GROOVE: Geometry-Guided Reduction of Operational-Space Jerk in VLA Execution

Sangho Yun[1], Minsoo Kim[2], Minwoo Cho[1], and Hwanjo Yu[1,2]

***Abstract*— Chunked vision–language–action (VLA) policies execute several commands per query, but jerk within chunks and across replanning boundaries can induce oscillatory motion and sharp actuator transients. We present GROOVE, an online regulator that searches directional correction regions around the raw three-dimensional end-effector (EEF) path, without retraining or additional VLA inference. It optimizes the new chunk using delivered commands as boundary conditions, reducing boundary and within-chunk jerk while bounding cumulative translation and local axis–angle deviation from the raw plan after every command. Using quadratic programs (QPs), GROOVE generates a cube reference and thirteen directional candidates, then selects the one with the lowest command-space jerk under a reference-relative deviation cap. On a held-out LIBERO benchmark, GROOVE achieves the largest reductions among the evaluated methods, reducing translational and rotational EEF jerk by 33.02% and 43.42%, respectively, with task success of 95.75% versus 93.75% for raw execution. Across 50 matched UR5e pairs with measured execution timing, it reduces translational and rotational tool-center-point (TCP) jerk by 16.39% and 19.49% and joint-current slew by 29.09%.**

***Index Terms*— vision–language–action models, action chunking, execution regulation, jerk reduction, robot manipulation**

I. INTRODUCTION

VLA policies execute single actions or multi-step chunks [1]–[4]. Chunking lets one query cover several control steps while promoting temporal coherence [2]–[5]. However, chunked execution can introduce jerk through two distinct mechanisms. Within a chunk, oscillations in predicted delta end-effector (EEF) commands can cause rapid motion changes. At a replanning boundary, the initial commands of a new chunk may not align with those most recently delivered to the controller [6]–[9]. Aligning only the replanning boundary therefore leaves within-chunk jerk unaddressed. Task success alone does not distinguish smooth execution from execution with abrupt motion and actuator transients. On physical manipulators, abrupt acceleration changes can excite structural vibration, degrade path tracking, and produce sharp torque and power transients [10], [11]. For a policy that already completes a task, reducing these transients is an additional objective that requires limiting changes to task-directed motion and accounting for execution time. We therefore evaluate measured motion and joint-current variation alongside task success and timing [12].

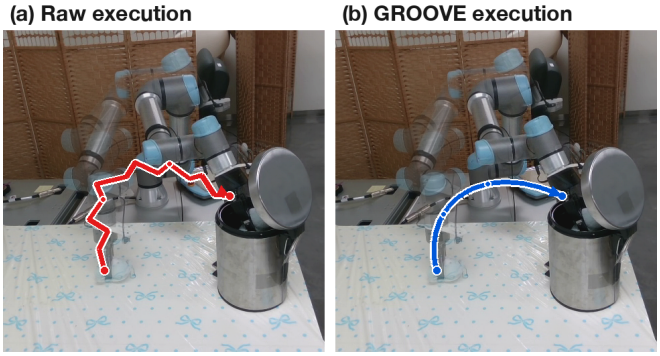

Fig. 1. Conceptual comparison of raw and GROOVE execution.

Existing approaches improve temporal consistency by modifying policy training or action generation [6]–[8], [13]. Training-time methods require retraining, while generation-time methods can require access to policy internals or new predictions during execution. Overlap-based methods depend on multiple unexecuted predictions, and ACT creates this overlap by querying the VLA at every step [2], [9]. Post-generation filters avoid these requirements by modifying commands after generation [14]. However, local corrections to delta-EEF commands accumulate as the commands are integrated into an EEF path. A filter can therefore reduce jerk while allowing the corrected path to drift from the raw task-directed path. Constraining only the chunk endpoint does not solve this problem because intermediate path deviations remain unconstrained.

We introduce GROOVE (Geometry-Guided Reduction of Operational-Space Jerk in VLA Execution), which regulates a new action chunk by searching the geometry of allowable corrections around its raw three-dimensional EEF path. A cube applies the same bound along each coordinate axis even when corrections that reduce jerk are concentrated along particular directions or planes. GROOVE constructs a cube reference and thirteen equal-volume directional constraint sets, each defining a different correction region at every raw-path timestamp (Figs. 1 and 2). A full-chunk quadratic program (QP) computes a candidate for each set using the last two delivered commands as boundary conditions. The selector chooses the candidate with the lowest command-space jerk under a reference-relative deviation cap. This geometric search runs after one VLA query, with every-

[1]Department of Computer Science and Engineering, POSTECH, Pohang, South Korea.
[2]Graduate School of Artificial Intelligence, POSTECH, Pohang, South Korea.
Correspondence to: Hwanjo Yu <hwanjoyu@postech.ac.kr>.
Project page: https://devsangho.github.io/GROOVE-public/.

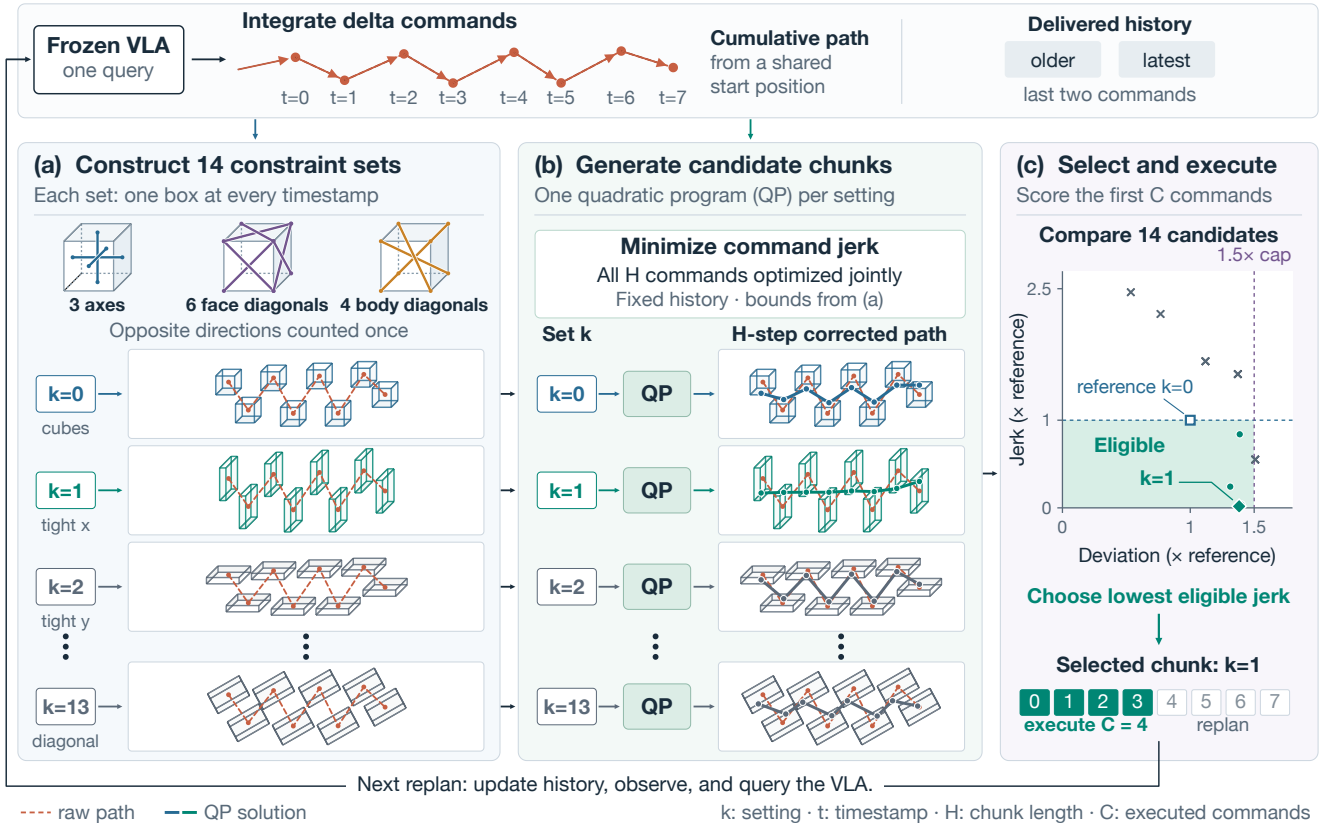

Fig. 2. GROOVE at one replan in a synthetic example computed by the QPs ($H = 8$, $C = 4$). (a) Construct 14 constraint sets, each placing a box at every raw-path timestamp. (b) Generate candidate chunks by jointly optimizing all H commands within the corresponding boxes, using delivered commands as boundary conditions. Corrected paths are overlaid on the red raw paths. (c) Select and execute using deviation and jerk over the first C commands. The blue square denotes the retained reference, crosses mark ineligible candidates, and the teal diamond marks the selected candidate $k = 1$. All fourteen score pairs are normalized by the reference, and some overlap.

prefix translation and local-coordinate rotation bounds and no retraining or additional prediction.

We make three contributions.

- We propose a geometric search over directional correction regions around the raw three-dimensional EEF path, with protected candidate selection under a reference-relative deviation cap.
- We implement this search through full-chunk optimization conditioned on delivered commands, with translation and rotation bounds at every prefix and no additional VLA inference.
- We evaluate the method with two frozen VLAs and on a UR5e. It yields the largest holdout jerk reductions among the evaluated methods and lowers TCP jerk and current slew on the robot, with higher aggregate success point estimates.

II. Related Work

A. Jerk-Aware Trajectory Generation

Minimum-jerk models optimize an entire trajectory by minimizing integrated squared jerk [15]. Jerk-limited motion is also used to suppress structural vibration, improve tracking, and reduce peak actuator demands on physical manipulators [10], [11]. Ruckig computes time-optimal motion under velocity, acceleration, and jerk limits [16]. These methods generate motion from boundary and target states. Regulating intermediate VLA delta-action paths requires additional deviation constraints.

B. Chunked Action Prediction and Aggregation

Action chunking reduces the average inference cost per control step by predicting several future commands in one query. OpenVLA predicts one action at a time, whereas $\pi_{0.5}$ and GR00T produce action chunks [1], [3], [4]. ACT predicts action chunks and temporally aggregates overlapping predictions at each control step [2]. BID samples multiple candidate chunks at test time and ranks them by consistency with past actions and contrast among future candidates [17]. ACT and BID combine multiple predictions to improve temporal consistency, but neither directly minimizes jerk or bounds cumulative EEF deviation from the raw path.

C. Training-Time Smoothness

SmoothVLA modifies a policy through reinforcement learning with a reward that combines task success and trajectory jerk [13]. Legato trains a flow policy to connect each new chunk smoothly to preceding actions [18]. RTR learns

a VAE-based latent action space for high-frequency continuous chunks and refines reused latent actions at inference time [19]. ChunkFlow uses seam and first- and second-order continuity losses during training and deterministic overlap blending during execution [20]. These approaches improve continuity by retraining the policy or learning an additional action representation.

D. Sampling-Time Action Consistency

Sampling-time methods modify how the policy samples actions to improve continuity across chunks. RTC uses flow inpainting to align the prefix of a new chunk with the unexecuted suffix of the previous chunk [7]. PAINT selects initial noise that improves prefix consistency before denoising [21]. SEAM replaces RTC's backward passes with a closed-form correction toward the previous suffix [6]. POTR modifies RTC guidance using prior correction and an orthogonal trust region [8]. ACG applies action-coherence guidance at selected attention layers during flow sampling [22]. These methods avoid retraining but require access to policy sampling. RTC, SEAM, and POTR require the previous unexecuted suffix, which is unavailable after full-chunk execution and contains planned rather than delivered commands. ACG instead evaluates an extra incoherent vector field during sampling.

E. Post-Generation Regulation

Post-generation methods regulate VLA outputs without retraining the policy. EMA uses a fixed decay, while One Euro adapts its low-pass cutoff to signal speed [14]. PACE sets execution length from low-speed transitions in the predicted chunk [23]. AAC shortens chunks under high action entropy and lengthens them under low entropy [24]. A2C2 applies a trained time-aware correction head at each control step [5]. LiPo minimizes jerk within per-step joint-position bounds around blended, overlapping trajectories [9]. Realtime-VLA V2 combines adaptive timing with model-based joint tracking [25]. GROOVE searches three-dimensional correction regions around the raw EEF path. A cube reference and equal-volume directional boxes define candidate paths, which are optimized from one new chunk and delivered-command history. A reference-relative selector chooses among these corrections using their jerk and path deviation.

III. METHOD

GROOVE reduces jerk by searching directional correction regions around the raw three-dimensional EEF path. Figure 2 shows how we define allowed corrections (a), optimize a candidate chunk under each setting (b), and select a candidate for execution (c). Each candidate contains H commands, of which the first C are executed before replanning ($1 \le C \le H$).

A. Construct 14 Constraint Sets

Smoothing individual delta commands can shift the path obtained by accumulating them. To limit this drift throughout the chunk, we place a box around each raw cumulative position and require the corrected position at that step to remain inside its corresponding box. Each row in Fig. 2(a) defines one constraint set containing H boxes. This sequence of boxes forms a tube around the raw path. The arrows represent commands, and the points represent cumulative positions.

Corrections that reduce jerk can be larger in some directions than others. We therefore compare a cube reference ($k = 0$) with thirteen settings that allow corrections along different directions ($k = 1, \dots, 13$). Each directional box is narrower along one axis and wider along the two perpendicular axes, with the same volume as the reference. For example, the $k = 1$ row restricts x deviation while allowing more correction along y and z. The fixed set of thirteen tight axes covers three coordinate axes, six face diagonals, and four body diagonals. It is independent of the raw chunk, which determines only the box centers. The box axes remain fixed throughout each row.

Near gripper transitions, all settings use smaller boxes to limit changes around opening or closing commands. The gripper commands themselves remain unchanged.

B. Generate Candidate Chunks

To reduce abrupt changes both within a chunk and at its start, we optimize all commands together using the last two delivered commands as boundary conditions. Only the box constraints vary across settings.

Let r be the raw arm chunk and v the chunk to optimize. At step t, each command $r_t, v_t \in \mathbb{R}^6$ contains controller-normalized translation and local axis–angle increments, denoted by superscripts tr and rot. The delivered commands h_{-2}, h_{-1} fix $v_{-2} = h_{-2}$ and $v_{-1} = h_{-1}$, with missing history initialized to zero. We penalize the second difference $\delta_h^2 v_t = v_t - 2v_{t-1} + v_{t-2}$. At $t = 0$, the second difference $v_0 - 2h_{-1} + h_{-2}$ connects the new chunk to delivered commands. Because translation commands are position increments, their second differences are proportional to position jerk under ideal integration at a fixed command period.

To express the boxes in (a), define the signed cumulative command difference $e_t(v) = \sum_{i=0}^{t}(v_i - r_i)$. Its translation component points from the raw cumulative position to the corrected position. For setting k, the columns of the orthonormal matrix Q_k are the three box axes, and $\eta_{k,t}$ contains their half-widths, measured from center to face. Thus $Q_k^\top e_t^{\mathrm{tr}}(v)$ expresses the deviation along those axes. Rotation uses a shared half-width ϵ_t, and $\mathbf{1}_3$ denotes the three-component all-ones vector. With a fixed weight $\lambda \ge 0$ on command magnitude, the QP returns candidate u_k,

$$
\begin{aligned}
u_k = \arg\min_{v_{0:H-1}} & \sum_{t=0}^{H-1} \left(\left\| \delta_h^2 v_t \right\|_2^2 + \lambda \| v_t \|_2^2 \right), \\
\text{s.t. } & \left| Q_k^\top e_t^{\mathrm{tr}}(v) \right| \preceq \eta_{k,t}, \\
& \left| e_t^{\mathrm{rot}}(v) \right| \preceq \epsilon_t \mathbf{1}_3, \quad t = 0, \dots, H-1.
\end{aligned}
\tag{1}
$$

Absolute values and inequalities apply componentwise. Second differences and cumulative constraints couple the H steps, so all $6H$ components are optimized jointly. Solving

this QP for each row produces the fourteen candidate paths in (b).

C. Select and Execute

A candidate can reduce jerk while increasing deviation, even when its boxes have the same volume as the reference. We therefore limit deviation relative to the reference and select the smoothest candidate that meets this limit. Only the first C commands enter this comparison because they will be executed before replanning. Rotation is shared across candidates, so we compare translation. For candidate u_k, with command $u_{k,t}$ at step t, define

$$B_k = \sum_{t=0}^{C-1} \left\| e_t^{\mathrm{tr}}(u_k) \right\|_1 , \qquad P_k = \frac{1}{C} \sum_{t=0}^{C-1} \left\| \delta_h^2 u_{k,t}^{\mathrm{tr}} \right\|_2 . \tag{2}$$

The deviation score B_k sums the L1 norms of cumulative translation deviations in the original coordinates, capturing intermediate drift even if later corrections cancel it. The jerk score P_k averages translation second-difference norms over the executed prefix, whereas the QP sums squared norms over all H arm commands.

The cube setting yields the QP-generated reference u_0, with scores B_0, P_0. If the reference fails numerical validation, we use the raw chunk. Otherwise, we retain it and admit a directional candidate only if it is numerically valid, satisfies $B_k \le \rho_B \max(B_0, 10^{-12})$, and has $P_k < P_0$. The fixed, dimensionless multiplier $\rho_B \ge 1$ sets the allowed reference-relative deviation, with 10^{-12} serving as a numerical floor. We choose the candidate with the smallest P_k among the reference and admitted candidates, using a fixed ordering for exact ties. This ensures that the selected candidate has a jerk score of at most P_0 and satisfies the deviation cap. These guarantees concern command-space scores, with physical motion evaluated separately. Panel (c) plots $(B_k/B_0, P_k/P_0)$ with positive reference denominators and selects $k = 1$ when $\rho_B = 1.5$.

The execution length C is shared with raw execution and all comparators for each policy. Regulation does not shorten C or query the VLA again. We send the first C arm commands from the chosen chunk with the corresponding original gripper commands g_t and discard the remaining $H - C$ commands. At the next replan, we update history from acknowledged commands and query the VLA using the new observation. Algorithm 1 summarizes the procedure.

Algorithm 1 GROOVE at one replan.

Input: raw chunk, delivered arm history, and last gripper state.
A. Construct 14 constraint sets. Prepare the boxes in Fig. 2(a) and shared rotation bounds.
B. Generate candidate chunks. For $k = 0, \ldots, 13$ **do**
  Solve Eq. (1) for the entire chunk to obtain u_k.
  Validate every prefix bound and unchanged gripper commands.
end for
C. Select and execute. If the reference is invalid, use the raw chunk.
Otherwise, compute B_k, P_k on the first C commands and admit valid candidates meeting the conditions in Sec. III-C.
  Choose the candidate with the smallest P_k among the reference and admitted candidates.
Send the first C commands of the chosen chunk with the original gripper commands.
Update delivered history and replan from the next observation.

Implementation Details

Outside gripper transitions, the reference uses the identity frame $Q_0 = I_3$ and half-widths $\eta_{0,t} = \epsilon_0 \mathbf{1}_3$. Directional settings use $\eta_{k,t} = [\epsilon_{\mathrm{tight}}, \epsilon_{\mathrm{wide}}, \epsilon_{\mathrm{wide}}]^\top$, with $0 < \epsilon_{\mathrm{tight}} < \epsilon_0 < \epsilon_{\mathrm{wide}}$ and $\epsilon_{\mathrm{tight}} \epsilon_{\mathrm{wide}}^2 = \epsilon_0^3$ to preserve box volume. Their tight axes are the normalized nonzero vectors in $\{-1, 0, 1\}^3$, counted once per opposite pair, and form the first columns of Q_k. Choose the coordinate axis least aligned with the first column, breaking ties in x, y, z order. Project this axis onto the plane perpendicular to the first column and normalize the projection to obtain the second column. The third column is the cross product of the first two.

Within w_g steps of a sign change in g_t, including transitions from the last delivered gripper command, each setting uses $\eta_{k,t} = \epsilon_g \mathbf{1}_3$ in its own frame. The shared rotation half-width ϵ_t equals ϵ_g in this window and ϵ_0 elsewhere. It bounds cumulative local axis–angle commands rather than exact orientation error.

In the orthonormal box frame, the objective is unchanged and translation separates into three scalar QPs spanning H steps. Reusing the common rotation solution gives $14 \times 3 + 3 = 45$ scalar solves. Fixed history makes the objective strictly convex even at $\lambda = 0$, and the raw chunk is feasible, so each QP has a unique exact solution. Numerical validation checks every prefix bound and exact preservation of gripper commands.

IV. EXPERIMENTS

We compare jerk and task success under matched query budgets, evaluate physical effects and timing, and examine ablations and runtime.

A. Simulation Benchmark

We evaluate frozen $\pi_{0.5}$ and GR00T N1.7 policies on all 40 tasks from LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10 at 20 Hz [3], [26], [27]. We use prediction horizons and execution lengths of $(H, C) = (10, 5)$ for the OpenPI pi05_libero checkpoint and $(16, 8)$ for GR00T N1.7. Development grids contain 800 matched raw–GROOVE pairs per policy for comparisons across policies under shared settings and for mechanism analyses. The primary comparison uses a separately frozen $\pi_{0.5}$ holdout of 400 matched episodes, with 100 per suite.

Within each matched block, methods share the policy checkpoint, task, initial state, sampling seed, and episode command limit. All holdout methods, including raw execution, use $(H, C) = (10, 5)$ and one VLA query per five commands, with no extra observations or predictions. Command limits are 220, 280, 300, and 520 for Spatial, Object, Goal, and LIBERO-10, respectively. Methods with

equal query budgets can have different inference and post-processing latencies, so we assess physical timing separately. We retain every assigned episode, and holdout outcomes were unavailable during method selection.

B. Comparators and Ablation Controls

The main holdout compares raw execution with RTC, a SEAM sweep over $\lambda \in \{0.05, 0.10, 0.20\}$, POTR, a query-matched overlap ensemble, a LiPo-QP adapter evaluated under the same protocol, and GROOVE [2], [6]–[9]. RTC follows the authors' ΠGDM prior-guidance formulation with a five-step prefix and maximum guidance of 5. SEAM guides all five available overlap steps, with $M = \min(20, H - C) = 5$ at each setting. POTR uses the paper's LIBERO values $\sigma_d = 0.4$, $\rho = 0.5$, and $\beta = 10$. Each configuration was frozen before evaluation. These generation-time methods leave the first chunk unguided because no previous suffix is available. We implement POTR in OpenPI following Eqs. 5–7 of the original paper [8].

The overlap ensemble averages the previous raw suffix and aligned new prefix with weights $\exp(-0.01i)$, where $i = 0$ denotes the previous suffix and $i = 1$ the new prefix [2]. The LiPo-QP adapter integrates delta-EEF commands, applies LiPo's overlap blend and ActionLiPo QP, and converts the result back to delta commands [9]. Its frozen overlap and path bounds are 0.4 and 0.06 in cumulative controller-normalized coordinates, preserving the original $0.02/0.003$ ratio at a scale of 20. Both adapters retain the same query rate, controller interface, and command budget. The LiPo adapter omits its controller-specific 400-Hz interpolation.

A follow-up candidate-set ablation reruns raw execution and four sets on the 400 initial-state assignments, using a separately frozen inference runtime and the same suite command limits. The sets comprise the cube alone, cube plus three axes, cube plus four body diagonals, and all fourteen candidates, with fixed QP parameters, selector, and $(H, C) = (10, 5)$. A separate 400-triplet rollout compares the cube with full sets at $\rho_B = 1$ and 1.5, with otherwise fixed settings. A post-hoc replay adjusts cube translation width to match selected first-C deviation B on stored full-set inputs and delivered history, retaining the objective and gripper guard.

On the development grid, we compare the full set with the cube reference, one direction sampled at each replan, and one candidate sampled uniformly from the eligible set. Both sampling controls use an episode-seeded generator. A sensitivity analysis restricted to the first 50 commands fixes the trajectory length. Fixed-window controls compare delivered-history, per-command, and endpoint-only variants with budget-matched EMA and One Euro filters [14]. Filter strengths are frozen on a disjoint 40-pair calibration set by matching the cube regulator's mean realized prefix deviation per replan without using success or jerk. Across all policies, GROOVE uses $\epsilon_0 = 0.15$, $\epsilon_{\text{tight}} = 0.05$, $\epsilon_{\text{wide}} = \sqrt{\epsilon_0^3/\epsilon_{\text{tight}}} \approx 0.2598$, $\epsilon_g = 0.001$, $w_g = 5$, $\lambda = 5 \times 10^{-9}$, and $\rho_B = 1.5$ in the controller-normalized coordinates defined above. GROOVE uses OSQP with absolute and relative tolerances of 10^{-6}, a 4,000-iteration limit, and a feasibility tolerance of 10^{-5} [28].

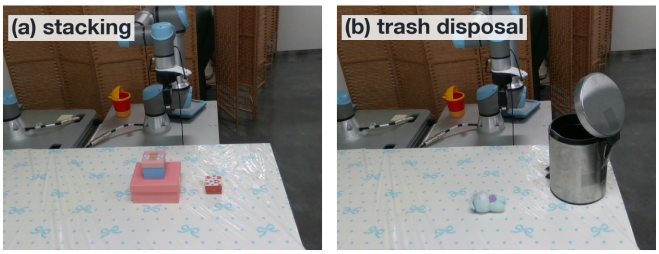

Fig. 3. Physical evaluation tasks on the UR5e. (a) Box stacking. (b) Long-range transport into a trash bin.

C. Physical-Robot Protocol

We deploy the frozen $\pi_{0.5}$ policy on a UR5e for box stacking and long-range transport into a trash bin (Fig. 3).

We collect 25 matched raw–GROOVE pairs per task, yielding 50 pairs and 100 attempts with alternating execution order. Each raw–GROOVE pair shares its policy seed, nominal object and receptacle locations, controller, safety limits, and 300-command budget.

Both conditions use $(H, C) = (10, 5)$, the same nominal 3.5-Hz Cartesian controller, and action scales of 0.05 m and 0.5 rad per normalized translation and rotation unit. An independent 125-Hz recorder captures TCP pose and speed, joint velocity, and joint current. The operator labels stable placement or release as success independently of metric computation. Each replan logs latency, solver failures, and raw fallbacks.

D. Metrics and Statistical Analysis

In simulation, translational jerk is the mean norm of third differences in executed EEF position divided by Δt^3. Rotational jerk is the mean norm of second differences in quaternion-derived local angular velocity divided by Δt^2, with $\Delta t = 0.05$ s for both metrics. Primary jerk endpoints are computed over each complete episode or trial. The first-50-command sensitivity analysis holds exposure length fixed.

We compute physical translational and rotational jerk from the linear- and angular-velocity triplets in the 125-Hz RTDE *actual_TCP_speed* signal. The analysis spans the first command through 250 ms after the final command. We resample in controller time, compute second derivatives with a fixed 11-sample third-order Savitzky–Golay filter, trim five samples at each boundary, and average the derivative norms. TCP travel is the integral of the linear-speed norm. Joint acceleration and current slew are the mean norms of the first Savitzky–Golay derivatives of RTDE joint velocity and current, using the same resampling, window, and trimming.

For positive endpoints, we report the paired geometric-mean reduction $100[1 - \exp(-\overline{\log(J^{\text{raw}}/J^{\text{method}})})]$. We form 95% confidence intervals from 10,000 paired bootstrap draws with seed 0. We resample tasks within suites in simulation and pairs within tasks on the robot, using equal group weights in both analyses. For task success, we report

TABLE I

LIBERO performance on the 400-pair $\pi_{0.5}$ holdout.

Executed-EEF jerk reduction from raw (%).

Method	Spatial		Object		Goal		LIBERO-10		Overall	
	ΔJ_{tr}	ΔJ_{rot}	ΔJ_{tr}	ΔJ_{rot}	ΔJ_{tr}	ΔJ_{rot}	ΔJ_{tr}	ΔJ_{rot}	ΔJ_{tr}	ΔJ_{rot}
Raw	–	–	–	–	–	–	–	–	–	–
RTC	6.19 [4.02, 8.44]	7.93 [6.08, 10.08]	8.23 [5.59, 10.86]	7.44 [4.96, 9.94]	4.92 [2.75, 7.22]	7.23 [4.78, 9.53]	6.11 [0.58, 9.83]	7.23 [3.82, 10.06]	6.37 [4.73, 7.81]	7.46 [6.15, 8.71]
SEAM ($\lambda = .05$)	10.20 [8.79, 11.69]	13.65 [12.54, 14.87]	11.25 [8.32, 14.12]	14.59 [12.20, 17.28]	8.95 [6.43, 11.75]	12.29 [10.17, 14.61]	11.78 [7.05, 14.95]	14.02 [10.81, 16.34]	10.55 [9.04, 11.96]	13.64 [12.49, 14.76]
SEAM ($\lambda = .10$)	17.95 [15.68, 20.41]	21.83 [20.06, 23.62]	17.29 [14.20, 20.16]	21.73 [18.33, 24.89]	14.90 [12.86, 17.53]	19.07 [16.74, 21.28]	19.44 [14.49, 23.05]	22.83 [19.11, 25.83]	17.41 [15.79, 18.94]	21.38 [19.92, 22.74]
SEAM ($\lambda = .20$)	20.25 [17.02, 23.74]	24.51 [22.18, 27.16]	24.94 [21.83, 27.96]	28.11 [25.55, 30.40]	18.51 [15.05, 22.30]	23.62 [20.39, 26.72]	24.29 [19.94, 27.89]	27.47 [23.62, 30.77]	22.05 [20.27, 23.81]	25.95 [24.43, 27.40]
POTR	10.50 [8.32, 13.15]	13.67 [12.18, 15.46]	13.27 [11.28, 15.47]	13.64 [11.33, 15.89]	8.27 [5.11, 11.53]	12.02 [9.46, 14.88]	11.55 [6.74, 15.15]	12.77 [8.83, 15.97]	10.92 [9.32, 12.42]	13.03 [11.66, 14.31]
Overlap ensemble	16.17 [13.67, 18.84]	20.36 [18.71, 22.02]	21.47 [18.73, 24.12]	23.63 [21.04, 26.03]	15.63 [13.69, 17.48]	19.79 [16.80, 22.49]	19.75 [14.90, 23.64]	22.67 [19.53, 25.25]	18.29 [16.73, 19.76]	21.63 [20.31, 22.83]
LiPo-QP adapter	14.26 [10.24, 18.48]	11.93 [5.97, 17.86]	8.99 [5.16, 12.38]	8.87 [4.24, 13.25]	9.14 [4.84, 13.08]	4.50 [-1.93, 10.85]	10.86 [6.73, 14.41]	7.21 [2.18, 12.34]	10.84 [8.81, 12.77]	8.17 [5.35, 10.97]
GROOVE	**35.32** [31.65, 38.65]	**46.60** [42.73, 49.53]	**29.00** [24.92, 33.07]	**39.47** [35.28, 43.66]	**31.03** [26.56, 35.24]	**41.80** [37.39, 45.66]	**36.44** [32.47, 40.54]	**45.54** [41.53, 49.82]	**33.02** [31.02, 34.97]	**43.42** [41.38, 45.36]

Success rate (%).

Method	Spatial	Object	Goal	LIBERO-10	Overall
Raw	96.00 [92.00, 100.00]	97.00 [94.00, 100.00]	95.00 [90.00, 100.00]	87.00 [72.00, 98.00]	93.75 [89.75, 97.00]
RTC	97.00 (+1.00) [94.00, 100.00]	99.00 (+2.00) [97.00, 100.00]	94.00 (-1.00) [86.00, 100.00]	90.00 (+3.00) [75.00, 100.00]	95.00 (+1.25) [90.75, 98.25]
SEAM ($\lambda = .05$)	94.00 (-2.00) [89.00, 99.00]	96.00 (-1.00) [93.00, 99.00]	97.00 (+2.00) [94.00, 100.00]	90.00 (+3.00) [81.00, 97.00]	94.25 (+0.50) [91.50, 96.75]
SEAM ($\lambda = .10$)	**98.00** (+2.00) [95.00, 100.00]	96.00 (-1.00) [92.00, 100.00]	97.00 (+2.00) [93.00, 100.00]	90.00 (+3.00) [83.00, 96.00]	95.25 (+1.50) [92.75, 97.50]
SEAM ($\lambda = .20$)	**98.00** (+2.00) [95.00, 100.00]	**100.00** (+3.00) [100.00, 100.00]	**98.00** (+3.00) [94.00, 100.00]	93.00 (+6.00) [87.00, 98.00]	**97.25** (+3.50) [95.25, 99.00]
POTR	96.00 (+0.00) [93.00, 99.00]	99.00 (+2.00) [97.00, 100.00]	95.00 (+0.00) [89.00, 100.00]	92.00 (+5.00) [84.00, 99.00]	95.50 (+1.75) [92.75, 97.75]
Overlap ensemble	96.00 (+0.00) [92.00, 100.00]	99.00 (+2.00) [97.00, 100.00]	96.00 (+1.00) [89.98, 100.00]	**94.00** (+7.00) [88.00, 99.00]	96.25 (+2.50) [93.75, 98.25]
LiPo-QP adapter	**98.00** (+2.00) [94.00, 100.00]	99.00 (+2.00) [97.00, 100.00]	97.00 (+2.00) [93.00, 100.00]	90.00 (+3.00) [78.00, 98.00]	96.00 (+2.25) [92.50, 98.75]
GROOVE	94.00 (-2.00) [89.00, 99.00]	**100.00** (+3.00) [100.00, 100.00]	95.00 (+0.00) [90.00, 100.00]	**94.00** (+7.00) [88.00, 99.00]	95.75 (+2.00) [93.25, 97.75]

paired percentage-point changes and use an exact McNemar test.

The primary physical analysis includes measured replanning delays. For the timing sensitivity analysis, we rescale each endpoint by its mean command interval raised to the derivative order, using three for jerk, two for joint acceleration, and one for current slew. We apply this standardization to full trials and to strict-interior intervals that exclude the intervals immediately before and after each replan.

V. Results and Analysis

Q1. How Does GROOVE Affect EEF Jerk and Task Success?

Table I summarizes jerk reduction and task success on the shared holdout. Parentheses show paired success-rate changes from raw execution, and bold and underlined values mark the best and second-best point estimates, respectively. Figure 4 shows a representative replan and its matched executed path. GROOVE achieves the largest translational and rotational jerk reductions for every LIBERO suite and overall, reducing aggregate EEF jerk by 33.02% and 43.42%, respectively. The strongest SEAM setting, $\lambda = 0.20$, reaches 22.05% and 25.95% jerk reduction with 97.25% task success. The reductions from GROOVE are 1.50 and 1.67 times as large under the same policy-query and command budgets. Task success increases from 93.75% for raw execution to 95.75% with GROOVE, a paired change of $+2.00$ percentage points with a 95% confidence interval of $[-0.25, 4.50]$. Among the 24 discordant pairs, 16 succeed only with GROOVE and 8 succeed only with raw execution, with an exact McNemar p-value of 0.152. Over the first 50 commands, GROOVE reduces translational and rotational jerk by 39.19% $[35.96, 42.69]$% and 49.08% $[46.29, 51.84]$%, respectively. All 400 assignments are completed for each direct comparator, with no solver fallbacks.

On the development grids, the same frozen regulator settings reduce translational and rotational jerk by 32.45% and 43.91% on $\pi_{0.5}$ and by 37.02% and 53.42% on GR00T N1.7. Task success changes by $+0.25$ pp and $+0.75$ pp, respectively.

Q2. Do the Gains Persist on the Physical Robot?

On the UR5e, GROOVE reduces TCP jerk and actuator-demand variation (Table II and Fig. 5).

TABLE II

UR5e reductions relative to raw execution.

Metric	Reduction [95% CI]
Translational TCP jerk	16.39% [11.63, 20.54]%
Rotational TCP jerk	19.49% [16.74, 21.95]%
TCP travel	6.89% [0.23, 12.74]%
Joint acceleration	15.07% [9.59, 19.53]%
Joint-current slew	29.09% [27.24, 30.82]%

Across 9-, 11-, and 15-sample Savitzky–Golay windows, the observed reductions are 16.01–16.59% in translational

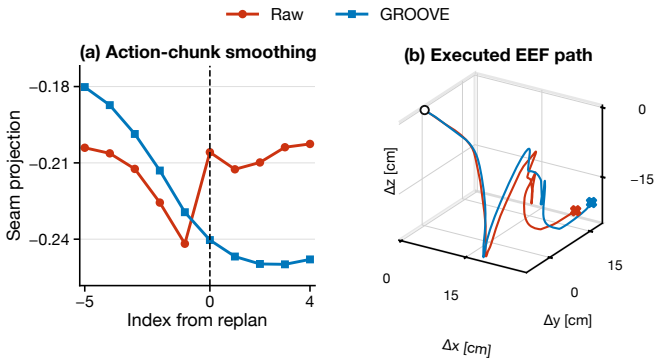

Fig. 4. Representative LIBERO pair nearest the median translational-jerk reduction. (a) Raw and delivered GROOVE delta-EEF commands projected onto the unit direction of the raw boundary jump. (b) Start-relative executed EEF paths. Circles and crosses denote starts and ends.

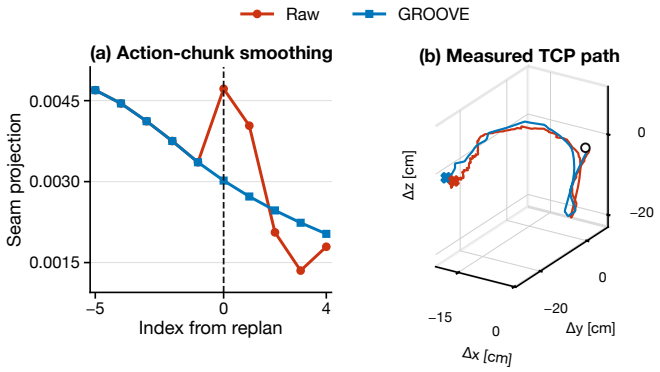

Fig. 5. Representative UR5e transport pair nearest the task-median translational-jerk reduction. (a) Raw and delivered GROOVE delta-EEF commands projected onto the unit direction of the raw boundary jump. (b) Start-relative measured TCP paths. Circles and crosses denote starts and ends.

TCP jerk, 17.89–21.88% in rotational TCP jerk, and 25.79–34.31% in joint-current slew.

Task success changes from 46% to 52%, yielding +6.0 pp with a 95% confidence interval of $[-14.0, 26.0]$ pp and an exact McNemar p-value of 0.690. Stacking improves by +12.0 pp, while long-range transport remains unchanged.

Timing sensitivity. Across the full trials, the mean command interval increases from 289.9 ms to 310.3 ms, a difference of 7.1%. The full-trial cadence-standardized estimates are -1.74% $[-8.17, 3.72]$% for translational jerk and 2.03% $[-2.11, 5.85]$% for rotational jerk, with both confidence intervals including zero. The primary effects in Table II include the deployed system's timing. We examine motion between replans separately.

A post-hoc strict-interior analysis excludes the intervals immediately before and after each replan. In the remaining intervals, the command-interval difference is 0.5%. After cadence standardization, translational and rotational jerk remain lower by 14.97% $[9.49, 19.84]$% and 18.65% $[15.39, 21.83]$%, respectively, while joint-current slew decreases by 29.44% $[27.51, 31.24]$%. Uniform time dilation at the observed interval ratio predicts only 1.17% for jerk and 0.39% for current slew, below the measured reductions.

Q3. Ablation Studies and Motion Fidelity

Candidate-set composition. Table III reports jerk reductions relative to the follow-up raw rerun, which achieves 95.50% success. Relative to the cube alone, adding three coordinate axes or four body diagonals reduces translational jerk by 2.72% and 3.73%, respectively. The full set reduces it by 5.33% $[3.25, 7.24]$. Relative to the full set, the four-diagonal subset has 1.69% $[-0.39, 3.73]$ higher translational jerk and a success difference of +0.25 $[-1.25, 1.75]$ percentage points. The full set's incremental benefit over four diagonals remains uncertain in LIBERO. Selection changes translation only, so rotational differences arise through closed-loop replanning.

Deviation allowance. In the 400-triplet rollout, the full set reduces translational jerk from cube-only execution by 1.60% $[0.08, 3.22]$ at $\rho_B = 1$ and 6.33% $[4.27, 8.41]$ at $\rho_B = 1.5$. Success is 96.0% and 95.5%, respectively, versus 95.5% for cube-only execution.

Matched-deviation replay. Across 11,356 matched replans (87.5%) from all 400 episodes, the full set lowers command-space jerk by 3.28% $[2.92, 3.65]$ relative to cubes with matched realized prefix deviation.

TABLE III

Candidate-set ablation on the $\pi_{0.5}$ LIBERO follow-up.

Setting	No. of candidates	SR [%]	ΔJ_{tr} [%] 95% CI	ΔJ_{rot} [%] 95% CI
Cube ($k = 0$)	1	95.50	29.28 [27.03, 31.39]	42.02 [39.68, 44.28]
Cube + 3 axes	4	94.50	31.21 [29.04, 33.39]	42.47 [40.34, 44.54]
Cube + 4 body diagonals	5	96.25	31.92 [29.67, 34.19]	42.91 [40.63, 45.18]
Full set	14	96.00	33.05 [31.06, 35.03]	43.74 [41.55, 45.93]

Candidate selection. On the $\pi_{0.5}$ development grid, the full set yields lower translational and rotational jerk than one random direction, and protected selection yields lower jerk than random eligible selection (Table IV, first two rows).

TABLE IV

Selection (development grid) and fidelity (fixed-50 holdout) ablations with $\pi_{0.5}$.

Contrast	ΔJ_{tr} [%] 95% CI	ΔJ_{rot} [%] 95% CI	ΔSR [pp] 95% CI
Full set vs. random direction	+7.83 [6.95, 8.63]	+5.85 [4.64, 6.98]	−0.75 [−1.88, 0.38]
Protected selector vs. random eligible	+3.29 [2.06, 4.53]	+2.66 [1.47, 3.89]	+0.25 [−1.00, 1.50]
Every-prefix vs. per-command	+10.28 [8.29, 12.25]	+2.75 [−1.27, 6.46]	+3.25 [0.00, 7.00]

History and fidelity constraints. On the separate 400-pair fixed-50-command holdout, the cube regulator reduces translational jerk by 38.52% $[35.58, 41.59]$ relative to raw execution and by 16.14% $[12.93, 19.37]$ and 16.53% $[13.40, 19.70]$ relative to budget-matched EMA and One Euro filters. The every-prefix guard reduces translational

jerk by 10.28% relative to the per-command guard. The rotational reduction is 2.75%, with a 95% interval spanning zero (Table IV, last row). A comparison with the no-history control shows a descriptive 35.20% $[31.53, 39.05]$ difference in favor of history conditioning. The endpoint-only guard lowers jerk by 25.14% relative to the every-prefix regulator but increases mean realized prefix deviation per replan by 90.7% and lowers success by 4.75 percentage points, illustrating the tradeoff between jerk reduction and motion fidelity.

Q4. What Is the Runtime Overhead?

With the shared-rotation implementation (Sec. III), mean delays from policy output to command readiness are 2.3 ms for raw execution and 155.5 ms for GROOVE. The 95th percentile of this delay for GROOVE is 301.2 ms. These delays include regulation, diagnostics, logging, and runtime contention. Exact-input replay of all 2,464 replans on the same A100 host reproduces every action. The regulator core averages 56.9 ms, with a 95th percentile of 69.0 ms (24.2% of the nominal command period). No solver failure or raw fallback occurs during deployment.

VI. CONCLUSION

GROOVE regulates frozen VLA chunks without retraining or additional inference, reducing boundary and within-chunk jerk under every-prefix translation and local-coordinate rotation bounds. It yields the largest jerk reductions among the evaluated methods on the shared LIBERO holdout and lowers measured TCP jerk and current slew across 50 matched UR5e pairs. Both evaluations show higher aggregate task-success point estimates. We will make the code and supporting materials publicly available upon publication.

ACKNOWLEDGMENTS

This work was supported by Samsung's Research Funding for Future Technology Program. It was also supported by the AI Star Fellowship Program funded by the Ministry of Science and ICT (MSIT), Republic of Korea.

REFERENCES

[1] M. J. Kim *et al.*, "OpenVLA: An open-source vision-language-action model," in *Proceedings of the 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 6–9 November 2025, pp. 2679–2713. [Online]. Available: https://proceedings.mlr.press/v270/kim25c.html

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023.

[3] K. Black *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," in *Proceedings of The 9th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, 27–30 September 2025, pp. 17–40. [Online]. Available: https://proceedings.mlr.press/v305/black25a.html

[4] NVIDIA *et al.*, "GR00T N1: An open foundation model for generalist humanoid robots," arXiv preprint arXiv:2503.14734, 2025.

[5] K. Sendai, M. Alvarez, T. Matsushima, Y. Matsuo, and Y. Iwasawa, "Leave no observation behind: Real-time correction for VLA action chunks," arXiv preprint arXiv:2509.23224, 2025.

[6] D. Zhan, X. Xu, J. Li, and J. Tang, "SEAM: Smooth execution of action-chunked motion for vision-language-action policies," arXiv preprint arXiv:2607.04609, 2026.

[7] K. Black, M. Y. Galliker, and S. Levine, "Real-time execution of action chunking flow policies," in *Advances in Neural Information Processing Systems*, vol. 38, 2025, pp. 37 596–37 620.

[8] K. Fang, H. Pei, and X. Chi, "Smoother action chunking flow policy via prior-corrected orthogonal trust-region guidance," arXiv preprint arXiv:2605.24433, 2026.

[9] D. Son and S. Park, "LiPo: A lightweight post-optimization framework for smoothing action chunks generated by learned policies," *International Journal of Control, Automation and Systems*, vol. 23, no. 11, pp. 3284–3292, 2025.

[10] R. Béarée, "New damped-jerk trajectory for vibration reduction," *Control Engineering Practice*, vol. 28, pp. 112–120, 2014.

[11] J.-E. Lee, A. Bylard, R. Sun, and L. Sentis, "On the performance of jerk-constrained time-optimal trajectory planning for industrial manipulators," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 9772–9778.

[12] Z. Li, H. Yang, Z. Chen, Y. Chen, Y. C. Lin, and C. Li, "From inference efficiency to embodied efficiency: Revisiting efficiency metrics for vision-language-action models," arXiv preprint arXiv:2603.19131, 2026.

[13] J. Li, X. Shi, H. Xie, M. Shang, and Y. Lu, "SmoothVLA: Aligning vision-language-action models with physical constraints via intrinsic smoothness optimization," arXiv preprint arXiv:2603.13925, 2026.

[14] G. Casiez, N. Roussel, and D. Vogel, "1 € Filter: A simple speed-based low-pass filter for noisy input in interactive systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Austin, Texas, USA: ACM, May 2012, pp. 2527–2530.

[15] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[16] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," in *Proceedings of Robotics: Science and Systems*, Virtual, Jul. 2021.

[17] Y. Liu, J. I. Hamid, A. Xie, Y. Lee, M. Du, and C. Finn, "Bidirectional decoding: Improving action chunking via guided test-time sampling," in *International Conference on Learning Representations*, 2025.

[18] Y. Liu *et al.*, "Learning native continuation for action chunking flow policies," in *Proceedings of Robotics: Science and Systems*, 2026. [Online]. Available: https://roboticsconference.org/program/papers/58/

[19] K. Wang, Y. Zheng, Y. Zheng, J. Zhao, and W. Ding, "Learning high-frequency continuous action chunks in latent space," in *Proceedings of the 43rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 306. PMLR, 2026. [Online]. Available: https://sjtu-zhao-lab.github.io/RTR/

[20] Z. Yang, Y. Shi, M. Yao, W. Xue, Y. Jueluo, and L. Liu, "ChunkFlow: Towards continuity-consistent chunked policy learning," arXiv preprint arXiv:2607.12992, 2026.

[21] T.-B. Ho *et al.*, "Start right, arrive right: Asynchronous execution via initial noise selection," arXiv preprint arXiv:2606.19774, 2026.

[22] M. Park *et al.*, "ACG: Action coherence guidance for flow-based vision-language-action models," in *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026. [Online]. Available: https://davian-robotics.github.io/ACG/

[23] J. Nie *et al.*, "PACE: Phase-aware chunk execution for robot policies with action chunking," arXiv preprint arXiv:2606.00537, 2026.

[24] Y. Liang *et al.*, "Adaptive action chunking at inference-time for vision-language-action models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2026, pp. 20 802–20 811.

[25] C. Yang, Y. Hu, Y. Ma, Y. Yang, J. Tan, and H. Fan, "Realtime-VLA V2: Learning to run VLAs fast, smooth, and accurate," arXiv preprint arXiv:2603.26360, 2026.

[26] B. Liu *et al.*, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 44 776–44 791.

[27] NVIDIA, "GR00T-N1.7-LIBERO," Hugging Face model card and checkpoint, 2026, accessed August 29, 2026. [Online]. Available: https://huggingface.co/nvidia/GR00T-N1.7-LIBERO

[28] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.