PCMP: LEARNING PACE-CONTROLLABLE MOTION PRIORS FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision–language–action (VLA) models connect visual–language understanding to robotic manipulation. However, motion-level generalization is limited in multi-task settings, and learning controllable execution pace from a single dataset remains difficult. These limitations motivate a reusable motion backbone that preserves task structure while responding to user-specified execution rates. We introduce Pace-Controllable Motion Priors (PCMP), a hierarchical framework that separates motion content, current progress, and desired future progression. PCMP encodes semantically aligned demonstrations into motion latents supervised by reconstruction and learns a progress-conditioned flow-matching prior for action generation. Complementary spatial and progress-schedule augmentations vary motion geometry and timing independently. A visual-language upper model predicts the motion latent and current progress, connecting the prior to closed-loop manipulation while retaining an explicit interface for pace commands. Experiments on RoboTwin show that the learned prior provides an effective motor backbone and retains useful execution capabilities when coupled to visual perception. Motion-latent ablations support the contribution of the trajectory representation, while speed-control experiments reveal a systematic response to user-specified rates: mean execution length decreases as the commanded rate increases. These findings support learning reusable motion structure separately from visual grounding and demonstrate a promising route toward manipulation with user-controlled execution pace.

1 INTRODUCTION

Learning robot manipulation policies requires connecting task understanding with reliable physical execution. Embodied multimodal models demonstrate how pretrained visual and linguistic knowledge can support perception and robotic reasoning (Driess et al., 2023). Vision–language–action (VLA) models bring these capabilities into action prediction, mapping visual observations and language instructions to robot actions (Zitkovich et al., 2023). However, building these models is challenging because perception, task understanding, and motion control must work together within a single policy. A common training recipe initializes the policy from a pretrained vision–language model, learns action prediction from robot demonstrations, and adapts the resulting model to target tasks through supervised fine-tuning (Kim et al., 2025). Demonstration quality and collection also shape policy learning (Mandlekar et al., 2021), motivating diverse data from BridgeData V2, DROID, and UMI (Walke et al., 2023; Khazatsky et al., 2024; Chi et al., 2024). Open X-Embodiment pools demonstrations across robot platforms, while Octo learns a generalist policy from heterogeneous data that can be adapted to new observation and action spaces (Open X-Embodiment Collaboration, 2024; Octo Model Team et al., 2024).

Multi-task manipulation requires both reusable motion skills and flexible execution pace. **(1) Difficulty learning controllable execution pace from a single dataset.** A single demonstration dataset may offer limited timing variation, leading policies to inherit its execution pace. Shortening execution intervals can increase tracking errors and encounter physical reachability constraints (Kim et al., 2026). Retiming and speed-conditioned training motivate separating motion content from its execution schedule. **(2) Limited motion-level generalization in multi-task settings.** Task- and scene-specific fine-tuning data constrain action generalization, particularly in multi-task and long-horizon settings; broad visual–language knowledge does not automatically yield transferable motion

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.

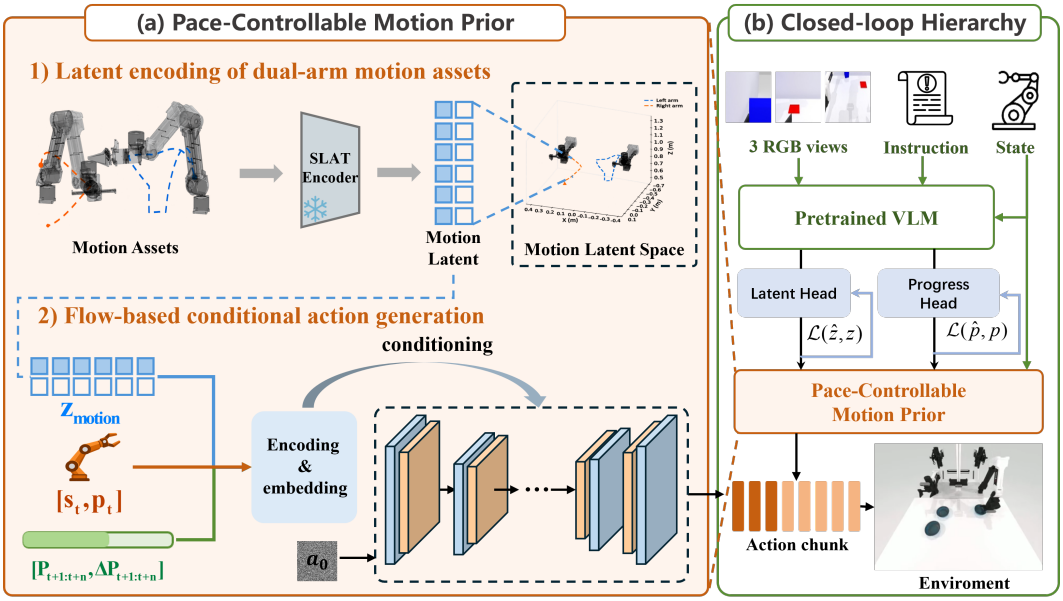


Figure 1: Overview of PCMP. (a) A frozen encoder supplies motion latents; the motion prior generates action chunks conditioned on the latent, current state and progress, and a future-progress command. (b) A visual-language model predicts the latent and progress, while a scheduler sets the requested pace for closed-loop execution.

skills. Implicit interfaces, such as GR00T N1's cross-attention conditioning (Bjorck et al., 2025), leave motion generation coupled to perceptual features. Explicit hierarchies expose language subtasks (Hi Robot (Shi et al., 2025)) or 2D end-effector paths (HAMSTER (Li et al., 2025)), but do not jointly specify complete 3D motion, gripper events, and execution progress. These limitations motivate a reusable motion prior that supports motion-level generalization and separates intended motion from its execution pace.

Advances in 3D asset generation suggest a way to build a structured motion representation for such a prior. A common approach first learns reconstructable shape representations with encoder–decoder models, then learns conditional mappings from images or text to these representations (Zhang et al., 2023; Zhao et al., 2023b; Xiang et al., 2025). This separation gives latent features a geometric referent and allows representation fidelity to be assessed separately from conditional prediction. We draw on this principle to treat a complete manipulation demonstration as a *motion asset*: an encoder–decoder learns latent tokens that retain its spatial paths, orientations, and gripper events, while an upper model learns to predict these tokens from task observations and instructions. However, reconstructing a motion reference does not by itself provide closed-loop execution. The robot must determine where it is within the motion and how that motion should advance from its current state. Our insight is therefore to pair the motion latent with explicit current and future progress, separating motion content from its execution schedule. This interface lets us train and evaluate the lower motion policy using encoded demonstrations before introducing errors from visual inference.

We introduce *Pace-Controllable Motion Priors* (PCMP), a hierarchical framework that separates visual task inference from pace-controlled motion execution (Figure 1). Building on a structured encoder–decoder, we encode semantically aligned manipulation trajectories into motion latents supervised by reconstruction. We then train a lower motion policy with consistency flow matching, using state–progress cross-attention to retrieve motion features and future-progress commands to condition action generation. Two complementary augmentations support this separation: semantic-anchor augmentation varies spatial motion assets, while progress-schedule augmentation varies their execution schedules. Finally, a visual-language upper model predicts the motion latent and current progress, connecting the learned motion prior to closed-loop manipulation. The resulting interface supports separate training and assessment of motion reconstruction, latent-conditioned execution, and visual task inference. Our main contributions are:

- We formulate manipulation demonstrations as reconstructable motion assets, grounding structured latent tokens in complete trajectories and providing a shared interface for visual prediction and motion execution.
- We develop a progress-conditioned motion prior that separates intended motion, current execution stage, and future progression. Complementary semantic-anchor and progress-schedule augmentations vary motion geometry and execution timing independently, supporting learning across spatial variations and progress schedules.
- We evaluate pace-controllable execution under different constant speed multipliers, assessing the ability to adjust execution pace at runtime while preserving the intended motion. This evaluation examines flexible temporal control beyond a single fixed execution rate.

2 Related Work

2.1 Multi-Task and Hierarchical VLA Models

RLBench, CALVIN, and LIBERO benchmark multi-task, long-horizon, and lifelong manipulation, respectively (James et al., 2019; Mees et al., 2021; Liu et al., 2023). Generalist policies share models across tasks and modalities (Brohan et al., 2022; Reed et al., 2022; Jiang et al., 2022), while OpenVLA and π_0 map observations and language to actions (Kim et al., 2025; Black et al., 2024). PerAct, Act3D, and RVT strengthen spatial grounding through 3D representations (Shridhar et al., 2022; Gervet et al., 2023; Goyal et al., 2023). Hierarchies instead condition control on language-selected skills (Ahn et al., 2023; Shi et al., 2025), latent plans from play (Lynch et al., 2019; Wang et al., 2023), or trajectory sketches (Gu et al., 2023). Generated programs and 3D value maps provide further planning interfaces (Liang et al., 2022; Huang et al., 2023); Long-VLA uses phase-aware masking for long-horizon behavior (Fan et al., 2025). These methods organize task intent or geometry without explicitly separating motion content from execution pace. PCMP conditions a separately trained motor backbone on a whole-episode motion latent, current progress, and a future progress reference, allowing motion learning and evaluation independently of visual grounding.

2.2 Learning and Controlling Execution Tempo

Movement primitives represent reusable motion structure (Ijspeert et al., 2013; Paraschos et al., 2013), and path parameterization optimizes timing along a fixed path (Pham & Pham, 2018). Learned approaches condition motion on completion time (Saigusa et al., 2021), selectively accelerate demonstrations (Guo et al., 2025), or combine tempo-diverse pretraining with reinforcement learning (Nam & Hwang, 2025). TempoVLA uses scalar speed conditions and trajectory retiming (Jing et al., 2026a); real-time chunking instead reduces inference-induced execution delays (Black et al., 2025). PCMP controls task advancement through a future semantic-progress reference: its values specify upcoming stages, and its increments specify pace. Progress-schedule augmentation pairs each motion latent with different schedules, supporting varying temporal commands without redefining motion content or uniformly scaling robot velocity.

2.3 Action Experts and Motion Priors

SPiRL learns skill priors for downstream reinforcement learning (Pertsch et al., 2020), while AMP learns motion-based style rewards for character control (Peng et al., 2021). Robot action experts acquire motor structure through joint VLA training in π_0, separate diffusion pretraining (Wen et al., 2025), latent actions from video (Ye et al., 2025), or trajectory autoencoding and latent distillation (Jing et al., 2026b). DreamZero jointly predicts video and actions (Ye et al., 2026). Complementary representations include action chunks (Zhao et al., 2023a), discretized actions with corrections (Shafiullah et al., 2022), and temporal or frequency-based compression (Zheng et al., 2024; Pertsch et al., 2025). Diffuser generates planning trajectories (Janner et al., 2022); DP3 combines point-cloud features with diffusion (Ze et al., 2024). Diffusion Policy and FlowPolicy model continuous action distributions (Chi et al., 2023; Zhang et al., 2024). PCMP uses consistency flow matching with explicit motion and progress conditions, making motion content and execution pace separately controllable.

3 PRELIMINARY AND TASK FORMULATION

Problem Formulation. Multi-task vision-language-action (VLA) models learn a shared, language-conditioned policy across manipulation tasks. At control step i, the policy receives an observation $o_i = (I_i, x_i)$, comprising visual input I_i and optional proprioceptive state x_i, together with the language instruction ℓ_c for task c, and predicts an action chunk:

$$\widehat{A}_i \sim \pi_\phi(\cdot \mid o_i, \ell_c), \qquad A_i = [a_i, a_{i+1}, \ldots, a_{i+H-1}], \tag{1}$$

where H denotes the prediction horizon.

Weakness. This formulation has two limitations. First, timing is implicit in the learned action sequences, with no independent input for commanding execution pace. Second, action generation remains coupled to perceptual features and task-specific supervision: pretrained visual–language knowledge does not by itself ensure motion-level generalization across tasks or long-horizon stages. These limitations motivate a reusable motion representation with separate conditions for motion identity, current progress, and the desired future progress schedule.

4 METHOD

Appendix A provides implementation details, architecture settings, and training hyperparameters.

4.1 MOTIVATION

PCMP separates three decisions in long-horizon manipulation: selecting *which motion to execute*, estimating *where execution is*, and specifying *how fast to advance*. For long-horizon manipulation, a structured motion latent z represents the intended motion and its semantic ordering, a scalar progress p locates the current stage of execution, and a future progress command C specifies the desired pace of advancement. This decomposition lets the same motion plan run under different progress schedules. A visual model grounds the motion and current progress in the scene, while a flow-matching motion policy translates these estimates and the requested schedule into robot actions. Figure 1(a) summarizes motion-prior learning, and Figure 1(b) shows the deployment interface.

4.2 PHYSICALLY GROUNDED MOTION LATENTS WITH TRELLIS

Why a trajectory-grounded task representation? A latent interface alone does not separate task representation from action execution. When a task code is defined only through downstream action prediction, semantic intent and motion can remain entangled, and its correspondence to a concrete reference trajectory is not explicitly constrained. This makes it difficult to assess the lower-level controller independently of the model that supplies its task code. We instead derive the task representation from an explicit manipulation trajectory and train it with a reconstruction objective. The resulting code has a measurable physical referent: the demonstrated motion. Once the encoder is fixed, demonstration-derived codes can condition the motion policy directly, allowing independent training and testing without visual task inference.

Manipulation trajectories as motion assets. Our design draws inspiration from 3D asset reconstruction and generation, where an encoder–decoder learns a latent representation from which an asset's geometry can be reconstructed. TRELLIS provides a structured latent representation for this purpose (Xiang et al., 2025). We treat a complete manipulation trajectory as a motion asset, $\mathcal{X}_e = (x_{e,i})_{i=0}^{N_e-1}$. For our dual-arm robot, each state is

$$x_{e,i} = \mathrm{concat}_{b\in\{L,R\}}\left[r^b_{e,i}, \rho^b_{e,i}, g^b_{e,i}\right] \in \mathbb{R}^{20}, \qquad r^b_{e,i} \in \mathbb{R}^3, \quad \rho^b_{e,i} \in \mathbb{R}^6, \quad g^b_{e,i} \in [0,1]. \tag{2}$$

Here r^b is the Cartesian end-effector position, ρ^b its continuous 6D rotation representation (Zhou et al., 2018), and g^b the normalized gripper opening for arm b. Each arm contributes ten dimensions, so the ordered asset captures both arms' spatial paths, orientations, and gripper events. We encode this physical state sequence into the motion latent z_e, while specifying the desired execution tempo separately through the progress command. The left branch of Figure 2 illustrates this encoding–reconstruction interface, while the right side connects the motion asset to semantic-anchor and progress-schedule augmentation.

Aligning multi-task dual-arm motion assets. The motion assets $\mathcal{X}_e$ vary in duration and execution pace. We first align demonstrations of each task using shared semantic phase anchors, assigning each dual-arm state $x_{e,i}$ a progress value $p_{e,i} \in [0, 1]$. Both arms use the same progress coordinate to retain their coordination. We then resample each asset on a uniform progress grid:

$$\widetilde{\mathcal{X}}_e = (\widetilde{x}_{e,k})_{k=0}^{K-1}, \qquad \widetilde{x}_{e,k} = Y_e\left(\frac{k}{K-1}\right), \tag{3}$$

where Y_e interpolates $\mathcal{X}_e$ with respect to progress. This gives all tasks a common K-state asset format, with semantic correspondence established within each task. The grasp, handover, and target stages in Figure 2 illustrate the ordered semantic structure retained by this alignment, even when stage durations differ.

Encoder and Decoder. We adapt TRELLIS's sparse Transformer encoder and decoder to reconstruct aligned motion assets $\widetilde{\mathcal{X}}_e$. The encoder uses self-attention (Vaswani et al., 2017) across trajectory points to map asset features F_e (normalized states and their local differences) and quantized state coordinates S_e to a Gaussian posterior $q_\psi(z \mid F_e, S_e)$. A matching Transformer decoder D_ω reconstructs the asset features from the sampled motion latent on the retained coordinates. We train the encoder–decoder jointly with a reconstruction loss and KL regularization, following variational autoencoding (Kingma & Welling, 2013):

$$\mathcal{L}_{\mathrm{VAE}} = \mathbb{E}_{z\sim q_\psi(\cdot|F_e,S_e)}\left[\mathrm{SmoothL1}(D_\omega(z; S_e), F_e)\right] + \beta D_{\mathrm{KL}}(q_\psi(z \mid F_e, S_e) \,\|\, \mathcal{N}(0, I)). \tag{4}$$

The first term supervises motion-asset reconstruction, while the second regularizes the posterior toward a standard Gaussian, with weight β. After training, the posterior mean $z_e \in \mathbb{R}^{K\times d_z}$ serves as the motion latent for the lower-level policy. The encoder is frozen when supplying targets for motion-policy and visual training, as indicated by the snowflake in Figure 1(a). The trajectory inset visualizes the physical reference associated with a code; its axes are Cartesian coordinates, not axes of the learned latent space.

4.3 Pace-Controllable Flow-Matching Motion Prior

Execution pace is controlled through a future-progress schedule: progress indexes the motion, while its requested increments determine how quickly execution should advance at a fixed control interval. We therefore use *pace-controllable* to describe the capability and *progress-conditioned* to describe the underlying mechanism.

Input Representation. The motion policy receives three complementary conditions: the motion latent z_e specifies *what motion to execute*; a single observation $O_i = (x_i, p_i)$ gives the current robot state and task progress, specifying *where execution currently is*; and the future progress command $C_i = [P_i^{\mathrm{ref}}, \Delta P_i^{\mathrm{ref}}]$ specifies *how the motion should advance*. Here P_i^{ref} contains desired future progress values and $\Delta P_i^{\mathrm{ref}}$ their successive increments. At a fixed control interval, these increments provide explicit information about the desired task-progression rate, complementing the single observation. A state–progress query retrieves relevant information from z_e through cross-attention; the retrieved features and an embedding of C_i condition the action decoder.

Flow-Matching Action Decoder. Our action decoder uses flow matching (Lipman et al., 2022) to generate action chunks from Gaussian noise, conditioned on $\kappa_i = (O_i, z_e, C_i)$. A temporal U-Net implements the conditional action generator in Figure 1(a) and parameterizes the vector field $v_\theta(A^\tau, \tau; \kappa_i)$ along

$$A^\tau = (1-\tau)A^0 + \tau A^\star, \qquad A^0 \sim \mathcal{N}(0, I), \quad \tau \in [0, 1], \tag{5}$$

where $A^\star$ is a demonstrated action chunk and τ denotes generative flow time. For a flow segment ending at η, the predicted endpoint is

$$F_\theta^\eta(A^\tau, \tau; \kappa_i) = A^\tau + (\eta - \tau)v_\theta(A^\tau, \tau; \kappa_i). \tag{6}$$

We train with a consistency flow-matching loss (Zhang et al., 2024) that enforces agreement between endpoint predictions and vector fields at neighboring flow times τ and τ':

$$\begin{aligned}\mathcal{L}_{\mathrm{motion}} = \mathbb{E}\Big[&\big\|F_\theta^\eta(A^\tau, \tau; \kappa_i) - \bar{F}^\eta(A^{\tau'}, \tau'; \kappa_i)\big\|_2^2 \\ &+ \lambda_v \chi(\tau, \tau', \eta)\big\|v_\theta(A^\tau, \tau; \kappa_i) - v_\theta(A^{\tau'}, \tau'; \kappa_i)\big\|_2^2\Big].\end{aligned} \tag{7}$$

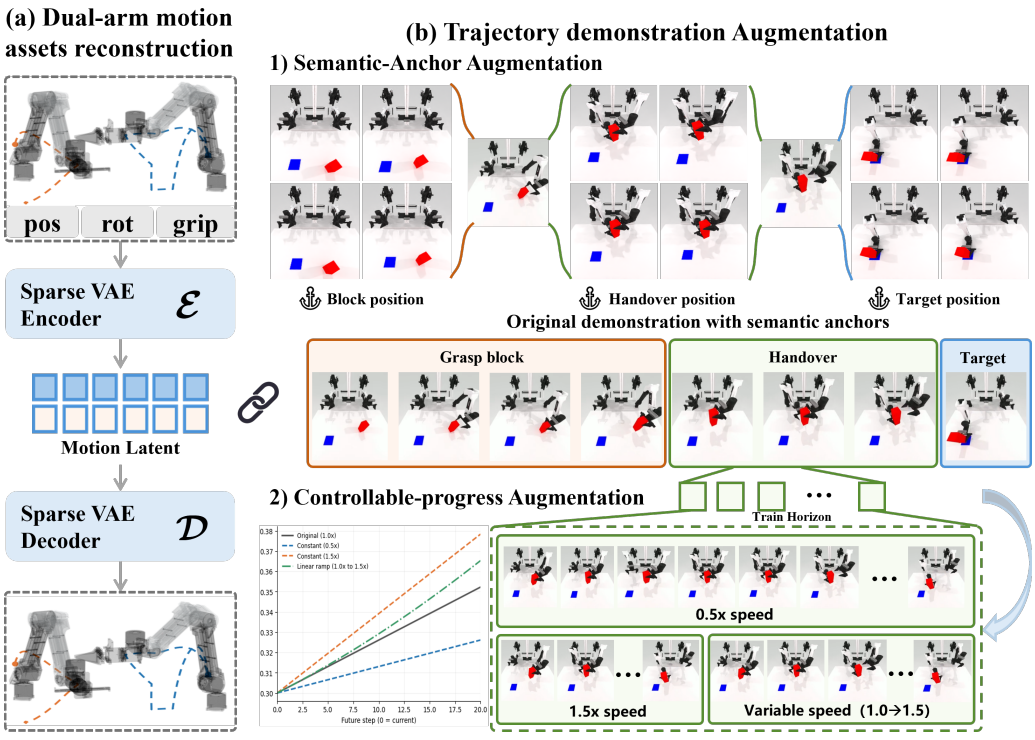


Figure 2: Motion encoding and augmentation. Left: an adapted TRELLIS VAE encodes and reconstructs dual-arm motion assets. Upper right: semantic-anchor augmentation varies motion geometry and recomputes the latent. Lower right: progress-schedule augmentation varies execution pace while preserving the motion latent.

The expectation is over training examples, Gaussian noise, and sampled flow times; both interpolated chunks share the same A^0 and $A^\star$. Here $\bar{F}^\eta$ is the neighboring endpoint prediction or the known interpolation endpoint at the configured boundary, χ selects valid vector-consistency pairs, and λ_v balances the two terms. The first term aligns the predicted endpoints; when χ is active, the second encourages consistent flow directions and magnitudes. This vector field describes the generative process; physical execution tempo is conditioned through C_i. At inference, the learned field transports noise to an action chunk; the controller executes a prefix before re-observing and replanning.

Controllability through Augmentation. As illustrated in Figure 2, we augment the motion asset and its traversal schedule separately: spatial augmentation changes the reference to be encoded, whereas temporal augmentation changes how that reference is traversed, complementing demonstration adaptation across scenes as in MimicGen (Mandlekar et al., 2023). *Semantic-anchor augmentation* applies smooth spatial perturbations to task-specific anchors to produce new motion assets. The transformed states define new action targets and are re-encoded to obtain matching motion latents, preserving consistency between the intended motion and its supervision (upper right of Figure 2). *Progress-schedule augmentation* instead retains the same motion asset and z_e, applies constant-rate or variable-speed progress schedules, and resamples states along $Y_e(p)$ to regenerate action targets and C_i. The lower right of Figure 2 illustrates constant-rate and linearly varying schedules applied to the same handover motion. Pairing one motion latent with different schedules teaches the decoder to respond to requested task-progression rates rather than reproduce a single demonstrated tempo.

4.4 Connecting Visual Grounding and Motion Generation

We connect the motion prior to the pretrained $\pi_{0.5}$ visual-language backbone (Physical Intelligence et al., 2025), using SigLIP image features (Zhai et al., 2023). Given fused features h_i of the current

images, robot state, and task instruction, two readout heads predict the standardized motion latent and current progress:

$$\widehat{\tilde{z}}_i = G^z_\varphi(h_i), \qquad \widehat{p}_i = G^p_\varphi(h_i, \widehat{p}_{i-1}). \tag{8}$$

Only the progress head uses its previous prediction; the latent branch uses current observations. The heads are trained against frozen-encoder latents and aligned progress targets, with predicted progress fed back between steps. Appendix A.5 provides architecture, loss, and training details.

At deployment, $\widehat{\tilde{z}}_i$ is denormalized to $\widehat{z}_i$, and a scheduler constructs C_i from $\widehat{p}_i$ and the requested pace:

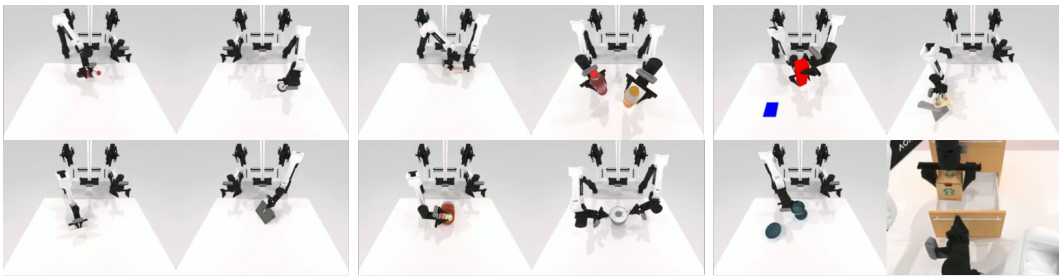
$$\widehat{A}_i = \pi_\theta((x_i, \widehat{p}_i), \widehat{z}_i, C_i)\,. \tag{9}$$

The controller executes a prefix and re-observes, grounding motion in vision while preserving explicit control of reference advancement.

5 Experiments

Our experiments investigate whether PCMP can modulate execution pace while preserving the intended motion and task completion. We address three questions: (1) Are augmented references executable, and does the motion representation support reconstruction and control (E1–E3)? (2) Can the lower motion prior respond to different pace commands while keeping the motion reference fixed (E4)? (3) Is task execution retained under pace commands when the motion latent and current progress are predicted from visual observations (E5)? The evaluation therefore progresses from reference validation and motion-prior analysis to visually grounded execution.

Experimental Setup. We use twelve RoboTwin 2.0 (Chen et al., 2025) tasks spanning free-space, contact-rich, and multi-stage bimanual manipulation (Figure 3), with episode-level data splits. Unless explicitly labeled as training-set results, all motion-prior evaluations use unseen trajectories within these tasks. We distinguish privileged motion-prior execution from visually grounded control. Slower pace commands receive proportionally longer rollout budgets. SR includes all attempted rollouts; reference-tracking errors assess motion fidelity. Detailed protocols and results are given in Appendices B and C.



(a) Easy tasks **(b) Medium tasks** **(c) Hard tasks**

Figure 3: Twelve RoboTwin 2.0 tasks covering diverse object interactions and bimanual manipulation.

5.1 E1: Reference Feasibility

We first assess whether the augmented references remain executable in simulation. For each of the twelve tasks, we replay 100 references generated by semantic-anchor augmentation and separately evaluate 50 source episodes at $0.5\times$ and $1.5\times$ through progress retiming. Table 1 reports mean SRs of 95.3% for spatial augmentation and 99.0%/98.0% for slow/fast retiming, with at least 90% SR on each task in each condition. These results support the feasibility of the augmented supervision under the tested replay settings; learned policy execution is evaluated separately below. Per-task results and replay protocols are provided in Appendix C.1.

Table 1: Replay SR (%) under spatial augmentation and progress retiming.

Task	Diff	Aug.	0.5×	1.5×
Adjust Bottle	Easy	94	100	98
Grab Roller	Medium	99	100	100
Hanging Mug	Hard	91	98	96
Put Object Cabinet	Hard	98	100	98
...	...	...	...	...
Mean (12 tasks)	/	95.3	99.0	98.0

Table 2: Motion asset reconstruction errors on held-out trajectories. Lower is better.

Task	Pos. (mm)	Rot. (deg)	Grip
Adjust Bottle	1.18	0.43	0.0027
Grab Roller	1.33	0.61	0.0030
Hanging Mug	1.52	0.48	0.0030
Put Object Cabinet	1.66	1.13	0.0041
...	...	...	...
Mean (12 tasks)	1.20	0.44	0.0028

5.2 E2: Motion Assets Reconstruction

We train a single trajectory encoder–decoder jointly on demonstrations from all twelve tasks. Each complete trajectory is resampled to $K = 128$ states on a uniform semantic-progress grid shared by both arms. After reconstruction, the trajectory is interpolated at the original progress values to compute position, rotation, and gripper errors. Table 2 reports errors on held-out episodes, with selected per-task results and an equal-task mean. Mean errors are 1.20 mm, $0.44°$, and 0.0028 for position, rotation, and normalized gripper opening, respectively. These results support accurate reconstruction on unseen trajectories when the shared decoder retains the state coordinates. Figure 4 compares reference and reconstructed Handover trajectories, showing closely aligned paths for both arms. Evaluation details and additional reconstruction visualizations (Figure 5) are provided in Appendices B.3 and C.2.

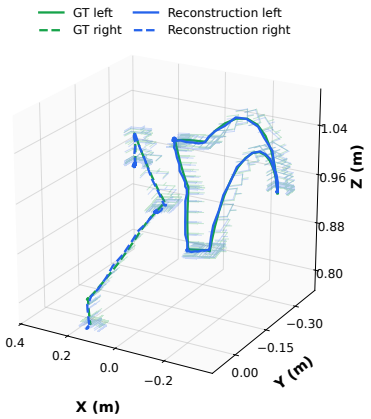


Figure 4: Handover: reference (green) and reconstruction (blue).

5.3 E3: Motion-Latent Ablation

We examine the role of motion-latent conditioning on the Handover task, independently of visual prediction, with 10 rollouts per condition. Table 3 compares four conditions: None is trained without the motion-latent conditioning branch; Zero and Random retain the branch but replace the latent input with zeros and random noise, respectively; Correct supplies the matching motion latent. The correct latent achieves 80% SR, compared with 0% without the conditioning branch, 50% with zero inputs, and 40% with random noise. It also yields the lowest position, rotation, and gripper tracking errors. These results support the importance of motion-latent conditioning for successful execution and reference tracking on this task.

Table 3: Motion-latent ablation on Handover. SR in %. Tracking errors are measured against the commanded reference.

Latent	SR ↑	Pos. ↓ (mm)	Rot. ↓ (deg)	Grip ↓
None	0	7.46	19.29	0.281
Zero	50	4.27	11.54	0.101
Random	40	5.38	14.93	0.126
Correct	**80**	**1.18**	**0.43**	**0.027**

5.4 E4: Speed Control

Training combines constant-rate augmentation at $s \in \{0.5, 0.75, 1.0, 1.5, 2.0\}$ with variable-speed schedules. We evaluate the five constant rates on unseen trajectories, with 10 rollouts per task, policy condition, and rate. The *w/o pace augmentation* model retains the future-progress inputs but is trained without constant-rate or variable-speed augmentation. Within each task, Speed ratio is the resampled reference length (511 steps) divided by the mean successful execution steps. When SR is zero, Steps and Speed ratio are reported as zero. Table 4 gives equal-task means including these zeros; Appendix B.5 details the reporting convention.

As shown in Table 4, the full policy exhibits an ordered response to the commanded pace: execution steps decrease from 849.1 to 262.2 as s increases from 0.5 to 2.0, while the realized speed ratio rises from 0.625 to 2.010. At $1.5\times$ and $2.0\times$, SR remains at 72.5% and 70.8%, respectively, compared with 77.5% at $1\times$. Without pace augmentation, SR falls to 45.8% and 18.3% at these faster commands. Its mean ratio decreases from 1.315 to 0.910, with the latter including zeros for three tasks with no successful rollout. These results support the role of pace augmentation in maintaining successful execution under pace changes. However, calibration remains imperfect, particularly at $0.75\times$, where the realized ratio is 0.982. The aggregate timing statistics should be interpreted alongside SR because they include zero entries for unsuccessful task conditions. Video timelines for all twelve tasks in Appendix C.3.1 (Figures 6–9) illustrate how manipulation unfolds over time at the five labeled pace settings, complementing these aggregate statistics.

Table 4: Pace-augmentation ablation. Equal-task means; SR in %. Ratio = Data/Policy steps (ideal: s). SR superscripts give changes from $1\times$ in percentage points before rounding (↑: increase; ↓: decrease).

	Commanded speed s				
Metric	$0.5\times$	$0.75\times$	$1.0\times$	$1.5\times$	$2.0\times$
w/o pace augmentation					
SR (%)	$36.7^{35.8\downarrow}$	$56.7^{15.8\downarrow}$	72.5	$45.8^{26.7\downarrow}$	$18.3^{54.2\downarrow}$
Steps	1005.1	596.7	507.0	411.8	345.7
Speed ratio	0.533	0.896	1.011	1.315	0.910
Policy (ours)					
SR (%)	$61.7^{15.8\downarrow}$	$66.7^{10.8\downarrow}$	77.5	$72.5^{5\downarrow}$	$70.8^{6.7\downarrow}$
Steps	849.1	541.6	499.6	314.2	262.2
Speed ratio	0.625	0.982	1.026	1.685	2.010

5.5 E5: Visual Closed-Loop Execution under Pace Commands

We evaluate four tasks at $s \in \{0.5, 1.0, 1.5\}$ (Table 5). The π_0 baseline (Black et al., 2024) and visual hierarchy each use 100 rollouts per evaluated task and rate from randomized initial states; π_0 is evaluated only at its nominal rate. The motion-prior reference and complete hierarchy use identical lower-policy weights. The former receives a latent encoded from the standard reference and manually supplied progress; the latter uses predictions of both from the $\pi_{0.5}$ visual frontend.

This comparison examines the execution cost of visual prediction. The motion prior achieves favorable success, while the visual hierarchy retains useful execution capability with lower SR on Handover and Hanging Mug and comparable performance on the other two tasks. These decreases are consistent with errors in the predicted motion latent and progress, motivating improved prediction heads and coupling with the motion prior. Table 5 measures task success under pace commands; realized timing is quantified for the motion prior in E4.

Table 5: Task success rates (%) at three commanded pace multipliers. "/" denotes unsupported rate control. Bold marks the best result per task and rate, including ties. Superscripts show percentage-point changes from π_0 at $1\times$.

Method	Handover Block			Hanging Mug			Grab Roller			Open Laptop		
	$0.5\times$	$1.0\times$	$1.5\times$	$0.5\times$	$1.0\times$	$1.5\times$	$0.5\times$	$1.0\times$	$1.5\times$	$0.5\times$	$1.0\times$	$1.5\times$
π_0	/	45	/	/	11	/	/	**96**	/	/	**85**	/
Motion prior (w/o $\pi_{0.5}$)	**65**	**$71^{26\uparrow}$**	**73**	**40**	**$43^{32\uparrow}$**	**46**	**95**	$91^{5\downarrow}$	**90**	**92**	$84^{1\downarrow}$	**86**
$\pi_{0.5}$ w/ prior	50	$55^{10\uparrow}$	57	21	$24^{13\uparrow}$	27	**95**	$93^{3\downarrow}$	**90**	**92**	**85**	**86**

6 Conclusion

We presented PCMP, a framework for learning pace-controllable motion priors for robotic manipulation. By representing demonstrations as reconstructable motion assets and separating motion content from progress schedules, PCMP learns a reusable motor backbone with complementary spatial and temporal augmentation. Our evaluation examines motion execution and pacing under different constant speed multipliers. The motion prior responds to pace commands, while adding visual prediction produces task-dependent changes in success. Future work will focus on improving the upper-level model to more fully realize the capabilities of the learned motion prior in visually grounded manipulation.

AI USAGE DISCLOSURE

We used generative AI tools to assist with drafting and refining the manuscript, including improving language, clarity, and organization. We also used these tools to search for and identify potentially relevant literature. We take responsibility for the final content of this work.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron 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 *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. *arXiv preprint arXiv:2506.07339*, 2025. URL https://arxiv.org/abs/2506.07339.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics Transformer for Real-World Control at Scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

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

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal Manipulation Interface: In-The-Wild Robot Teaching Without In-The-Wild Robots. *arXiv preprint arXiv:2402.10329*, 2024. URL https://arxiv.org/abs/2402.10329.

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

Yiguo Fan, Shuanghao Bai, Xinyang Tong, Pengxiang Ding, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, et al. Long-VLA: Unleashing long-horizon capability of vision language action model for robot manipulation. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2018–2037, 2025. URL https://proceedings.mlr.press/v305/fan25a.html.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3D: 3D Feature Field Transformers for Multi-Task Robotic Manipulation. *arXiv preprint arXiv:2306.17817*, 2023. URL https://arxiv.org/abs/2306.17817.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic View Transformer for 3D Object Manipulation. *arXiv preprint arXiv:2306.14896*, 2023. URL https://arxiv.org/abs/2306.14896.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, et al. RT-Trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023. URL https://arxiv.org/abs/2311.01977.

Lingxiao Guo, Zhengrong Xue, Zijing Xu, and Huazhe Xu. DemoSpeedup: Accelerating visuomotor policies via entropy-guided demonstration acceleration. *arXiv preprint arXiv:2506.05064*, 2025. URL https://arxiv.org/abs/2506.05064.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. *arXiv preprint arXiv:2307.05973*, 2023. URL https://arxiv.org/abs/2307.05973.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25 (2):328–373, 2013. doi: 10.1162/NECO_a_00393. URL https://www.pure.ed.ac.uk/ws/portalfiles/portal/7874487/NECO_a_00393.pdf.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *arXiv preprint arXiv:1909.12271*, 2019. URL https://arxiv.org/abs/1909.12271.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. *arXiv preprint arXiv:2205.09991*, 2022. URL https://arxiv.org/abs/2205.09991.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General Robot Manipulation with Multimodal Prompts. *arXiv preprint arXiv:2210.03094*, 2022. URL https://arxiv.org/abs/2210.03094.

Dong Jing, Jingchen Nie, Tianqi Zhang, Jiaqi Liu, Huaxiu Yao, Zhiwu Lu, and Mingyu Ding. TempoVLA: Learning speed-controllable vision-language-action policies. *arXiv preprint arXiv:2606.06491*, 2026a. URL https://arxiv.org/abs/2606.06491v2.

Dong Jing, Tianqi Zhang, Jiaqi Liu, Jinman Zhao, Zelong Sun, Li Erran Li, Zhiwu Lu, and Mingyu Ding. Learning action priors for cross-embodiment robot manipulation. *arXiv preprint arXiv:2606.26095*, 2026b. URL https://arxiv.org/abs/2606.26095.

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 P. Foster, Pannag R. Sanketi, Quan Vuong, et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Sunwoo Kim, Jeongjun Kim, and Joseph Lim. Time optimal execution of action chunk policies beyond demonstration speed. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fab80bb9d97e9b9ff5c19f91f72838c6-Abstract-Conference.html.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. URL https://arxiv.org/abs/1312.6114.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3bfee3bc6639c36e6e7b058db909f760-Abstract-Conference.html.

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/2209.07753.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. *arXiv preprint arXiv:2306.03310*, 2023. URL https://arxiv.org/abs/2306.03310.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning Latent Plans from Play. *arXiv preprint arXiv:1903.01973*, 2019. URL https://arxiv.org/abs/1903.01973.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martin-Martin. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. *arXiv preprint arXiv:2108.03298*, 2021. URL https://arxiv.org/abs/2108.03298.

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. *arXiv preprint arXiv:2310.17596*, 2023. URL https://arxiv.org/abs/2310.17596.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *arXiv preprint arXiv:2112.03227*, 2021. URL https://arxiv.org/abs/2112.03227.

Taewook Nam and Sung Ju Hwang. SpeedAug: Policy acceleration via tempo-enriched policy and RL fine-tuning. *arXiv preprint arXiv:2512.00062*, 2025. URL https://arxiv.org/abs/2512.00062.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *Proceedings of Robotics: Science and Systems*, 2024. URL https://octo-models.github.io/.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation*, 2024. URL https://robotic-transformer-x.github.io/.

Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL https://papers.nips.cc/paper/2013/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *arXiv preprint arXiv:2104.02180*, 2021. URL https://arxiv.org/abs/2104.02180.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating Reinforcement Learning with Learned Skill Priors. *arXiv preprint arXiv:2010.11944*, 2020. URL https://arxiv.org/abs/2010.11944.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. *arXiv preprint arXiv:2501.09747*, 2025. URL https://arxiv.org/abs/2501.09747.

Hung Pham and Quang-Cuong Pham. A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, 34(3):645–659, 2018. doi: 10.1109/TRO.2018.2819195. URL https://arxiv.org/abs/1707.07239.

Physical Intelligence, Kevin Black, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A Generalist Agent. *arXiv preprint arXiv:2205.06175*, 2022. URL https://arxiv.org/abs/2205.06175.

Yuki Saigusa, Ayumu Sasagawa, Sho Sakaino, and Toshiaki Tsuji. Imitation learning for variable speed motion generation over multiple actions. In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021. doi: 10.1109/IECON48115.2021.9589303. URL https://arxiv.org/abs/2103.06466.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior Transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022. URL https://arxiv.org/abs/2206.11251.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation. *arXiv preprint arXiv:2209.05451*, 2022. URL https://arxiv.org/abs/2209.05451.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 2017. URL https://arxiv.org/abs/1706.03762.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, et al. BridgeData V2: A Dataset for Robot Learning at Scale. *arXiv preprint arXiv:2308.12952*, 2023. URL https://arxiv.org/abs/2308.12952.

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. MimicPlay: Long-horizon imitation learning by watching human play. *Proceedings of the 7th Conference on Robot Learning*, 229:201–221, 2023. URL https://proceedings.mlr.press/v229/wang23a.html.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. DexVLA: Vision-Language Model with Plug-In Diffusion Expert for General Robot Control. *arXiv preprint arXiv:2502.05855*, 2025. URL https://arxiv.org/abs/2502.05855.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21469–21480, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Xiang_Structured_3D_Latents_for_Scalable_and_Versatile_3D_Generation_CVPR_2025_paper.html.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45d74e190008c7bff2845ffc8e3facd3-Abstract-Conference.html.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations. *arXiv preprint arXiv:2403.03954*, 2024. URL https://arxiv.org/abs/2403.03954.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid Loss for Language Image Pre-Training. *arXiv preprint arXiv:2303.15343*, 2023. URL https://arxiv.org/abs/2303.15343.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics*, 2023. URL https://arxiv.org/abs/2301.11445.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. *arXiv preprint arXiv:2412.04987*, 2024. URL https://arxiv.org/abs/2412.04987.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. *arXiv preprint arXiv:2304.13705*, 2023a. URL https://arxiv.org/abs/2304.13705.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Advances in Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=xmxgMij3LY.

Ruijie Zheng, Ching-An Cheng, Hal Daumé III, Furong Huang, and Andrey Kolobov. PRISE: LLM-Style Sequence Compression for Learning Temporal Action Abstractions in Control. *arXiv preprint arXiv:2402.10450*, 2024. URL https://arxiv.org/abs/2402.10450.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the Continuity of Rotation Representations in Neural Networks. *arXiv preprint arXiv:1812.07035*, 2018. URL https://arxiv.org/abs/1812.07035.

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 *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Implementation Details

The experimental workstation is equipped with an NVIDIA GeForce RTX 4070 SUPER GPU with 12 GB of VRAM. This appendix describes the motion representation, policy architecture, training, and execution interface. Implementation defaults are documented below. The motion-prior checkpoint is selected according to its flow-matching training loss.

A.1 State, Action, and Policy Interface

A demonstration e contains a task instruction ℓ_e, synchronized multi-view images $I_{e,i}$, robot state $x_{e,i}$, and semantic phase label $m_{e,i}$ at physical time $t_i = i\Delta t$. We use $\Delta t = 0.06$ seconds. For the dual-arm platform, each arm contributes Cartesian position $r_i^b \in \mathbb{R}^3$, a continuous 6D rotation representation, and normalized gripper opening $g_i^b \in [0,1]$, giving $x_i \in \mathbb{R}^{20}$. The 14D action is reconstructed from adjacent states,

$$a_i = \mathrm{Diff}(x_i, x_{i+1}) = \mathrm{concat}_{b\in\{L,R\}} \left[r_{i+1}^b - r_i^b,\ \mathrm{Log}\big((R_i^b)^\top R_{i+1}^b\big)^\vee,\ g_{i+1}^b - g_i^b \right]. \tag{10}$$

Rotation increments are expressed in the current end-effector frame. This definition allows actions to be regenerated after trajectory transformation instead of treating them as independent labels.

At step i, the motion policy observes one state–progress pair $O_i = (x_i, p_i)$, a motion latent z_e, and a future progress command C_i. It predicts an H-step action chunk,

$$\widehat{A}_i = \pi_\theta(O_i, z_e, C_i), \qquad \widehat{A}_i \in \mathbb{R}^{H\times 14}, \tag{11}$$

with $H = 24$. The dataset retrieves $H+1$ states internally so that all H actions are valid; no artificial zero action is appended at the endpoint. We use t for physical time, p for task progress, and τ for generative flow time.

A.2 Motion Assets, Semantic Progress, and Autoencoding

For task c, let $\mathcal{D}_c$ be its training demonstration set. Each motion asset $\mathcal{X}_e = (x_{e,i})_{i=0}^{N_e-1}$ is accompanied by semantic phase labels $m_{e,i}$, with the state and action conventions in Appendix A.1. Semantic progress is constructed as follows.

Normalized frame index is not comparable across demonstrations when operators pause or execute semantic stages at different rates. We therefore define progress from task-level semantic anchors. For task c with M_c ordered phases, let $n_{e,m}$ denote the number of control transitions assigned to phase m in episode e. The mean phase durations and the corresponding task-shared anchors are

$$\bar{n}_{c,m} = \frac{1}{|\mathcal{D}_c|} \sum_{e\in\mathcal{D}_c} n_{e,m}, \qquad \alpha_{c,m} = \frac{\sum_{j=1}^{m} \bar{n}_{c,j}}{\sum_{j=1}^{M_c} \bar{n}_{c,j}}, \tag{12}$$

where $\alpha_{c,0} = 0$ and $\alpha_{c,M_c} = 1$. Thus, the same anchor denotes the same semantic event across all demonstrations of a task. Let $F_{e,m}$ be the number of sampled states in phase m of episode e. Within that phase, progress advances uniformly with the local index $j \in \{0, \dots, F_{e,m} - 1\}$,

$$p_{e,m,j} = \alpha_{c,m-1} + (\alpha_{c,m} - \alpha_{c,m-1}) \begin{cases} j/F_{e,m}, & m < M_c, \\ j/(F_{e,m}-1), & m = M_c,\ F_{e,m} > 1, \\ 1, & m = M_c,\ F_{e,m} = 1. \end{cases} \tag{13}$$

The non-final interval excludes its right endpoint, which belongs to the next phase, while the final phase includes $p = 1$. Progress therefore encodes task advancement rather than accumulated Cartesian distance or frame count. Its nominal task rate is $\bar{v}_c^p = 1/(\Delta t \sum_m \bar{n}_{c,m})$, but the execution rate may be changed without changing the progress coordinate itself.

The progress-indexed interpolant for a given processing path is

$$Y_e(p) = \mathrm{Interp}\{(p_{e,i}, x_{e,i})\}_{i=0}^{N_e-1}. \tag{14}$$

Here Y_e denotes the interpolant for the relevant processing path: Y_e^{rec} for reconstruction and Y_e^{aug} for augmentation. At repeated progress values, the former retains the last corresponding state and

the latter retains the first, as detailed below. For reconstruction, states are resampled at K uniformly spaced progress values, using the same grid for both arms. The original control-rate sequences are retained for action supervision. Translation and gripper channels use cubic interpolation; rotations use an $\mathrm{SO}(3)$ spline, with spherical linear interpolation when too few samples are available. Orientations are then converted back to their 6D representation. Per-channel min–max normalization maps states to $[-1, 1]$. Feature differences use centered differences internally and one-sided differences at the endpoints. Normalized states are quantized to form the sparse coordinates retained by both encoder and decoder.

For each aligned state, the input feature is $F_{e,k} = [\bar{x}_{e,k}, \delta_k \bar{x}_e] \in \mathbb{R}^{2d_x}$, where $\bar{x}_{e,k}$ is the normalized state and δ_k is a local difference along the progress grid, not a physical-time velocity. Following TRELLIS's feature–coordinate structure, the sparse input is $\mathcal{A}_e = \{(F_{e,k}, s_{e,k})\}_{k=0}^{K-1}$, where $s_{e,k} \in \{0, \dots, Q-1\}^{20}$ is the quantized normalized state and Q is the quantization resolution. Thus, $d_x = 20$ yields 40-dimensional features, and $S_e = (s_{e,k})_{k=0}^{K-1}$ supplies robot-state coordinates. The decoder retains these coordinates while reconstructing the features; the lower-level motion policy receives only the exported motion latents.

The default multi-task trajectory VAE configuration uses $K = 128$, $d_x = 20$, $d_z = 8$, and eight full-attention blocks of width 256 with eight heads in both encoder and decoder, with MLP ratio four. Training uses SmoothL1 reconstruction and $\beta = 10^{-3}$, with the KL term averaged over tokens and channels. Posterior sampling is used during training; motion latent export uses the posterior mean. These are implementation settings, not requirements of the formulation. Reconstruction is conditioned on retained coordinates and therefore does not, by itself, establish that the exported motion latents alone determine a physically feasible trajectory. Independent lower-policy tests use the exported features without those coordinates.

Progress Calibration and Degenerate Stages The phase-uniform progress convention is defined above. True dwell duration cannot in general be recovered from a single-valued path $Y(p)$.

For repeated progress values, the reconstruction resampler retains the last corresponding state, whereas the augmentation interpolator retains the first. These conventions are applied consistently within their respective processing paths. They coincide when repeated states are identical; when the poses differ at a stage boundary, the two paths use their respective endpoint-selection rules.

Stage anchors, metric scales, feature and motion-latent normalization, and rate bounds are computed once using training data only and then fixed. The same calibration and processing rules are reused for unseen trajectories without re-estimating these quantities from each evaluation trajectory. Unless explicitly labeled as training-set evaluation, all motion-prior tests use new trajectories not used to train the motion prior.

A.3 Motion-Prior Architecture and Training

At step i, the motion policy conditions on a single state–progress observation $O_i = (x_i, p_i)$, the motion latent z_e, and the future progress command C_i to generate an H-step action chunk.

The lower-level model must answer a conditional control question: given the current state on a reference motion, what action sequence follows if the reference is traversed under a requested future rate? A single current progress or scalar speed is insufficient because it cannot describe an upcoming acceleration or deceleration. We therefore condition the policy on a short future schedule. Let p_i be the progress of the latest observation and $H_r = H - 1$. We form

$$P_i^{\mathrm{ref}} = (p_{i+1}^{\mathrm{ref}}, \dots, p_{i+H_r}^{\mathrm{ref}}), \tag{15}$$

$$\Delta P_i^{\mathrm{ref}} = (p_{i+1}^{\mathrm{ref}} - p_i, \dots, p_{i+H_r}^{\mathrm{ref}} - p_{i+H_r-1}^{\mathrm{ref}}), \tag{16}$$

$$C_i = [P_i^{\mathrm{ref}}, \Delta P_i^{\mathrm{ref}}]. \tag{17}$$

With $H = 24$ and one observation, the command contains 23 absolute future progress values and 23 increments, giving a 46D raw input. Absolute values identify the semantic region to be reached; increments specify the local execution rate. A positive multiplier schedule s_h is converted to a

task-calibrated bounded rate and integrated as

$$v_h^{\mathrm{cmd}} = \mathrm{clip}(s_h v^p_{\mathrm{base}}, v^p_{\mathrm{min},c}, v^p_{\mathrm{max},c}), \qquad p^{\mathrm{ref}}_{i+h} = \min\left(1, p_i + \Delta t \sum_{j=1}^{h} v_j^{\mathrm{cmd}}\right). \tag{18}$$

The same z_e can consequently be paired with different C_i: trajectory identity is preserved while timing is changed explicitly.

State-dependent retrieval from the motion latent. State and current progress are first embedded by separate two-layer MLPs. Our primary *cross-attention-plus* adapter treats z_e as 128 eight-dimensional tokens rather than flattening it prematurely. For the current observation, the concatenated state and progress embedding forms a query; LayerNorm-normalized trajectory tokens provide keys and values. A four-head cross-attention block followed by a feed-forward block returns a 256D trajectory readout for that frame. Hence, two observations can retrieve different regions of the same whole-episode motion, and a similar robot pose can induce different actions when it occurs in different semantic stages. The normalized future command C_i is independently mapped to a 64D embedding. The per-frame state, progress, and trajectory readouts together with the future-command embedding form the global condition of a temporal U-Net, injected at each resolution through FiLM.

Local Consistency Flow-Matching Objective The endpoint map and complete objective are defined in Eqs. (6) and (7). The conditioning $\kappa_i = (O_i, z_e, C_i)$ is held fixed across both flow-time evaluations.

For Eq. (7), sample $\tau \sim \mathcal{U}(\epsilon_\tau, 1)$ and $\tau' = \min(\tau + \delta_\tau, 1)$. Divide $[0, 1]$ into S equal segments and let η be the upper endpoint of the segment selected by τ. Let b_τ be the configured boundary threshold and $A_\eta = (1-\eta)A^0 + \eta A^\star$. The local implementation uses

$$\overline{F}^{\eta}(A^{\tau'}, \tau'; \kappa_i) = \begin{cases} F^{\eta}_\theta(A^{\tau'}, \tau'; \kappa_i), & \tau' < b_\tau, \\ A_\eta, & \tau' \geq b_\tau, \end{cases} \qquad \chi = \mathbf{1}[\tau < b_\tau]\,\mathbf{1}[\eta - \tau > 1.01\delta_\tau]. \tag{19}$$

When $b_\tau = 0$, the endpoint target is always A_η and the vector-field consistency term is disabled. FlowPolicy (Zhang et al., 2024) formulates its target branch using an EMA network. In our implementation, both predicted branches use the current network, without stop-gradient or a separate EMA teacher in this loss, even if an EMA copy is maintained elsewhere for evaluation.

The default temporal loss weights are $2.0, 1.5, 1.0, 1.0$ over four five-action blocks of the 20-action execution slice. Remaining action slots have weight one. The weighted mean is divided by the sum of temporal weights and averaged over batch and action dimensions. Flow training and inference use the configuration associated with the selected motion-prior checkpoint, including S, b_τ, δ_τ, λ_v, and the number of flow-integration steps.

A.4 Spatial and Progress-Schedule Augmentation

Semantic-anchor augmentation applies smooth, task-specific spatial offsets to selected anchors of a complete motion asset. In simulation, associated objects or goals are relocated consistently. Actions are regenerated from adjacent transformed states, and the frozen TRELLIS encoder produces a new motion latent for the transformed asset. Training can periodically refresh a configured subset from original demonstrations; validation uses fixed transformed assets and frozen normalization statistics. The spatial perturbation protocol is described in Section 5.1.

Progress-schedule augmentation samples the demonstrated schedule, a constant rate multiplier, or a variable-speed schedule with a linear rate transition. States are resampled along the progress-indexed asset $Y_e^{\mathrm{aug}}(p)$, and both action targets and future progress commands are regenerated. The motion latent is retained because the underlying asset is unchanged. The sampling mixture, constant multipliers, and rate bounds are specified in Appendix A.4.

Exact Window Indexing and Rate Augmentation The current motion network uses $H_w = 24$ action slots. A training item loads $H_w + 1 = 25$ states to compute every adjacent action, including the last one. With n_o observation slots, the current state has local index $j_c = n_o - 1$. The future

reference contains state progress at local indices $n_o, \ldots, H_w - 1$, together with the differences starting from j_c. Thus

$$H_r = H_w - n_o, \qquad \dim[P^{\text{ref}}, \Delta P^{\text{ref}}] = 2(H_w - n_o). \tag{20}$$

Our single-observation formulation uses $n_o = 1$, so $j_c = 0$ and the reference has 46 dimensions when $H_w = 24$. The final auxiliary action depends on state x_{24}, which is not represented in this reference vector. The returned 20-action execution slice is $A[j_c : j_c + 20]$ and is covered by the supplied reference. Generating 20 actions per query does not imply querying the model at every 0.06-second action interval: executing the whole chunk would delay the next query by approximately 1.2 seconds, excluding computation.

The sampler requests 48 source states to accommodate faster progress-space traversals. This length does not guarantee coverage for every nonuniform source rate. References are clipped to the available progress range rather than extrapolating poses, which can create a constant-state suffix before actual task completion. This clipping should be measured separately from a genuine terminal stop.

The default augmentation mixture is $(\pi_{\text{orig}}, \pi_{\text{const}}, \pi_{\text{ramp}}) = (0.50, 0.20, 0.30)$. Training includes both constant-rate and variable-speed augmentation. The constant-rate branch uses multipliers $\{0.5, 0.75, 1.0, 1.5, 2.0\}$, while variable-speed schedules change the requested rate over the course of a motion. Positive source rates are bounded by task-specific 1st and 99th percentiles. The median positive rate is used only when the source estimate is invalid or nearly zero. Head- or tail-padded windows bypass rate augmentation.

A.5 Visual Upper Model and Autoregressive Training

The upper model reuses the SigLIP image encoder and Gemma language backbone of $\pi_{0.5}$ without constructing its action expert. Each step contains three RGB views, task language, and robot state encoded into the language prompt. Two learned readout queries are appended to the prefix. For the motion latent branch, K learned slot queries cross-attend to the fused image and language/state tokens outside the backbone. Each slot predicts d_z standardized latent channels. The progress readout is passed to a separate autoregressive head. History enters this head only, not the VLM prefix or the motion latent branch.

The two prediction branches are defined in Eq. (8). The frozen motion encoder supplies latent targets, while semantic alignment supplies progress targets. The training objective is

$$\mathcal{L}_{\text{upper}} = \lambda_z \mathcal{L}_z + \lambda_p \mathcal{L}_p + \lambda_c \mathcal{L}_{\text{cons}} + \lambda_o \mathcal{L}_{\text{order}}, \tag{21}$$

The default upper-model configuration uses the token decoder and autoregressive progress head. The motion latent decoder uses $K = 128$, $d_z = 8$, width 256, two cross-attention/feed-forward blocks, eight attention heads, and feed-forward width 1024. The backbone feature width is 2048. The progress head maps the current readout to 256 features and the previous scalar prediction to 32 features, concatenates them, and applies a $288 \to 256 \to 1$ MLP followed by sigmoid. It predicts absolute progress in $[0, 1]$, not a progress increment.

Let $\mathcal{V}$ contain valid sequence steps and $\mathcal{E}$ contain adjacent valid step pairs from the same episode. With ℓ_{H} denoting elementwise-averaged SmoothL1, the terms in Eq. (21) are

$$\mathcal{L}_z = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell_{\text{H}}(\widehat{\bar{z}}_i, \bar{z}_e), \tag{22}$$

$$\mathcal{L}_p = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell_{\text{H}}(\widehat{p}_i, p_i), \tag{23}$$

$$\mathcal{L}_{\text{cons}} = \frac{1}{|\mathcal{E}|} \sum_{(i-1,i) \in \mathcal{E}} \ell_{\text{H}}(\widehat{\bar{z}}_{i-1}, \widehat{\bar{z}}_i), \tag{24}$$

$$\mathcal{L}_{\text{order}} = \frac{1}{|\mathcal{E}|} \sum_{(i-1,i) \in \mathcal{E}} \max(0, \widehat{p}_{i-1} - \widehat{p}_i + \mu). \tag{25}$$

Pairwise terms are zero when no valid pair is available. The default training configuration uses SmoothL1 transition 0.1, $\mu = 0$, and $(\lambda_z, \lambda_p, \lambda_c, \lambda_o) = (1, 1, 0.05, 0.1)$. The order penalty allows

unchanged progress and does not impose hard monotonicity. Latent targets are standardized using motion-latent normalization statistics computed from training data and then fixed, with one motion latent per episode; predictions are unstandardized before being passed to the lower policy.

Training sequences contain four consecutive frames by default and never cross episode boundaries. Each image is encoded independently; only the progress head is evaluated sequentially. At the first step, missing history is represented by -1. Later steps receive the model's own previous prediction, detached from the preceding computation graph. No target progress is fed back. Padded steps are excluded from all losses and adjacent pairs. During training, history dropout is 0.1 and Gaussian perturbations use standard deviation $\min(0.25\,\mathrm{EMA}(\mathrm{MAE}_{\mathrm{train}}), 0.02)$, with EMA decay 0.99. Valid perturbed predictions are clipped to $[0, 1]$; missing history is kept distinct. These operations are disabled during evaluation. Evaluation carries predictions through complete episodes and resets history at episode boundaries.

The default upper-model configuration freezes the backbone and optimizes the queries and prediction heads. The motion prior is trained separately.

A.6 Closed-Loop Execution Interface

At each observation, the visual model predicts a motion latent and current progress as in Eq. (8). The scheduler constructs C_i from the predicted progress and the requested pace, and the lower policy generates an action chunk according to Eq. (9). After executing a prefix, the controller acquires the next observation and repeats this process. The previous progress prediction is carried across calls and reset at an episode boundary. The 24-step model horizon and the 20-action returned slice described in Appendix A.4 are distinct quantities.

To execute Eq. (10), translation is added in the state frame, orientation is composed as $R_{\mathrm{next}} = R_{\mathrm{current}}\,\mathrm{Exp}([\Delta\omega]_\times)$, and gripper opening is incremented and bounded. Here $\Delta\omega$ denotes the rotational increment in the action. An IK or tracking controller must realize these targets with a declared timing policy. Kinematic interpolation does not supply collision, acceleration, or contact-feasibility guarantees. Those properties require controller checks and physical or simulated rollouts.

B Experimental Setup and Protocols

This appendix specifies the data, metrics, and protocols for the five main-text experiments. Detailed replay, reconstruction, and pace-control results are collected in Appendix C. Hardware and implementation details are given in Appendix A.

B.1 Tasks, Data Splits, and Evaluation Metrics

The evaluation covers twelve RoboTwin 2.0 tasks, grouped into four easy, four medium, and four hard tasks as listed in Table 6. Demonstrations are split by episode. Unless explicitly labeled as training-set results, motion-prior evaluations use new trajectories not used in motion-prior training. The motion-prior checkpoint is selected by the lowest flow-matching training loss. Stage anchors, normalization statistics, and rate bounds are computed from the training split only and reused unchanged for unseen trajectories (Appendix A.2).

Task success is determined by the environment's task predicate. Tracking metrics report position error in millimeters, rotation error in degrees, and normalized gripper error. Reconstruction and closed-loop tracking use different alignments, specified below. SR includes all attempted rollouts. For pace evaluation, slower commands receive proportionally longer rollout budgets. Missing measurements use $--$; the zero-success timing convention is specified in Appendix B.5.

B.2 Reference Replay Protocol

The semantic-augmentation test uses 50 expert episodes per task with two variants each (1,200 rollouts total). Semantic anchors and scene entities are sampled from task-specific intervals widened fivefold. References are replayed with the strict CuRobo controller at 16.67 Hz from the source scene; success requires the task predicate after the final target. Progress retiming uses 50 source episodes per task and speed, a 0.06 s control interval, and fixed inverse-kinematics interpolation

with 15 simulation steps per target. Temporal resampling preserves reference geometry. The tested replay multipliers are $0.5\times$ and $1.5\times$, relative to the unified reference rather than the original expert duration.

B.3 Motion Assets Reconstruction Protocol

A single trajectory encoder–decoder is trained jointly on all twelve tasks, using 2,160 training episodes in total. Each trajectory is resampled to 128 progress-uniform states, encoded, decoded, and inverse-resampled at the source progress values before physical-error evaluation. The results are then grouped by task; these groups do not correspond to separately trained encoders. Evaluation selects 50 training and 20 validation episodes per task. Place Can Basket and Put Object Cabinet each exclude one validation episode outside the fixed normalizer's XYZ bounds, giving 238/240 evaluated validation episodes. Reported means average the displayed task-level values with equal task weights; the mean validation position error is $14.45/12 = 1.20417$ mm. Table 7 contains the complete results underlying Table 2.

B.4 Motion-Latent Ablation Protocol

This experiment evaluates motion-latent conditioning on the Handover task, independently of visual prediction, with 10 rollouts per condition. None is trained with the motion-latent conditioning branch removed. Zero and Random retain the branch but replace its input with zeros and random noise, respectively. Correct supplies the matching motion latent. Tracking errors are measured against the commanded reference. Table 3 reports SR, position error, rotation error, and gripper error for these four conditions. The scope of this ablation is the Handover task, not a twelve-task mean.

B.5 Execution-Pace Control Protocol

The motion prior is trained with both constant-rate augmentation, using $s \in \{0.5, 0.75, 1.0, 1.5, 2.0\}$, and variable-speed schedules. The reported constant-rate experiment evaluates these five multipliers on unseen trajectories, with 10 rollouts per task, policy condition, and rate. Both models receive absolute future progress values and their increments. The *w/o pace augmentation* model is trained without constant-rate or variable-speed progress-schedule augmentation. It retains the same conditioning interface and accepts pace commands during evaluation. Table 4 reports equal-task means, while Tables 8–10 report individual tasks. The tested rates are included in the full policy's training augmentation; they are not held-out rates for that model. No cross-command average is reported.

Reference length and ratio aggregation. The data reference is a uniformly resampled trajectory with 511 control transitions. This length is shared across tasks and does not denote the original demonstration's frame count. It is distinct from the $K = 128$ progress grid used by the trajectory encoder. For task c and rate s, let $\mathcal{S}_{c,s}$ denote the successful rollouts. When this set is nonempty, the reported execution length is the mean completion step count $\overline{N}_{c,s}$, and the speed ratio is computed after that averaging:

$$\overline{N}_{c,s} = \frac{1}{|\mathcal{S}_{c,s}|} \sum_{j \in \mathcal{S}_{c,s}} N_j, \qquad R_{c,s} = \frac{511}{\overline{N}_{c,s}}. \tag{26}$$

The ratio is relative to the uniformly resampled reference. It is not an average of per-rollout ratios. SR is computed over all attempted rollouts.

Zero-success convention. If no rollout succeeds, both Steps and Speed ratio are reported as zero. These zeros indicate the absence of a successful completion, not a measured zero-duration execution; the ratio formula is not applied to a zero denominator. Main-text means give each of the twelve tasks equal weight, including these zero entries. Thus, aggregate timing statistics should be read alongside SR, since failures affect these means as well as execution pace. Missing or inapplicable measurements retain $--$.

Table 6: Reference replay under semantic augmentation ($N = 100$ per task) and progress retiming ($N = 50$ per task and speed). Rate columns report SR (%); speed multipliers are relative to the unified reference. Steps includes the initial state. The augmentation and retiming protocols use different controllers; see Appendix B.2.

		Semantic augmentation		Progress retiming	
Task	Difficulty	Steps	SR (%)	$0.5\times$	$1.5\times$
Adjust Bottle	Easy	127.5	94	100	98
Open Laptop	Easy	192.1	100	100	100
Click Bell	Easy	68.8	100	100	100
Turn Switch	Easy	90.0	92	100	96
Grab Roller	Medium	85.5	99	100	100
Place Can Basket	Medium	226.3	94	100	96
Lift Pot	Medium	101.5	93	100	100
Pick Diverse Bottles	Medium	108.6	100	100	100
Hanging Mug	Hard	299.0	91	98	96
Handover Block	Hard	250.6	93	98	98
Stack Bowls Three	Hard	416.6	90	92	94
Put Object Cabinet	Hard	236.4	98	100	98

B.6 Visual Closed-Loop Execution under Pace Commands

The comparison covers Handover Block, Hanging Mug, Grab Roller, and Open Laptop at commanded multipliers $s \in \{0.5, 1.0, 1.5\}$. The baseline and visual hierarchy each use 100 rollouts per evaluated task and rate condition, starting from randomized initial states. This protocol differs from the 10-rollout evaluations in E3 and E4; their success rates should not be treated as repeated measurements on an identical evaluation set. Table 5 compares three settings: the π_0 baseline at its nominal rate, the motion prior supplied with a motion latent computed by the frozen encoder from the standard reference and manually specified current progress, and the complete hierarchy using the $\pi_{0.5}$ frontend to predict motion latent and current progress. The baseline does not accept explicit pace commands; its other rate entries are therefore inapplicable rather than zero-success evaluations.

The prior-only setting and complete hierarchy use identical lower-policy weights. The former evaluates the motion model with privileged conditioning. The comparison assesses performance when supplied motion conditions are replaced by predictions of the motion latent and progress. The observed SR reductions on Handover and Hanging Mug are consistent with this added prediction burden; performance is largely retained on Grab Roller and Open Laptop. The experiment does not separately quantify the contribution of each prediction head. It reports task success under pace commands rather than realized execution rates.

C Additional Quantitative Results

This appendix provides the complete per-task results for reference replay (E1), motion asset reconstruction (E2), and constant-rate pace control (E4). Video timelines complement the pace-control statistics in Appendix C.3.1. Protocols for all five experiments are given in Appendix B. The Handover latent ablation and the four-task visual comparison are reported in Tables 3 and 5, respectively.

C.1 Full Reference Replay Results

Table 6 reports all twelve tasks corresponding to Table 1, including task difficulty and reference step counts. The main-text means use all twelve tasks, not only the four selected rows. Replay protocols are given in Appendix B.2.

C.2 Full Motion Assets Reconstruction Results

Table 7 reports per-task training and validation errors from the single encoder–decoder trained jointly on all twelve tasks. Table 2 displays only validation errors for the same four tasks as Table 1,

with the mean computed over all twelve. Evaluation sample counts and exclusions are described in Appendix B.3.

Figure 5 compares reference and reconstructed trajectories, complementing the quantitative errors below.

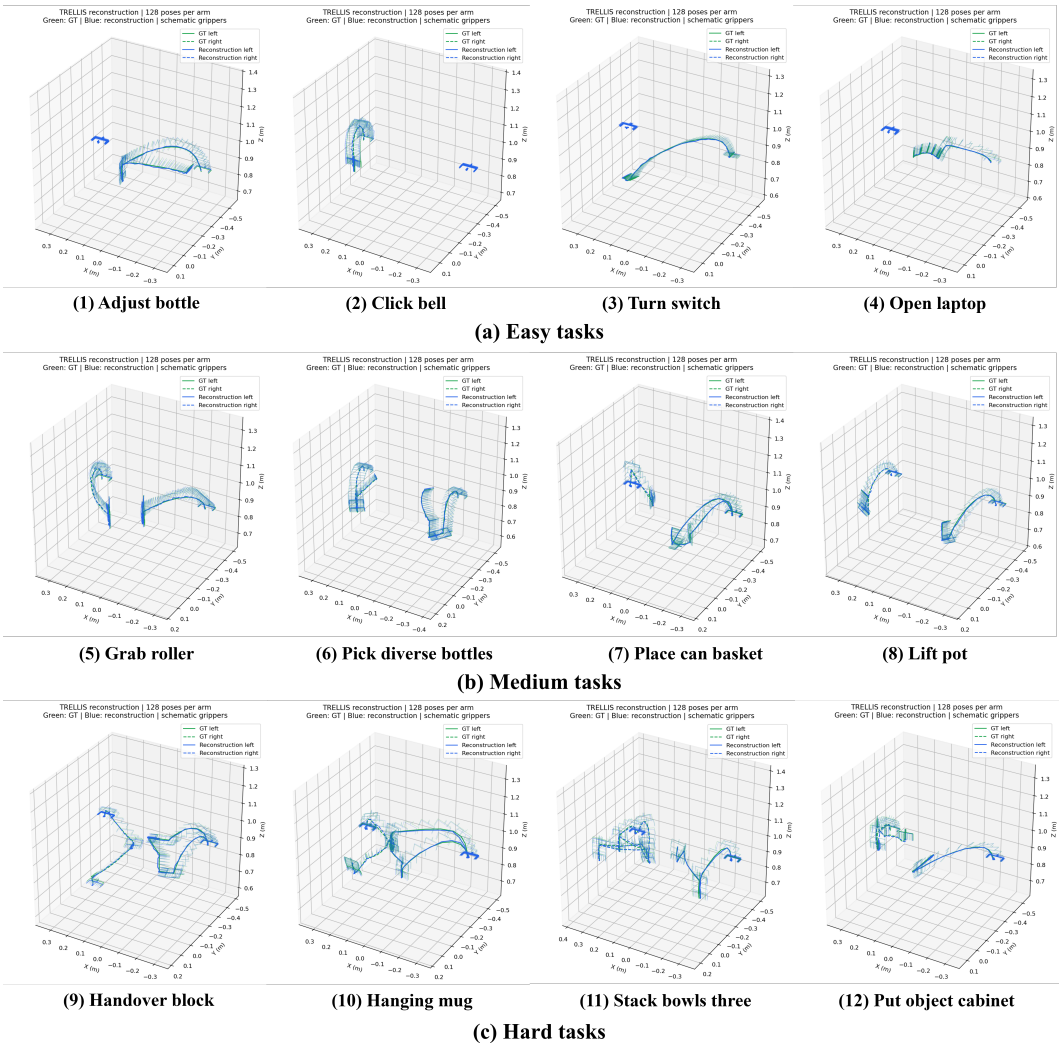


Figure 5: Qualitative motion-asset reconstruction. Green and blue show reference and reconstructed trajectories, respectively, with 128 poses per arm. Solid and dashed lines distinguish the left and right arms; gripper glyphs illustrate end-effector poses.

C.3 Execution-Pace Control Results

Tables 8–10 report the twelve tasks in three difficulty groups. Each task lists the uniformly resampled data reference, *w/o pace augmentation*, and the full policy at five commanded multipliers $s \in \{0.5, 0.75, 1.0, 1.5, 2.0\}$. Table 4 summarizes these results using equal task weights. SR superscripts indicate changes relative to the same policy's $1.0\times$ result.

The reference length is 511 control transitions for every task. Speed ratio is this reference length divided by the mean successful completion steps within each task and condition. If SR is zero, Steps and Speed ratio are both reported as zero and included in the equal-task means. These zeros indicate no successful completion. Appendix B.5 defines aggregation and the zero-success convention. All

Table 7: Per-task reconstruction error: position MAE (mm), rotation geodesic error (degrees), and normalized gripper MAE. Each task uses 50 training and 20 validation episodes, except Place Can Basket and Put Object Cabinet (19 validation episodes each after excluding one out-of-bounds reference). The mean weights all twelve tasks equally. Lower is better.

		Training episodes			Validation episodes		
Task	Diff.	Pos. (mm)	Rot. (deg)	Grip	Pos. (mm)	Rot. (deg)	Grip
Adjust Bottle	Easy	0.64	0.14	0.0019	1.18	0.43	0.0027
Open Laptop	Easy	0.48	0.16	0.0022	1.30	0.57	0.0039
Click Bell	Easy	0.43	0.11	0.0017	0.54	0.13	0.0019
Turn Switch	Easy	0.59	0.13	0.0018	1.22	0.32	0.0026
Grab Roller	Medium	0.47	0.13	0.0018	1.33	0.61	0.0030
Place Can Basket	Medium	0.65	0.14	0.0018	1.02	0.34	0.0023
Lift Pot	Medium	0.42	0.14	0.0016	0.79	0.29	0.0018
Pick Diverse Bottles	Medium	0.55	0.07	0.0020	2.18	0.51	0.0036
Hanging Mug	Hard	0.67	0.13	0.0020	1.52	0.48	0.0030
Handover Block	Hard	0.59	0.10	0.0017	0.73	0.16	0.0019
Stack Bowls Three	Hard	0.50	0.12	0.0025	0.98	0.34	0.0031
Put Object Cabinet	Hard	0.48	0.14	0.0021	1.66	1.13	0.0041
Mean (12 tasks)		0.54	0.13	0.0019	1.20	0.44	0.0028

motion-prior results here use unseen trajectories; the motion-prior checkpoint is selected by flow-matching training loss.

Table 8: Per-task speed-control ablation: easy tasks. SR is in %. *w/o pace aug.* removes constant- and variable-rate augmentation while retaining future-progress conditioning. Data reference is the unified resampled reference length of 511 steps; Speed ratio is this length divided by mean policy steps. For zero SR, Steps and Speed ratio are reported as 0 by convention, not as measured zero duration; these zeros enter the equal-task means. −− denotes missing or inapplicable values. SR superscripts show percentage-point changes relative to 1.0×: blue ↑ indicates an increase and red ↓ a decrease. Equal or missing values are unmarked.

		Commanded speed s				
Condition	Metric	0.5×	0.75×	1.0×	1.5×	2.0×
Adjust Bottle						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$70^{30\downarrow}$	$90^{10\downarrow}$	100	$80^{20\downarrow}$	$40^{60\downarrow}$
	Steps	892.0	546.2	500.0	355.3	382.5
	Speed ratio	0.573	0.936	1.022	1.438	1.336
Policy (ours)	SR (%)	$90^{10\downarrow}$	100	100	100	100
	Steps	796.4	515.3	492.6	306.3	254.9
	Speed ratio	0.642	0.992	1.037	1.668	2.005
Open Laptop						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$80^{20\downarrow}$	$90^{10\downarrow}$	100	$80^{20\downarrow}$	$50^{50\downarrow}$
	Steps	768.0	469.3	516.9	305.3	327.6
	Speed ratio	0.665	1.089	0.989	1.674	1.560
Policy (ours)	SR (%)	100	100	100	100	100
	Steps	685.7	442.7	509.3	263.1	218.3
	Speed ratio	0.745	1.154	1.003	1.942	2.341

Continued on next page

Table 8 (continued)

Condition	Metric	Commanded speed s 0.5×	0.75×	1.0×	1.5×	2.0×
Click Bell						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$50^{40\downarrow}$	$80^{10\downarrow}$	90	$60^{30\downarrow}$	$30^{60\downarrow}$
	Steps	1123.6	699.5	524.2	397.5	423.7
	Speed ratio	0.455	0.731	0.975	1.286	1.206
Policy (ours)	SR (%)	$70^{20\downarrow}$	90	90	$100^{10\uparrow}$	$100^{10\uparrow}$
	Steps	1003.1	659.9	516.6	342.7	282.4
	Speed ratio	0.509	0.774	0.989	1.491	1.809
Turn Switch						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$30^{30\downarrow}$	$50^{10\downarrow}$	60	$40^{20\downarrow}$	$20^{40\downarrow}$
	Steps	552.7	328.2	514.7	214.3	245.0
	Speed ratio	0.925	1.557	0.993	2.385	2.086
Policy (ours)	SR (%)	$60^{10\downarrow}$	$60^{10\downarrow}$	70	70	$60^{10\downarrow}$
	Steps	493.3	309.7	507.1	184.7	163.3
	Speed ratio	1.036	1.650	1.008	2.767	3.129

Table 9: Per-task speed-control ablation: medium tasks. Metrics and superscript conventions follow Table 8.

Condition	Metric	Commanded speed s 0.5×	0.75×	1.0×	1.5×	2.0×
Grab Roller						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$30^{70\downarrow}$	$70^{30\downarrow}$	100	$60^{40\downarrow}$	$20^{80\downarrow}$
	Steps	1125.7	651.6	534.5	457.0	541
	Speed ratio	0.454	0.784	0.956	1.118	0.945
Policy (ours)	SR (%)	$40^{60\downarrow}$	$90^{10\downarrow}$	100	100	100
	Steps	954.0	592.4	526.6	351.5	292.5
	Speed ratio	0.536	0.863	0.970	1.454	1.747
Place Can Basket						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$40^{40\downarrow}$	$60^{20\downarrow}$	80	$50^{30\downarrow}$	$20^{60\downarrow}$
	Steps	1040.5	637.3	509.8	444.6	534
	Speed ratio	0.491	0.802	1.002	1.149	0.957
Policy (ours)	SR (%)	$60^{20\downarrow}$	$70^{10\downarrow}$	80	80	$70^{10\downarrow}$
	Steps	881.8	579.4	502.3	342.0	288.7
	Speed ratio	0.579	0.882	1.017	1.494	1.770
Lift Pot						
Data reference	Steps	511	511	511	511	511
w/o pace aug.	SR (%)	$50^{30\downarrow}$	$70^{10\downarrow}$	80	$50^{30\downarrow}$	$20^{60\downarrow}$
	Steps	901.6	539.6	507.3	370.6	418.5
	Speed ratio	0.567	0.947	1.007	1.379	1.221
Policy (ours)	SR (%)	80	80	80	$90^{10\uparrow}$	80
	Steps	764.1	490.5	499.8	285.0	226.3
	Speed ratio	0.669	1.042	1.022	1.793	2.258

Continued on next page

Table 9 (continued)

Condition	Metric	Commanded speed s 0.5×	0.75×	1.0×	1.5×	2.0×
Pick Diverse Bottles						
Data reference	Steps	511	511	511	511	511
	SR (%)	$10^{10\downarrow}$	$10^{10\downarrow}$	20	$10^{10\downarrow}$	$0^{20\downarrow}$
w/o pace aug.	Steps	1206.0	750	572.0	508	0
	Speed ratio	0.424	0.681	0.893	1.006	0
	SR (%)	20	$10^{10\downarrow}$	20	$30^{10\uparrow}$	$40^{20\uparrow}$
Policy (ours)	Steps	1022.0	682.0	563.5	391.0	320.5
	Speed ratio	0.500	0.749	0.907	1.307	1.594

Table 10: Per-task speed-control ablation: hard tasks. Metrics and superscript conventions follow Table 8.

Condition	Metric	Commanded speed s 0.5×	0.75×	1.0×	1.5×	2.0×
Hanging Mug						
Data reference	Steps	511	511	511	511	511
	SR (%)	$30^{60\downarrow}$	$70^{20\downarrow}$	90	$50^{40\downarrow}$	$10^{80\downarrow}$
w/o pace aug.	Steps	1112.7	647.3	502.1	494.0	612.0
	Speed ratio	0.459	0.789	1.018	1.034	0.835
	SR (%)	$60^{40\downarrow}$	$90^{10\downarrow}$	100	$90^{10\downarrow}$	100
Policy (ours)	Steps	897.2	567.8	494.7	340.7	278.2
	Speed ratio	0.570	0.900	1.033	1.500	1.837
Handover Block						
Data reference	Steps	511	511	511	511	511
	SR (%)	$10^{30\downarrow}$	$20^{20\downarrow}$	40	$20^{20\downarrow}$	$0^{40\downarrow}$
w/o pace aug.	Steps	1047	587.0	475.0	453	0
	Speed ratio	0.488	0.871	1.076	1.128	0
	SR (%)	$40^{40\downarrow}$	$30^{50\downarrow}$	80	$30^{50\downarrow}$	$30^{50\downarrow}$
Policy (ours)	Steps	844.0	515.0	468.0	312.7	260.0
	Speed ratio	0.605	0.992	1.092	1.634	1.965
Stack Bowls Three						
Data reference	Steps	511	511	511	511	511
	SR (%)	$30^{40\downarrow}$	$50^{20\downarrow}$	70	$40^{30\downarrow}$	$10^{60\downarrow}$
w/o pace aug.	Steps	1113.0	677.2	456.0	516.5	664
	Speed ratio	0.459	0.755	1.121	0.989	0.770
	SR (%)	70	$50^{20\downarrow}$	70	70	$60^{10\downarrow}$
Policy (ours)	Steps	897.7	594.0	449.3	356.3	301.7
	Speed ratio	0.569	0.860	1.137	1.434	1.694
Put Object Cabinet						
Data reference	Steps	511	511	511	511	511
	SR (%)	$10^{30\downarrow}$	$20^{20\downarrow}$	40	$10^{30\downarrow}$	$0^{40\downarrow}$
w/o pace aug.	Steps	1178	627.5	472.0	426	0
	Speed ratio	0.434	0.814	1.083	1.200	0
	SR (%)	$50^{10\uparrow}$	$30^{10\downarrow}$	40	$10^{30\downarrow}$	$10^{30\downarrow}$
Policy (ours)	Steps	950.4	550.3	465.0	294.0	259
	Speed ratio	0.538	0.929	1.099	1.738	1.973

C.3.1 VIDEO TIMELINES

Figures 6–9 show the twelve tasks at five labeled pace settings. Rows arrange sampled frames along elapsed video time; endpoint labels give the displayed durations. Higher labeled rates correspond to shorter timelines in these examples. The frames illustrate how object interactions unfold over time, complementing the success and step-count statistics in Tables 8–10.

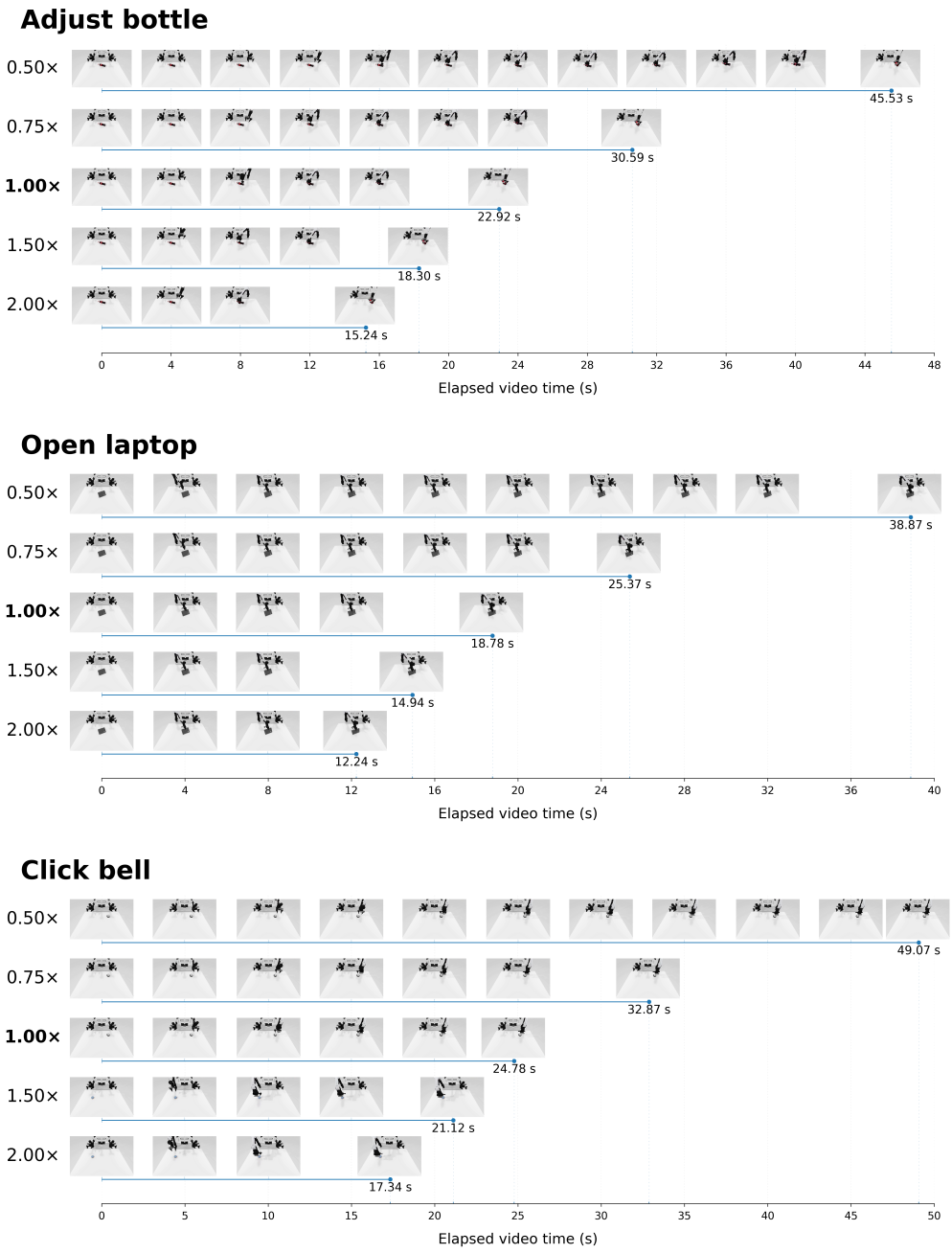


Figure 6: Video timelines for Adjust Bottle, Open Laptop, and Click Bell (top to bottom). Each panel compares five pace settings; time is in seconds.

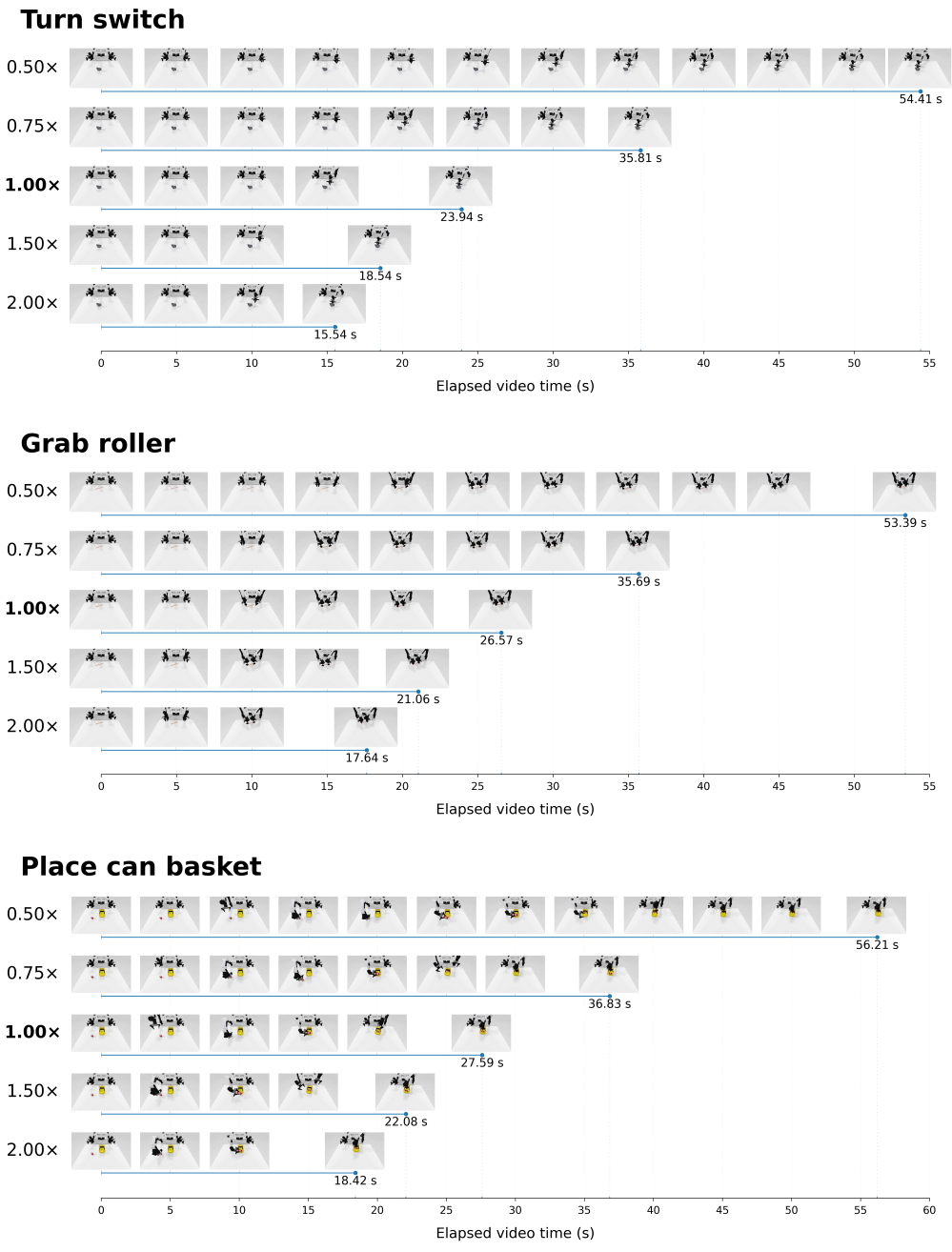


Figure 7: Video timelines for Turn Switch, Grab Roller, and Place Can Basket (top to bottom). Frames show the progression of the illustrated manipulation sequences at five pace settings.



Figure 8: Video timelines for Lift Pot, Pick Diverse Bottles, and Hanging Mug (top to bottom). The panels compare the temporal arrangement of bimanual lifting, bottle handling, and mug placement.

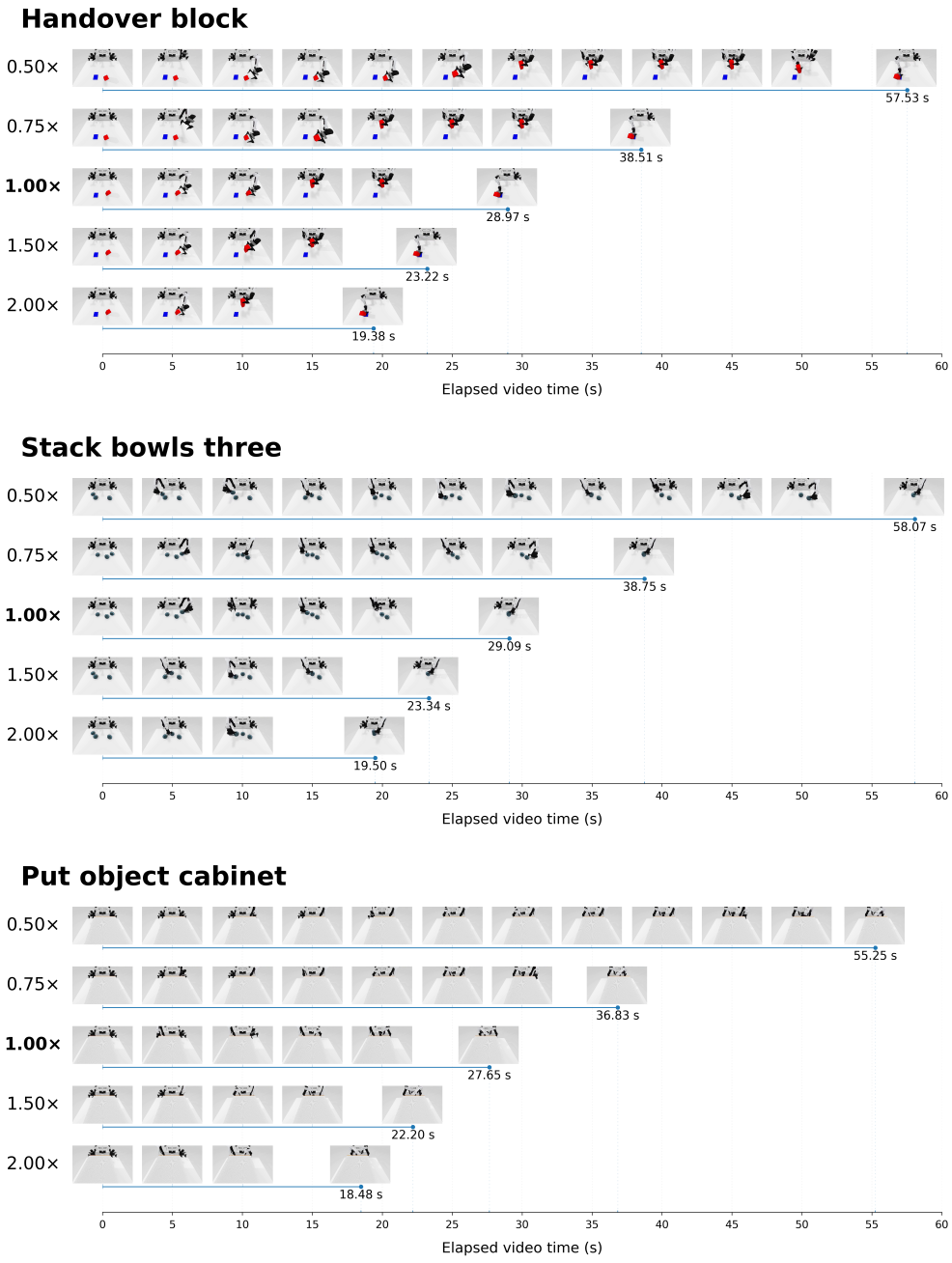


Figure 9: Video timelines for Handover Block, Stack Bowls Three, and Put Object Cabinet (top to bottom). The frames provide a temporal view of handover, stacking, and cabinet placement at five pace settings.