DA-VLA: A DYNAMICS-AWARE VISION-LANGUAGE-ACTION SYSTEM FOR HIGH-SPEED MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing VLA systems remain limited in task execution speed on real robots, making it difficult to meet the throughput demands of practical applications. To improve execution throughput while reliably completing tasks, a VLA system needs to naturally connect continuously updated policy outputs with ongoing motion, preserve motion stability and execution reliability, and fully exploit the robot's dynamics. To this end, we present *DA-VLA*, a dynamics-aware VLA system for high-speed manipulation that coordinates continuous policy action generation with the robot's actual motion capabilities. On the policy side, we introduce an dynamics-aware objective that promotes smooth motion within each action chunk during training while constraining continuity between newly generated chunks and already committed motion. The same objective guides action generation during inference, allowing successive policy updates to naturally extend the motion already being executed. On the execution side, *DA-VLA* evaluates the robot's available motion capability from its current state and continuously adapts execution according to upcoming action trends, avoiding unnecessary conservative slowdowns between successive policy targets and enabling fast, stable, and accurate motion within the robot's dynamic capabilities. Further speed-success comparisons show that, at a matched task success rate, *DA-VLA* achieves nearly twice the execution throughput of $\pi_{0.5}$. These results show that *DA-VLA* can better exploit the robot's physical capabilities to improve practical execution throughput while preserving task reliability.

1 INTRODUCTION

Recent vision-language-action (VLA) policies have demonstrated strong task-level competence across diverse manipulation tasks by mapping multimodal observations and language instructions to action sequences. However, real-robot execution remains relatively slow for applications that demand high task throughput. Continuous action heads and real-time chunking approaches have improved action generation and online responsiveness through continuous action modeling and inference-execution overlap (Physical Intelligence et al., 2025; Yang et al., 2026; Black et al., 2025). But making policy outputs available sooner is not sufficient. Successive outputs must remain compatible with ongoing motion, and the robot must exploit the motion capability available from its current state without sacrificing task success. High-throughput VLA execution therefore requires coordinating continuous policy action generation with the robot's actual motion capabilities.

Such coordination is challenging because policy updates occur while the robot is moving, whereas faster task completion imposes greater dynamic demands on physical execution. Under asynchronous inference, the next action chunk is generated before the current one finishes execution and must therefore continue from motion already committed for execution. If the new chunk is inconsistent with this committed motion, the reference trajectory becomes discontinuous, increasing the corrective burden on the low-level execution. Even if local interpolation removes position-level discontinuities, independently constructed trajectory segments still do not guarantee high-order continuity across updates; abrupt changes in velocity, acceleration, or jerk can induce robot vibration and degrade continuous tracking. Meanwhile, executing the same motion over a shorter horizon requires higher velocities and accelerations, changing the torque demand relative to the robot's state-dependent actuation margins. High-throughput execution therefore requires new policy actions to naturally

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.

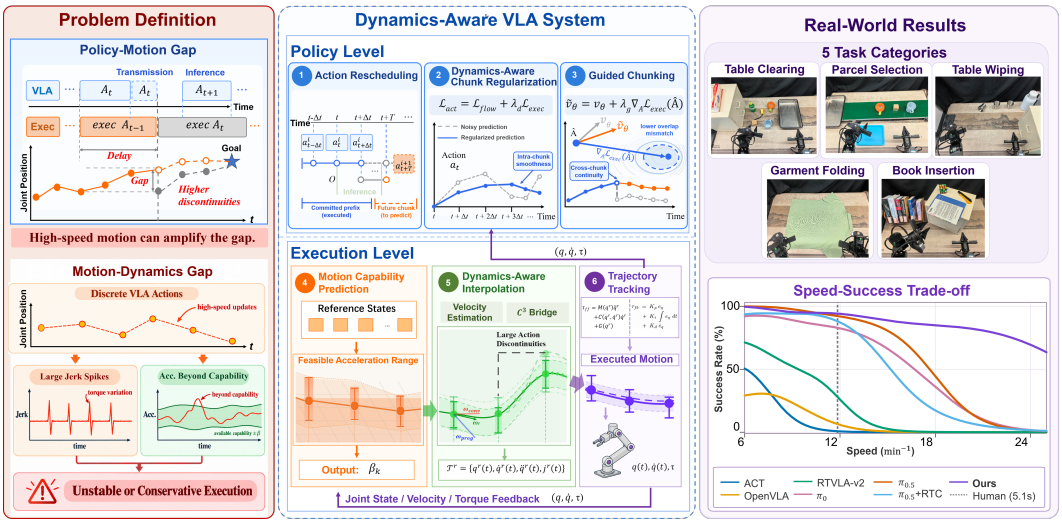


Figure 1: **Overview of *DA-VLA*.** Left: chunk transitions and state-dependent actuation demands motivate the design. Center: *DA-VLA* combines dynamics-aware action generation, asynchronous scheduling, motion capability prediction, trajectory interpolation, and tracking on *DA-Arm*. Right: five tasks and estimated local success versus reciprocal trial duration.

extend committed motion while execution adapts continuously to the robot's current state, preserving stability and accuracy as physical motion is accelerated.

To this end, we present *DA-VLA*, a dynamics-aware VLA system for high-speed manipulation that coordinates continuous policy action generation with the robot's actual motion capabilities, as illustrated in Figure 1. On the policy side, a dynamics-aware objective promotes intra-chunk motion smoothness and continuity between new chunks and committed motion during training, and guides action generation during inference. On the execution side, *DA-VLA* estimates nominal acceleration bounds from the current reference state and interpolates trajectories using subsequent action trends and inherited motion states. Dynamics-compensated tracking then follows the transmitted references with feedback correction. Together, these designs translate continuous policy updates into reliable high-throughput task execution. Our contributions are threefold:

1. **Dynamics-Guided VLA Action Chunking.** At the VLA policy level, we introduce a dynamics-aware objective on predicted clean action chunks to regularize intra-chunk smoothness and cross-chunk continuity. The same objective guides sampling during inference, linking action generation to committed motion during asynchronous execution and encouraging chunks that better respect the robot's dynamic execution capability for high-speed manipulation.
2. **Motion-Feasible Trajectory Interpolation and Tracking.** At the execution level, we propose a motion-feasible execution stack. It derives state-dependent feasible acceleration bounds from robot states and torque margins, interpolates look-ahead jerk-limited joint references within these bounds, and tracks the references with dynamics-compensated execution, thereby enabling higher joint speed and task throughput while preserving tracking quality and motion continuity.
3. **A Complete System for High-Speed VLA Execution.** We implement and evaluate the complete policy-to-execution pathway on *DA-Arm* across five real-world manipulation tasks, achieving an average SR of 80.6% and retaining 90% success at 100 Hz action-reference dispatch. Our open robotic arm platform provides the modeling and execution interfaces required for dynamics-aware VLA tuning and local execution. We will release its hardware design, execution stack, training pipeline, and model weights to support reproducible deployment.

2 Related Work

Vision-Language-Action Models and Efficient Execution. Vision-language-action (VLA) models pair vision-language backbones with discrete or continuous action heads to map observations and

instructions to robot actions (Kim et al., 2024; Octo Model Team et al., 2024; Wen et al., 2025; Shukor et al., 2025). Large-scale robot datasets support their multi-task generalization (Open X-Embodiment Collaboration et al., 2023; Walke et al., 2024; Khazatsky et al., 2025), while flow matching and diffusion provide continuous action models (Black et al., 2024; Physical Intelligence et al., 2025; Chi et al., 2024; Hou et al., 2025). Efficient execution has been pursued through both faster sampling and asynchronous policy updates. Consistency, shortcut, and mean-flow methods reduce sampling steps (Yang et al., 2024; Frans et al., 2025; Geng et al., 2025), while Recursive Flow Matching uses recursive supervision and cross-scale consistency (Huang et al., 2026). Related VLA approaches include SnapFlow, Mean-Flow based One-Step VLA, and FASTER (Luan et al., 2026; Chen et al., 2026; Lu et al., 2026). Real-time chunking and action conditioning further reduce inference-induced stalls (Black et al., 2025; Tang et al., 2025; Ma et al., 2025). We adopt efficient sampling as a component of our system and focus on connecting policy updates to committed motion and state-dependent trajectory interpolation.

Low-level Trajectory Interpolation and Execution. Low-level execution in robotic manipulators commonly relies on PID and MPC (Di Carlo et al., 2018) control with gravity compensation and trajectory interpolation. For high-speed motion, polynomial trajectory interpolation is widely adopted to constrain velocity, acceleration and jerk, improving smoothness and reducing vibration and impact. However, such controllers typically assume that the upper layer provides a continuous, executable reference trajectory, whereas VLA-based systems often output low-frequency action chunks as sparse targets. When execution is implemented by geometric interpolation followed by position tracking, the controller has limited ability to proactively adjust trajectories based on the current joint state and actuator margins, which can lead to tracking lag and conservative slowdowns during fast manipulation.

3 METHODOLOGY

We address high-speed VLA execution through a complete pipeline of dynamics-aware action generation, asynchronous chunk updates, and low-level interpolation and tracking. On the VLA policy side, the dynamics-aware objective encourages intra-chunk smoothness and cross-chunk continuity during training and inference. On the execution side, we interpolate low-rate action chunks into high-frequency continuous joint references under dynamics constraints, enabling faster physical execution while maintaining smooth and trackable motions.

3.1 FEW-STEP DYNAMICS-GUIDED VLA ACTION CHUNKING

We design a dynamics-aware VLA policy that generates action chunks suitable for high-speed execution. Built on a pretrained Qwen3-VL-4B backbone (Bai et al., 2025), our policy extracts compact task-conditioned features with learnable queries and models continuous action chunks using a scale-conditioned flow-matching action head trained with recursive trajectory supervision. A dynamics-aware objective regularizes cross-chunk continuity and intra-chunk smoothness during training and guides sampling during inference. As a result, policy learning is more closely aligned with the actual execution process, thereby reducing the execution correction burden caused by abrupt action changes during high-speed manipulation.

Given an observation o_t comprising visual inputs, language instructions, and robot state, *DA-VLA* predicts an action chunk $A_t = \{a_{t,1}, \ldots, a_{t,H}\} \in \mathbb{R}^{H \times d_a}$. Learnable feature-query tokens are appended after the visual and language tokens, and their final-layer hidden states h_q provide a fixed-length multimodal readout for the action head. We first pretrain on public robot datasets and then fine-tune on target-embodiment demonstrations. For a data source r, *DA-VLA* predicts an action chunk of length H in its native action representation, with dimension d_a^r. During pretraining, source-specific state and action interfaces connect native representations to the shared VLM and action-head feature space. Target adaptation combines action learning with a language-modeling objective to preserve multimodal understanding.

Dynamics-Aware Action Generation Training We model continuous action chunks using a flow-matching-based action head. Given the target action sequence A_t, we sample Gaussian noise

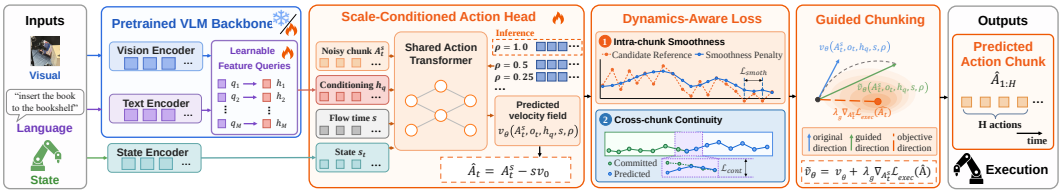


Figure 2: **Dynamics-aware action generation and guided sampling.** Visual, language, and robot-state inputs condition a continuous action head. Dynamics-aware regularization is evaluated on the predicted clean chunk during target adaptation, and the same objective guides inference to promote intra-chunk smoothness and continuity with committed motion. Recursive multi-scale supervision is used during training; inference uses three sampling steps at the primary scale.

$\varepsilon \sim \mathcal{N}(0, I)$ and construct a noise-perturbed action chunk A_t^s at flow matching timestep $s \in [0, 1]$ as

$$A_t^s = (1 - s)A_t + s\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I) \tag{1}$$

Conditioned on o_t and h_q, the action head predicts a velocity field from the noisy chunk, flow time, and scale condition. At the primary scale, the standard flow-matching objective is

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{t,s,\epsilon}\left[\text{MSE}\left(v_\theta(A_t^s, o_t, h_q, s, 1), \epsilon - A_t\right)\right]. \tag{2}$$

$$v_{\theta,r}(A_t^s, o_t, h_q, s, \rho) = D_r^a(F_\theta(E_r^a(A_t^s), E_r^x(x_t), h_q, s, \rho)) \in \mathbb{R}^{H \times d_a^r}. \tag{3}$$

Here, E_r^a and E_r^x are source-specific action and state encoders, D_r^a decodes shared features into the native velocity field, and F_θ is the shared action transformer. The robot state x_t is included in o_t; MSE is computed over valid action entries. Perturbation and flow supervision remain in the native action space of dimension d_a^r. We omit the source index below.

To support few-step sampling, we adapt Recursive Flow Matching (Huang et al., 2026) to the action head. For $R > 1$ training scales, we construct recursively rescaled time-scale pairs that share the same noisy action:

$$\rho_i = \rho^i, \qquad s_i = s/\rho_i, \qquad v_i = v_\theta(A_t^s, o_t, h_q, s_i, \rho_i), \quad i = 0, \ldots, R-1. \tag{4}$$

Here, ρ is the base scale factor, i indexes the R training scales, and s_i is the rescaled flow time. The primary scale is $\rho_0 = 1$, and the supervised velocity target is $v^\star = \epsilon - A_t$. The training combines velocity supervision and cross-scale consistency, whereas inference uses the primary scale. The corresponding clean estimates are

$$\widehat{A}_t^{(i)} = A_t^s - s_i v_i, \qquad \widehat{A}_t = \widehat{A}_t^{(0)} = \{\widehat{a}_{t,k}\}_{k=1}^H. \tag{5}$$

However, cross-scale agreement does not ensure smooth asynchronous execution: independently generated chunks can introduce abrupt transitions and intra-chunk oscillations. We therefore regularize the primary clean estimate $\hat{A}_t$ using the robot model. During target adaptation, the primary clean-action estimate provides joint-position references $\hat{q}_{t,k} = g_r(\hat{a}_{t,k})$, where g_r specifies the target joint representation. Joint velocity and acceleration are estimated by finite differences:

$$\hat{\dot{q}}_{t,k} = \frac{\hat{q}_{t,k} - \hat{q}_{t,k-1}}{\Delta t}, \qquad \hat{\ddot{q}}_{t,k} = \frac{\hat{\dot{q}}_{t,k} - \hat{\dot{q}}_{t,k-1}}{\Delta t}. \tag{6}$$

Here, Δt is the action-reference interval: $0.02\,\text{s}$ during training and $1/f_{\text{dispatch}}$ during inference, where f_{dispatch} is the action-reference dispatch rate. The corresponding nominal torque is evaluated using the inverse-dynamics model in Section 3.2. Using these candidate references and nominal torques, we define the dynamics-aware regularizer as

$$\mathcal{L}_{\text{exec}} = \mathcal{L}_{\text{smooth}} + \lambda_c \mathcal{L}_{\text{cont}}, \tag{7}$$

$$\mathcal{L}_{\text{smooth}} = \sum_{k=3}^{H} \left\| \hat{q}_{t,k} - 2\hat{q}_{t,k-1} + \hat{q}_{t,k-2} \right\|^2, \tag{8}$$

$$\mathcal{L}_{\text{cont}} = \sum_{k=1}^{H} \eta_{k-1} \left[\alpha_q \|\widehat{q}_{t,k} - q_{t,k}^{\text{ref}}\|^2 + \mathbf{1}_{k \le d_1} \left(\alpha_{\dot{q}} \|\hat{\dot{q}}_{t,k} - \dot{q}_{t,k}^{\text{ref}}\|^2 + \alpha_\tau \|\widehat{\tau}_{t,k} - \tau_{t,k}^{\text{ref}}\|^2 \right) \right]. \tag{9}$$

$$\eta_k = \mathbf{1}_{k < d_1} + \mathbf{1}_{d_1 \le k < d_1 + d_2}\, c_k \frac{e^{c_k} - 1}{e - 1}, \quad c_k = \frac{d_1 + d_2 - k}{d_2 + 1}, \quad k \in \{0, \ldots, H-1\}. \tag{10}$$

Here, $\mathcal{L}_{\text{smooth}}$ and $\mathcal{L}_{\text{cont}}$ are the intra-chunk smoothness and cross-chunk continuity losses, and λ_c weights the continuity term. The hatted quantities describe the predicted chunk, while $q_{t,k}^{\text{ref}}$, $\dot{q}_{t,k}^{\text{ref}}$, and $\tau_{t,k}^{\text{ref}}$ denote the reference quantities within the committed overlap. During training, the reference prefix uses recorded actions preceding the sampled frame, with its length d_1 randomly sampled from $\{0, \ldots, 20\}$. The RTC-inspired (Black et al., 2025) mask η_k assigns unit weight to the committed prefix of length d_1, decaying position-consistency weights to the subsequent d_2 steps, and zero weight to the remaining tail, with $d_1 + d_2 < H$. The position reference beyond the prefix is obtained by holding its terminal action. Velocity and nominal-torque consistency are evaluated only within the committed overlap. The symbol $\mathbf{1}$ denotes an indicator, and c_k parameterizes the decay within the transition region. The coefficients α_q, $\alpha_{\dot{q}}$, and α_τ balance these terms, while $\mathcal{L}_{\text{smooth}}$ penalizes second-order position differences throughout the predicted chunk. The joint action-training objective combines flow supervision, cross-scale consistency, and dynamics-aware regularization:

$$\mathcal{L}_{\text{act}} = \mathbb{E}\left[\frac{1}{R}\sum_{i=0}^{R-1}\text{MSE}(v_i, \rho_i v^\star) + \frac{\lambda_{\text{cs}}}{R-1}\sum_{i=1}^{R-1}\frac{\text{MSE}(\widehat{A}_t^{(i)}, \widehat{A}_t)}{s_i^2} + \lambda_d \mathcal{L}_{\text{exec}}(\widehat{A}_t)\right]. \tag{11}$$

The terms supervise action generation, enforce cross-scale consistency, and regularize execution, with weights λ_{cs} and λ_d for the latter two. The expectation is over training samples, flow times, and noise. For $s_i > 0$, the normalized clean-action residual equals the corresponding velocity-consistency residual. The same dynamics-aware objective guides inference.

Dynamics-Aware Guided Inference Sampling During inference, the same dynamics-aware objective guides sampling to encourage continuity with committed motion. We evaluate it on the primary clean estimate $\hat{A}_t = A_t^s - sv_0$ and modify the velocity field as

$$\widetilde{v}_\theta = v_0 + \lambda_g \nabla_{A_t^s}\mathcal{L}_{\text{exec}}(A_t^s - sv_0), \qquad v_0 = v_\theta(A_t^s, o_t, h_q, s, 1). \tag{12}$$

$$\text{where} \quad \lambda_g = \min\left(c, \frac{1-s}{s \cdot r_s^2}\right), r_s^2 = \frac{(1-s)^2}{s^2 + (1-s)^2}. \tag{13}$$

Here, v_0 and $\widetilde{v}_\theta$ are the primary and guided velocity fields, and the gradient is taken with respect to A_t^s. The factor r_s^2 defines the time-dependent guidance schedule, capped by c with limiting value $\lambda_g(1) = c$. This coefficient follows pseudoinverse guidance used in training-free inpainting and real-time action chunking. Integrating from $s = 1$ to $s = 0$ makes the positive gradient term a descent-directed execution correction. We use uniform reverse-time Euler updates:

$$A_t^{s_{j+1}} = A_t^{s_j} - \frac{1}{K}\tilde{v}_\theta(A_t^{s_j}, o_t, h_q, s_j), \qquad s_j = 1 - \frac{j}{K}, \quad j = 0, \ldots, K-1. \tag{14}$$

Here, j indexes the K sampling steps and s_j is the flow time, with $K = 3$ by default. The committed motion provides a soft reference during asynchronous inference; at the scheduled handoff, execution resumes after the new chunk's committed prefix.

3.2 Motion-Feasible Trajectory Interpolation and Tracking

Dynamics-aware action generation promotes smooth and consistent chunks but does not by itself enforce state-dependent actuation limits during tracking. We therefore introduce a motion-feasible local execution layer that converts the low-rate action chunk A_t into a high-frequency reference trajectory $\{\mathbf{q}_k^r, \mathbf{q}_k{}^r, \ddot{\mathbf{q}}_k{}^r\}$ for closed-loop execution. The framework consists of three coupled components: motion capability prediction estimates state-dependent acceleration bounds from actuation margins, dynamics-aware interpolation reconstructs jerk-limited references under these constraints, and dynamics-compensated trajectory tracking realizes the generated trajectory while feeding executed states back for online updates.

Motion Capability Prediction. Policy-generated motions need to satisfy the physical constraints of the robot. Given the current reference state $\mathbf{x}_k^r = [\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r, \ddot{\mathbf{q}}_k^r, \dddot{\mathbf{j}}_k^r]^T$, Motion capability prediction establishes the relationship between joint torque and acceleration through robot dynamics:

$$\boldsymbol{\tau}_k = M(\mathbf{q}_k^r)\ddot{\mathbf{q}}_k + h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r). \tag{15}$$

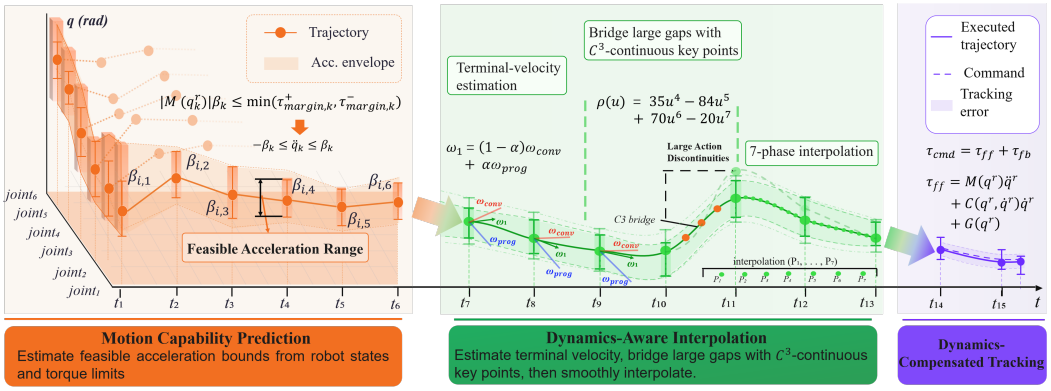


Figure 3: **Local execution framework for motion-feasible trajectory interpolation and tracking.** Given low-frequency action chunks from the VLA policy, the motion capability prediction estimates the state-dependent acceleration envelope $\boldsymbol{\beta}_k$ from the current robot state and torque margins. Dynamics-aware interpolation reconstructs high-frequency jerk-limited reference trajectories $\mathcal{T}^r = \{q^r(t), \dot{q}^r(t), \ddot{q}^r(t), j^r(t)\}$ under these dynamic constraints. A dynamics-compensated controller executes the interpolated trajectory, with the measured robot states fed back for subsequent trajectory interpolation and policy execution.

where $M(\mathbf{q}_k^r)$ is the joint-space inertia matrix and $h(\mathbf{q}_k^r, \dot{\mathbf{q}}k^r)$ collects the nonlinear dynamics. Under actuator limits and safety margins, the admissible torque satisfies $\boldsymbol{\tau}_{\text{safe}}^- \le \boldsymbol{\tau}_k \le \boldsymbol{\tau}_{\text{safe}}^+$. Combining the dynamics and torque constraints yields the feasible acceleration set:

$$\mathcal{A}_k = \left\{ \ddot{\mathbf{q}} \mid \boldsymbol{\tau}_{\text{safe}}^- \le M(\mathbf{q}_k^r)\ddot{\mathbf{q}} + h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r) \le \boldsymbol{\tau}_{\text{safe}}^+ \right\}. \tag{16}$$

Since $\mathcal{A}_k$ couples all joints, motion capability prediction extracts a joint-wise acceleration envelope $\boldsymbol{\beta}_k$ for online trajectory generation. The available torque margins are defined as

$$\boldsymbol{\tau}_{\text{margin},k}^{\pm} = \pm(\boldsymbol{\tau}_{\text{safe}}^{\pm} - h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r)). \tag{17}$$

The envelope satisfies

$$|M(\mathbf{q}_k^r)|\,\boldsymbol{\beta}_k \le \min(\boldsymbol{\tau}_{\text{margin},k}^+, \boldsymbol{\tau}_{\text{margin},k}^-), \tag{18}$$

and is obtained by

$$\max_{\boldsymbol{\beta}_k} \sum_i w_i \log(\beta_{i,k} + \epsilon), \tag{19}$$

subject to the above constraint. The resulting envelope $\boldsymbol{\beta}_k$ is provided to dynamics-aware interpolation for trajectory interpolation.

Dynamics-Aware Interpolation. Given the acceleration envelope $\boldsymbol{\beta}_k$ from motion capability prediction, dynamics-aware interpolation reconstructs continuous jerk-limited reference trajectories from discrete VLA action targets under the robot's instantaneous dynamic capability.

Since VLA policies update actions in discrete chunks, directly connecting consecutive targets may introduce velocity discontinuities during execution. Dynamics-aware interpolation therefore estimates transition states using a local temporal window, where (q_0, ω_0) denotes the current reference state, q_1 denotes the target, and q_2 denotes a preview target from the following action point or future chunk.

Using this temporal window and the dynamic constraint $\beta_{i,k}$, dynamics-aware interpolation determines the terminal velocity for trajectory interpolation:

$$\omega_{1,i}^{\text{raw}} = (1 - \alpha_i)\omega_{\text{conv},i} + \alpha_i \omega_{\text{prog},i}. \tag{20}$$

Here, $\omega_{\text{conv},i}$ follows the future motion tendency indicated by the preview target q_2, while $\omega_{\text{prog},i}$ drives the state toward the current target q_1 according to the acceleration capability $\beta_{i,k}$ estimated by motion capability prediction. The coefficient α_i balances future progression and target convergence, and the resulting terminal velocity defines the boundary condition for subsequent trajectory generation.

With the boundary condition obtained, dynamics-aware interpolation densifies large target transitions with intermediate reference points and connects them to the inherited reference state through a C^3-continuous bridge.

$$\rho(u) = 35u^4 - 84u^5 + 70u^6 - 20u^7, \tag{21}$$

where u denotes the normalized transition variable. During the transition, the generated acceleration is constrained by the dynamic envelope estimated by motion capability prediction:

$$\underline{\ddot{q}}_{i,k} \le \ddot{q}_{i,k} \le \overline{\ddot{q}}_{i,k}. \tag{22}$$

The resulting transition states are then parameterized by jerk-limited seven-phase interpolation to obtain the high-frequency reference trajectory:

$$\mathcal{T}^r = \{q^r(t), \dot{q}^r(t), \ddot{q}^r(t), j^r(t)\}. \tag{23}$$

Appendix B provides detailed derivations of terminal velocity estimation and trajectory construction.

Dynamics-Compensated Trajectory Tracking The interpolated trajectory $\mathcal{T}^r$ serves as the reference for the low-level execution, where $(q, \dot{q})$ denotes the measured feedback state. The commanded torque is computed using inverse-dynamics feedforward and feedback compensation:

$$\tau_{\text{cmd}} = \tau_{\text{ff}} + \tau_{\text{fb}}, \tag{24}$$

$$\tau_{\text{ff}} = M(q^r)\ddot{q}^r + C(q^r, \dot{q}^r)\dot{q}^r + G(q^r), \tag{25}$$

$$\tau_{\text{fb}} = K_p e_q + K_d \dot{e}_q, \qquad e_q = q^r - q, \qquad \dot{e}_q = \dot{q}^r - \dot{q}. \tag{26}$$

where $M(q^r)$, $C(q^r, \dot{q}^r)\dot{q}^r$, and $G(q^r)$ denote the inertia, Coriolis/centrifugal, and gravity terms of the manipulator dynamics. The feedforward term compensates nominal dynamics along the reference trajectory, while feedback corrects residual tracking errors and disturbances. The executed joint states are fed back to the execution layer for subsequent feasibility estimation and to the policy layer as part of the robot state.

4 Experiments

4.1 Setup and metrics

Platform and tasks. All real-robot experiments use *DA-Arm*, our task-adapted 6+1-DoF arm with joint position, velocity, and torque feedback and low-level control access. Appendix C describes the hardware and robot-model interface, and Appendix D describes more execution protocols. We evaluate five tasks spanning general manipulation, bimanual deformable-object handling, contact-rich execution, and dynamic interception.

Baselines. We compare with ACT (Zhao et al., 2023), OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024), and $\pi_{0.5}$ (Physical Intelligence et al., 2025) using the same target-task demonstration data as *DA-VLA* and the same task and evaluation protocols. Other deployment settings are kept consistent wherever compatible with the methods' native interfaces. $\pi_{0.5}$+RTC (Black et al., 2025) and RealtimeVLA-v2 (RTVLA-V2) (Yang et al., 2026) provide comparisons with policy-side real-time scheduling and inference-execution overlap.

Metrics. We report success rate (SR) and execution throughput for evaluation, defined as:

$$\text{SR} \triangleq \frac{N_{\text{success}}}{N}, \quad \text{Throughput} \triangleq \frac{N_{\text{success}}}{\sum_{i=1}^{N} T_i}, \tag{27}$$

where N is total trials, N_{success} is successful trials, and T_i is trial duration in minutes. For failed trials, we set $T_i = T_{\max} = \lambda \frac{1}{N_{\text{success}}} \sum_{i:\text{success}_i=1} T_i$, with $\lambda = 1.5$, fixed before final aggregation, and set execution throughput to zero when no trial succeeds. Thus, execution throughput is a failure-penalized throughput metric expressed in successful completions per minute. For aggregate reporting, each task metric is min–max normalized across the methods compared on each task and then averaged across the five-task suite; *Parcel Selection* uses SR as its task metric. Trial counts, normalization, and motor-level metrics are detailed in Appendix D.

Table 1: **Performance across five real-world tasks.** For *Table Clearing*, we evaluate action-reference dispatch at 30, 50, and 100 Hz, together with dual-object and out-of-distribution (OOD) settings. Entries report SR (%) / execution throughput, except the OOD settings and *Parcel Selection*, which report SR only. The final row reports average SR and normalized execution throughput over the standard task settings. Higher is better.

Setting	ACT	OpenVLA	RTVLA-v2	π_0	$\pi_{0.5}$	$\pi_{0.5}$+RTC	Ours
Slow Execution (30 Hz)	30% / 1.07	20% / 0.95	52% / 2.95	98% / 8.20	98% / 8.46	82% / 8.28	**100% / 8.93**
Nominal Execution (50 Hz)	26% / 1.75	16% / 0.76	40% / 1.92	62% / 4.99	**96%** / 10.37	80% / 8.39	92% / **10.76**
Accelerated Execution (100 Hz)	0% / 0.00	10% / 0.58	26% / 2.07	60% / 6.55	78% / 8.20	60% / 6.28	**90% / 15.92**
Dual-object Operation	0% / 0.00	0% / 0.00	20% / 0.71	50% / 2.84	**90% / 5.91**	70% / 4.39	80% / 5.20
Unseen Placement Position [OOD]	0%	10%	10%	40%	40%	30%	**60%**
Multi-object Operation [OOD]	0%	0%	0%	0%	0%	0%	**20%**
Open-vocabulary Instruction [OOD]	–	0%	0%	50%	40%	40%	**60%**
Garment Folding	0% / 0.00	0% / 0.00	0% / 0.00	45% / 1.02	**80%** / 1.69	50% / 0.91	**80% / 1.84**
Table Wiping	45% / 0.88	10% / 0.17	5% / 0.12	55% / 1.43	70% / **3.16**	65% / 2.33	**80%** / 3.03
Book Insertion	0% / 0.00	5% / 0.05	0% / 0.00	15% / 0.31	55% / 1.29	50% / 1.11	**65% / 1.59**
Parcel Selection	4%	0%	0%	54%	**86%**	76%	**86%**
Average	15.0% / 0.079	6.2% / 0.010	9.0% / 0.023	46.2% / 0.446	77.4% / 0.938	64.2% / 0.713	**80.6% / 0.991**

Table 2: **Feedback-level analysis of the local execution framework.** Metrics are computed from transmitted references and real-robot feedback under identical execution settings. Tracking error measures the deviation between measured robot states and interpolated reference trajectories, while acceleration and jerk characterize motion smoothness and continuity.

Execution Strategy	Tracking Error (rad) ↓	Acc. RMS (rad/s^2) ↓	Jerk RMS (rad/s^3) ↓	Lag (ms)↓
Linear Interpolation	0.0088	11.10	1522.97	95.7
MinJerk	0.0102	54.46	10361.77	**10.0**
Ours	0.0085	**6.25**	**935.78**	76.6
w/o Chunk Fusion	0.0095	7.38	1002.21	78.7
w/o Inheritance	**0.0084**	6.69	1112.88	173.2

4.2 Main task results

Table 1 evaluates task-level success and throughput. It reports success rate / execution throughput over 50 trials per method at each dispatch rate, 10 trials for dual-object operation, and success rate over 10 trials for each OOD condition. *DA-VLA* achieves success rate / execution throughput of 100%/8.93, 92%/10.76, and 90%/15.92 at 30, 50, and 100 Hz, respectively. At 50 Hz, $\pi_{0.5}$ achieves higher success rate (96%), while *DA-VLA* still achieves higher execution throughput (10.76 versus 10.37). At 100 Hz, *DA-VLA* achieves the highest success rate and execution throughput among the evaluated methods. These comparisons establish system-level performance. Across the three OOD settings, *DA-VLA* achieves the highest success rate under unseen placement, multi-object, and open-vocabulary variations, showing that the complete system retains task performance under these spatial and language variations.

For other tasks, the first three tasks use 20 trials per method, while *Parcel Selection* uses 50 trials. The five-task average includes *Table Clearing* at 50 Hz and reports mean SR and task-wise min-max normalized execution throughput. Across these evaluations, *DA-VLA* achieves 80.6%/0.991, compared with 77.4%/0.938 for $\pi_{0.5}$ and 64.2%/0.713 for $\pi_{0.5}$+RTC. It achieves the highest or tied-highest SR on all four tasks, with the highest execution throughput on *Garment Folding* and *Book Insertion*. These results establish a strong aggregate speed-success trade-off across continuous, contact-rich, and dynamic manipulation tasks.

4.3 Trajectory Interpolation Analysis

Execution-level experiments verify that the proposed lower-level execution framework can reliably convert high-speed action chunks into continuous and executable robot motions. We evaluate its performance from three aspects: tracking accuracy, motion smoothness, and execution responsiveness.

Table 2 compares the execution performance of different trajectory interpolation strategies, where Linear interpolation and MinJerk represent direct action interpolation and segment-wise smooth

Table 3: **Module-level ablation of the *DA-VLA* policy-to-execution pathway.** The full method uses 3 sampling steps; the retrained variant without RecFM uses 10. Avg. reports five-task mean SR and task-wise normalized execution throughput within the five variants. Ablation configurations are described in Appendix D.

Variant	Table Clearing	Garment Folding	Table Wiping	Book Insertion	Parcel Selection	Avg.
Full method	**92% / 10.76**	**80% / 1.84**	80% / 3.03	**65% / 1.59**	86%	**80.6% / 0.976**
w/o RecFM (10 steps)	**92%** / 10.46	75% / 1.50	75% / 2.81	**65%** / 1.55	**88%**	79.0% / 0.867
w/o motion-feasible interpolation	80% / 8.85	60% / 1.33	70% / 2.23	60% / 1.32	84%	70.8% / 0.646
w/o asynchronous chunking	82% / 6.51	70% / 1.06	**85% / 3.19**	**65%** / 1.10	0%	60.4% / 0.387
w/o dynamics-aware regularization and guidance	64% / 5.92	45% / 0.81	60% / 1.58	30% / 0.45	60%	51.8% / 0.136

trajectory generation, respectively. The proposed execution framework maintains comparable tracking error while reducing the feedback acceleration RMS and jerk RMS by 43.7% and 38.6% compared with Linear, and by 88.5% and 91.0% compared with MinJerk, demonstrating that continuously updated VLA action chunks require both trajectory smoothness and temporal state consistency.

Ablation studies further verify the contribution of individual components. Removing Chunk Fusion increases the acceleration and jerk RMS by 18.1% and 7.1%, respectively, while removing Inheritance increases the alignment lag by 126.2%, highlighting the importance of cross-chunk information fusion and state preservation for high-speed continuous execution.

4.4 Ablation Study

Table 3 evaluates four components under matched per-task data and trial protocols. The full method achieves the strongest aggregate success rate and normalized execution throughput of 80.6% and 0.976. Removing dynamics-aware regularization and guidance causes the largest aggregate degradation, while removing motion capability prediction and dynamics-aware interpolation reduces execution throughput across all four tasks reporting this metric. Without asynchronous chunking, *Parcel Selection* fails completely, although *Table Wiping* improves slightly. These results support the contributions of the combined dynamics-aware regularization and guidance, asynchronous updates, and state-dependent interpolation.

The retrained variant without RecFM uses ten sampling steps instead of three, with complete inference latency increasing from approximately 60 to 160 ms. Its mean success rate remains close to that of the full method (79.0% versus 80.6%), with at most one successful trial of difference per task, while execution throughput decreases on all four tasks reporting this metric. This comparison supports the use of RecFM for inference efficiency with comparable observed task success under the evaluated sampling configurations.

5 Conclusion and Limitations

This work proposes *DA-VLA*, a dynamics-aware system for high-speed VLA execution combining few-step action generation, dynamics-aware asynchronous chunk updates, state-dependent trajectory interpolation, and dynamics-compensated tracking on *DA-Arm*. Across five real-world tasks, our method achieves the highest observed aggregate success rate and normalized execution throughput of 80.6% and 0.991 among the compared methods. Motor-log analysis shows lower feedback acceleration and jerk RMS than the interpolation baselines while maintaining comparable tracking error. The open platform supports reproduction of the complete policy-to-execution pathway.

The current evaluation covers diverse tabletop tasks, dynamic parcel interception, and direct motor-level execution analysis. The analytical model serves as a nominal feasibility prior together with safety margins and feedback correction; large structural or load changes require updating the robot-specific configuration. The embodiment-parameterized interface provides a concrete deployment path for compatible manipulators, while empirical zero-shot cross-robot transfer remains to be evaluated.

AI USE STATEMENT

In this work, we use generative AI tools to assist with language polishing, grammar correction, and improving the clarity and readability of the manuscript. All methodological decisions, implementations, experiments, analyses, and reported results were verified. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025.

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, 2024. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies, 2025. URL https://arxiv.org/abs/2506.07339.

Yang Chen, Xiaoguang Ma, and Bin Zhao. Mean-flow based one-step vision-language-action, 2026. URL https://arxiv.org/abs/2603.01469.

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, 2024. URL https://arxiv.org/abs/2303.04137.

Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, 2018. doi: 10.1109/IROS.2018.8594448. URL https://ieeexplore.ieee.org/document/8594448.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2025. URL https://arxiv.org/abs/2410.12557.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Hengjun Pu, Chengyang Zhao, Ronglei Tong, Yu Qiao, Jifeng Dai, and Yuntao Chen. Diffusion transformer policy, 2025. URL https://arxiv.org/abs/2410.15959.

Jiahe Huang, Sihan Xu, Sharvaree Vadgama, and Rose Yu. Recursive flow matching, 2026. URL https://arxiv.org/abs/2605.26535.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon

Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

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

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. Faster: Rethinking real-time flow vlas, 2026. URL https://arxiv.org/abs/2603.19199.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. Snapflow: One-step action generation for flow-matching vlas via progressive self-distillation, 2026. URL https://arxiv.org/abs/2604.05656.

Yunchao Ma, Yizhuang Zhou, Yunhuan Yang, Tiancai Wang, and Haoqiang Fan. Running vlas at real-time speed, 2025. URL https://arxiv.org/abs/2510.26742.

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

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen,

Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

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, 2025. URL https://arxiv.org/abs/2504.16054.

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

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, 2025. URL https://arxiv.org/abs/2512.01031.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale, 2024. URL https://arxiv.org/abs/2308.12952.

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.

Chen Yang, Yucheng Hu, Yunchao Ma, Yunhuan Yang, Jing Tan, and Haoqiang Fan. Realtime-vla v2: Learning to run vlas fast, smooth, and accurate, 2026. URL https://arxiv.org/abs/2603.26360.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency, 2024. URL https://arxiv.org/abs/2407.02398.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

A Coupling recursive flow matching with execution regularization

For the shared noisy action $A_t^s = (1-s)A_t + s\epsilon$, let $v^\star = \epsilon - A_t$, $\rho_i = \rho^i$, and $s_i = s/\rho_i$. The scales are chosen so that the rescaled times remain within the training interval. With $v_i = v_\theta(A_t^s, o_t, h_q, s_i, \rho_i)$, the recursive flow-matching objective is

$$\mathcal{L}_{\mathrm{RecFM}} = \mathbb{E}\left[\frac{1}{R}\sum_{i=0}^{R-1}\mathrm{MSE}(v_i, \rho_i v^\star) + \frac{\lambda_{\mathrm{cs}}}{R-1}\sum_{i=1}^{R-1}\mathrm{MSE}(v_i, \rho_i v_0)\right]. \tag{28}$$

Define the clean action estimate at each scale as $\widehat{A}_t^{(i)} = A_t^s - s_i v_i$, with $\widehat{A}_t = \widehat{A}_t^{(0)}$. Since $s_i\rho_i = s$, we obtain

$$\widehat{A}_t^{(i)} - \widehat{A}_t = -s_i(v_i - \rho_i v_0). \tag{29}$$

Therefore, for $s_i > 0$ and the same valid-action mask,

$$\mathrm{MSE}(v_i, \rho_i v_0) = \frac{1}{s_i^2}\,\mathrm{MSE}(\widehat{A}_t^{(i)}, \widehat{A}_t). \tag{30}$$

Substituting this identity and adding $\lambda_d\mathcal{L}_{\mathrm{exec}}(\widehat{A}_t)$ yields Eq. 11. The clean-action form exposes the shared prediction used by cross-scale consistency and dynamics-aware regularization without changing the velocity-consistency weighting. For numerical evaluation, the original velocity residual avoids explicit division by s_i^2 near zero.

Although demonstrations provide executable task intent at the collection time scale, the generated chunks require extra continuity during asynchronous execution. The dynamics-aware objective therefore regularizes predicted clean chunks rather than recorded demonstrations.

Dynamics-aware regularization is evaluated on the primary estimate only. Its position reference extends beyond the committed overlap by terminal holding, whereas velocity and nominal-torque consistency are restricted to the true overlap. During inference, only the primary scale is evaluated, and the same dynamics-aware objective provides the correction in Eq. 12.

B Additional control derivations

VLA policies typically generate discrete joint targets at a relatively low frequency, whereas the low-level controller requires high-frequency reference states that are motion-feasible and temporally smooth. Direct interpolation between consecutive VLA targets with fixed motion limits is insufficient for real-world execution, since the available motion capability of a manipulator varies with its configuration and velocity. The inertial coupling, Coriolis and centrifugal effects, gravity terms, and remaining motor torque margin jointly determine the acceleration range that can be safely applied at the current state. Meanwhile, consecutive VLA targets may correspond to different local motion trends. Treating each target as an independent static endpoint may introduce unnecessary deceleration during continuous motion, while excessive pass-through velocity near a directional change can increase the subsequent tracking difficulty.

To address these challenges, we formulate trajectory interpolation as a two-stage process consisting of motion capability prediction and dynamics-aware interpolation. Motion capability prediction estimates the instantaneous dynamic capability of the manipulator and provides a state-dependent acceleration boundary for subsequent trajectory generation. Given the current reference state, motion capability prediction considers the complete rigid-body dynamics and the available motor torque margin to construct a feasible joint-wise acceleration envelope. This envelope preserves the coupling information contained in the full mass matrix while providing an explicit acceleration constraint for each joint.

Based on the dynamic capability provided by motion capability prediction, dynamics-aware interpolation determines the temporal evolution between consecutive VLA targets. Instead of treating

every target as a zero-velocity terminal state, dynamics-aware interpolation estimates an appropriate terminal velocity according to the current reference state, the current target waypoint, and the preview waypoint. The estimated terminal velocity balances motion progress and local convergence while remaining consistent with the available dynamic capability. Subsequently, dynamics-aware interpolation generates a fixed-horizon C^3-continuous trajectory under the prediction-derived acceleration boundary. The generated trajectory continuously inherits position, velocity, acceleration, and jerk states across consecutive replanning cycles.

Let i denote the joint index, k denote the high-frequency execution cycle, and ℓ denote the VLA update index. The reference state maintained by the controller at cycle k is

$$\mathbf{x}_k^r = [\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r, \ddot{\mathbf{q}}_k^r, \mathbf{j}_k^r]^{\mathrm{T}}, \tag{31}$$

where $\mathbf{q}_k^r$, $\dot{\mathbf{q}}_k^r$, $\ddot{\mathbf{q}}_k^r$, and $\mathbf{j}_k^r$ denote the reference joint position, velocity, acceleration, and jerk, respectively.

The overall interpolation process can be summarized as

$$\mathbf{x}_k^r, \boldsymbol{\theta}_\ell, M(\mathbf{q}_k^r) \rightarrow \boldsymbol{\beta}_k \rightarrow \boldsymbol{\omega}_1 \rightarrow \{\mathbf{q}^r, \dot{\mathbf{q}}^r, \ddot{\mathbf{q}}^r, \mathbf{j}^r\}, \tag{32}$$

where $\boldsymbol{\theta}_\ell$ denotes the target sequence generated by the VLA policy, $M(\mathbf{q}_k^r)$ denotes the current joint-space mass matrix, $\boldsymbol{\beta}_k$ denotes the acceleration envelope estimated by motion capability prediction, and $\boldsymbol{\omega}_1$ denotes the terminal velocity estimated by dynamics-aware interpolation.

Therefore, motion capability prediction determines the dynamically admissible motion capability at the current robot state, while dynamics-aware interpolation determines how to continuously transition between VLA targets within this capability boundary. Together, they convert low-frequency discrete VLA commands into high-frequency reference trajectories respecting nominal motion constraints and maintaining high-order temporal continuity.

B.1 Motion Capability Prediction

The executable acceleration capability of a manipulator is determined by its instantaneous dynamics rather than a fixed acceleration limit. Given the reference joint state at execution cycle k, the rigid-body dynamics are written as

$$\boldsymbol{\tau}_k = M(\mathbf{q}_k^r)\ddot{\mathbf{q}}_k + h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r), \tag{33}$$

where $\boldsymbol{\tau}_k \in \mathbb{R}^n$ denotes the joint torque vector, $M(\mathbf{q}_k^r) \in \mathbb{R}^{n\times n}$ is the joint-space mass matrix, and

$$h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r) = C(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r)\dot{\mathbf{q}}_k^r + g(\mathbf{q}_k^r) \tag{34}$$

represents the velocity-dependent and gravity-induced torque terms. Here, $C(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r)$ denotes the Coriolis and centrifugal matrix, and $g(\mathbf{q}_k^r)$ denotes the gravity torque.

In practical execution, the entire motor torque capability cannot be allocated to trajectory generation, since part of the torque capacity must be reserved for feedback stabilization, model mismatch, and external disturbances. Therefore, we define the available torque interval as

$$\boldsymbol{\tau}_{\mathrm{safe}}^- = -\rho_\tau \boldsymbol{\tau}_{\mathrm{rated}} + \boldsymbol{\tau}_{\mathrm{reserve}}, \tag{35}$$

$$\boldsymbol{\tau}_{\mathrm{safe}}^+ = \rho_\tau \boldsymbol{\tau}_{\mathrm{rated}} - \boldsymbol{\tau}_{\mathrm{reserve}}, \tag{36}$$

where $\rho_\tau \in (0, 1)$ denotes the usable torque ratio, $\boldsymbol{\tau}_{\mathrm{rated}}$ is the rated torque vector, and $\boldsymbol{\tau}_{\mathrm{reserve}}$ is the reserved torque margin.

Substituting the rigid-body dynamics into the torque constraint gives the feasible acceleration set at cycle k:

$$\mathcal{A}_k = \left\{ \ddot{\mathbf{q}} \middle| \boldsymbol{\tau}_{\text{safe}}^- \leq M(\mathbf{q}_k^r)\ddot{\mathbf{q}} + h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r) \leq \boldsymbol{\tau}_{\text{safe}}^+ \right\}. \tag{37}$$

Although $\mathcal{A}_k$ provides the complete coupled acceleration constraint, it is not directly suitable for high-frequency trajectory interpolation because dynamics-aware interpolation requires an explicit joint-wise acceleration boundary. Therefore, motion capability prediction constructs an axis-aligned acceleration envelope inside the feasible set:

$$-\boldsymbol{\beta}_k^- \leq \ddot{\mathbf{q}} \leq \boldsymbol{\beta}_k^+, \tag{38}$$

where $\boldsymbol{\beta}_k^+$ and $\boldsymbol{\beta}_k^-$ denote the positive and negative acceleration capability vectors, respectively.

To preserve the dynamic coupling among different joints, the envelope computation is performed using the full mass matrix rather than independent single-joint limits. Considering the acceleration directions that increase and decrease each joint velocity, the available torque margin is defined as

$$\boldsymbol{\tau}_{\text{margin},k}^+ = \boldsymbol{\tau}_{\text{safe}}^+ - h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r), \tag{39}$$

$$\boldsymbol{\tau}_{\text{margin},k}^- = h(\mathbf{q}_k^r, \dot{\mathbf{q}}_k^r) - \boldsymbol{\tau}_{\text{safe}}^-. \tag{40}$$

The acceleration envelope is then constrained by

$$|M(\mathbf{q}_k^r)|\, \boldsymbol{\beta}_k \leq \min\left(\boldsymbol{\tau}_{\text{margin},k}^+, \boldsymbol{\tau}_{\text{margin},k}^- \right), \tag{41}$$

where $\boldsymbol{\beta}_k$ denotes the symmetric acceleration magnitude envelope. The absolute operator is applied element-wise to the mass matrix, providing a conservative axis-aligned approximation while retaining the coupling effect among joints.

To obtain the largest feasible acceleration envelope, motion capability prediction formulates the envelope estimation as a weighted logarithmic volume maximization problem:

$$\max_{\boldsymbol{\beta}_k} \sum_{i=1}^{n} w_i \log(\beta_{i,k} + \epsilon), \tag{42}$$

subject to

$$0 \leq \boldsymbol{\beta}_k \leq \bar{\boldsymbol{\beta}}, \tag{43}$$

and

$$|M(\mathbf{q}_k^r)|\, \boldsymbol{\beta}_k \leq \min\left(\boldsymbol{\tau}_{\text{margin},k}^+, \boldsymbol{\tau}_{\text{margin},k}^- \right). \tag{44}$$

Here, $w_i > 0$ denotes the weight of joint i, ϵ is a small positive constant for numerical stability, and $\bar{\boldsymbol{\beta}}$ denotes the predefined acceleration ceiling.

The resulting solution $\boldsymbol{\beta}_k$ provides the instantaneous acceleration capability available for trajectory generation. Since the same dynamic boundary is used for both acceleration and braking, the actual motion direction is determined later by dynamics-aware interpolation according to the target displacement and local motion trend. Therefore, motion capability prediction only characterizes the current physical capability of the manipulator and does not impose a specific motion direction.

B.2 Dynamics-Aware Interpolation for Feasible Trajectories

Although the acceleration envelope provided by motion capability prediction characterizes the instantaneous dynamically admissible acceleration magnitude, it does not explicitly consider whether the corresponding acceleration can be reached within a single execution interval under jerk and snap constraints. Therefore, dynamics-aware interpolation further combines the motion capability prediction envelope with the C3 reachability constraints induced by the current reference jerk, obtaining the instantaneous executable acceleration interval for high-frequency trajectory updates.

Trend-Aware Terminal Velocity Estimation Given the dynamic acceleration envelope provided by motion capability prediction, dynamics-aware interpolation determines an appropriate terminal velocity for each interpolation segment. The terminal velocity serves as the boundary condition of the subsequent fixed-horizon trajectory interpolation process. A suitable terminal velocity should simultaneously satisfy three requirements: maintaining sufficient motion progress when the target is beyond the current dynamic reachability, following the local target trend when the trajectory enters the reachable region, and reducing the pass-through velocity before an upcoming directional change.

For joint i, let the current interpolation segment be defined by three consecutive VLA targets:

$$(\theta_{0,i}, \theta_{1,i}, \theta_{2,i}), \tag{45}$$

where $\theta_{0,i}$ denotes the current executed joint position, $\theta_{1,i}$ denotes the current target waypoint, and $\theta_{2,i}$ denotes the preview waypoint after the current target. The current target displacement and preview displacement are defined as

$$e_i = \theta_{1,i} - \theta_{0,i}, \tag{46}$$

$$e_{+,i} = \theta_{2,i} - \theta_{1,i}. \tag{47}$$

The current motion direction is determined by

$$\sigma_i = \mathrm{sign}(e_i), \tag{48}$$

where $\sigma_i \in \{-1, 0, 1\}$ indicates the moving direction of joint i. The current velocity projected onto the target direction is therefore

$$\omega_{\parallel,i} = \sigma_i \omega_{0,i}, \tag{49}$$

where $\omega_{0,i}$ denotes the current joint velocity.

To characterize the effective remaining distance, we compensate the geometric error using the current motion state:

$$d_{\mathrm{eff},i} = \max\left(0, |e_i| - \mathrm{clip}\left(k_v \omega_{\parallel,i} T_h, -k_d d_{\max,i}, k_d d_{\max,i}\right)\right), \tag{50}$$

where $d_{\max,i}$ denotes the maximum local interpolation distance, k_v controls the velocity compensation strength, and k_d limits the maximum correction magnitude. When the current velocity points toward the target, the effective distance decreases; when the velocity points away from the target, the effective distance increases.

Based on the effective distance, dynamics-aware interpolation continuously interpolates between a progress-oriented regime and a convergence-oriented regime. We define the convergence distance and progress distance as

$$d_{\mathrm{conv},i} = d_{\max,i}, \tag{51}$$

$$d_{\mathrm{prog},i} = \mu_{\mathrm{prog}} d_{\max,i}, \qquad \mu_{\mathrm{prog}} > 1. \tag{52}$$

The progress weight is then computed as

$$\alpha_i = \mathrm{clip}\left(\frac{d_{\mathrm{eff},i} - d_{\mathrm{conv},i}}{d_{\mathrm{prog},i} - d_{\mathrm{conv},i}}, 0, 1 \right). \tag{53}$$

When $\alpha_i = 1$, the current segment is dominated by motion progress, while $\alpha_i = 0$ corresponds to local convergence behavior.

The convergence velocity is estimated from the preview trend. The local trend velocity is defined as

$$\omega_{\mathrm{trend},i} = \frac{\theta_{2,i} - \theta_{1,i}}{T_h}, \tag{54}$$

where T_h denotes the trajectory interpolation horizon. To ensure that the pass-through velocity remains consistent with the nominal motion constraints, we first define the local reachable distances:

$$\ell_{1,i} = |\theta_{1,i} - \theta_{0,i}|, \tag{55}$$

$$\ell_{2,i} = |\theta_{2,i} - \theta_{1,i}|. \tag{56}$$

Using the acceleration capability estimated by motion capability prediction, the admissible velocity bound is

$$\omega_{\mathrm{cap},i} = \min\left(\omega_{\max,i}, \sqrt{2\beta_{i,k}\ell_{1,i}}, \sqrt{2\beta_{i,k}\ell_{2,i}} \right), \tag{57}$$

where $\beta_{i,k}$ denotes the acceleration capability of joint i obtained from motion capability prediction at execution cycle k. The convergence velocity is then

$$\omega_{\mathrm{conv},i} = \mathrm{clip}\left(\omega_{\mathrm{trend},i}, -\omega_{\mathrm{cap},i}, \omega_{\mathrm{cap},i} \right). \tag{58}$$

When the target remains outside the current interpolation capability, dynamics-aware interpolation maintains forward motion by introducing a progress-oriented velocity. The corresponding velocity magnitude is

$$\omega_{\mathrm{prog,mag},i} = \lambda_p \min\left(\omega_{\max,i}, \sqrt{2\beta_{i,k} d_{\max,i}} \right), \tag{59}$$

where $\lambda_p > 0$ is the progress coefficient. The progress velocity is

$$\omega_{\mathrm{prog},i} = \sigma_i \omega_{\mathrm{prog,mag},i}. \tag{60}$$

A separate treatment is applied when the preview trajectory indicates a forthcoming directional reversal. A sign change alone does not necessarily require stopping, since the target transition may still be outside the local reachable region. Therefore, a turning event is activated only when both geometric reversal and local proximity are satisfied:

$$d_{\mathrm{turn},i} = \eta_{\mathrm{turn}} d_{\max,i}, \tag{61}$$

$$\mathcal{T}_i = 1 \iff (e_i e_{+,i} < 0) \wedge (d_{\mathrm{eff},i} \le d_{\mathrm{turn},i}). \tag{62}$$

When the turning condition is satisfied, the terminal velocity becomes

$$\omega_{\mathrm{turn},i} = \sigma_i \lambda_{\mathrm{turn}} \omega_{\mathrm{cap},i}, \tag{63}$$

where $\lambda_{\mathrm{turn}} \in [0,1]$ controls the remaining velocity at the turning point.

The raw terminal velocity estimate is obtained by combining the above behaviors:

$$\omega_{1,i}^{\mathrm{raw}} = \begin{cases} \omega_{\mathrm{turn},i}, & \mathcal{T}_i = 1, \\ (1-\alpha_i)\omega_{\mathrm{conv},i} + \alpha_i \omega_{\mathrm{prog},i}, & \mathcal{T}_i = 0. \end{cases} \tag{64}$$

To suppress high-frequency fluctuations caused by local target variations, a first-order temporal filter is applied. Let $\omega_{1,i}^{\mathrm{prev}}$ denote the previous terminal velocity estimate. The filtered velocity is

$$\omega_{1,i}^{*} = (1-\gamma_\omega)\omega_{1,i}^{\mathrm{prev}} + \gamma_\omega \omega_{1,i}^{\mathrm{raw}}, \tag{65}$$

where $\gamma_\omega \in [0,1]$ is the filtering coefficient. The final terminal velocity is constrained by the velocity limit:

$$\omega_{1,i} = \mathrm{clip}\left(\omega_{1,i}^{*}, -\omega_{\max,i}, \omega_{\max,i}\right). \tag{66}$$

In addition to terminal velocity estimation, dynamics-aware interpolation also adjusts the acceleration utilization according to the current motion stage. Specifically, the effective acceleration capability used by trajectory interpolation is

$$\beta_{\mathrm{eff},i,k} = \beta_{i,k}\left(\lambda_{\beta,\min} + \alpha_i(1-\lambda_{\beta,\min})\right), \tag{67}$$

where $\lambda_{\beta,\min} \in [0,1]$ defines the minimum acceleration utilization ratio during the convergence regime. Therefore, the trajectory generator can exploit more of the available dynamic capability during long-range progress while reducing acceleration aggressiveness near local convergence regions.

The terminal velocity estimator outputs two quantities for the following C^3 trajectory generation: the terminal velocity $\omega_{1,i}$ and the effective acceleration bound $\beta_{\mathrm{eff},i,k}$. The former defines the terminal boundary condition, while the latter determines the dynamic capability used by the trajectory generator.

C^3 Feasibility Refinement and Seven-Phase Trajectory Interpolation After determining the terminal velocity ω_1 and the effective acceleration capability β_{eff} from the previous stage, dynamics-aware interpolation generates a fixed-horizon trajectory that connects the current reference state to the local terminal state. Different from conventional interpolation schemes that restart each segment from manually specified boundary conditions, dynamics-aware interpolation directly inherits the complete reference state maintained by the high-frequency controller, including position, velocity, acceleration, and jerk. This state inheritance enables consecutive replanning cycles to preserve temporal continuity throughout the generated trajectory.

Although the acceleration envelope provided by motion capability prediction characterizes the instantaneous dynamically admissible acceleration magnitude, it does not explicitly consider whether the corresponding acceleration can be reached within a single execution interval under jerk and snap constraints. Therefore, dynamics-aware interpolation further combines the motion capability prediction envelope with the C3 reachability constraints induced by the current reference jerk, obtaining the instantaneous executable acceleration interval for high-frequency trajectory updates.

Let $j_{\max,i} > 0$ denote the jerk limit of joint i, $s_{\max,i} > 0$ denote the snap limit, and Δt denote the high-frequency execution period. The jerk of the next execution cycle must satisfy

$$-j_{\max,i} \le j_{i,k+1} \le j_{\max,i}. \tag{68}$$

Meanwhile, the jerk variation is constrained by the snap bound:

$$-s_{\max,i}\Delta t \le j_{i,k+1} - j^r_{i,k} \le s_{\max,i}\Delta t. \tag{69}$$

The intersection of the jerk magnitude constraint and the snap continuity constraint gives the reachable jerk interval:

$$j^{\min}_{i,k+1} = \max\left(-j_{\max,i}, j^r_{i,k} - s_{\max,i}\Delta t\right), \tag{70}$$

$$j^{\max}_{i,k+1} = \min\left(j_{\max,i}, j^r_{i,k} + s_{\max,i}\Delta t\right). \tag{71}$$

The reference acceleration follows the discrete jerk integration relation:

$$\ddot{q}^r_{i,k+1} = \ddot{q}^r_{i,k} + j_{i,k+1}\Delta t. \tag{72}$$

Therefore, the acceleration interval determined only by C3 continuity is

$$\ddot{q}^{C3,-}_{i,k} = \ddot{q}^r_{i,k} + j^{\min}_{i,k+1}\Delta t, \tag{73}$$

$$\ddot{q}^{C3,+}_{i,k} = \ddot{q}^r_{i,k} + j^{\max}_{i,k+1}\Delta t. \tag{74}$$

Combining the C3 reachable interval with the motion capability prediction acceleration envelope yields the actual executable acceleration range:

$$\underline{\ddot{q}}_{i,k} = \max\left(-\beta^-_{i,k}, \ddot{q}^{C3,-}_{i,k}\right), \tag{75}$$

$$\overline{\ddot{q}}_{i,k} = \min\left(\beta^+_{i,k}, \ddot{q}^{C3,+}_{i,k}\right). \tag{76}$$

This interval is recomputed at every high-frequency cycle using the current reference state, motion capability prediction envelope, and high-order derivative constraints. The velocity estimation stage uses the original motion capability $\beta_{i,k}$ for finite-distance motion prediction, while the above interval is applied when updating the actual acceleration reference.

Continuous VLA commands may exhibit abrupt trend changes between consecutive low-frequency updates. To avoid directly replacing the current trajectory with a discontinuous new target, dynamics-aware interpolation detects local trend changes from consecutive VLA targets.

Let q^{VLA}_{ℓ} denote the VLA target at update index ℓ. The target increment is defined as

$$\Delta q_{\ell} = q^{\mathrm{VLA}}_{\ell} - q^{\mathrm{VLA}}_{\ell-1}, \tag{77}$$

and the trend variation indicator is

$$I_{\ell} = \max_i \left|\Delta q_{\ell,i} - \Delta q_{\ell-1,i}\right|. \tag{78}$$

When I_ℓ exceeds a predefined threshold and sufficient history is available, a new motion segment is treated as a transition event. The first VLA frame and cases without complete history skip this detection.

Let t_h denote the bridge activation time and T_f denote the fixed bridge duration. The normalized bridge time is

$$u = \mathrm{clip}\left(\frac{t - t_h}{T_f}, 0, 1\right). \tag{79}$$

The smooth bridge weight is defined as

$$\rho(u) = 35u^4 - 84u^5 + 70u^6 - 20u^7. \tag{80}$$

Let q_{old} denote the local target retained at bridge activation and q_{new} the incoming target. During the bridge, we update the target as $q_{\text{bridge}}(t) = (1 - \rho(u))q_{\text{old}} + \rho(u)q_{\text{new}}$. This blended target replaces q_{target} in the subsequent local target projection, while the trajectory generator inherits the current reference position, velocity, acceleration, and jerk as its initial state. During bridging, dynamics-aware interpolation fully inherits the current reference state

$$(q_k^r, \dot{q}_k^r, \ddot{q}_k^r, j_k^r), \tag{81}$$

and uses this state as the initial boundary condition of the newly generated seven-phase trajectory. The bridge only updates the trajectory boundary transition; the final position, velocity, acceleration, and jerk are always produced by the same continuous trajectory.

Even after terminal velocity estimation and trend bridging, the original VLA target may exceed the displacement achievable within the current planning horizon. Let T_h denote the trajectory horizon and $q_{\text{target},i}$ denote the original VLA target. The short-term reachable displacement is approximated as

$$\Delta q_i^+ = \max(\dot{q}_{i,k}^r, 0)T_h + \frac{1}{2}\beta_{\text{eff},i}^+ T_h^2, \tag{82}$$

$$\Delta q_i^- = \max(-\dot{q}_{i,k}^r, 0)T_h + \frac{1}{2}\beta_{\text{eff},i}^- T_h^2. \tag{83}$$

The local terminal target is then projected into the reachable region:

$$q_{\text{feasible},i} = q_{i,k}^r + \text{clip}\left(q_{\text{target},i} - q_{i,k}^r, -\Delta q_i^-, \Delta q_i^+\right). \tag{84}$$

The projected target only serves as the local endpoint of the current trajectory interpolation cycle. The original VLA target remains unchanged and is progressively tracked in subsequent replaning cycles.

After determining the inherited initial state, feasible terminal position, and terminal velocity, dynamics-aware interpolation generates a seven-phase trajectory using eight jerk nodes over the planning horizon. Let $t_0, \ldots, t_7$ denote the phase boundaries and $m \in \{0, \ldots, 6\}$ denote the phase index.

$$\Delta T_m = t_{m+1} - t_m. \tag{85}$$

Let j_m and j_{m+1} denote the jerk values at the two boundaries of phase m. The phase snap is

$$s_m = \frac{j_{m+1} - j_m}{\Delta T_m}. \tag{86}$$

For local phase time $\tau \in [0, \Delta T_m]$, jerk is linearly interpolated as

$$j(\tau) = j_m + s_m\tau. \tag{87}$$

Given the phase initial position, velocity, and acceleration (q_m, v_m, a_m), acceleration evolves as

$$a(\tau) = a_m + j_m\tau + \frac{1}{2}s_m\tau^2. \tag{88}$$

The corresponding velocity is

$$v(\tau) = v_m + a_m\tau + \frac{1}{2}j_m\tau^2 + \frac{1}{6}s_m\tau^3, \tag{89}$$

and the position is

$$q(\tau) = q_m + v_m\tau + \frac{1}{2}a_m\tau^2 + \frac{1}{6}j_m\tau^3 + \frac{1}{24}s_m\tau^4. \tag{90}$$

Adjacent phases share identical boundary jerk values, and the terminal state of each phase is directly propagated as the initial state of the next phase. Therefore, q, v, a, and j remain continuous throughout the complete trajectory.

The initial boundary condition is inherited from the current reference state:

$$q(0) = q_k^r, \qquad \dot{q}(0) = \dot{q}_k^r, \tag{91}$$

$$\ddot{q}(0) = \ddot{q}_k^r, \qquad j(0) = j_k^r. \tag{92}$$

The terminal conditions are

$$q(T_h) = q_{\text{feasible}}, \qquad \dot{q}(T_h) = \omega_1, \tag{93}$$

and

$$\ddot{q}(T_h) = 0, \qquad j(T_h) = 0. \tag{94}$$

To suppress unnecessary high-frequency variation of jerk nodes, we introduce the jerk variation regularization:

$$R_{\Delta j} = \sum_{m=0}^{6} \|j_{m+1} - j_m\|_2^2. \tag{95}$$

To retain the local motion intent of the policy targets, we penalize deviations from a fixed local reference $q_{\text{ref}}(t)$ through $R_{\text{traj}} = \frac{1}{N_s}\sum_{n=1}^{N_s} \|q(t_n) - q_{\text{ref}}(t_n)\|_2^2$, where t_n are uniformly spaced evaluation times over the planning horizon and N_s is their number. The reference is constructed from the local policy targets before trajectory optimization and remains fixed during the solve. Together with the trajectory regularization term R_{traj}, the final objective is

$$R = \lambda_{\Delta j} R_{\Delta j} + \lambda_{\text{traj}} R_{\text{traj}}, \tag{96}$$

where $\lambda_{\Delta j}$ and λ_{traj} are non-negative weights.

All jerk nodes and phase snaps satisfy

$$|j_m| \le j_{\max}, \qquad |s_m| \le s_{\max}. \tag{97}$$

The generated seven-phase trajectory provides the nominal reference trajectory. However, since the robot configuration and velocity continuously change during execution, the dynamic condition satisfied at the planning instant cannot replace online feasibility checking.

Let $\ddot{q}_{i,k+1}^{\text{plan}}$ denote the planned acceleration obtained from the seven-phase trajectory at the next high-frequency execution instant. The acceleration is projected into the currently executable interval:

$$\ddot{q}_{i,k+1}^r = \text{clip}\left(\ddot{q}_{i,k+1}^{\text{plan}}, \underline{\ddot{q}}_{i,k}, \overline{\ddot{q}}_{i,k}\right). \tag{98}$$

After projection, all reference states are propagated consistently. The updated jerk is

$$j^r_{i,k+1} = \frac{\ddot{q}^r_{i,k+1} - \ddot{q}^r_{i,k}}{\Delta t}, \tag{99}$$

the reference velocity is

$$\dot{q}^r_{i,k+1} = \mathrm{clip}(\dot{q}^r_{i,k} + \ddot{q}^r_{i,k+1}\Delta t, -\omega_{\max,i}, \omega_{\max,i}), \tag{100}$$

and the reference position is updated as

$$q^r_{i,k+1} = q^r_{i,k} + \dot{q}^r_{i,k+1}\Delta t. \tag{101}$$

This recursive update guarantees that the executed position, velocity, acceleration, and jerk always originate from the same reference state sequence. Finally, the feedforward torque for the next execution cycle is computed as

$$\tau_{\mathrm{ff},k+1} = M(q^r_{k+1})\ddot{q}^r_{k+1} + h(q^r_{k+1}, \dot{q}^r_{k+1}), \tag{102}$$

and the reference position, velocity, and feedforward torque are sent to the original MIT controller. The reference acceleration and jerk are retained internally for the next dynamics-aware interpolation update cycle.

B.3 Notation of Low-level Motion Generation

All variables in the low-level trajectory interpolation framework are defined in the joint space. Table 4 summarizes the major symbols used throughout this section. Here, q, $\dot{q}$, $\ddot{q}$, j, and s denote joint angle, joint angular velocity, joint angular acceleration, joint jerk, and joint snap, respectively.

C *DA-Arm* Hardware and Embodiment-Parameterized Interface

C.1 Technical specifications

Tables 5 and 6 summarize the kinematic parameters and technical specifications of *DA-Arm*.

C.2 Robot-Model Interface and Re-instantiation

DA-VLA separates the reusable policy prior from the physical quantities required by the deployed robot. The robot-model interface supplies the kinematic tree, link inertial parameters, joint states, actuation limits, rated torques, residual-friction allowance, and safety margins used by the dynamics-aware objective and motion capability prediction. A separate action adapter maps policy outputs into the target joint representation, while the command-feedback adapter connects interpolated references to the available low-level controller and returns executed joint states. The dynamics-aware objective, motion capability prediction, and dynamics-aware interpolation do not change their algorithmic form across compatible articulated manipulators.

Instantiating *DA-VLA* on another compatible manipulator consists of: (1) loading its robot description and inertial parameters; (2) configuring joint, velocity, acceleration, and torque limits; (3) defining the policy-to-joint action mapping; (4) connecting command and joint-feedback interfaces; and (5) adapting the policy on target-embodiment demonstrations, while the experiments evaluate the complete policy-to-execution pathway on *DA-Arm*.

D Experiment Details

Execution Protocol. Demonstrations are collected at 50 Hz. *Table Clearing* evaluates 30/50/100 Hz action-reference dispatch with the checkpoint and controller settings fixed within each method; 100 Hz halves the demonstration reference interval. *Parcel Selection* additionally imposes a finite

interception window. The complete three-step inference pipeline, including the VLM, action head, and guidance, takes approximately 60 ms on one RTX 4090 with CUDA Graph optimization, within the 200 ms committed window at 100 Hz.

Within each task, all compared methods use the same workspace, object set, initial-condition range, instruction protocol, success criterion, timeout, and controller-side safety configuration. *Table Clearing* nominal and high-speed single-object settings use 10 trials for each of five objects (50 trials per method); dual-object and each OOD setting use 10 trials per method. *Garment Folding*, *Table Wiping*, and *Book Insertion* use 20 trials per method in the revised main evaluation. *Parcel Selection* uses 10 trials for each of five parcels (50 trials per method).

Metrics. We report success rate (SR) and execution throughput score (ETS) for evaluation, and use time-to-success (TTS) as a reference variable, defined as:

$$\mathrm{SR} \triangleq \frac{N_{\mathrm{success}}}{N}, \quad \mathrm{TTS} \triangleq \frac{1}{N_{\mathrm{success}}} \sum_{i:\mathrm{success}_i=1} T_i, \quad \mathrm{ETS} \triangleq \frac{N_{\mathrm{success}}}{\sum_{i=1}^{N} T_i}, \tag{103}$$

where N is the total number of trials, N_{success} is the number of successful trials, and T_i is the duration of the i-th trial in minutes.

We fix $\lambda = 1.5$ before final aggregation. If at least one trial succeeds, each failed trial is assigned $T_{max} = \lambda\,\mathrm{TTS}$; if no trial succeeds, success rate and execution throughput are both zero. Execution throughput therefore has units of successful completions per minute. For aggregate reporting, the exact task set and method set used for task-wise normalization are listed beside the corresponding table, preventing the normalization reference from changing implicitly across analyses.

For aggregate execution throughput reporting, we independently apply task-wise min–max normalization across all methods compared on the same task:

$$\mathrm{nETS}_m^{(j)} = \frac{\mathrm{ETS}_m^{(j)} - \min_{m'} \mathrm{ETS}_{m'}^{(j)}}{\max_{m'} \mathrm{ETS}_{m'}^{(j)} - \min_{m'} \mathrm{ETS}_{m'}^{(j)}}, \tag{104}$$

where m indexes the method and j indexes the task. The best-performing method on each task is therefore normalized to 1, and the lowest-performing method is normalized to 0. For *Parcel Selection*, we apply the same task-wise min–max normalization to success rate because this task does not report execution throughput. The normalized values are then averaged across the five-task suite. The normalization reference set is defined independently for the main-method comparison and the ablation comparison; therefore, nETS values should be compared within, rather than across, the two tables.

Motor logs are first restricted by an optional human-provided coarse time window and then refined by the same automatic motion-activity rule for all trials. Rows with invalid status, non-finite values, or duplicate timestamps are removed; large tracking errors are not removed solely because of magnitude. Signals are uniformly resampled and differentiated using the same filtering pipeline. Metrics cover the six arm joints and exclude the gripper.

Position RMSE compares the transmitted MIT position reference q_{snd} with motor feedback q without temporal alignment. Velocity RMSE compares the velocity derived from consecutive transmitted references with measured joint velocity. Feedback-jerk RMS/P95 quantify measured motion continuity. Physical alignment lag is the nonnegative lag maximizing the normalized cross-correlation between transmitted and feedback velocity; the median reliable lag across joints is reported per trial. Peak ρ is the correlation strength at the selected lag and is used to assess estimate reliability rather than as a primary performance metric.

Experiment resources. All experiments were conducted on a local GPU cluster. Target-adaptation runs used eight NVIDIA H200 141-GB GPUs, 160 vCPUs, and 1,800 GB of host memory. A complete target-adaptation run required approximately one day. The complete inference pipeline takes approximately 60 ms with the three-step full method and 160 ms with the ten-step variant without RecFM.

Task descriptions and evaluation scope The evaluation covers *Table Clearing*, *Garment Folding*, *Table Wiping*, *Book Insertion*, and *Parcel Selection*, spanning general manipulation, deformable-object handling, contact-rich execution, and dynamic interception.

Table Clearing. The robot arm is required to place tabletop objects into a designated container following natural-language instructions. Objects are randomly distributed with varying positions and counts within a predefined region, and the robot must execute either single-object pick-and-place or sequentially pick multiple objects in a specified order. Representative items include regular wooden blocks, soft round plush toys, plastic cups, a relatively heavy tape measure, and a half-filled water bottle, covering diverse geometries, materials, and mass distributions. The evaluation further includes multiple out-of-distribution (OOD) conditions: target/container placements outside the training range, irrelevant distractor objects already inside the container, generalization to longer pick sequences than those seen during demonstrations, and open-ended or under-specified language commands that require semantic grounding and disambiguation from the scene. This task evaluates the model's ability to imitate and reliably execute high-speed manipulation trajectories, as well as its generalization across spatial configurations, object counts, and instruction styles.

Garment Folding. The dual-arm system must fold a garment laid flat on the table and then move the folded garment to a designated side area. A single task-specific policy predicts a joint 14-dimensional action for both arms at each step, compared with the 7-dimensional action used for single-arm tasks. This task emphasizes bimanual coordination (e.g., synchronized grasping, alignment, and crease formation) and robustness to the uncertainty induced by deformable-object shape changes. It is used to assess stability and robustness in long-horizon, contact-rich manipulation of deformable materials.

Table Wiping. The robot arm must pick up a cloth placed at the side of the table, wipe the water stains on the tabletop until clean, and return the cloth to its original location. This task requires the model to identify transparent or low-contrast water stains, adapt the wiping path to their changing boundaries, and maintain stable surface contact with sufficient coverage. It reflects the capability to handle fine-grained scene variations and to manipulate deformable tools.

Book Insertion. The robot arm must insert two books placed on the upper edge of a drawer into reserved slots on a bookshelf. Since the books have non-negligible weight and the slots provide only a small clearance, successful insertion demands precise pose alignment, smooth pushing, and proper handling of contact constraints. Moreover, inserting the first book alters the force and spatial distribution within the shelf, potentially affecting the strategy for inserting the second book. This task evaluates execution smoothness, precision, and stability in multi-step contact-rich manipulation.

Parcel Selection. Following language instructions, the robot arm must grasp a specified parcel from a high-speed moving conveyor belt and place it at a fixed location. Because the target moves continuously and the grasping window is short, the robot must estimate the target pose in real time, track it online, and dynamically correct the end-effector trajectory to achieve stable grasp-and-place. This task imposes strict requirements on execution latency and closed-loop responsiveness, and it also tests object selection, trajectory adaptation, and language-conditioned execution in dynamic environments.

Ablations. We retrain the variant without RecFM using standard flow matching, while retaining dynamics-aware regularization, inference guidance, asynchronous chunking, and the remaining experimental settings. This variant uses ten sampling steps, whereas the full *DA-VLA* policy uses three. Both variants are evaluated on the same five-task suite.

The variant without dynamics-aware regularization and guidance is trained on the same target-task demonstrations, but removes the training-time dynamics-aware objective and disables inference-time execution guidance. The variant without dynamics-aware interpolation disables both motion capability prediction and dynamics-aware interpolation and uses MinJerk interpolation instead. The variant without asynchronous chunking uses synchronous execution: it completes the current chunk, acquires a new observation, and waits for inference to finish before executing the next chunk.

E Data Composition and Training Configuration

Data Composition. Public pretraining and target-embodiment adaptation serve complementary roles. Public pretraining learns reusable visual-language-action representations from the robot-data mixture in Table 7. Source-specific state and action interfaces accommodate the original action

representations and dimensions, and the DA-Arm dynamics-aware objective is inactive at this stage. During target adaptation, task-specific interfaces map the shared features to the DA-Arm action representation, where joint references and nominal torque demands are evaluated for dynamics-aware regularization. We adapt a separate model for each target task, and jointly optimize the action objective on embodied-action batches $\mathcal{B}_{\mathrm{act}}$ and the next-token prediction objective on multimodal batches $\mathcal{B}_{\mathrm{lm}}$:

$$\mathcal{L}_{\mathrm{total}} = \lambda_{\mathrm{act}}\mathcal{L}_{\mathrm{act}}(\mathcal{B}_{\mathrm{act}}) + \lambda_{\mathrm{lm}}\mathcal{L}_{\mathrm{lm}}(\mathcal{B}_{\mathrm{lm}}). \tag{105}$$

Here, λ_{act} and λ_{lm} weight the action and language-modeling objectives, respectively; $\mathcal{B}_{\mathrm{act}}$ and $\mathcal{B}_{\mathrm{lm}}$ denote the corresponding training batches. The action objective supervises velocities at the primary and auxiliary scales in the corresponding source action space, while dynamics-aware regularization is evaluated only on the primary clean-action estimate during target adaptation. We use $R = 4$ recursive training scales with cross-scale consistency weight $\lambda_{\mathrm{cs}} = 1.0$. The multimodal batch is sampled from COCO, VQA-v2, PixMo-Points, and PixMo-Point-Explanations with within-mixture proportions of 25%, 40%, 25%, and 10%, respectively. These proportions determine the composition of the multimodal stream, while $\lambda_{\mathrm{lm}} = 0.3$ weights its objective relative to action learning. Multimodal examples are supervised as text outputs and do not receive the dynamics-aware loss.

Inference and asynchronous execution. Inference evaluates the primary scale using $K = 3$ uniform reverse-time Euler steps, with guidance cap $c = 10$. The dynamics evaluation uses $\Delta t = 0.02\,\mathrm{s}$ during training and $\Delta t = 1/f_{\mathrm{dispatch}}$ during inference; this reference interval is distinct from the high-frequency execution period. At test time, we use $H = 50$ and a fixed committed prefix of $d_1 = 20$ actions across dispatch rates, corresponding to a 200 ms inference window at 100 Hz.

Inference is launched after dispatching 10 of the active chunk's 30 retained actions, while its final 20 actions continue to execute. The new chunk is conditioned on the observation captured at inference launch, and execution switches to its 21st action at the scheduled handoff. The committed actions provide the reference prefix, whose terminal action supplies the subsequent soft position reference. The continuity mask retains a zero-weight tail, and velocity and nominal-torque consistency are evaluated only within the committed overlap. Prefix consistency remains a soft constraint.

Contact-rich execution. For contact-rich tasks, *DA-VLA* does not explicitly estimate the environment contact wrench. Real demonstrations provide task-level contact intent, including approach direction, contact location, and manipulation strategy. The dynamics-aware pathway improves the conditions under which this strategy reaches contact by reducing chunk-transition discontinuities, respecting state-dependent actuation margins, and interpolating jerk-limited approach motion. During contact, dynamics compensation accounts for the nominal robot model and feedback control rejects bounded residual disturbances. The evaluated gains therefore do not require an exact contact model, while large or rapidly varying contacts remain outside the present scope. The detailed task descriptions are provided in Appendix D.

Table 4: **Notation and parameter summary for feasible trajectory interpolation.**

Symbol	Meaning	Value	Unit / Range
State and indices			
i	Joint index	1–6	integer
k	High-frequency execution step	-	$\mathbb{N}$
ℓ	VLA update index	-	$\mathbb{N}$
q_i	Joint angle	-	rad
$\dot{q}_i$	Joint angular velocity	-	rad/s
$\ddot{q}_i$	Joint angular acceleration	-	rad/s^2
j_i	Joint jerk	-	rad/s^3
s_i	Joint snap	-	rad/s^4
$\mathbf{x}_k^r$	Reference state	-	$[q, \dot{q}, \ddot{q}, j]^T$
VLA targets and Dynamics-Aware Interpolation variables			
q_0	Current executed joint target	-	rad
q_1	Current target waypoint	-	rad
q_2	Preview waypoint	-	rad
e	Current target error	-	rad
e_+	Preview error	-	rad
σ	Motion direction indicator	-	$\{-1, 0, 1\}$
$\dot{q}_{\parallel}$	Velocity projected along motion direction	-	rad/s
d_{eff}	Effective distance	-	rad
$d_{\max,i}$	Maximum interpolation distance	0.04	rad
k_v	Velocity compensation coefficient	0.5	> 0
k_d	Distance correction coefficient	0.5	> 0
μ_{prog}	Progress distance coefficient	2.0	> 1
α	Progress-convergence weight	-	$[0, 1]$
$\dot{q}_{\text{trend}}$	Preview trend angular velocity	-	rad/s
$\dot{q}_{\text{conv}}$	Convergence angular velocity	-	rad/s
$\dot{q}_{\text{prog}}$	Progress angular velocity	-	rad/s
$\dot{q}_{\text{turn}}$	Turning angular velocity	-	rad/s
$\dot{q}_1$	Terminal angular velocity	-	rad/s
$\dot{q}_{\max,i}$	Angular velocity limit	2.0	rad/s
$\gamma_{\dot{q}}$	Angular velocity filtering coefficient	0.5	$[0, 1]$
Motion Capability Prediction dynamics			
$\boldsymbol{\tau}$	Joint torque	-	N·m
$M(q)$	Mass matrix	-	-
$C(q, \dot{q})$	Coriolis matrix	-	-
$g(q)$	Gravity torque	-	N·m
$h(q, \dot{q})$	Bias dynamics	-	N·m
$\boldsymbol{\tau}_{\text{rated}}$	Rated torque vector	$(18, 18, 18, 6, 6, 6)$	N·m
$\boldsymbol{\tau}_{\text{reserve}}$	Torque reserve	$(1.5, 1.5, 1.5, 0.5, 0.5, 0.5)$	N·m
ρ_τ	Torque utilization ratio	0.82	$(0, 1)$
$\mathcal{A}_k$	Feasible acceleration set	-	-
$\ddot{q}_k^{\max}$	Predicted motion acceleration envelope	-	rad/s^2
$\bar{\ddot{q}}$	Maximum acceleration bound	$(100, 100, 100, 100, 100, 100)$	rad/s^2
w_i	Envelope optimization weight	1.0	> 0
C^3 trajectory interpolation and projection			
T_h	Planning horizon	-	s
T_f	Bridge duration	0.05	s
$j_{\max,i}$	Jerk limit	3000	rad/s^3
$s_{\max,i}$	Snap limit	60000	rad/s^4
$\lambda_{\ddot{q},\min}$	Minimum acceleration ratio	0.8	$[0, 1]$
$\ddot{q}_{\text{eff}}$	Effective acceleration bound	-	rad/s^2
q_{feasible}	Reachable target position	-	rad
$\underline{\ddot{q}}, \bar{\ddot{q}}$	Executable acceleration interval	-	rad/s^2
t_p	Local phase time	-	s
ΔT_m	Duration of phase m	-	s
j_m	Phase jerk	-	rad/s^3
s_m	Phase snap	-	rad/s^4

Table 5: Modified D–H parameters of the robotic arm.

Link	θ_i (deg)	a_i (mm)	d_i (mm)	α_i (deg)
Base rotation	0	0	106.26	0
Shoulder pitch	180	19	0	-90
Elbow flexion	$180 + \theta$	269	0	180
Wrist deflection	$-\theta$	236.12	0	0
Wrist pitch	90	80	0	90
End-effector rotation	0	0	29	90

Table 6: Technical specifications of the robotic arm.

Quantity	Specification
Active DoF	6
Maximum reach	636.7 mm
Main body weight	3.4 kg
Payload	2 kg at end-effector
Communication interface	CAN
Supply voltage	24 V DC

Table 7: **Public datasets used for VLA pretraining.** Sampling ratios denote the probability of drawing samples from each dataset.

Dataset	Trajectories	Frames	Robot embodiment	Sampling ratio (%)
RT-1	59,591	3,073,209	Google Everyday Robots	15.09
ManiSkill	30,037	4,510,748	Franka (simulation)	9.69
BC-Z	36,609	5,426,963	Google Robot	11.59
Mobile ALOHA	275	463,650	Mobile ALOHA	8.35
DROID	46,544	12,991,419	Franka Panda	16.87
RoboSet	16,834	1,310,439	Franka Panda	7.51
BridgeData V2	23,690	1,012,886	WidowX 250	12.47
RH20T	23,789	11,073,707	Franka Panda	18.40
Total	**237,369**	**39,863,021**	–	**100.00**

Table 8: **Task dataset size.**

Task	Demonstrations	Frames
Table Clearing	470	166,390
Garment Folding	100	79,067
Table Wiping	100	66,003
Book Insertion	100	102,964
Parcel Selection	180	64,436
Total	950	478,860

Table 9: Optimization and training configuration

Parameter	Public pretraining	Target adaptation
Feature-query tokens	20	20
Action horizon / chunk length	50	50
Optimizer	AdamW	AdamW
Base learning rate	1.5×10^{-5}	1.5×10^{-5}
Adam coefficients (β_1, β_2)	$(0.9, 0.995)$	$(0.9, 0.995)$
Adam numerical stability ϵ	1.0×10^{-8}	1.0×10^{-8}
Weight decay	1.0×10^{-2}	1.0×10^{-2}
Per-device batch size	2	2
Number of GPUs	32	8
Global batch size	64	16
Training steps	20,000	80,000
Learning-rate schedule	Cosine	Cosine with warmup and decay
Warmup steps	5,000	1,000
Decay steps	100,000	60,000
Minimum learning rate	1×10^{-5}	1.5×10^{-6}
Gradient clipping	Global norm 1.0	Global norm 1.0
Numerical precision	FP32	FP32
VLM parameters	Frozen	Trainable
Random seed	0	0
Training hardware	32× NVIDIA H200	8× NVIDIA H200
Inference hardware	Single NVIDIA RTX 4090	

Table 10: Recursive flow-matching, dynamics-aware training, and guided-inference settings.

Parameter	Description	Value
R	Recursive training scales	4
λ_{cs}	Cross-scale consistency weight	1.0
λ_{act}	Action-objective weight	1.0
λ_{lm}	Multimodal language-objective weight	0.3
λ_d	Dynamics-aware objective weight	1.0×10^{-3}
λ_c	Continuity weight	0.5
α_q	Position-reference consistency weight	1.0
$\alpha_{\dot{q}}$	Overlap velocity-consistency weight	0.5
α_τ	Overlap nominal-torque consistency weight	1.0×10^{-9}
d_1	Training prefix length	Random integer in $\{0, \ldots, 20\}$
d_1	Test-time committed prefix length	20
d_2	Soft transition length	Configurable, $0 < d_2 < H - d_1$
c	Inference-guidance cap	10
Δt	Training reference interval	$0.02\,\text{s}$
Δt	Inference reference interval	$1/f_{\text{dispatch}}$
Flow-time distribution	Flow-matching training-time distribution	$\text{Beta}(1.5, 1.0)$
Inference integration steps	Guided flow-matching denoising steps	3