PEREGRINE: VISION-LANGUAGE-ACTION POLICIES FOR CONTACT-RICH DYNAMIC MANIPULATION THROUGH ACTION CORRECTION WITH FUTURE LATENT REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Action chunks amortize the online inference cost of vision-language-action (VLA) policies by generating multistep action sequences, but robot and object motion can render later actions in a chunk stale before execution. We propose Peregrine, a lightweight framework that conditions action correction on latent representations predicted for the target execution time. A Dynamic Tokenizer learns structured historical dynamics through robot-motion and region supervision, encoding observed changes in the robot and objects into compact dynamics tokens. A Future Predictor combines these tokens with the latest observation, the action timeline, and temporal conditions to predict latent representations at the target action's execution time. A Residual Corrector uses the execution-time prediction and the corresponding base action to generate a bounded residual correction, without requiring future-image generation or complete action replanning. The pretrained flow-based VLA remains frozen. All auxiliary modules are trained offline, and their weights remain fixed during deployment, while predictions and corrections are updated as new observations arrive.We evaluate Peregrine in dynamic simulations and real-robot manipulation, assessing task success rates and future latent prediction accuracy, and examining component contributions through ablation studies.

1 INTRODUCTION

Vision-language-action (VLA) models map visual observations and language instructions to robot actions, providing a practical interface for general-purpose manipulation [1, 2, 3, 4]. Chunked policies, including ACT and Diffusion Policy, predict multistep action sequences [5, 33]; flow-based VLAs such as π_0 and $\pi_{0.5}$ also use action chunks [22, 35]. Executing several commands between model calls amortizes inference costs, but later commands depend on a scene observed earlier and may become stale before execution.

In dynamic and contact-rich manipulation, robot tracking error, contact-induced displacement, and sustained target motion can change the robot–object state while an action chunk is being generated or executed. For example, a grasp planned from an earlier image may reach a moving object after it has left the intended contact region. We call the resulting discrepancy between an action's planned conditions and its execution state action-time mismatch. The same issue can arise during long-horizon tasks with nominally static objects when small execution errors accumulate. This motivates conditioning correction on a representation of the state the target action will encounter.

Real-time execution and correction methods improve responsiveness through asynchronous action generation and learned residual updates [6, 7, 8]. Anticipatory approaches already compensate for latency using rolled-forward robot states or predicted visual context [9, 29, 30, 31]. Peregrine studies how to use such anticipation for a bounded residual on an already generated command: a residual based solely on the latest measured state leaves motion during correction computation and transmission unaccounted for. We align the prediction horizon with the full observation-to-correction interval and condition the residual on the predicted state immediately before the corresponding action executes. Section 3 defines the timing and available-information constraints.

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.

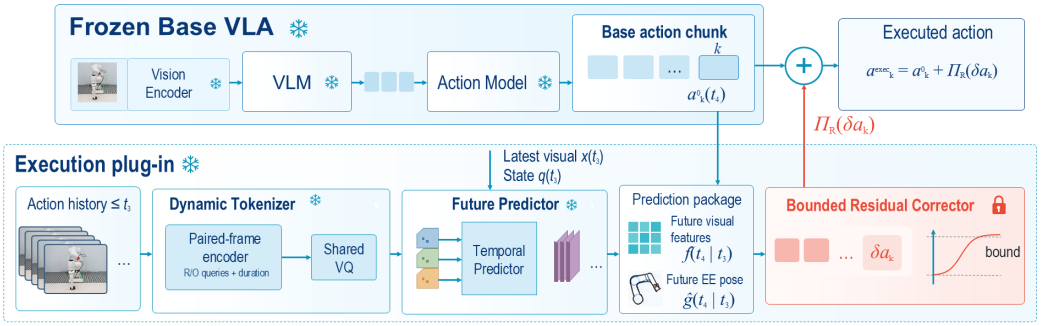

Figure 1: Overview of Peregrine. The frozen base VLA generates an action chunk. The Dynamic Tokenizer encodes historical robot and object dynamics, and the Future Predictor uses this history and the latest observation at t_3 to predict visual features and end-effector pose at execution time t_4. The Residual Corrector conditions on this prediction and the corresponding base action to produce a bounded correction. All model weights remain fixed online.

We propose Peregrine, a framework that adds execution-time prediction and bounded action correction to a frozen, task-adapted VLA (Figure 1). A Dynamic Tokenizer encodes completed observation intervals into compact robot and object dynamics tokens, with their roles guided by motion and region supervision. An independent Future Predictor combines historical dynamics, the latest observation, and available task and timed-action context to estimate visual features and robot pose at the target execution time. A Residual Corrector uses this prediction to adjust the corresponding pending base command. All auxiliary modules are trained offline and remain fixed during deployment; new observations update predictions and corrections without changing model weights.

Our contributions are threefold:

1. **Execution-time prediction and correction.** We formulate action-time mismatch and couple future latent prediction with a bounded residual for the same pending action, preserving the frozen base policy and its action-chunk interface.

2. **Structured historical dynamics.** We introduce compact robot and object dynamics tokens whose different informational roles are guided by measured-motion and region supervision, supplying historical context for action-conditioned future prediction.

3. **Evaluation across simulation and real robots.** We evaluate Peregrine with two base policies across six controlled simulation conditions and six Franka tasks, covering sustained motion, collision-induced trajectory changes, and long-horizon manipulation.

2 Related Work

VLA Execution and Residual Correction. RT-1, RT-2, OpenVLA, and Octo establish foundations for robot policies conditioned on vision and language [1, 2, 3, 4]; π_0, $\pi_{0.5}$, and GR00T N1 extend generative VLA control across tasks and embodiments [22, 28, 35]. ACT and Diffusion Policy generate action sequences [5, 33]. RTC overlaps execution with generation through action inpainting [6], AsyncVLA selectively refines action tokens through asynchronous flow matching [8], and A2C2 learns a residual from the latest observation, base action, chunk index, and policy features [7].

VLASH rolls robot states forward under issued actions [29]; F2F-AP predicts object flow to synthesize future observations and align their features [9]. FutureRTC predicts execution-time visual and proprioceptive inputs for a frozen policy [30], while Jetson-PI uses committed actions to predict future representations for an action expert and optimizes onboard scheduling [31]. These methods already address prediction–execution alignment. Peregrine combines compact historical robot/object dynamics with a joint visual/pose forecast to condition a bounded residual on a specific pending base command, with the horizon covering the correction pipeline through command adoption.

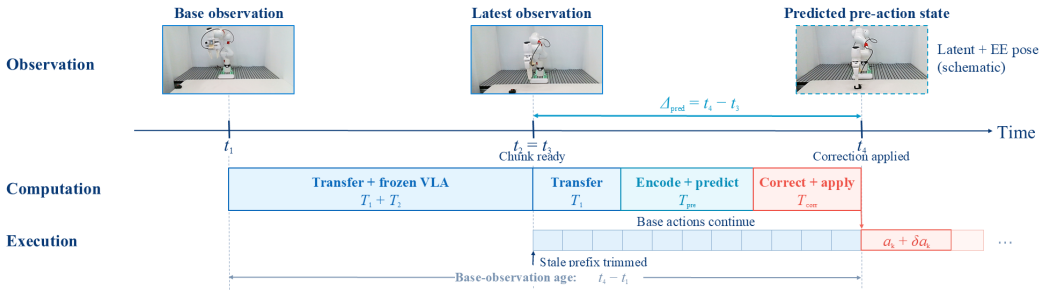

Figure 2: Observation, computation, and execution timeline for the first correction update, with $t_2 = t_3$. After the base action chunk becomes available, the stale prefix is discarded and base actions continue during transfer, prediction, and correction. The horizon $\Delta_{pred} = t_4 - t_3$ targets the pre-action state when the bounded correction is applied at t_4. The future-state photograph is schematic: Peregrine predicts latent visual features and end-effector pose.

Dynamic Robotic Manipulation. Dynamic manipulation requires responses to changing scene and contact conditions. Grasping methods use target-referenced representations [10], long-term grasp tracking [11], reachability- and motion-aware decisions [12], and temporally robust perception [13]. VLA-based approaches include DynamicVLA's continuous inference and action streaming [14] and the optical-flow and world-query motion cues of DOMINO/PUMA [15].

Future Prediction and Latent Representations. World models support state prediction and decision-making [16, 17]. WorldVLA autoregressively models observations and actions [18], DreamZero jointly models video and actions [19], and GigaBrain-0.5M* combines future-state conditioning with world-model-based reinforcement learning [20]. DINO-WM predicts pretrained visual patch features for planning [32]; V-JEPA 2 learns predictive video representations and an action-conditioned latent model for robot planning [36]. DynVLA uses quantized dynamics tokens for action reasoning in autonomous driving [21]. Peregrine uses robot-motion and region supervision to encode completed intervals into historical dynamics tokens, then predicts visual features and pose at a pending command's execution time for residual correction, without generating future images.

3 Problem Formulation

We study how available observations and action information can be used to predict the state when a correction takes effect and to adjust the base action at that same time.

Let O_{t_1} contain the images, robot state, and task instruction acquired at t_1. The frozen base policy π_0 generates an action chunk of length H. Observation transfer and base inference take T_1 and T_2, respectively, and the chunk becomes available at t_2:

$$A^0_{t_1} = \pi_0(O_{t_1}) = (a^0_0, \ldots, a^0_{H-1}), \qquad t_2 = t_1 + T_1 + T_2. \tag{1}$$

At t_2, the system discards the action prefix that has become stale over $T_1 + T_2$ and begins executing the remaining actions, retaining each action's execution timestamp on the original planned timeline. For the first correction update considered here, the latest observation is acquired when the action chunk is generated, so $t_3 = t_2$. In subsequent updates, t_3 advances with the acquisition time of each new observation, without waiting for another base-policy inference to finish.

Between observation acquisition at t_3 and the application of its correction, the system must perform transfer, prediction, and correction; the latter two stages take T_{pre} and T_{corr}, respectively. We select the target action's execution time t_4 using measured end-to-end latency. Let $\widehat{L}$ denote the latency estimate after alignment to that action's timestamp. The prediction horizon is then

$$t_4 = t_3 + \widehat{L}, \qquad \Delta_{pred} = t_4 - t_3, \qquad \widehat{L} \approx \widehat{T}_1 + \widehat{T}_{pre} + \widehat{T}_{corr}. \tag{2}$$

The transfer, prediction, and correction stages in Figure 2 provide a simplified timing model. End-to-end measurement must also account for the required feature processing, correction transmission, and

controller application, without counting the already completed base inference T_2 again. The target t_4 is selected in advance, while its actual application time is checked against execution logs. Base actions continue to execute during the attached modules' computation. The prediction horizon must cover the correction computation after prediction finishes, extending until the correction is applied to the corresponding action.

We therefore formulate the problem as finding a local correction for a pending action in the currently available chunk that is appropriate for the state at its application time, while keeping the base policy frozen and using only available observation history and action information. The central constraint is that the predicted state and corrected action share the same execution time t_4. The objective is to reduce action mismatch caused by state changes between observation and execution.

4 PEREGRINE

Peregrine uses historical dynamics and execution-time prediction to produce bounded corrections for pending actions of a frozen base VLA (Figure 1). The modules are trained sequentially offline, freezing previously trained upstream components at each stage.

4.1 ENCODING ROBOT AND OBJECT DYNAMICS

The Dynamic Tokenizer uses measured robot-motion supervision and start-conditioned reconstruction to guide two groups of dynamics tokens toward different informational roles, drawing on compact dynamics representations [21]. For a completed interval $[a, b]$ of duration $\delta = b - a$, frozen DINOv2 extracts endpoint features X_a, X_b [24]. After shared projection and the addition of spatial encodings and frame identifiers, a shared Transformer [37] encodes the paired features together with robot and object queries, conditioned on δ. Each group contains two queries. Their outputs U_R, U_O are mapped by G_R, G_O to continuous representations and then quantized using a shared codebook $\mathcal{C}$, following vector quantization in VQ-VAE [26]:

$$\begin{aligned} E_R = G_R(U_R), \quad E_O = G_O(U_O), \quad E = [E_R; E_O], \\ k_j = \underset{1 \le k \le K}{\operatorname{argmin}} \|e_j - \mathcal{C}_k\|_2^2, \quad z_j = \mathcal{C}_{k_j}, \quad Z_{a,b} = [Z_R; Z_O]. \end{aligned} \tag{3}$$

Here, e_j is the j-th vector in E, and Z_R, Z_O each contain two quantized vectors. Both query groups access the same scene changes, preserving interaction information; their different roles are learned through the supervision below rather than assigned by the query labels.

Start-conditioned reconstruction encourages the limited tokens to encode changes within the interval. The robot readout D_R uses only Z_R, the normalized initial state $\bar{q}_a$, and δ to estimate the seven-dimensional motion target $m_{a,b}$, comprising end-effector translation, relative rotation, and gripper-opening change. These labels come from measured state changes, not action commands. The visual and region readouts, D_X and D_S, use both token groups to reconstruct endpoint visual features and the layout of background, robot, and object regions:

$$\widehat{m}_{a,b} = D_R(\bar{q}_a, Z_R, \delta), \qquad \widehat{X}_b = D_X(X_a, \bar{q}_a, Z_{a,b}, \delta), \qquad \widehat{S}_b = D_S(S_a, M_a, Z_{a,b}, \delta). \tag{4}$$

S_a and M_a denote the initial region map and its validity mask. Because both endpoint images enter the encoder, these tasks reconstruct observed history rather than predict an unseen future. Robot-motion supervision directly constrains the robot group, whereas joint scene reconstruction permits shared interaction information. This asymmetric supervision encourages different emphases without requiring strict separation of robot and object motion. The joint objective is

$$\mathcal{L}_T = \mathcal{D}_{fg}(\widehat{X}_b, X_b) + \lambda_R \mathcal{L}_R(\widehat{m}_{a,b}, m_{a,b}) + \lambda_S \mathcal{L}_S(\widehat{S}_b, S_b; M_b) + \lambda_{VQ} \mathcal{L}_{VQ}. \tag{5}$$

$\mathcal{D}_{fg}$ combines full-image feature error with balanced errors over robot and object regions. $\mathcal{L}_R$ is the Huber loss on normalized motion targets, $\mathcal{L}_S$ is class-balanced cross-entropy over valid regions, and $\mathcal{L}_{VQ}$ combines codebook fitting and encoder consistency. Only valid labels contribute to each supervised term, and the λ coefficients set their weights. Quantization uses a straight-through estimator; encoding details, label construction, and the quantization objective are provided in Appendix A.1.

At runtime, only the frozen encoder and codebook are used to encode the endpoint images and duration of completed intervals. Motion labels, region maps, and auxiliary readouts are used only during offline training. The latest interval can end at the current observation without waiting for another image; its tokens and temporal information are passed to the predictor.

4.2 Future Prediction at Action Execution Time

Given the target execution time t_4 selected using the full end-to-end latency in Section 3, the Future Predictor anchors its estimate at the latest observation time t_3 and conditions on $\Delta_{pred} = t_4 - t_3$. It jointly predicts visual features and end-effector pose immediately before the target command begins to act. The predictor is independent of the base VLA and shares parameters across prediction horizons:

$$\left(\widehat{X}_{t_4|t_3}, \widehat{q}_{t_4|t_3}\right) = P_\theta\left(X_{t_3}, \bar{q}_{t_3}, c_\ell, \mathcal{H}_{t_3}, \mathcal{B}_{[t_3,t_4)}, \Delta_{pred}\right). \tag{6}$$

X_{t_3} and $\bar{q}_{t_3}$ are the current visual features and end-effector pose with normalized position; c_ℓ is the task condition and $\mathcal{H}_{t_3}$ the historical dynamics. The action condition $\mathcal{B}_{[t_3,t_4)}$ contains commands and their relative times of effect from execution records and plans available when prediction begins. It covers only the pre-action interval and excludes the correction not yet generated in the current update.

Conditions are organized into scene and history pathways. Current visual features, pose, task, and time-annotated actions form the scene condition $\mathcal{S}$. For each completed historical interval $[a_m, b_m]$, dynamics tokens retain role and within-group markers and encode both duration $\delta_m = b_m - a_m$ and age $\rho_m = t_3 - b_m$, with $b_m \le t_3$. Spatial queries for the visual grid and a pose query successively cross-attend to scene and history conditions, then exchange information through self-attention. Every query can read both token groups, allowing robot–object interactions to inform both outputs. Readout heads predict changes relative to the current representation and scale them by $g(\Delta) = \Delta/\delta_{ref}$, where δ_{ref} is a fixed reference duration. Quaternion normalization makes the orientation output valid, and a zero horizon recovers the current representation. Position is inverse-normalized before the predicted pose is passed to the corrector. Complete embeddings, attention layers, and readouts are given in Appendix A.2.

Training pairs an observation anchor t_3 with a recorded future endpoint t_f at a sampled horizon. History ends at t_3, and the action condition contains controls actually executed over $[t_3, t_f)$ in the same trajectory. Targets are frozen features of the future image and the measured pre-command pose at that endpoint. The time condition, action interval, and supervision target use matched actual timestamps, with t_f serving as the training counterpart of t_4. A single predictor is trained across horizons with the joint objective

$$\begin{aligned} \mathcal{L}_P &= \mathcal{D}_{fg}\left(\widehat{X}_{t_f|t_3}, sg(X_{t_f})\right) + \lambda_p \mathcal{L}_p + \lambda_{rot}\mathcal{L}_{rot}, \\ \mathcal{L}_p &= MSE_{valid}\left(\widehat{\bar{p}}_{t_f|t_3}, sg(\bar{p}_{t_f})\right), \qquad \mathcal{L}_{rot} = Mean_{valid}\left[1 - \left\langle \widehat{r}_{t_f|t_3}, sg(r_{t_f})\right\rangle^2\right]. \end{aligned} \tag{7}$$

$\mathcal{D}_{fg}$ follows Section 4.1; $\mathcal{L}_p$ measures normalized-position error on valid components, and $\mathcal{L}_{rot}$ uses squared unit-quaternion inner products so that r and $-r$ are equivalent. The λ coefficients weight the pose terms, and sg denotes stop-gradient. Only the predictor and its input/output mappings are updated. Online conditioning uses available records and plans, so prediction error also reflects plan–execution discrepancies. Sampling details and the distinction from corrector training are specified in Appendix A.3.

4.3 Action Correction Based on Future States

The Residual Corrector conditions on the single future prediction $\widehat{Y}_{t_4|t_3} = \left(\widehat{X}_{t_4|t_3}, \widehat{q}_{t_4|t_3}\right)$ and the base command a_k^0 at the same target time in the original action chunk. Together with task and action-timing context, these inputs produce a bounded local correction:

$$\delta a_k = C_\psi\left(\widehat{Y}_{t_4|t_3}, a_k^0, c_{t_3}\right), \qquad a_k^{exec} = a_k^0 + \Pi_{\mathcal{R}}(\delta a_k). \tag{8}$$

c_{t_3} contains the available task condition, original base action chunk, and the corrected command's original chunk index and execution time. $\Pi_{\mathcal{R}}$ constrains residual magnitude to a predefined range. Actions are composed in a common control coordinate system, with rotations combined according to

the action interface. The latest measured visual features affect the correction through the predictor, rather than entering the corrector as a separate input.

Offline residual supervision uses the difference between temporally aligned expert and base actions [7]. We cache chunks generated by the frozen VLA from earlier observations, select the command for t_4 along the original timeline, and pair it with the expert command a^*_{k,t_4} at the same demonstration time. After aligning action semantics and scales, we optimize

$$\delta a^*_k = a^*_{k,t_4} - a^0_k, \qquad \mathcal{L}_C = Huber(\delta a_k, \delta a^*_k) + \lambda_r \|\delta a_k\|_2^2. \tag{9}$$

The Huber term fits the residual target, while magnitude regularization encourages small local changes. During training, the frozen predictor uses only observed history and the base plan available at that time. The corrector consumes its actual predictions, including prediction errors, rather than ground-truth future features. Future expert commands are used only to construct residual targets. This input construction preserves the prediction interface used at deployment; the distinction between executed-action supervision and conditioning on unexecuted base plans is detailed in Appendix A.3.

On each new observation, the system updates the target time, prediction, and residual following Section 3, while previously available actions continue to execute. A correction applies only to its corresponding pending command. Every revision of the same command is relative to the fixed original a^0_k, without accumulating residuals across updates. Late corrections are not reassigned to other actions, and applied corrections enter subsequent execution records for the next prediction update.

5 Experimental Evaluation

We evaluate execution-time prediction, closed-loop manipulation in simulation and on Franka, and the contributions of history representations and supervision.

5.1 Compared Methods and Common Protocol

We compare GR00T, $\pi_{0.5}$ [22], DynamicVLA [14], PUMA [15], RTC [6], and Peregrine attached to GR00T or $\pi_{0.5}$. The GR00T N1 reference [28] provides model-family background; the exact GR00T release and checkpoint underlying the supplied results remain to be specified. Our primary backbone is $\pi_{0.5}$ adapted with LoRA [27], frozen after task adaptation and shared with RTC. Each base policy and its Peregrine counterpart use the same checkpoint; the GR00T add-on is trained separately. Baselines retain their original architectures and scheduling mechanisms.

All models learn from offline trajectories. Data are split by complete episodes and shared scene origins before constructing temporal windows. Normalization uses training-set statistics, and checkpoints are selected on the development set. Prediction and simulation share fixed weights across all horizons, with no horizon-specific retraining or online updates.

At 20 Hz, the four observation-to-correction intervals satisfy

$$\Delta_h = h\delta_c \in \{50, 100, 150, 200\}\,\text{ms}, \quad t_4 = t_3 + \Delta_h, \quad h \in \{1, 2, 3, 4\}. \tag{10}$$

For Peregrine, Δ_h includes transmission, encoding, prediction, correction, and command adoption; base-policy inference time T_2 is not added again. Predicted features, pose, and the corrected action share target time t_4. Franka retains its own protocol rather than this interval sweep.

Success criteria and timeouts are fixed in advance; macro-averages weight conditions equally. New evaluations use paired simulation states and perturbation seeds, randomized method order within physical trial blocks, and success counts with 95% confidence intervals based on independent trials or scenes. Existing simulation and Franka values are retained from supplied tables; previously estimated entries still require verification against trial records. Trial counts are not inferred from percentages. A dash marks a pending measurement.

5.2 Prediction Accuracy Across Four Horizons

We evaluate future visual-feature prediction on held-out offline trajectories, using a common set of observation anchors with valid future targets at all four horizons. The same predictor checkpoint

Table 1: Future visual-feature prediction error across four horizons (Latent MSE ↓). Persistence retains the current features; Peregrine uses the same checkpoint at all horizons.

Method	1 step 50 ms	2 steps 100 ms	3 steps 150 ms	4 steps 200 ms
Persistence	2.8	4.6	5.9	8.3
Peregrine	2.5	3.6	4.4	6.6

Table 2: Isaac Sim success rates (%). (a) Six-condition macro-averages by interval. (b) Linear (Low/Mid/High), arc (Acc/Dec), and rebound (Reb) results at 150 ms.

	(a) Macro SR (ms)				(b) Condition SR at 150 ms					
Method	50	100	150	200	Low	Mid	High	Acc	Dec	Reb
GR00T	43.50	39.00	37.17	34.00	35.00	17.00	5.00	68.00	74.00	24.00
$\pi_{0.5}$	62.17	59.00	57.83	51.83	89.00	46.00	22.00	72.00	78.00	40.00
DynamicVLA	65.50	60.17	59.33	51.50	83.00	46.00	27.00	73.00	79.00	48.00
PUMA	64.83	61.50	58.83	52.17	85.00	42.00	24.00	76.00	81.00	45.00
RTC	65.33	61.33	58.50	56.17	88.00	46.00	23.00	72.00	78.00	44.00
GR00T + Peregrine	66.50	63.67	59.83	54.83	81.00	32.00	14.00	82.00	86.00	64.00
$\pi_{0.5}$ + Peregrine	76.17	71.17	71.00	67.33	91.00	56.00	32.00	86.00	89.00	72.00

is evaluated at 50, 100, 150, and 200 ms. Inputs comprise the history available at each anchor and the recorded actions over the corresponding interval. Actual timestamps are matched using a fixed tolerance; anchors failing this requirement at any horizon are excluded from all four evaluations. Evaluation records specify the held-out trajectory count, common anchor count, and matching tolerance.

Future images are encoded with the same frozen DINOv2 encoder [24], preprocessing, and feature scale used during training. We report mean squared error over the complete predicted visual feature grid (Latent MSE):

$$E_{\text{latent}}(h) = \frac{1}{NN_p d_v} \sum_{i=1}^{N} \left\| \widehat{X}_{i,h} - X^*_{i,h} \right\|_F^2 . \tag{11}$$

Here, N is the number of common evaluation anchors, N_p is the number of spatial patches, and d_v is the feature dimension. $\widehat{X}_{i,h}$ is the predictor's final future visual output, and $X^*_{i,h}$ is the encoded image at the corresponding future time. Future images serve only as evaluation targets.

A Persistence reference uses the current features as the future prediction. Both methods use identical anchors and future targets (Table 1), testing whether prediction improves on retaining the current representation. This evaluation measures overall visual-feature accuracy; it does not separately establish pose accuracy or closed-loop success.

5.3 Controlled Dynamic Simulation

Isaac Sim [23] includes low-, medium-, and high-speed linear motion, accelerating and decelerating arcs, and collision-induced rebound (Figure 3). Methods share the robot, camera, object geometry, motion parameters, and initial phases.

A controlled clock restricts command adoption to implement the four intervals while the scene evolves and previously available actions execute. All methods share observation availability and delay rules: future observations and retroactive command insertion are prohibited, and the interval is counted once. This tests closed-loop performance at specified delays, not a measured 50 ms hardware budget.

At 150 ms, the supplied values in Table 2 show macro-averages changing from 37.17% to 59.83% for GR00T and from 57.83% to 71.00% for $\pi_{0.5}$ with Peregrine. For $\pi_{0.5}$, rebound success changes from 40.00% to 72.00%.

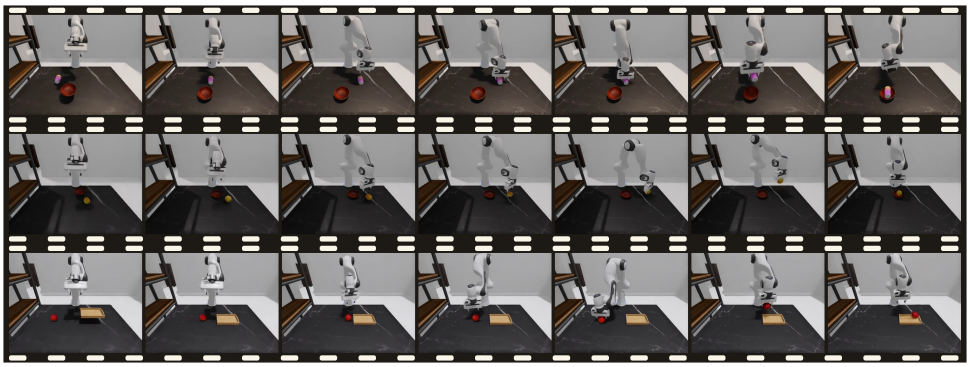

Figure 3: Linear, arc, and rebound motion (rows); time advances from left to right.

Table 3: Franka task success rates (%) and their unweighted mean (Avg.).

Method	Tank	Volle	Can	Sort	Cup	Ball	Avg.
GR00T	10.00	30.00	10.00	60.00	70.00	50.00	38.33
$\pi_{0.5}$	20.00	40.00	10.00	60.00	70.00	60.00	43.33
DynamicVLA	30.00	50.00	20.00	50.00	60.00	60.00	45.00
PUMA	30.00	50.00	20.00	50.00	70.00	60.00	46.67
RTC	20.00	40.00	20.00	60.00	70.00	60.00	45.00
GR00T + Peregrine	50.00	50.00	30.00	70.00	80.00	70.00	58.33
$\pi_{0.5}$ + Peregrine	60.00	60.00	40.00	70.00	80.00	80.00	65.00

5.4 Franka Manipulation

We consider remotely driven tank grasping (Tank), conveyor-shuttlecock grasping (Volle), rolling-can grasping (Can), three-object sorting (Sort), cup hanging (Cup), and conveyor-tennis-ball grasping (Ball). Figure 4 shows the platform and camera views.

In the supplied results (Table 3), Peregrine changes the six-task macro-average from 38.33% to 58.33% for GR00T and from 43.33% to 65.00% for $\pi_{0.5}$. These compare complete systems; the following ablations isolate component contributions.

5.5 Module Ablations at 150 ms

All six configurations use frozen LoRA-$\pi_{0.5}$ on the six simulation conditions. **Base** executes base actions directly. **Reactive** trains a separate corrector on current visual features, pose, and base actions, retaining the timing context and residual interface but omitting the tokenizer and predictor. **Raw-history** replaces dynamic tokens with patches from the same three historical frames, adds projection and spatial/temporal encodings, and retrains the predictor and corrector.

w/o Robot supervision removes only the dedicated robot-motion loss; **w/o Object supervision** removes only $\mathcal{L}_O$. Both retain both token groups and all other supervision, retraining the tokenizer and downstream modules. **Full** uses the complete design. Data access, training budgets, and model selection are matched; Raw-history has longer visual memory, so attention computation is not assumed equal.

Table 4 reports condition-wise success and the equally weighted macro-average. Full versus Reactive tests the anticipatory pathway; Full versus Raw-history tests the history representation. Loss ablations test the contributions of supervision to task performance without establishing strict motion disentanglement. Main-table success rates are reusable only with identical checkpoints, trials, seeds, intervals, and scoring. Prediction accuracy is evaluated separately in Section 5.2.

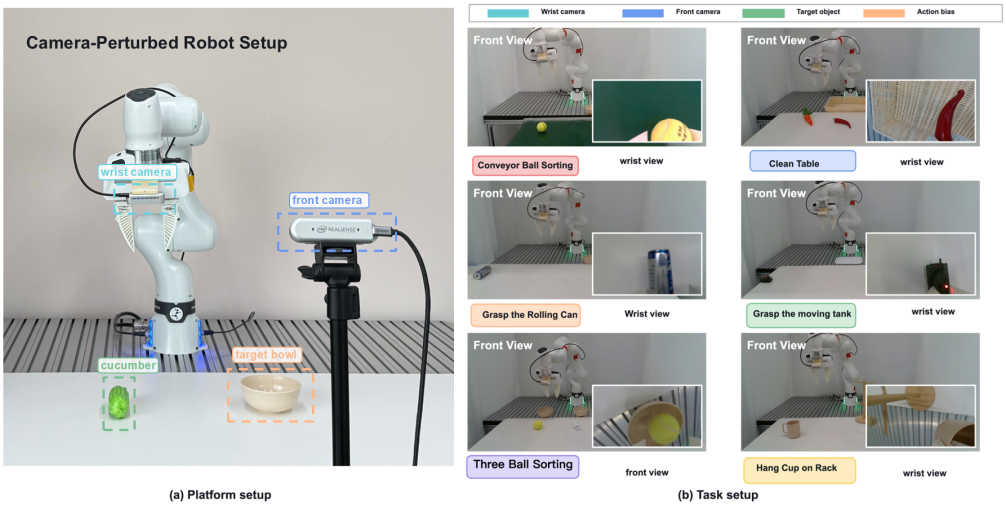

Figure 4: Franka setup and camera views. Peregrine uses a fixed external view; the base VLA retains its camera inputs.

Table 4: Ablation success rates (%) across six motion conditions at 150 ms. Avg. is the equally weighted mean; higher is better.

	Linear motion			Arc motion			
Configuration	Low	Medium	High	Accel.	Decel.	Rebound	Avg. ↑
Base VLA	89.00	46.00	22.00	72.00	78.00	40.00	57.83
Reactive correction	90.00	49.00	23.00	74.00	81.00	45.00	60.33
Raw-history replacement	90.00	54.00	26.00	78.00	86.00	63.00	66.17
w/o Robot supervision	91.00	51.00	24.00	76.00	85.00	60.00	64.50
w/o Object supervision	90.00	53.00	28.00	79.00	87.00	65.00	67.00
Full Peregrine	91.00	56.00	32.00	86.00	89.00	72.00	71.00

6 Limitations

Peregrine assumes that short-term states at action execution time can be predicted from the latest observation, robot state, and base-action context. The method may fail under unobserved target motion, prolonged occlusion, disturbances occurring shortly before execution, inaccurate alignment of the time at which corrections take effect, or semantic errors in the base policy that cannot be repaired by bounded residuals.

The tokenizer requires completed historical intervals, while predictor supervision requires recorded actions to be paired with actual future observations. Accuracy conditioned on demonstrations may not transfer to nominal base plans. More accurate execution-time estimation and broader coverage of motion in offline data are natural directions for extension.

7 Conclusion

We present Peregrine, a correction framework that aligns predictions with action execution time for chunked, flow-based VLAs. The framework predicts latent representations at execution time and applies bounded residuals while preserving the base VLA interface, supporting contact-rich dynamic manipulation.

AI USE STATEMENT

Generative AI assisted with literature search, conceptual and methodological discussion, drafting mathematical formulations, experimental design, translation, manuscript writing, and image generation. The method overview contains AI-generated imagery; simulation scenario images are taken from archived simulation recordings. Numerical results in the experimental tables were supplied by the authors. AI assistance with drafting and formatting does not replace experimental measurement or author verification. The authors remain responsible for the results, methodological claims, references, implementation, and final content. Final author review and disclosure in the submission forms remain pending.

ETHICS STATEMENT

Physical robot manipulation can introduce hazards through tracking errors, unexpected object motion, or inaccurate predictions. Peregrine constrains residual action changes but does not provide a general collision-avoidance or safety guarantee. Evaluation is limited to the reported simulation conditions and Franka tasks. We report no research involving human participants.

REPRODUCIBILITY STATEMENT

Appendix A details the model configuration, data adaptation, loss functions, training procedure, and online execution protocol. Tables 2–4 report simulation, real-robot, and ablation success rates; Table C.1 provides additional condition-wise results. Exact code versions, trial counts, random seeds, statistical uncertainty, and complete training and evaluation logs remain to be documented.

REFERENCES

[1] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://doi.org/10.15607/RSS.2023.XIX.025.

[2] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. 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*, pages 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

[3] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[4] Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems XX*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://doi.org/10.15607/RSS.2024.XX.090.

[5] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://doi.org/10.15607/RSS.2023.XIX.016.

[6] Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, volume 38, pages 33383–33407, 2025. doi: 10.52202/085713-1122. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/300ccb2187dedd4edcc07f7e76d8e553-Abstract-Conference.html.

[7] Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for VLA action chunks. *arXiv preprint arXiv:2509.23224*, 2025. URL https://arxiv.org/abs/2509.23224.

[8] Yuhua Jiang, Shuang Cheng, Yan Ding, Feifei Gao, and Biqing Qi. AsyncVLA: Asynchronous flow matching for vision-language-action models. *arXiv preprint arXiv:2511.14148*, 2025. URL https://arxiv.org/abs/2511.14148.

[9] Haoyu Wei, Xiuwei Xu, Ziyang Cheng, Hang Yin, Angyuan Ma, Bingyao Yu, Jie Zhou, and Jiwen Lu. F2F-AP: Flow-to-future asynchronous policy for real-time dynamic manipulation. *arXiv preprint arXiv:2604.02408*, 2026. URL https://arxiv.org/abs/2604.02408.

[10] Jirong Liu, Ruo Zhang, Hao-Shu Fang, Minghao Gou, Hongjie Fang, Chenxi Wang, Sheng Xu, Hengxu Yan, and Cewu Lu. Target-referenced reactive grasping for dynamic objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8824–8833, 2023. doi: 10.1109/CVPR52729.2023.00852. URL https://doi.org/10.1109/CVPR52729.2023.00852.

[11] Nuo Chen, Xiao-Ming Wu, Guohao Xu, Jian-Jian Jiang, Zibo Chen, and Wei-Shi Zheng. MotionGrasp: Long-term grasp motion tracking for dynamic grasping. *IEEE Robotics and Automation Letters*, 10 (1): 796–803, 2025. doi: 10.1109/LRA.2024.3504792. URL https://doi.org/10.1109/LRA.2024.3504792.

[12] Iretiayo Akinola, Jingxi Xu, Shuran Song, and Peter K. Allen. Dynamic grasping with reachability and motion awareness. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 9422–9429, 2021. doi: 10.1109/IROS51168.2021.9636057. URL https://doi.org/10.1109/IROS51168.2021.9636057.

[13] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. AnyGrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39 (5): 3929–3945, 2023. doi: 10.1109/TRO.2023.3281153. URL https://doi.org/10.1109/TRO.2023.3281153.

[14] Haozhe Xie, Beichen Wen, Jiarui Zheng, Zhaoxi Chen, Fangzhou Hong, Haiwen Diao, and Ziwei Liu. DynamicVLA: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026. URL https://arxiv.org/abs/2601.22153.

[15] Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. *arXiv preprint arXiv:2603.15620v3*, 2026. Accepted to ECCV 2026. URL https://arxiv.org/abs/2603.15620v3.

[16] Xinqing Li, Xin He, Le Zhang, Min Wu, Xiaoli Li, and Yun Liu. A comprehensive survey on world models for embodied AI. *arXiv preprint arXiv:2510.16732*, 2025. URL https://arxiv.org/abs/2510.16732.

[17] Jingtao Ding, Yunke Zhang, Yu Shang, Jie Feng, Yuheng Zhang, Zefang Zong, Yuan Yuan, Hongyuan Su, Nian Li, Jinghua Piao, Yucheng Deng, Nicholas Sukiennik, Chen Gao, Fengli Xu, and Yong Li. Understanding world or predicting future? a comprehensive survey of world models. *arXiv preprint arXiv:2411.14499*, 2024. URL https://arxiv.org/abs/2411.14499.

[18] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025. URL https://arxiv.org/abs/2506.21539.

[19] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

[20] GigaBrain Team, Boyuan Wang, Bohan Li, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, Lv Feng, Mingming Yu, Peng Li, Qiuping Deng, Tianze Liu, Xinyu Zhou, Xinze Chen, Xiaofeng Wang, Yang Wang, Yifan Li, Yifei Nie, Yilong Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaBrain-0.5M*: A VLA that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*, 2026. URL https://arxiv.org/abs/2602.12099.

[21] Shuyao Shang, Bing Zhan, Yunfei Yan, Yuqi Wang, Yingyan Li, Yasong An, Xiaoman Wang, Jierui Liu, Lu Hou, Lue Fan, Zhaoxiang Zhang, and Tieniu Tan. DynVLA: Learning world dynamics for action reasoning in autonomous driving. *arXiv preprint arXiv:2603.11041*, 2026. URL https://arxiv.org/abs/2603.11041.

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

[23] NVIDIA Corporation. Isaac Sim: Robotics simulation and synthetic data generation. Online software documentation. URL https://developer.nvidia.com/isaac/sim. Accessed September 26, 2026.

[24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL https://arxiv.org/abs/2304.07193.

[25] Haozhe Xie et al. DOM public dataset: Metadata and feature specification. Hugging Face dataset, 2026. URL https://huggingface.co/datasets/hzxie/DOM/blob/main/meta/info.json. Accessed September 26, 2026.

[26] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL

https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

[27] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[28] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

[29] Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. VLASH: Real-Time VLAs via Future-State-Aware Asynchronous Inference. *arXiv preprint arXiv:2512.01031*, 2025. URL https://arxiv.org/abs/2512.01031.

[30] Hai Jiang, Yixian Zou, Binbin Liang, Boqian Liu, Fanman Meng, and Shuaicheng Liu. FuturERTC: Real-Time Robot Execution with Anticipatory-Conditioned Action Chunking. *arXiv preprint arXiv:2607.24008*, 2026. URL https://arxiv.org/abs/2607.24008.

[31] Zebin Yang, Qi Wang, Yunhe Wang, Xiurui Guo, Bo Yu, Shaoshan Liu, Jiafeng Xu, Hao Dong, and Meng Li. Jetson-PI: Towards Onboard Real-Time Robot Control via Foresight-Aligned Asynchronous Inference. *arXiv preprint arXiv:2607.12659*, 2026. URL https://arxiv.org/abs/2607.12659.

[32] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. *arXiv preprint arXiv:2411.04983*, 2024. URL https://arxiv.org/abs/2411.04983.

[33] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

[34] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[35] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

[36] Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

A Implementation Details and Reproducibility Protocol

This appendix specifies the tensors, supervision signals, and execution rules. Table A.1 presents the reference architecture; run configurations are documented in Appendix A.4. Neither the reference settings nor checks using synthetic computations constitute training results.

A.1 Model Configuration and Data Adaptation

Table A.1: Reference model configuration. The state interface and effective action interface determine d_q and d_a, respectively.

Component	Configuration
Visual encoder	Frozen DINOv2 ViT-S/14; 224×224 input; $N = 256, d_v = 384$
Tokenizer	516 sequence positions; 2 pre-LN SA+FFN layers; width 256, 4 attention heads, FFN width 1024
Role-specific readouts	2 Robot queries and 2 Object queries; role-specific LN+Linear; code dimension $d_z = 32$
Quantization	Shared 64×32 codebook; nearest-neighbor distances computed in float32; straight-through (ST) gradients; $\beta = 0.25$
Visual reconstruction	Initial visual features/state, all codes, and time; 256 queries; 2 SA→CA→FFN layers
Robot head	Two Robot codes, initial state, and time; an MLP predicts 7 normalized motion components
Region reconstruction	Initial three-class label map and validity mask; 256 queries; 2 SA→CA→FFN layers attending to all codes
Predictor	256 spatial queries and 1 state query; width 256, 4 attention heads, FFN width 1024; 3 scene CA→history CA→SA→FFN layers
Historical memory	8 codes from the two most recent complete intervals; role, slot, duration, and age relative to the current time; 2 role-specific missing-history entries when no history is available
Time encoding	Raw time in seconds and sine/cosine features at 1, 2, 4, and 8 Hz; phase $2\pi ft$; projection through a two-layer MLP
Initial T/P optimizer settings	AdamW [34]; learning rate 10^{-4}, weight decay 10^{-4}, gradient-norm clipping threshold 1, dropout 0

Visual and task conditioning. We obtain normalized DINOv2 patch features [24] by calling forward_features and reading x_norm_patchtokens, excluding CLS and other non-spatial positions. The features form a 16×16 grid and are scaled by $1/\sqrt{384}$ exactly once. The backbone remains frozen and in evaluation mode. RGB frames and region labels share deterministic spatial transformations, with fixed color-channel order, resizing/cropping, interpolation, and backbone normalization. The initial implementation disables independent random augmentation across frames and uses interpolation that preserves discrete labels for class maps. Cache keys include the camera, frame, data revision, preprocessing, and model version.

Task conditioning uses only instructions or metadata known at the anchor time. The minimal implementation for a fixed task set uses 256-dimensional embeddings and a common task dictionary; a constant token can also be used for a single task. If frozen text features are used instead, their encoder, pooling method, and dimensionality must be recorded, and generalization to unseen language conditions must be evaluated separately.

Paired-frame encoding. For a completed interval $[a, b]$ of duration $\delta = b - a$, the frozen visual encoder produces $X_a, X_b \in \mathbb{R}^{N \times d_v}$. A shared projection W_V maps these features to width d; spatial encodings P and frame identifiers f_a, f_b preserve location and temporal order. Two robot queries

Q_R and two object queries Q_O read the paired features through a shared Transformer [37]:

$$\begin{aligned} V_a = X_a W_V + P + f_a, \qquad V_b = X_b W_V + P + f_b + \phi_T(\delta), \\ U = Enc_\omega\left([V_a; V_b; Q_R + \phi_T(\delta); Q_O + \phi_T(\delta)]\right), \\ E_R = G_R(U_R), \qquad E_O = G_O(U_O). \end{aligned} \tag{A.1}$$

Here, ϕ_T embeds interval duration, U_R, U_O are the query-group outputs, and G_R, G_O map them to code dimension d_z. Both groups can access all paired-frame features. The continuous vectors are quantized by Eq. (3), using the shared codebook in Table A.1.

States, actions, and kinematics. State normalization and action adaptation are configured separately. When Franka states contain joint angles, physical units must be restored before forward kinematics (FK) is computed. The robot model, joint order, zero offsets and directions, base frame, and tool transform are fixed. Recorded measurements of the end-effector pose may also be used directly under the corresponding coordinate conventions. The end-effector position p_t, rotation R_t, and measured total gripper opening g_t define the Robot supervision target:

$$m_{a,b} = \rho_R\left(\begin{bmatrix} p_b - p_a \\ vee\left(Log(R_a^\top R_b)\right) \\ g_b - g_a \end{bmatrix}\right), \qquad \rho_R(v) = v \oslash \sigma_R, \quad \sigma_R \in \mathbb{R}^7_{>0}. \tag{A.2}$$

The matrix logarithm uses the principal branch, and the vee operator returns a three-dimensional rotation vector, with numerical handling for zero angles and angles close to π. Translation and gripper opening are measured in meters, and rotation in radians. The scaling vector σ_R is determined from the training set and then frozen. Robot target commands cannot substitute for measured states, and the position of a single finger does not automatically equal the total gripper opening. Missing motion components are masked individually.

Internal padding and transformations relative to the current state must first be handled by the base policy's output adapter. Only the resulting effective actions enter P as conditioning or C as operands in residual correction. Under the shared affine normalization $N(a) = (a - \mu_a) \oslash \sigma_a$, execution follows $a^{phys} = N^{-1}(N(a^0) + \Pi_{\mathcal{R}}(\delta a))$; the mean is not separately added to the residual. Absolute rotation matrices or quaternions require a valid composition interface: elementwise addition does not compose rotations. The implementation configuration must specify the Corrector architecture, the organization of its context, and the scales of its constraints.

GR00T and DOM adaptation. We fix the GR00T version and checkpoint, adapt the policy to the target domain, and freeze the backbone. A separate action adapter produces effective commands. P uses recorded transitions with the corresponding state/action interfaces; C uses candidate actions from that policy and imitation residuals. Transitions conditioned on expert actions remain distinct from nominal plans used during deployment. Sharing a frozen Tokenizer within a domain requires identical visual encoders, preprocessing, history sampling, and data splits. Changes to normalization and interfaces must be recorded; complete reuse of the auxiliary modules' weights is not assumed.

DOM provides LeRobot v2.1 data at a nominal frequency of 25 Hz: six-dimensional measured end-effector poses and seven-dimensional pose/gripper commands. The adapter follows the official rotation order and coordinate conventions, converts Euler-angle states to rotation matrices, and constructs Eq. (A.2); missing measurements of gripper opening are masked. Training and deployment must agree on absolute versus relative definitions and units, not merely vector dimensions. Conversion to LeRobot v3 preserves episode identifiers, timing information, and action semantics; changing the storage format alone does not establish compatibility. See [14, 25] for metadata and code.

Quantization and gradients. All four codes share a single codebook. We use the VQ-VAE codebook and commitment losses with a straight-through estimator [26], averaged over the batch, slots, and channels:

$$\begin{aligned} \mathcal{L}_{VQ} = \tfrac{1}{BJd_z} \textstyle\sum_{n,j,d} \left[(sg(e_{n,j,d}) - \mathcal{C}_{k_{n,j},d})^2 + \beta(e_{n,j,d} - sg(\mathcal{C}_{k_{n,j},d}))^2\right], \\ Z_{ST} = E + sg(\mathcal{C}[k] - E). \end{aligned} \tag{A.3}$$

Here, B is the batch size, $J = 4$ the number of code vectors, and $\beta = 0.25$. All offline task heads in Eq. (4) read Z_{ST}. Task gradients propagate to the encoder, the first loss term updates the codebook, and the commitment term constrains the encoder readouts. Once the Tokenizer is frozen, P reads vectors obtained by codebook lookup, or indices tied to this fixed codebook, rather than continuous vectors before quantization.

A.2 Complete Loss and Attention Definitions

Region reconstruction. Labels are 224×224 integer maps: background, robot, and object are encoded as 0, 1, and 2, respectively; the ignore label is 255. Validity masks are stored separately. Three valid one-hot channels and one validity channel are concatenated, partitioned into 14×14 patches, and flattened into 256 vectors of dimension 784, each projected to width 256. After position and time encodings are added, the queries pass through two layers, each consisting of query SA, CA over all four codes, and an FFN. Each position produces $14 \times 14 \times 3 = 588$ logits, which are rearranged into a $3 \times 224 \times 224$ map. Code vectors are projected from 32 to 256 dimensions.

Let $\Omega_{n,c}$ denote the valid endpoint pixels in sample n belonging to class c, $\mathcal{C}_n$ the set of nonempty classes, and $\mathcal{N}_S$ the set of samples with at least one valid endpoint pixel. The class-balanced region loss is

$$\mathcal{L}_S = \frac{1}{|\mathcal{N}_S|} \sum_{n \in \mathcal{N}_S} \frac{1}{|\mathcal{C}_n|} \sum_{c \in \mathcal{C}_n} \frac{1}{|\Omega_{n,c}|} \sum_{u \in \Omega_{n,c}} -logsoftmax(\widehat{S}_{b,n}(u))_c. \tag{A.4}$$

For ignored pixels at the start of an interval, both the one-hot channels and the validity channel are zero; ignored endpoint pixels are excluded from the loss. Absent classes are excluded from the denominator. If a batch contains no valid labels, this term is skipped and its coverage is recorded. Visible occluders receive their actual class labels; no contours are fabricated for unknown hidden regions. The procedures for generating pseudo-labels and performing sampled quality checks are fixed during development.

Spatially balanced feature supervision. Let $e_{n,i} = d_v^{-1} \|\widehat{X}_{n,i} - X_{n,i}\|_2^2$. The weight $w_{n,i,c}$ for endpoint patch i is the number of valid pixels of class c divided by the patch area. Let $\mathcal{F}_n \subseteq \{R, O\}$ denote the foreground classes with valid pixels. The error for each sample is

$$\mathcal{D}_{fg,n} = \frac{1}{N} \sum_i e_{n,i} + \frac{\eta_{fg}}{|\mathcal{F}_n|} \sum_{c \in \mathcal{F}_n} \frac{\sum_i w_{n,i,c} e_{n,i}}{\sum_i w_{n,i,c}}. \tag{A.5}$$

When no valid foreground is present, only the first term is retained. The result is then averaged over samples with valid images. Area-based aggregation preserves the contribution of foreground regions smaller than a patch. Positive area denominators retain their actual values, while zero denominators are handled before division. T and P use region labels from their respective observed endpoints. The Robot Huber loss uses a threshold of 1 in normalized units. The Robot and state losses are first averaged over valid dimensions and then over valid samples; each supervision target maintains its own validity set.

Historical and scene conditions. For a completed interval $[a_m, b_m]$, let $\delta_m = b_m - a_m$ and $\rho_m = t_3 - b_m$, with $b_m \le t_3$. Each quantized dynamics vector is projected to the predictor width and combined with role, within-group slot, duration, and age encodings:

$$h_{m,j} = W_H z_{m,j} + e_{group(j)} + e_{index(j)} + \phi_H(\delta_m, \rho_m), \qquad \mathcal{H}_{t_3} = \{h_{m,j}\}_{m,j}. \tag{A.6}$$

The scene condition $\mathcal{S}$ contains the current visual features, normalized end-effector pose, task condition, and timed action sequence. Action values use a separate projection and are combined with encodings of their start and end times relative to the anchor. The neural network inputs do not include learned embeddings for actual/locked/nominal status. Invalid padding values are zeroed before projection, and the corresponding keys are masked. When all history is missing, valid missing-history entries prevent every softmax position from being masked.

Predictor queries and updates. We construct one spatial query for each location in the visual grid and one pose query. Each spatial query combines a shared learnable visual embedding, the projected current feature at that location, its spatial encoding, and a prediction-horizon embedding. The pose query combines a separate learnable embedding, the projected current normalized pose, and the same horizon embedding. Stacking these queries gives $Q^{(0)} \in \mathbb{R}^{(N+1) \times d}$. Each pre-LN layer performs the following residual updates:

$$\begin{aligned} Q_1 &= Q + CA_S(LN(Q), \mathcal{S}), & Q_2 &= Q_1 + CA_H(LN(Q_1), \mathcal{H}_{t_3}), \\ Q_3 &= Q_2 + SA(LN(Q_2)), & Q^+ &= Q_3 + FFN(LN(Q_3)). \end{aligned} \tag{A.7}$$

Here, CA and SA denote cross-attention and self-attention, respectively, while LN and FFN denote layer normalization and a feed-forward network. LN parameters are independent across layers and sublayers. Every prediction query can attend to both robot and object dynamics groups. After three layers, the final LN and linear heads produce $U_X \in \mathbb{R}^{N\times d_v}$, $U_p \in \mathbb{R}^3$, and $U_r \in \mathbb{R}^4$.

Prediction readouts. Let the end-effector pose be $q = (p, r)$, where $p \in \mathbb{R}^3$ is position and $r \in \mathbb{R}^4$ is a unit quaternion. Position is normalized using training-set statistics to obtain $\bar{p}$; orientation retains its unit-quaternion representation. Omitting the prediction subscript $t_4 \mid t_3$, the outputs are

$$\begin{aligned} \widehat{X} &= X_{t_3} + g(\Delta_{pred})U_X, \qquad \widehat{\bar{p}} = \bar{p}_{t_3} + g(\Delta_{pred})U_p, \\ \widehat{r} &= \mathcal{N}\left(r_{t_3} + g(\Delta_{pred})U_r;\, r_{t_3}\right), \qquad g(\Delta) = \Delta/\delta_{ref}. \end{aligned} \tag{A.8}$$

The reference duration δ_{ref} is fixed. The operator $\mathcal{N}(v; r_0)$ normalizes v to unit length and returns the reference unit quaternion r_0 when the norm is too small. These readouts recover the current representation at a zero horizon. The position output is inverse-normalized and combined with $\widehat{r}$ to form the seven-dimensional pose supplied to the corrector with the predicted visual features.

Feature-channel MSE, region cross-entropy, and physical motion errors have different scales; $g(\Delta_{pred})$ also changes the gradient scale of prediction updates. Training records must include each loss term and gradient ranges in the shared layers. The weights $\lambda_R, \lambda_S, \lambda_{VQ}, \lambda_p, \lambda_{rot}, \eta_{fg}$ are fixed before evaluation, and errors are examined separately for each prediction horizon. The reference optimizer settings do not replace this numerical calibration.

A.3 Training Samples and Online Execution

Prediction interface. Let X_t denote the image features from a frozen visual encoder, and let $\bar{q}_t$ denote the robot state normalized according to the specified state interface. Anchored at the latest observation time t_3, the single future prediction is written as

$$\widehat{Y}_{t_4|t_3} = (\widehat{X}_{t_4|t_3}, \widehat{q}_{t_4|t_3}) = P_\theta\left(\mathcal{I}_{\le t_3}, \mathcal{B}_{[t_3,t_4)}, \Delta_{pred}\right). \tag{A.9}$$

Here, $\mathcal{I}_{\le t_3}$ comprises the latest observation available for this update, the task instruction, and the observation history through t_3. The action information $\mathcal{B}_{[t_3,t_4)}$ consists of execution records and action plans available when prediction begins, covering $[t_3, t_4)$ and excluding the correction that has not yet been generated in this update. The predicted visual features and robot state jointly describe the state immediately before the target action begins to take effect at t_4.

Offline windows. Data are split by episode or source scene before Tokenizer intervals and multi-horizon Predictor samples are constructed. Offline sampling retains the frame index t and offset h: under the convention that observations precede command dispatch, the transition from t to $t + h$ is conditioned on actions $a_t, \ldots, a_{t+h-1}$, with the actual observation at $t + h$ as the target. Their acquisition timestamps correspond to t_3 and t_4 in the deployment notation, respectively, and their difference is Δ_{pred}; frame indices are not treated directly as continuous time. Under a 20 Hz action timing convention, the four nominal horizons of 1, 2, 3, and 4 steps are sampled in a balanced manner, corresponding to 50, 100, 150, and 200 ms. The time condition, action interval, and supervision target use matched actual timestamps, and all horizons share one predictor. The data adapter verifies the acquisition order. Historical sampling intervals and future prediction horizons are configured separately. Missing history can be masked; targets beyond episode boundaries are excluded rather than fabricated by repeating the final frame.

LeRobot v3 specifies storage and indexing, while action units, postprocessing, and hardware clocks remain part of the dataset configuration. Times inferred solely from the nominal frame rate must be identified as nominal, and synchronization errors among video, states, and actions are recorded separately. Offline expert-conditioned samples are checked for consistency among trajectory identifiers, action windows, and observed futures; artificial availability timestamps are not assigned to make later expert actions appear known at the anchor time. The online action snapshot $\mathcal{B}_{[t_3,t_4)}$ includes only commands available before the current prediction begins, with interval coverage and action versions verified. The provenance of these two types of samples is checked separately.

Corrector samples cache action chunks generated by the base policy from observations at t_1. A candidate corresponding to target t_4 is selected on the original action timeline and subtracted from the corresponding expert command. Anchored at the latest observation t_3, P reads the observed

demonstration history and the base plan available for the current update, and generates a single prediction for t_4. Actual future observations are used only for supervision associated with the corresponding executed actions and for offline error measurement; they are never treated as inputs available at deployment. Training first freezes T and trains P, then freezes P and trains C. Replacing T or its codebook requires rebuilding the cache and adapting the corresponding P. Replacing the history pathway also requires retraining C for the resulting prediction inputs.

Each online update. For pending actions in the available action chunk, the controller performs temporal alignment as follows:

1. Read the latest observation and its acquisition time t_3, and update the frozen visual-feature cache and the dynamic codes for newly completed intervals. For the first update, $t_3 = t_2$; at t_2, the base action chunk has already had its stale prefix corresponding to $T_1 + T_2$ removed, and execution of the remaining actions has begun.

2. Select the correction target t_4 using an estimate based on measured end-to-end latency of the plug-in, identify the corresponding base action a_k^0 from the original action timestamps, and fix the available $\mathcal{B}_{[t_3,t_4)}$ and its version for this update. The prediction horizon is $\Delta_{pred} = t_4 - t_3$, covering transfer, prediction, correction, and the actual overhead of dispatch and adoption by the controller.

3. Conditioned on visual information, measured state, historical dynamics, task information, and action information at t_3, invoke P once to predict the visual features and robot state immediately before the action associated with t_4 takes effect.

4. Feed the single future prediction and the base action and context associated with t_4 into C. The latest measured visual information affects C through the predictor, without a separate visual input to C. Form a new residual relative to the fixed base action and apply the constraints. Base actions continue to execute during prediction and correction computation.

5. Before the correction takes effect, verify the identifier and version of the corresponding action, and apply the correction to its assigned target action. Record the deviation of the actual effective time t_4^{act} from the target; a correction that misses its target action must not be directly reassigned to another action. Continue updating from new observations, without introducing a separate waiting stage after correction computation finishes.

Each update computes only one future prediction, anchored at the latest observation. The reported cost includes the complete pipeline of history encoding, prediction, correction, dispatch, and adoption by the controller. An update is not guaranteed for every camera frame; the effective rate depends on the actual computation and execution pipeline. Residual constraints, verification of action identifiers and versions, and continuous feedback limit local command changes, conditioning mismatches, and observation staleness, respectively. None alone establishes global closed-loop stability.

A.4 Experimental Records and Interpretation

Each run records the data revision; lists of training, development, and test trajectories; robot and action configurations, including the Franka model and calibration; cameras and preprocessing; label versions; normalization statistics; history/action windows and the horizon distribution; random seeds; training budget and number of steps; loss weights; and checkpoints. Corrector records include its architecture, normalization space, $\Pi_{\mathcal{R}}$, and λ_r. Deployment logs specify the hardware used for the complete pipeline, the control frequency, t_1, t_2, t_3, t_4 and the actual effective time t_4^{act}, original action indices, end-to-end latency estimates, timing of individual stages, history-feature cache hits, and statistics on missed target times. RTC records additionally specify the action-chunk length, number of denoising steps, scheduling parameters, and guidance overhead included in the measured end-to-end latency. All settings not yet specified here, including the remaining evaluation trial counts, must be determined from the final configurations and logs before experimental results are reported. Neither demonstration counts nor earlier experimental settings can substitute for these records.

The Latent MSE values in Table 1 use the same held-out anchors and future targets for both methods. Errors are averaged over spatial patches and feature dimensions within each anchor, then equally across anchors, as defined in Section 5.2. When uncertainty is estimated, complete independent trajectories or scene groups are resampled and the same estimator is recomputed. Tables 2 and 4 compare methods using fixed scenario lists, reproducible initial states, and seeds for exogenous

disturbances. When trials in Table 3 cannot be paired trajectory by trajectory, methods are randomized within predefined task blocks, and uncertainty intervals are computed using the corresponding independent sampling units. Intervals for task macro-averages must be distinguished from binomial intervals obtained by pooling all trials.

These evaluations support different levels of conclusions: gains from the Tokenizer do not establish the independent effects of individual supervision terms or strict physical disentanglement; prediction accuracy under expert conditioning does not establish intervention accuracy under arbitrary plans; and residual fitting does not establish closed-loop stability. The final claims remain bounded by the evidence available at each level.

B Retained Material from the Original Manuscript

This appendix distinguishes material retained from the supplied manuscript from the algorithm revised in Sections 3–4.3. The retained experimental images appear in Section 5, and the original architecture diagram is retained in Figure B.1. The current architecture is shown in Figure 1.

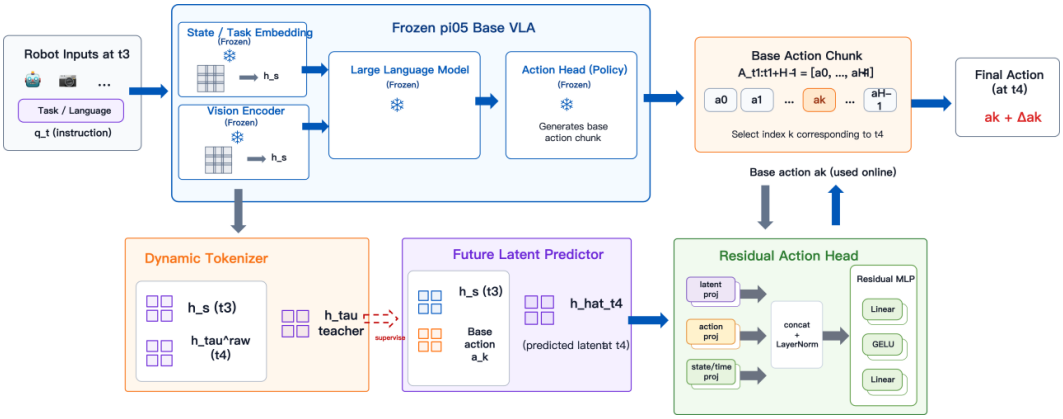

Figure B.1: Original Peregrine architecture diagram, retained unchanged to preserve continuity with the source manuscript. The diagram depicts the earlier teacher-only Tokenizer and reconstruction-based supervision of the Predictor. These connections have been replaced by the historical-token interface and direct supervision from actual future observations described in Section 4; the diagram does not represent another currently active model.

Updated numerical tables. Tables 2 and 3 have been updated using the two numerical tables supplied for this revision. Thirteen displayed averages agree, after rounding to two decimal places, with the equally weighted means of their respective six component values. The $\pi_{0.5}$ simulation row retains the average reported in the source manuscript. The supplied values have not been independently verified as results of completed evaluations; their provenance remains to be confirmed. Earlier manuscripts and their table inventories remain archived separately to avoid overwriting historical evidence.

The GR00T simulation average remains 37.17, calculated from its six condition-specific values. At the author's request, the $\pi_{0.5}$ simulation row has been restored to its original values, including the reported average of 57.80. The arithmetic mean of its six displayed condition-specific values is approximately 57.83; 57.80 is retained as the value reported in the source manuscript. The Franka table also includes revised values and the column labels Tank, Volle, Can, Sort, Cup, and Ball. The meanings of Volle and the final Ball column still require confirmation before they can be associated with the task descriptions or images in the source manuscript. The earlier manuscript's counts of approximately 200 demonstrations per task and approximately 1,200 in total are historical descriptions, not verified data inventories for any modified task set.

Original diagnostics. For traceability, the source manuscript also reported a +0.11 increase in the cosine margin of future latent representations, an 18% reduction in future-latent loss relative to

directly copying the current latent representation, a 31% reduction in action MSE, and a 98% residual acceptance rate. These are claims from the earlier manuscript. They do not validate the revised historical Tokenizer, the single-prediction Predictor, or the current deployment protocol, and they are not used as results for the new ablations.

Evaluation evidence to be added. The median placeholder values previously marked E have been removed from the two main tables and replaced with the supplied revised values. This replacement alone does not establish that methods not previously measured have now been evaluated. No confidence intervals, trial counts, significance claims, or state-of-the-art conclusions are inferred from the supplied percentages. Table 4 has also been populated with the supplied 150 ms ablation success rates.

C Supplementary Experimental Tables

Table C.1 provides condition-wise success rates at one, two, and four steps. Three-step details appear in Table 2(b), and four-horizon macro-averages in Table 2(a). All horizons share the same checkpoint. "—" denotes a pending measurement.

Table C.1: Condition-wise simulation success rates (%) at one, two, and four steps. Three-step details are reported in Table 2(b).

(a) 1 step / 50 ms

Method	Linear low	Linear mid	Linear high	Arc accel.	Arc decel.	Rebound
GR00T	41.00	24.00	12.00	72.00	84.00	28.00
$\pi_{0.5}$	89.00	54.00	25.00	79.00	77.00	49.00
DynamicVLA	85.00	53.00	34.00	81.00	84.00	56.00
PUMA	92.00	45.00	30.00	85.00	84.00	53.00
RTC	90.00	56.00	28.00	79.00	85.00	54.00
GR00T + Peregrine	84.00	43.00	19.00	88.00	90.00	75.00
$\pi_{0.5}$ + Peregrine	92.00	65.00	38.00	91.00	94.00	77.00

(b) 2 steps / 100 ms

Method	Linear low	Linear mid	Linear high	Arc accel.	Arc decel.	Rebound
GR00T	40.00	18.00	9.00	65.00	80.00	22.00
$\pi_{0.5}$	85.00	49.00	24.00	76.00	76.00	44.00
DynamicVLA	80.00	45.00	25.00	79.00	82.00	50.00
PUMA	88.00	47.00	23.00	79.00	85.00	47.00
RTC	89.00	45.00	27.00	78.00	85.00	44.00
GR00T + Peregrine	80.00	38.00	18.00	87.00	87.00	72.00
$\pi_{0.5}$ + Peregrine	91.00	58.00	32.00	87.00	88.00	71.00

(c) 4 steps / 200 ms

Method	Linear low	Linear mid	Linear high	Arc accel.	Arc decel.	Rebound
GR00T	34.00	14.00	1.00	59.00	76.00	20.00
$\pi_{0.5}$	86.00	43.00	15.00	63.00	79.00	25.00
DynamicVLA	73.00	37.00	21.00	62.00	71.00	45.00
PUMA	82.00	34.00	13.00	67.00	78.00	39.00
RTC	88.00	48.00	21.00	66.00	81.00	33.00
GR00T + Peregrine	75.00	26.00	16.00	78.00	76.00	58.00
$\pi_{0.5}$ + Peregrine	89.00	55.00	29.00	84.00	85.00	62.00

The six-condition ablation success rates at 150 ms and their equally weighted averages are reported together in Table 4.