Candidate Selection Across Object Interactions

Figure 5 traces two-object placement with the 16×2 controller. At $t^* = 260$, the configured weighted-medoid rule selects C1 for prefix execution; C1 has the lower total planning cost J in Eq. (4).

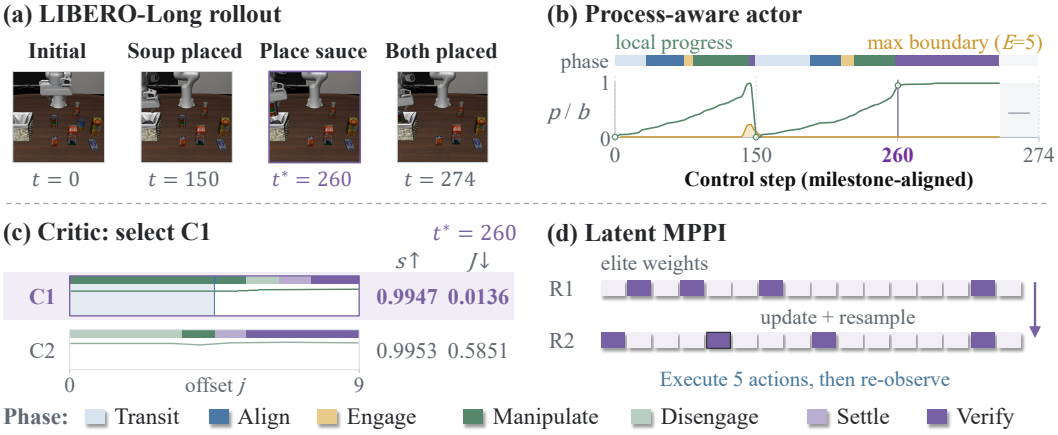


Figure 5: **Candidate selection on LIBERO-Long.** (a) Two-object placement milestones. (b) Process forecasts; gray marks termination. (c) At $t^* = 260$, the selected candidate C1 has the lower planning cost J. (d) Two refinement rounds followed by five-action execution with weighted-medoid selection.

Across four executions of the same 500 initial states, this controller averages 95.50% SR versus 94.55% for the same-weights Process Actor and 94.40% for the official checkpoint (Table A4). Replanning jump MSE is 1.475 versus 2.392 in units of 10^{-3}. Figure A3 aligns the smaller command changes with replanning boundaries. Appendix G provides aligned traces for all four garment categories and ten LIBERO-Long tasks.

5 Conclusion

PatchVLA pairs each candidate action chunk with aligned forecasts of phase, local progress, and subtask boundaries. A sequence critic scores the resulting action–process sequence, and the score refines the latent proposal before one evaluated prefix is executed and the controller replans. With

a shared Actor, Critic, execution rule, and eight-candidate budget, two-round refinement reaches 73.96% success versus 70.21% for one-round selection. In the repeated 16×2 LIBERO-Long evaluation, the controller averages 95.50% SR and has lower replanning jump MSE than the same-weights Process Actor.

AI USE STATEMENT

GPT-5.6-sol assisted with generating the human-and-AI-in-the-loop data-generation workflow shown in Figure 3. GPT-6-Astra assisted with manuscript polishing. The authors reviewed all outputs and take responsibility for the paper's content.

ETHICS STATEMENT

The reported evaluations are conducted in simulation. Deployment on physical robots should be preceded by task-specific safety testing for collisions and unsafe contact, with appropriate human oversight.

REPRODUCIBILITY STATEMENT

Sections 3 and 4.1 describe the model and evaluation. Algorithms 1 and 2 give the training and inference loops. Appendix B details the models, losses, action transforms, and calibration. Appendix C documents the evaluation protocols and checkpoint selection; Appendix F records baseline scope. Appendix E defines the behavioral metrics and their aggregation.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings*

of Robotics: Science and Systems, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005. doi: 10.1007/s10479-005-5724-z. URL https://doi.org/10.1007/s10479-005-5724-z.

Xiaofeng Han, Shunpeng Chen, Zenghuang Fu, Zhe Feng, Lue Fan, Dong An, Changwei Wang, Li Guo, Weiliang Meng, Xiaopeng Zhang, Rongtao Xu, and Shibiao Xu. Multimodal fusion and vision-language models: A survey for robot vision. *Information Fusion*, 126:103652, 2026. doi: 10.1016/j.inffus.2025.103652. URL https://doi.org/10.1016/j.inffus.2025.103652.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. URL https://arxiv.org/abs/2207.05608.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. URL https://arxiv.org/abs/2403.12945.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

LeHome Challenge Organizers. LeHome Challenge 2026: Garment manipulation skill learning in household scenarios. Official challenge website and software repository, 2026. URL https://lehome-challenge.com/. ICRA 2026 challenge. Software: https://github.com/lehome-official/lehome-challenge. Accessed 2026-09-20.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026a. URL https://arxiv.org/abs/2601.21998.

Wenhao Li, Xiu Su, Dan Niu, Yichao Cao, Hongyan Xu, Zhe Qu, Lei Fan, Shan You, and Chang Xu. VLA-ATTC: Adaptive test-time compute for VLA models with relative action critic model. *arXiv preprint arXiv:2605.01194*, 2026b. URL https://arxiv.org/abs/2605.01194v1.

Zeyi Li, Yushi Yang, Shawn Xie, Kyle Xu, Tianxing Chen, Yuran Wang, Zhenhao Shen, Yan Shen, Yue Chen, Wenjun Li, Yukun Zheng, Chaorui Zhang, Siyi Lin, Fei Teng, Hongjun Yang, Ming Chen, Steve Xie, and Ruihai Wu. LeHome: A simulation environment for deformable object manipulation in household scenarios. *arXiv preprint arXiv:2604.22363*, 2026c. URL https://arxiv.org/abs/2604.22363v1.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022. URL https://arxiv.org/abs/2209.07753v1.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024. URL https://arxiv.org/abs/2410.13816.

Pengjun Niu, Yujia Xie, Rui Peng, Hang Zhao, and Ke Liu. SkelWAM: A skeleton-guided world-action model for zero-shot cross-embodiment manipulation. *arXiv preprint arXiv:2609.21983*, 2026. doi: 10.48550/arXiv.2609.21983. URL https://arxiv.org/abs/2609.21983.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. URL https://arxiv.org/abs/2405.12213.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023. URL https://arxiv.org/abs/2310.08864.

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

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 2025. URL https://proceedings.mlr.press/v305/wagenmaker25a.html.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. URL https://arxiv.org/abs/2208.06193.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017. doi: 10.2514/1.G001921. URL https://doi.org/10.2514/1.G001921.

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. *arXiv preprint arXiv:2312.13139*, 2023. URL https://arxiv.org/abs/2312.13139.

Rongtao Xu, Jian Zhang, Minghao Guo, Youpeng Wen, Haoting Yang, Min Lin, Jianzheng Huang, Zhe Li, Kaidong Zhang, Liqiong Wang, Yuxuan Kuang, Meng Cao, Feng Zheng, and Xiaodan Liang. A_0: An affordance-aware hierarchical model for general robotic manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13491–13501, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Xu_A0_An_Affordance-Aware_Hierarchical_Model_for_General_Robotic_Manipulation_ICCV_2025_paper.html.

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. URL https://arxiv.org/abs/2603.16666.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024. URL https://arxiv.org/abs/2407.08693.

Kaidong Zhang, Jian Zhang, Rongtao Xu, Yu Sun, Youpeng Wen, Shuoshuo Xue, Xiaoyu Guo, Minghao Guo, Weijia Liufu, Liu Zihou, Kangyi Ji, Zihang Li, Ruiyi Chen, Meng Cao, Jingming Zhang, Shen Zhao, Xiaojun Chang, Feng Zheng, Ivan Laptev, and Xiaodan Liang. A1: Adaptive truncated vision-language-action model from affordance to action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 1503–1514, 2026. URL https://openaccess.thecvf.com/content/CVPR2026F/html/Zhang_A1_Adaptive_Truncated_Vision-Language-Action_Model_from_Affordance_to_Action_CVPRF_2026_paper.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://www.roboticsproceedings.org/rss19/p016.html.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025. URL https://arxiv.org/abs/2510.10274.

Appendix A gives the algorithms; Appendices B and C specify learning and evaluation; Appendices D–G provide quantitative comparisons, behavioral analyses, and aligned traces.

A Complete algorithms

Algorithms 1 and 2 summarize training and inference using the notation of Section 3. Helper interfaces and optimization boundaries are defined in Appendix B.5.

Algorithm 1 PatchVLA training

Input: $\mathcal{D}_{\mathrm{demo}}, \mathcal{A}_{\mathrm{exp}}, \mathcal{D}_{\mathrm{adapt}}, \mathcal{D}_{\mathrm{crit}}, \mathcal{D}_{\mathrm{cal}}$; pretrained ϑ, initial Ψ, trainable subset Ω
Output: Frozen deployment bundle $\mathcal{B} = (\vartheta, \Psi, \alpha, \beta)$

$\mathcal{D} \leftarrow \mathrm{ReviewedTargets}(\mathcal{D}_{\mathrm{demo}}, \mathcal{A}_{\mathrm{exp}})$ [Figure 3a]
Actor/process supervision
for minibatches $(x_t, A, Y, \mathcal{M}) \sim \mathcal{D}$ **do**
  $(h_t, \mathcal{K}_t, c_t) \leftarrow \mathrm{Encode}_{\vartheta}(x_t)$
  $\epsilon \sim \mathcal{N}(0, I), \quad \tau \sim p_{\mathrm{flow}}, \quad A_\tau \leftarrow (1-\tau)A + \tau\epsilon$
  $\hat{v} \leftarrow v_\theta(A_\tau, \tau; \mathcal{K}_t, c_t)$
  $(Q_t, P, \chi) \leftarrow \mathrm{Process}_{\phi}(x_t, A)$
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{actor}}(\hat{v}, Q_t, P, \chi; \epsilon - A, Y, \mathcal{M})$ [Eq. (6)]
  $\vartheta_\Omega \leftarrow \mathrm{OptStep}(\vartheta_\Omega, \nabla_{\vartheta_\Omega}\mathcal{L})$
end for
Optional fixed-action readout adaptation
if $\mathcal{D}_{\mathrm{adapt}} \neq \varnothing$ **then**
  $(\phi_{\mathrm{ro}}, \omega) \leftarrow \mathrm{Adapt}_{\mathrm{ro}}(\mathcal{D}_{\mathrm{adapt}}; \mathrm{sg}(\vartheta))$ [Eq. (8)]
  $\vartheta[\phi_{\mathrm{ro}}] \leftarrow \phi_{\mathrm{ro}}$
end if
Detached sequence-critic learning
for same-start groups $\mathcal{G} \sim \mathcal{D}_{\mathrm{crit}}$ **do**
  $(h_t, \mathcal{K}_t, c_t) \leftarrow \mathrm{Encode}_{\vartheta}(x_t)$
  $(\tilde{h}_t, \tilde{A}^{(k)}, \tilde{P}^{(k)}) \leftarrow \mathrm{sg}(h_t, A^{(k)}, F_\phi(x_t, A^{(k)}))$
  $s_m^{(k)} \leftarrow C_{\psi_m}(\tilde{h}_t, \tilde{A}^{(k)}, \tilde{P}^{(k)}), \quad m = 1, \ldots, M$
  $\psi_m \leftarrow \mathrm{OptStep}(\psi_m, \nabla_{\psi_m}\mathcal{L}_{\mathrm{critic}}(s_m, \mathcal{G})), \quad m = 1, \ldots, M$ [Eq. (7)]
end for
Affine calibration
$\bar{s}^{\mathrm{cal}} \leftarrow \mathrm{sg}\big(M^{-1}\sum_m C_{\psi_m}(\mathcal{D}_{\mathrm{cal}}; \vartheta)\big)$
$(\alpha, \beta) \leftarrow \mathrm{FitAffine}(\bar{s}^{\mathrm{cal}}, y^{\mathrm{cal}})$ [Eq. (9)]
return $\mathcal{B} = (\vartheta, \Psi, \alpha, \beta)$

Algorithm 2 PatchVLA inference

Input: Frozen bundle $\mathcal{B}$; action budget B; configuration Γ

$t \leftarrow 0, \quad d \leftarrow \mathrm{false}, \quad \xi \leftarrow \varnothing$
while $t < B$ and $\neg d$ **do**
  $x_t \leftarrow \mathrm{Observe}(), \quad (h_t, \mathcal{K}_t, c_t) \leftarrow \mathrm{Encode}_{\vartheta}(x_t)$
  $(\mu_0, \sigma_0^2) \leftarrow \mathrm{InitOrReuse}_\Gamma(\xi)$
  for $r = 0, \ldots, R-1$ **do**
    $\epsilon_{r,k} \sim \mathcal{N}(\mu_r, \mathrm{diag}(\sigma_r^2)), \quad k = 1, \ldots, K$
    $A_{r,k} \leftarrow G_\theta(x_t, c_t, \epsilon_{r,k}), \quad P_{r,k} \leftarrow F_\phi(x_t, A_{r,k})$
    $(S_{r,k}, u_{r,k}) \leftarrow \mathrm{Score}_{\Psi,\alpha,\beta}(h_t, A_{r,k}, P_{r,k})$ [Eq. (3)]
    $J_{r,k} \leftarrow \mathrm{Cost}_\Gamma(S_{r,k}, u_{r,k}, A_{r,k}; \xi)$ [Eq. (4)]
    $\mathcal{E}_r \leftarrow \mathrm{Lowest}_{\lceil \eta K \rceil}(J_r)$
    $(w_r, \mu_{r+1}, \sigma_{r+1}^2) \leftarrow \mathrm{Update}_\Gamma(\mu_r, \sigma_r^2, \mathcal{E}_r, \epsilon_r, J_r)$ [Eq. (5)]
  end for
  $k^\star \leftarrow \mathrm{Select}_\Gamma(\mathcal{E}_{R-1}, A_{R-1}, J_{R-1}, w_{R-1}), \quad A^\star \leftarrow A_{R-1,k^\star}$
  $E_t \leftarrow \min\{\mathrm{Prefix}_\Gamma(P_{R-1,k^\star}),\ B - t\}$
  $(e_t, d) \leftarrow \mathrm{Execute}(A^\star_{0:E_t-1})$
  $\xi \leftarrow \mathrm{Advance}_\Gamma(\xi, \mu_R, \sigma_R^2, A^\star, P_{R-1,k^\star}, e_t), \quad t \leftarrow t + e_t$
end while

B MODEL IMPLEMENTATION AND TRAINING

B.1 SUPERVISION AND OPTIMIZATION STAGES

Expert temporal anchors and reviewed VLM-assisted subtask segments define the process targets (Figure 3). Interaction cues label phase, accepted anchors define progress, and changes of subtask instance mark boundaries.

The flow-time law is $\tau = 0.001 + 0.999u$, $u \sim \mathrm{Beta}(1.5, 1)$. The labeled actor/process stage uses $A_\tau = (1-\tau)A + \tau\epsilon$ and velocity target $\epsilon - A$. Invalid action offsets and unavailable process targets are masked before reduction. Current/future phase terms use classification cross-entropy, progress uses sigmoid-output MSE, and boundary uses weighted logit BCE. Phase and progress each average their current and trajectory losses with equal weights; boundary supervision applies to the trajectory only. The auxiliary term sums current uncertainty BCE against $1-$confidence and future contact-intent cross-entropy, with $\lambda_{\mathrm{aux}}^{\mathrm{actor}} = 0.05$ in the process-supervised configurations. It is zero when unused. $\mathrm{Process}_\phi(x_t, A)$ returns current phase/progress readouts Q_t, the candidate forecast $P = F_\phi(x_t, A)$, and auxiliary outputs χ. Current readouts are $Q_t = (\hat{z}_t, \hat{p}_t)$. Their labels and masks are included in $(Y, \mathcal{M})$; auxiliary outputs do not alter the deployed (z, p, b) interface. Figure 3b shows the core loss terms; Eq. (6) includes the auxiliaries.

The jointly trained actor/process checkpoint initializes optional readout adaptation of ϕ_{ro} and the state head ω, defined below. $\mathcal{D}_{\mathrm{adapt}}$ is empty when adaptation is unused. Readout and critic updates may be interleaved at the configured cadence.

B.2 FIXED ACTION GENERATOR AND DETACHED CRITIC

LeHome-Fold adaptation freezes the vision–language backbone, action DiT, action projections, time MLPs, process-context encoder, and process-to-action modulation. Policy-side updates affect only current-phase, current-progress, phase-trajectory, progress-trajectory, and boundary readouts, together with the current-success head.

The LeHome-Fold sequence tokens have $12+7+1+1 = 21$ channels: actions, phase probabilities, progress and boundary probability. A projected observation token and positional sequence tokens have width 128 and pass through two four-head Transformer layers. The observation-token output supplies success and coverage heads. Three independently initialized critic members are calibrated by an affine transform of their mean success logit. Group-balanced BCE, within-group ranking and valid coverage regression have weights 1, 0.25 and 0.2. The ranking hinge margin is 0.2.

B.3 CANONICAL ACTION TRANSFORMS

The flow uses padded model coordinates: A_τ and ϵ have shape $H \times d_z$, with d_z the padded dimension and $d_a \leq d_z$ effective action coordinates. Fixed checkpoint transforms retain these coordinates and map the normalized chunk to executable commands.

Actor and scorer share checkpoint normalization and executable-action transforms. The LeHome-Fold scorer converts candidate actions to normalized canonical coordinates and reconstructs zero padding before process forecasting; the critic reads effective coordinates only. Each Actor/process snapshot is paired with its corresponding feature cache, critic, and calibration parameters. Candidate selection returns one intact decoded chunk in executable coordinates.

B.4 PROCESS IMPLEMENTATIONS ACROSS BENCHMARKS

The LIBERO-Long experiments use the process-aware Actor from the corresponding checkpoint family. That checkpoint uses a shared MLP over each action embedding and the current observation context, whereas LeHome-Fold uses a shifted causal action-prefix representation with a state-fusion projection. Both instantiate the candidate-conditioned interface in Eq. (2); the downstream sequence critic aggregates action and process tokens across the complete horizon. The critic deployed in the LIBERO-Long controller is trained on the auxiliary 40-task branch cache with group-balanced classification and within-group ranking, without LeHome-Fold coverage supervision. Closed-loop eval-

uation uses the ten-task LIBERO-Long benchmark. The shared interface supports these benchmark-specific process heads with the same sequence-evaluation and latent-refinement procedure.

B.5 Algorithm interfaces and optimization boundaries

Training. Critic updates recompute observation features and process forecasts for each group's recorded actions. Independently initialized members use separate optimizers on detached inputs, supervised by the branch's recorded continuation outcome and any quality targets from the same record. No critic gradient reaches ϑ.

For fixed-action adaptation, $\mathrm{Adapt}_{\mathrm{ro}}$ snapshots a frozen teacher $\bar{\vartheta}$ and updates only $\zeta = (\phi_{\mathrm{ro}}, \omega)$, leaving the action generator and process-context path fixed. The two streams are normalized separately before their gradients are summed. The objective relevant to these trainable parameters is

$$\mathcal{L}_{\mathrm{adapt}} = \lambda_{\mathrm{keep}}(\mathcal{L}^{\mathrm{act}}_{\mathrm{keep}} + \mathcal{L}^{\mathrm{state}}_{\mathrm{keep}}) + \lambda_{\mathrm{state}}\,\mathrm{BCE}(H_\omega, y^{\mathrm{state}}). \tag{8}$$

Each optimization step applies $\zeta \leftarrow \mathrm{OptStep}(\zeta, \nabla_\zeta \mathcal{L}_{\mathrm{adapt}})$. Consistency uses detached teacher predictions, phase KL, progress Huber, and soft boundary BCE. Current-state BCE uses separate recorded state labels. Its head H_ω reads frozen context, so BCE updates ω only. Process readouts receive consistency gradients. The implemented weights are $\lambda_{\mathrm{keep}} = 0.5$ and $\lambda_{\mathrm{state}} = 0.1$. Flow and reference-velocity terms have zero gradient with respect to ζ when the action path is fixed. The auxiliary state head is separate from the sequence critic and is not used for candidate evaluation. The assignment $\vartheta[\phi_{\mathrm{ro}}] \leftarrow \phi_{\mathrm{ro}}$ stores the adapted readouts in the model snapshot.

With the actor/process snapshot and critic ensemble frozen, affine calibration fits mean logits $\bar{s}_i$ to recorded outcomes y_i^{cal}:

$$\min_{a,\beta} \frac{1}{N_{\mathrm{cal}}} \sum_i \mathrm{BCE}_{\mathrm{logits}}(\alpha \bar{s}_i + \beta, y_i^{\mathrm{cal}}) + 10^{-4}(a^2 + \beta^2), \qquad \alpha = \exp(\mathrm{clip}(a, -5, 5)). \tag{9}$$

The fitted parameters are bound to the corresponding model snapshot. Data separation follows Appendix C.

Inference. Planning state ξ stores the optional proposal, previous executed command, and reset state. An empty state initializes the configured zero-mean Gaussian. Otherwise, planning reuses the stored proposal. Continuity cost is zero when no previous command is retained.

Select returns either the minimum-cost elite or the elite weighted medoid,

$$k^\star = \arg\min_{k \in \mathcal{E}} \sum_{i \in \mathcal{E}} w_i \|A_k - A_i\|_2,$$

Distances use flattened action chunks. Execution applies the checkpoint's action transforms to a positive fixed or boundary-limited prefix, respecting the remaining budget and environment termination. It returns the actual count e_t and termination flag d. Advance stores the last executed command and, when proposal reuse is enabled, shifts (μ_R, σ_R^2) by e_t with configured variance floors and tail initialization. Boundary or command-edge resets clear this memory. The final update serves subsequent reuse, while current execution uses an evaluated final-round candidate.

C Evaluation configurations and protocols

C.1 Evaluation configurations and result mapping

Table A1 organizes primary evaluations, repeated executions, and checkpoint comparisons. The rows identify the evaluation group used by each result table and its recorded execution scale.

Alias	Controller	Episodes/execution	Executions	Results
LH-main	4×2	240	Primary	Tables 1, 3
LH-repeat	4×2 / IID8	240	2	Appendix D.1
LH-checkpoint	epoch 1–5	240	2; e2: 4	Appendix C.2
LB-main	4×2	500	Primary	Tables 2, 4
LB-repeat	16×2	500	4	Table A4

Table A1: **Result-group navigation.** LH and LB denote LeHome-Fold and LIBERO-Long. Main rows identify the full controller; the component tables also include single-candidate and 8×1 configurations. LH-repeat uses the selected epoch-2 controller. The checkpoint summary records the original two executions and the two additional epoch-2 executions.

C.2 Benchmarks, repetitions, and checkpoint selection

LeHome-Fold evaluates four garment categories: long/short tops and long/short pants. Each seed-42 execution contains 40 seen and eight unseen assets with five consecutive resets per asset, yielding 240 episodes (200 Seen, 40 Unseen). For the checkpoint sweep, the process-conditioned Actor has two complete repetitions. All five controller epochs have two repetitions; epoch 2 has two additional repetitions. A separate LIBERO-Long controller cohort covers the ten official LIBERO-10 tasks with 50 initial states per task and four seed-7 executions.

Epochs 1–5 correspond to steps 536, 1071, 1606, 2141, and 2676. The equal-budget study uses epoch 2, selected after inspecting the five-checkpoint evaluation results. The controller's original two executions, two additional executions, and pooled four-execution result have overall success rates of 73.96%, 71.25%, and 72.60%, respectively. The original two Process Actor executions reach 71.88%, and the final epoch-5 controller reaches 70.63%. The additional executions do not add synchronized Process Actor baseline repeats. In the repeated LIBERO-Long cohort, Actor-only and PatchVLA share action weights and normalization. The official $\pi_{0.5}$ checkpoint is a separately trained policy.

C.3 Information access and data separation

Runtime inputs are permitted images, language, and robot state. Hidden task predicates and ground-truth process labels are used only for supervision or evaluation. Critic labels follow each candidate's own recorded continuation; process targets follow the corresponding temporally annotated trajectory.

Sibling branches and source episodes remain together when splitting data, and calibration is disjoint from training and evaluation. LeHome-Fold classification uses 297 held-out branches; ranking uses 57 complete four-candidate groups and excludes 20 incomplete groups. The auxiliary LIBERO critic cache covers 40 tasks and 640 source episodes: 4,784 training records and 960 each for validation, calibration, and test. Its test set contains 240 groups, seven with mixed outcomes.

C.4 Controller settings and matched comparisons

Setting	LeHome-Fold	LIBERO-Long
Chunk horizon H / action dimension d_a	50 / 12	10 / 7
Latent shape / flow steps	50×32 / 10	10×32 / 10
Candidates K / rounds R	4 / 2	4 / 2
Executed prefix / episode budget	5 / 600	1–5 / 520
Elite fraction / update rate	0.5 / 0.5	0.25 / 1
Variance bounds	$[0.25, 4]$	$[10^{-4}, 100]$
Final selection	Minimum-cost elite	Weighted medoid
Proposal reuse / boundary-limited prefix	Disabled / disabled	Enabled / enabled

Table A2: **A shared controller interface across action spaces.** Settings for the primary PatchVLA configuration in Tables 1 and 2. The separate LIBERO-Long 16×2 cohort and qualitative example use their original candidate budget. With $K = 4$ and elite fraction 0.25, LIBERO-Long has one elite, so its weighted-medoid selector returns that intact candidate.

LIBERO-Long uses temperature 0.25. The earliest boundary probability at least 0.5 limits the prefix. Proposal reuse shifts by the executed prefix, applies a variance floor of 1, and resets at a predicted boundary or gripper-command sign transition. Uncertainty, roughness, and continuity penalties have weights 0.25, 0.05, and 1. LeHome-Fold logs uncertainty without penalizing it and uses neither warm starts nor transition gates.

LeHome-Fold IID8 and MPPI 4×2 each evaluate eight candidates with identical Actor, critic, flow steps, prefix, and costs, and no proposal reuse. Full-input and process-masked variants each have two complete executions; their process-input comparison is summarized in Appendix D.1.

Controller–Actor comparisons evaluate the full deployed configurations, including candidate budgets, execution-prefix rules, proposal reuse, and planning costs. Table 4 compares the complete controller configurations listed in its component columns.

C.5 Controller resources and paired evaluation

With 10 flow steps, $H = 50$, and five-action prefixes, the observed post-warm-up server-request median/p95 durations are 86.9/178.9 ms for the single-candidate Actor (36,423 requests), and 743.0/945.0 ms for MPPI 4×2 (eight candidates, 35,815 requests). Timings reflect recorded server loads, which were not matched across methods. LIBERO-Long timing quantiles use episode-level aggregation.

LeHome-Fold pairs match garment, repetition, and nominal reset index, without verified full simulator-state equality. LIBERO-Long pairs additionally match initial-state hashes. Paired comparisons count both failure-to-success conversions and success-to-failure regressions.

D Additional quantitative comparisons

This appendix reports equal-budget process-input comparisons, task-level outcomes, and paired controller–Actor outcomes. The LIBERO-Long analyses use the LB-repeat (16×2) cohort.

D.1 Equal-budget repeats and process inputs

The equal-budget repeat compares one-round selection (IID8, 8×1) with two-round refinement (MPPI 4×2). Both configurations retain process-conditioned action generation, evaluate eight candidates per decision, use the same Actor, Critic, flow steps, prefix, and costs, and disable proposal reuse. Masks zero only the fixed critic's process inputs without retraining. Each repeat covers 200 Seen and 40 Unseen episodes (60 per garment category) under the same seed/reset protocol. Aggregate scores combine complete repeats under this shared protocol.

Configuration	Repeat 1	Repeat 2	Seen	Unseen	**Overall SR**
Process Actor	72.50	71.25	74.75	57.50	<u>71.88</u>
IID8 + full Critic	72.08	68.33	71.75	<u>62.50</u>	70.21
MPPI4×2 + full Critic	**74.17**	<u>73.75</u>	<u>76.00</u>	**63.75**	**73.96**
IID8 + P-mask	71.67	69.58	73.50	56.25	70.63
MPPI4×2 + P-mask	<u>73.75</u>	**74.17**	**78.00**	53.75	**73.96**

Table A3: **Equal-budget refinement and process-input sensitivity.** The shaded **Overall SR** column reports aggregate success rate; all other metric columns are also success rates (%) over complete repetitions. Denominators are 240 for each repeat, 400 Seen, 80 Unseen and 480 Overall. Bold/underlined values denote best/second-best per column, with ties sharing rank.

Additional epoch-2 controller repeats are reported in Table A5.

D.2 Task-level gains and unseen garments

On LeHome-Fold, PatchVLA improves aggregate Unseen success by 6.25 percentage points ($46/80$ to $51/80$), compared with 1.25 points on Seen assets ($299/400$ to $304/400$). On LIBERO-Long, task 09 improves by 9.5 points across four repeats, while the mean over all ten tasks is $+0.95$ points.

Figure A1 resolves these average gains by garment category and task. Repeat-level aggregate results appear in Appendix E.1.

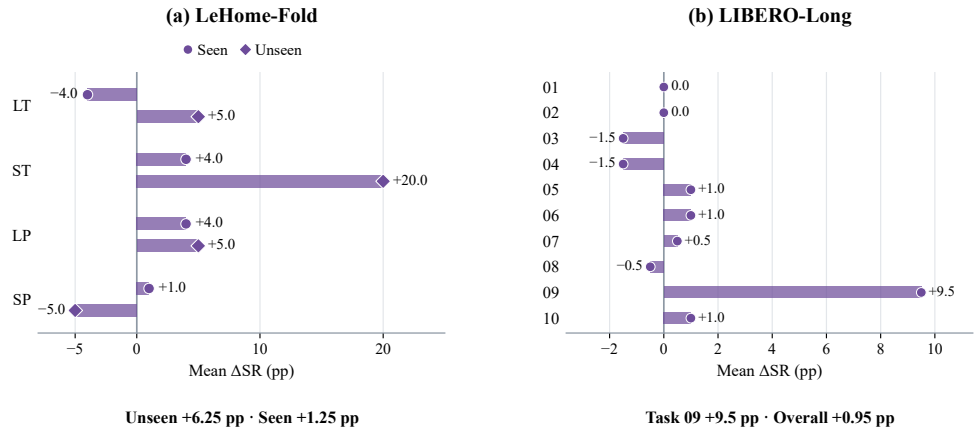


Figure A1: **Task-level gains and unseen garments.** Horizontal bars and points show mean PatchVLA–Actor success-rate differences in percentage points from the zero baseline. LeHome-Fold separates four garment categories into seen and unseen assets in the nominally paired two-repeat cohort; circles/diamonds mark seen/unseen garments. LIBERO-Long includes all ten tasks and four repeated evaluations of fixed weights on the same initial-state set. Purple denotes the mean controller–Actor difference. Task identifiers are defined in Table A6.

D.3 Paired outcomes relative to the Process Actor

Figure A2 reports failure-to-success conversions and success-to-failure regressions relative to the Process Actor. Their net effects are success-rate gains of 2.08 and 0.95 percentage points on LeHome-Fold and LIBERO-Long, respectively.

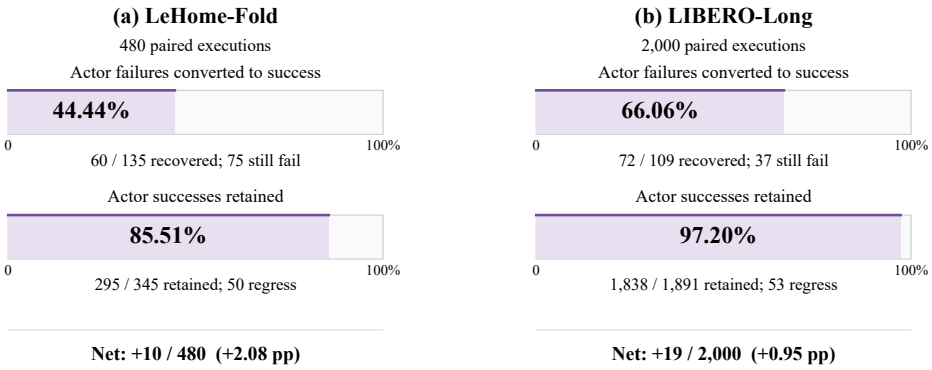


Figure A2: **Paired outcomes relative to the Process Actor.** Top: the fraction of Actor failures converted to success. Bottom: the fraction of Actor successes retained. Net SR change divides failure-to-success minus success-to-failure counts by all pairs. LeHome-Fold uses 480 nominal reset-index pairs over two repeats; LIBERO-Long uses 2,000 initial-state-matched pairs over four repeats. Pairing details are given in Appendix C.

E Behavioral and diagnostic analysis

LB-repeat evaluates the 16×2 controller in four fixed-weight executions of the same 500 official initial states under seed 7. The primary 4×2 results are reported in Tables 2 and 4.

E.1 Repeated gains, goal persistence, and command continuity

Across four executions, PatchVLA averages 95.50% SR, compared with 94.55% for the same-weights Process Actor and 94.40% for the official checkpoint. Table A4 reports completion, goal persistence, and command-continuity metrics.

Table A4: **Completion, goal persistence, and command continuity with the 16×2 controller.** Results over 2,000 episodes per method. Actor uses PatchVLA's process-actor weights, and Official is the locally evaluated official $\pi_{0.5}$. Panel (a) groups completion, goal coverage/persistence, and command-continuity metrics. Panel (b) retains all four SR repeats. Bold/underline mark best/second distinct values within each comparable panel, with ties sharing rank.

(a) Behavioral metrics: four-execution means

Metric	PatchVLA	Actor	Official
Completion			
SR (%) ↑	**95.50**	<u>94.55</u>	94.40
SR@300 (%) ↑	**82.20**	<u>82.00</u>	81.00
SR@400 (%) ↑	89.20	**90.05**	<u>89.85</u>
B-AUC ↑	**47.899**	47.754	<u>47.883</u>
Goal coverage and persistence			
NGC (%) ↑	**97.225**	<u>96.325</u>	<u>96.325</u>
NG-AUC ↑	**59.505**	59.061	<u>59.117</u>
Final goal coverage (%) ↑	**97.608**	96.958	<u>97.025</u>
PG-AUC ↑	**59.346**	58.850	<u>58.957</u>
NR-SR$_1$ (%) ↑	**89.95**	<u>88.60</u>	86.45
NR-SR$_2$ (%) ↑	**92.10**	<u>91.70</u>	89.80
NR-SR$_5$ (%) ↑	**95.20**	<u>94.00</u>	93.30
Command continuity			
Command MSE ($\times 10^{-3}$) ↓	**0.891**	<u>1.058</u>	1.234
Replanning jump MSE ($\times 10^{-3}$) ↓	**1.475**	<u>2.392</u>	3.230

(b) SR (%) by repeated execution

Method	Repeat 1	Repeat 2	Repeat 3	Repeat 4	Mean
$\pi_{0.5}$ checkpoint	94.8	93.4	<u>94.4</u>	**95.0**	94.40
Process Actor	<u>95.2</u>	<u>94.6</u>	<u>94.4</u>	94.0	<u>94.55</u>
PatchVLA controller	**96.2**	**95.6**	**95.8**	<u>94.4</u>	**95.50**

E.2 Command changes around replanning

Figure A3 localizes the command-continuity difference to the replanning event. In repeat 1, episode-mean boundary MSE is 2.411 for Actor and 1.474 for PatchVLA ($\mathrm{MSE} \times 1000$), while the smaller command changes are concentrated at replanning boundaries. Each controller is aligned to its own replanning events.

E.3 Goal acquisition, loss, and recovery

Figure A4 distinguishes terminal completion from the trajectory of environment goal predicates. It shows all 50 trial pairs of the two-object basket task in repeat 1. In the highlighted pair, both policies succeed, but Actor loses a previously satisfied predicate for 33 steps. This example records a recovery pattern beyond terminal SR. The persistence metrics over the existing four-repeat cohort in Table A4 provide the corresponding aggregate comparison beyond final success.

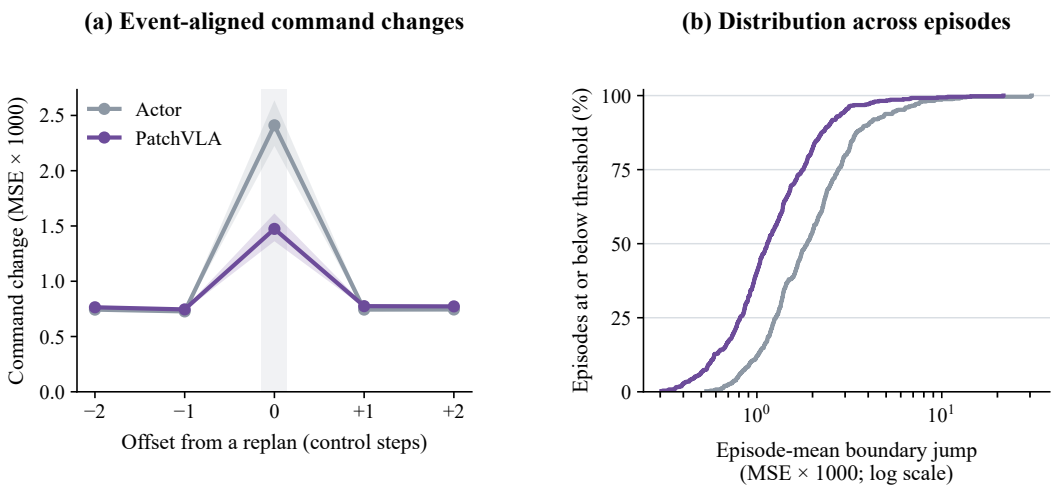


Figure A3: **Where command changes occur.** All ten LIBERO-Long tasks and 500 matched initial states from repeat 1. (a) Offset zero spans two successive plans. Differences of six clipped non-gripper command components are averaged within each episode, then equally across episodes. Bands are marginal 95% intervals from 2,000 task-stratified paired-episode bootstrap draws and quantify episode-level sampling variation. (b) Empirical distribution of episode-mean boundary jumps. They summarize command continuity at replanning boundaries.

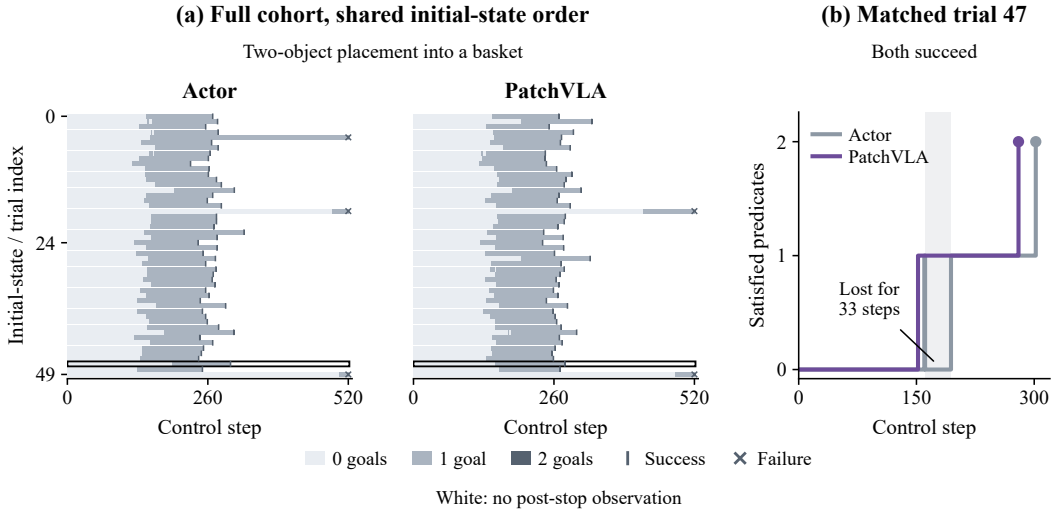


Figure A4: **Goal dynamics beyond final success.** (a) All 50 initial-state-matched trials, ordered by trial index. Gray intensity gives the number of satisfied environment predicates; ticks/crosses mark terminal success/failure, and white denotes unobserved time after termination. Black outlines identify the pair shown in (b). (b) Trial 47: Actor loses the soup-in-basket predicate at step 161 and recovers it at 194. The example is the longest recovered Actor loss among pairs where both succeed and PatchVLA has no recorded loss. Aggregate retention is reported in Table A4. Raw predicates differ from predicted process signals and from the confirmation rule in NR-SR_2.

E.4 Metric definitions and aggregation

Each evaluation has ten tasks with 50 official initial states each. The control budget is $L = 520$ environment steps, distinct from the action-chunk horizon. Ten settling actions precede $t = 0$ and are excluded. These post hoc behavioral diagnostics were defined after inspecting the evaluations and fixed before computation.

Success under a control budget. Let $T_i \le L$ be the terminal control step and $s_i \in \{0,1\}$ the official task-success indicator for episode i. Write $\mathcal{M}$ for an episode mean within each task followed by an equally weighted mean over the ten tasks. Define

$$\mathrm{SR@}b = 100\,\mathcal{M}[s_i \mathbf{1}\{T_i \le b\}], \qquad \mathrm{SR} = 100\,\mathcal{M}[s_i],$$
$$\text{B-AUC} = \frac{1}{L}\sum_{b=1}^{L} \mathrm{SR@}b = 100\,\mathcal{M}\left[s_i \frac{L+1-T_i}{L}\right]. \tag{10}$$

The sum includes the success step, and failures contribute zero. Here the budget is measured in control steps, and area scores range from 0 to 100.

Completion and persistence of new goals. Let $g_{i,t,j} \in \{0,1\}$ be the simulator's native terminal-goal predicate j. Only predicates false after settling are new goals: $J_i = \{j : g_{i,0,j} = 0\}$ and $p_{i,t} = |J_i|^{-1} \sum_{j \in J_i} g_{i,t,j}$. The scores are

$$\mathrm{NGC} = 100\,\mathcal{M}[p_{i,T_i}],$$
$$\text{NG-AUC} = \frac{100}{L}\,\mathcal{M}\left[\sum_{t=1}^{T_i} p_{i,t} + s_i(L - T_i)\right]. \tag{11}$$

For persistence, replace $g_{i,t,j}$ by $q_{i,t,j} = \min_{u=t,\ldots,T_i} g_{i,u,j}$ in Eq. (11) to obtain PG-AUC. PG-AUC measures goal persistence retrospectively until observed episode termination. Final goal coverage instead includes all terminal predicates, including those initially satisfied.

Success is padded with ones to L using an absorbing-state convention. Evaluation checks require failures to reach L; abnormal early termination requires separate handling. Empty J_i makes new-goal metrics undefined: exclude the episode from that within-task mean and report valid counts (500 per method/repeat here). These outcomes are computed from external simulator predicates. Phase, progress, and boundary prediction use separate targets.

Regression and command diagnostics. NR-SR_2 counts official successes with no subsequently lost new goal; acquisition/loss requires two consecutive identical samples. Failures contribute zero. NR-SR_1/NR-SR_5 provide one-/five-sample sensitivity checks for the same event definition. Command MSE averages squared adjacent-command differences over six continuous native components after common clipping to $[-1, 1]$, excluding the gripper and terminal padding. Replanning jump MSE restricts this calculation to actual replanning boundaries, excluding the initial call. Command MSE and replanning jump MSE quantify command variation and are reported alongside task success.

F Baseline details and checkpoint sensitivity

F.1 Baseline evaluation scope

Table 2 compares baselines under the 500-episode protocol (Zheng et al., 2025; Yuan et al., 2026; Li et al., 2026a; Shukor et al., 2025) and reports the same PatchVLA 4×2 result as Table 4. X-VLA command metrics are omitted because its action representation is incompatible with that comparison. The official $\pi_{0.5}$ baseline is separately trained, while the same-weights Process Actor belongs to LB-repeat (Table A4).

F.2 LeHome-Fold checkpoint and repeat results

Table A5 reports LeHome-Fold success rates for the original and additional epoch-2 executions, alongside the final epoch-5 controller. Checkpoint selection is described in Appendix C.2.

Method / evaluation block	Seen	Unseen	**Overall SR**
Process Actor, repeats 1–2	<u>74.75</u>	<u>57.50</u>	<u>71.88</u>
PatchVLA, final epoch 5	73.25	<u>57.50</u>	70.63
PatchVLA e2, repeats 1–2	**76.00**	**63.75**	**73.96**
PatchVLA e2, repeats 3–4	73.00	62.50	71.25
PatchVLA e2, pooled 1–4	74.50	63.13	72.60

Table A5: **LeHome-Fold checkpoint and repeat results (SR, %).** The shaded **Overall SR** column reports aggregate success rate; Seen and Unseen are also success percentages. Runs use the same frozen Actor and seed 42. Actor results use repeats 1–2; the epoch-2 controller is additionally evaluated on repeats 3–4. Bold/underline mark best/second within the original two-repeat block, with ties sharing rank.

G Aligned execution traces

G.1 Task index and replay selection

Table A6 defines the task identifiers used in Figure A1 and the traces below. LeHome-Fold fixes Seen asset 0 in each category and takes its first successful episode from the epoch-2, step-1071, 4×2 archive: the second attempt for long pants and the first for the other categories. LIBERO-Long takes the first success in repeat/trial order from LB-repeat, which is repeat 1, trial 00 for every task. Four frames show initialization, two intermediate interactions, and first success.

ID	LIBERO-Long task
01	Soup and tomato sauce in the basket
02	Cream cheese and butter in the basket
03	Turn on stove; place moka pot on it
04	Black bowl in bottom drawer; close drawer
05	White mug on left plate; yellow-and-white mug on right plate
06	Book in rear caddy compartment
07	White mug on plate; pudding to its right
08	Soup and cream cheese in the basket
09	Both moka pots on the stove
10	Yellow-and-white mug in microwave; close microwave

Table A6: **Task identifiers.** LT/ST denote long-/short-sleeved tops; LP/SP denote long/short pants. LIBERO-Long follows the official task order.

G.2 Query-aligned process and critic signals

Figures A5 and A6 cover all four garment categories and all ten LIBERO-Long tasks. Control steps count executed policy actions. At each query, the phase strip and progress p report current readouts; b is the maximum boundary probability over the selected candidate's actually executed prefix, truncated at termination. These query-time readouts form a sequence across the episode. Phase, p, and b use piecewise-constant displays. The critic trace connects successive query readouts, and frame guides mark the corresponding control steps. The terminal interval has no further query; LeHome-Fold excludes the evaluator's post-success settling tail.

The selected candidate's calibrated S (Eq. (3)) and the final-round candidate min–max band use the right axis. Progress and boundary use the left 0–1 axis. The critic axis is 0–1 for LeHome-Fold; each LIBERO-Long panel labels its own local range covering all candidate scores. The phase abbreviations Align (mechanism figures) and Pre-contact (traces) both denote pre-contact alignment.

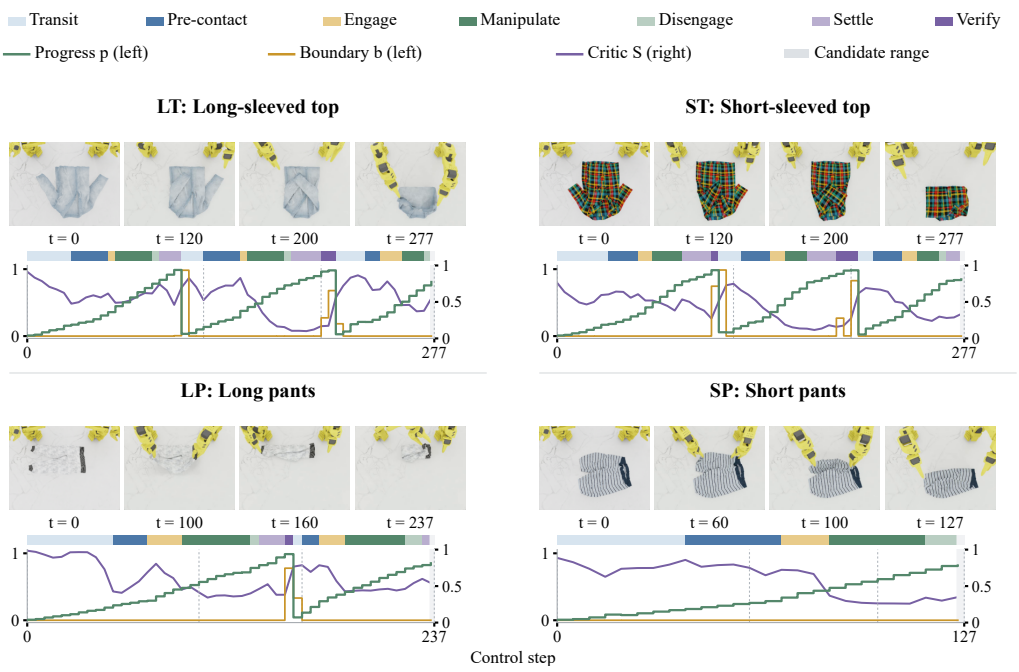


Figure A5: **LeHome-Fold: keyframes and predictions across garment categories.** Original top-view frames show one successful Seen-asset replay per category, selected by the rule above. Timestamps and vertical guides align keyframes with recorded predictions on the control-step axis.

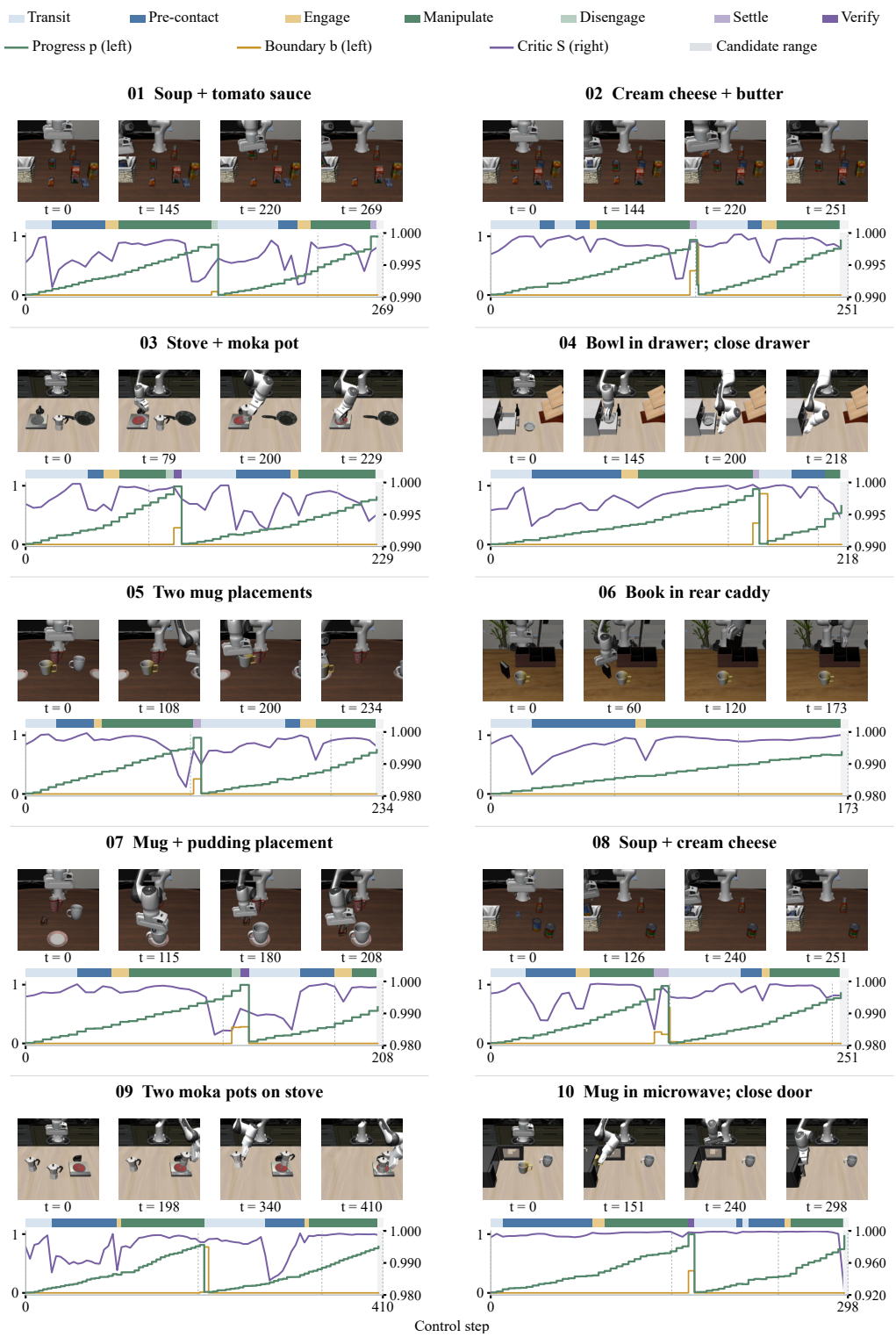


Figure A6: **LIBERO-Long aligned traces: all ten tasks (01–10).** Each panel uses repeat 1, trial 00 of the archived 16×2 controller. The right axis is local to each panel and covers its candidate score range; all tasks share one legend.