KI-VLA: KINEMATICS-INFORMED ACTION DECODING FOR FLOW-MATCHING ROBOT POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching robot policies can generate operational-space action chunks without explicitly modeling the underlying joint-space motion required for execution. As a result, decoded chunks can contain jitter, micro-reversals, and locally inconsistent commands that degrade closed-loop performance. We introduce Kinematics-Informed VLA (KI-VLA), a predictor-corrector decoder that treats the flow policy's intermediate end-effector action chunks as noisy kinematic observations, and uses them to infer a smooth latent joint-increment trajectory. A kinematic observation model and Kalman/RTS smoothing infer temporally coherent joint motion, which is mapped back to action space via bounded trust-region updates. By interleaving these corrections with flow-matching steps, KI-VLA refines intermediate proposals while keeping the base policy frozen. On RoboTwin 2.0, KI-VLA improves the π_0 baseline from 47.5% to 54.6% in the benchmark's Easy setting and from 26.8% to 32.6% in its Hard setting. On real hardware, clean-condition success is comparable (68.5% to 71.0%), while success under a controlled disturbance injected into the action proposal rises from 38.5% to 57.0%. Further studies show that the gains are concentrated in the few-step decoding regime and can diminish under severe measurement-covariance or process-noise mis-scaling, clarifying the operating regime in which decoding-time kinematic correction is most useful.

1 INTRODUCTION

Vision-language-action (VLA) policies map visual observations and language instructions to robot actions. To support multimodal behavior and reduce the need to recompute actions at every control step, recent policies generate action chunks with diffusion or flow-matching decoders (Ghosh et al., 2024; Chi et al., 2025; Black et al., 2025b;a), in contrast to earlier models that emit one discretized action per step (Brohan et al., 2023b;a; Kim et al., 2025; O'Neill et al., 2024). Although chunking supports long-horizon imitation learning, iterative decoding can produce jitter, micro-reversals and discontinuities between consecutive actions. These artifacts could be weakly penalized by offline training but can accumulate during closed-loop execution (Zhao et al., 2023; Ross et al., 2011). The challenge is especially clear for end-effector delta actions where the policy specifies motion in Cartesian space, while the robot executes it through joint motion, whose relation to Cartesian motion varies with configuration (Khatib, 1987; Lynch & Park, 2017). Thus, smoothing a decoded action chunk requires more than temporal consistency; it should also consider how the proposed end-effector motion maps to the robot's joints.

Applying a fixed smoother to the final action chunk can suppress useful rapid motions along with unwanted jitter. Existing work addresses action coherence via post-optimization, execution scheduling, and test-time guidance (Son & Park, 2025; Black et al., 2025c; Park et al., 2025), while FAST changes the action representation used during training (Pertsch et al., 2025). For flow-matching policies, this motivates refining the action chunk during decoding when intermediate samples are available. The key distinction from generic action-space smoothing is to regularize decoded actions through a configuration-dependent mapping to latent joint motion rather than filtering Cartesian commands directly. Such correction should also remain conservative so that kinematic regularization does not overwrite the task intent encoded by the frozen policy.

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.

In response, we propose Kinematics-Informed VLA (KI-VLA), a predictor-corrector decoder for flow-matching robot policies. It treats the policy's end-effector action chunk as a noisy kinematic observation of an underlying joint-increment trajectory. A state-dependent Jacobian connects joint increments to end-effector motion. Kalman filtering (Kalman, 1960) and Rauch-Tung-Striebel (RTS) smoothing (Rauch et al., 1965) then infer a temporally coherent trajectory. A trust-region update bounds the correction in action space. The corrector can be applied between Euler flow updates without retraining the VLA backbone.

As a result, KI-VLA addresses within-chunk jitter and locally inconsistent operational-space deltas using differential kinematics and a constant-velocity temporal prior. It does not model full dynamics, contact, friction, collision avoidance, torque feasibility or hardware safety. Our contributions are:

- We use a Jacobian observation model and differentiable Kalman/RTS inference to estimate smooth joint-increment trajectories, then map them back to action space with bounded corrections.
- We interleave correction with flow-matching updates and support sparse schedules that trade computation for performance, while keeping the base policy frozen.
- We evaluate the controlled end-pose VLA comparison on eight RoboTwin 2.0 tasks and four real-robot tasks, measuring task success and command-level motion.
- We examine uncertainty and sensitivity, report confidence intervals, sweep the number of function evaluations (NFE), and analyze when the corrector helps and where its assumptions limit it.

2 Related Work

VLA policies and chunked control. VLA policies map visual and language observations to robot actions (Brohan et al., 2023b;a; Jiang et al., 2023; Kim et al., 2025; O'Neill et al., 2024), with related systems grounding language in pretrained skills or high-level plans (Ichter et al., 2023; Driess et al., 2023); those that also condition on proprioceptive state and emit multi-step action chunks (Ghosh et al., 2024; Zhao et al., 2023; Black et al., 2025b) reduce the effective control horizon. RTC and FAST address high-frequency execution and action representation for chunked robot control (Black et al., 2025c; Pertsch et al., 2025). KI-VLA is complementary: it keeps the trained policy fixed and refines the decoded operational-space chunk through a latent kinematic model.

Generative robot action decoders. Diffusion and flow-matching policies model multimodal action distributions through iterative generation (Chi et al., 2025; Ho et al., 2020; Song et al., 2021; Karras et al., 2022; Lipman et al., 2023; Black et al., 2025b; Gao et al., 2025). Because flow-matching inference exposes explicit intermediate samples through an ODE-style trajectory, it offers natural insertion points for predictor-corrector refinement. Our method exploits this structure rather than modifying only the final chunk.

Test-time refinement and classical robotics. Temporal filters, trajectory smoothing, inverse kinematics (IK), and operational-space control are classical tools for stabilizing robot motion (Khatib, 1987; Lynch & Park, 2017; Flash & Hogan, 1985; Berscheid & Kröger, 2021; Ma et al., 2021). Learning-based post-optimization and guidance methods similarly encourage smoother decoded trajectories (Son & Park, 2025; Park et al., 2025). Our Kalman measurement update is itself equivalent to a regularized implicit-IK step (Appendix B.1); thus classical estimation is a component of the method rather than an unrelated competitor. We directly compare learned and analytic Jacobians in Sec. 4.4, where the analytic model provides a strong reference for assessing the role of J_θ.Our Kalman measurement update is itself equivalent to a regularized implicit-IK step (Appendix B.1); thus classical estimation is a component of the method rather than an unrelated competitor. We directly compare learned and analytic Jacobians in Sec. 4.4, where the analytic model provides a strong reference for assessing the role of J_θ.

Structured priors and physics-informed learning. Physics-informed neural networks inject governing equations through training losses (Raissi et al., 2019), while hybrid approaches such as APHYNITY augment incomplete physical dynamics with learned residuals (Yin et al., 2021), and Phy-SSM incorporates partially known dynamics into state-space models with physics regularization (Wang et al., 2025). Physics-Based Flow Matching (PBFM) instead enforces physical constraints during flow-model training to balance physical and distributional objectives (Baldan et al., 2026). Our use of structure is intentionally narrower: KI-VLA leaves the base VLA frozen and injects

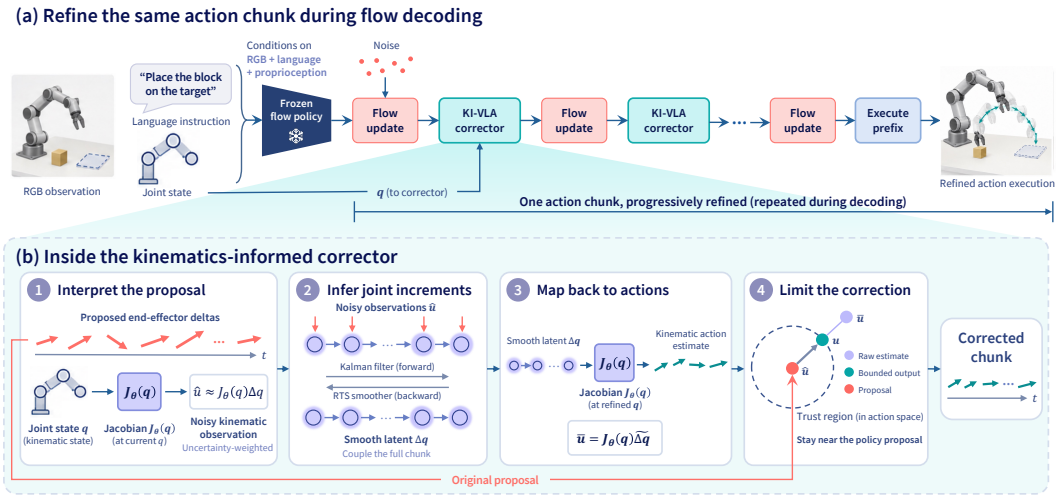


Figure 1: Overview of KI-VLA. **(a)** A frozen flow-matching policy progressively decodes an operational-space action chunk from multimodal observations, while KI-VLA repeatedly refines intermediate proposals before receding-horizon execution. **(b)** At each correction step, the proposed end-effector deltas are treated as uncertainty-weighted noisy observations of latent joint motion through the learned kinematic model $J_\theta(q)$. Kalman filtering and RTS smoothing infer a smooth latent joint trajectory, which is mapped back to operational space and conservatively combined with the original policy proposal through a trust-region update.

only differential-kinematic and temporal structure at inference through an observation model and Bayesian smoothing. Thus, it is complementary to methods that learn or enforce richer dynamics during training rather than a replacement for them.

3 Method

A flow-matching VLA decodes an operational-space action chunk, e.g., end-effector translation and rotation deltas. Such chunks are task-aligned but can contain jitter, reversals, or locally inconsistent motions. Instead of applying fixed smoothing directly in action space, KI-VLA treats the decoded deltas as noisy observations of an underlying smooth joint-increment trajectory. A learned Jacobian maps this latent joint motion to operational-space deltas, Kalman/RTS smoothing infers a temporally coherent trajectory, and a trust-region update projects the result back to the original action space while staying close to the base policy proposal. The following subsections formalize the setup, the learned kinematic model, the corrector, and its interleaving with flow-matching decoding, with an overview shown in Fig. 1.

3.1 Problem Setup

We consider receding-horizon imitation learning with chunked execution. At control step t, the robot receives an observation: $o_t = \{I_t, \text{language}, s_t\}$, where I_t denotes camera observations and $s_t \in \mathbb{R}^{d_s}$ is the proprioceptive state. Let $q_t \in \mathbb{R}^n$ denote the arm joint configuration extracted from s_t. A flow-matching policy predicts an action chunk $\mathbf{A}_t = [a_t, \dots, a_{t+H-1}]$, $a_\tau \in \mathbb{R}^{d_a}$, over horizon H. We decompose each action as $a_\tau = [u_\tau \;\; g_\tau]$, where $u_\tau \in \mathbb{R}^\alpha$ is an operational-space motion delta, consisting of translation and axis-angle rotation, and g_τ denotes gripper commands. For a single end effector, $\alpha = 6$.

Our goal is to improve test-time action decoding without changing the underlying flow-matching policy. Given an intermediate action chunk $\widehat{\mathbf{A}}_t$, we introduce a differentiable corrector $\mathcal{C}_{\theta,\phi}(q_t, \widehat{\mathbf{A}}_t, \xi)$, where $\xi \in [0, 1]$ denotes normalized decoding progress. At decoding step s, the flow predictor first proposes

$$\widehat{\mathbf{A}}_t^{(s)} = \mathcal{P}_\psi^{\mathrm{FM}}\left(\mathbf{A}_t^{(s-1)}, o_t, \tau_s\right), \tag{1}$$

and the corrector refines it as

$$\mathbf{A}_t^{(s)} \leftarrow \mathcal{C}_{\theta,\phi}\Big(q_t, \widehat{\mathbf{A}}_t^{(s)}, \xi_s\Big). \tag{2}$$

In our implementation, the flow time follows the π_0-style convention where $\tau = 1$ is noise and $\tau = 0$ is data, and we set $\xi_s = 1 - \tau_s$ so that ξ_s increases as the sample becomes more action-like. The remainder of this section describes the learned kinematic model, the latent smoothing-based corrector, and the predictor-corrector decoding procedure.

3.2 Kinematics-Informed Observation Model

The corrector connects operational-space motion deltas to joint-space behavior through differential kinematics, $v_e = J(q)\dot{q}$, where $J(q) \in \mathbb{R}^{\alpha\times n}$ is the geometric Jacobian (Khatib, 1987; Lynch & Park, 2017). With control period Δt and finite difference $\Delta q_t = q_{t+1} - q_t$, we use the discrete approximation $u_t \approx J(q_t)\Delta q_t$.

Rather than requiring an analytic Jacobian online, we learn a differentiable surrogate $J_\theta : \mathbb{R}^n \to \mathbb{R}^{\alpha\times n}$. This provides a lightweight observation model trained from logged joint states and end-effector pose changes, while avoiding an online analytic Jacobian or inverse-kinematics solver.

In practice, J_θ is parameterized as a lightweight MLP that maps normalized joint angles to a vector in $\mathbb{R}^{\alpha n}$. Given end-effector pose (p_t, R_t), we define the pose-derived operational-space target

$$u_t^{\text{data}} \triangleq \begin{bmatrix} p_{t+1} - p_t \\ \text{Log}(R_t^\top R_{t+1}) \end{bmatrix} \in \mathbb{R}^\alpha, \tag{3}$$

where $\text{Log} : SO(3) \to \mathbb{R}^3$ is the standard rotation log map (Lynch & Park, 2017).

We train J_θ with a kinematic-consistency objective and stability regularizers, thereby injecting known differential-kinematic structure into the learned observation model without modeling the full robot dynamics:

$$\mathcal{L}_{\text{kin}}(\theta) = \mathbb{E}_t\big[\|u_t^{\text{data}} - J_\theta(q_t)\Delta q_t\|_2^2\big], \tag{4}$$

$$\mathcal{L}_{\text{smooth}}(\theta) = \mathbb{E}_t\big[\|\nabla_q J_\theta(q_t)\|_F^2\big], \tag{5}$$

$$\mathcal{L}_{\text{cond}}(\theta) = -\mathbb{E}_t\big[\log\det(J_\theta(q_t)J_\theta(q_t)^\top + \epsilon I)\big], \tag{6}$$

$$\mathcal{L}_J(\theta) = \mathcal{L}_{\text{kin}}(\theta) + \lambda_{\text{reg}}\Big(\mathcal{L}_{\text{smooth}}(\theta) + \beta\,\mathcal{L}_{\text{cond}}(\theta)\Big). \tag{7}$$

Here, $\mathcal{L}_{\text{kin}}$ fits a local linearization from joint increments to pose-derived task-space motion, while $\mathcal{L}_{\text{smooth}}$ and $\mathcal{L}_{\text{cond}}$ stabilize relinearization by discouraging abrupt Jacobian variation and near-singular mappings. Appendix C.1 provides formal justification. In the remainder of the paper, J_θ serves as the observation operator linking decoded operational-space actions to latent joint-space increments.

3.3 Kinematics-Informed Action Corrector

A learned Jacobian alone does not ensure temporal coherence across an action chunk. We therefore interpret a candidate motion chunk as noisy measurements of a smooth trajectory. The corrector (i) builds a kinematic observation model from J_θ, (ii) infers a temporally coupled latent trajectory with Kalman filtering and RTS smoothing, and (iii) maps the result back to operational space under a trust-region constraint.

Latent joint-increment smoother. For each horizon index $k \in \{0, \dots, H-1\}$, we define

$$x_{t+k} = \begin{bmatrix} \Delta q_{t+k} \\ \Delta\dot{q}_{t+k} \end{bmatrix} \in \mathbb{R}^{2n}, \qquad x_{t+k+1} = Ax_{t+k} + w_{t+k}, \qquad w_{t+k} \sim \mathcal{N}(0, Q), \tag{8}$$

with

$$A = \begin{bmatrix} I & \Delta t\, I \\ 0 & I \end{bmatrix}.$$

This constant-velocity prior encourages smooth latent joint increments while allowing deviations through process noise.

Given a candidate motion chunk $\hat{u}_{t:t+H-1}$, we treat each $\hat{u}_{t+k}$ as a noisy observation:

$$z_{t+k} \triangleq \hat{u}_{t+k} = H_{t+k} x_{t+k} + \eta_{t+k}, \qquad \eta_{t+k} \sim \mathcal{N}(0, \Sigma^u_{t+k}), \tag{9}$$

where

$$H_{t+k} = [J_{t+k} \quad 0], \qquad J_{t+k} \triangleq J_\theta(\bar{q}_{t+k}). \tag{10}$$

Here $\bar{q}_{t+k}$ is the nominal joint configuration for linearization. We initialize $\bar{q}_t = q_t$ and update it between relinearization iterations by rolling the corrected increments forward, so the Jacobian is re-evaluated along the corrected motion while remaining fixed within each smoothing pass.

Implicit inverse kinematics. The Kalman measurement update for the Δq block is equivalent to a regularized least-squares implicit IK step:

$$\Delta q^+_{t+k} = \arg\min_{\Delta q} \left\| \hat{u}_{t+k} - J_{t+k} \Delta q \right\|^2_{(\Sigma^u_{t+k})^{-1}} + \left\| \Delta q - \mu^-_{t+k} \right\|^2_{(P^-_{t+k})^{-1}}, \tag{11}$$

where (μ^-_{t+k}, P^-_{t+k}) are the predicted mean and covariance before the measurement update. Thus, the corrector infers the joint-space motion that best explains the decoded operational-space actions under both a kinematic observation model and a smooth trajectory prior. Appendix B.1 gives the proof.

Uncertainty-aware measurement model. To avoid trusting all decoded actions equally, we parameterize a diagonal measurement covariance:

$$\Sigma^u_{t+k} = s(\xi_s) \cdot \mathrm{diag}\Big(\mathrm{softplus}(h_\phi(\varphi_{t+k})) + \epsilon\Big), \tag{12}$$

where h_ϕ is a lightweight uncertainty module and $s(\xi_s)$ modulates measurement trust according to decoding progress. The reliability feature vector is

$$\varphi_{t+k} = \Big[\bar{q}_{t+k},\ \|\hat{u}_{t+k}\|_2,\ \|\hat{u}_{t+k} - \hat{u}_{t+k-1}\|_2,\ \kappa_{t+k}\Big], \tag{13}$$

with

$$\kappa_{t+k} = -\log\det(J_{t+k} J^\top_{t+k} + \epsilon I).$$

κ_{t+k} is a kinematic log-volume proxy that grows as the reachable operational-space volume collapses. It enters φ_{t+k} as an input to h_ϕ, allowing the learned measurement model to adapt its covariance according to local kinematic degeneracy. Appendix B.2 explains the interpretation of this proxy and how measurement covariance affects posterior trust.

RTS smoothing and projection back to action space. Within each iteration we evaluate $H_{t+k} = [J_\theta(\bar{q}_{t+k})\ 0]$ and Σ^u_{t+k} at the current nominal trajectory and hold them fixed for the entire smoothing pass, so the model is linear-Gaussian and Rauch-Tung-Striebel smoothing returns the exact fixed-interval MAP estimate for that linearization:

$$\tilde{x}_{t:t+H-1} = \mathrm{RTS}\big(z_{t:t+H-1}; A, \{H_{t:t+H-1}\}, Q, \{\Sigma^u_{t:t+H-1}\}\big). \tag{14}$$

RTS smoothing couples the full chunk and suppresses within-chunk jitter; Appendix C.2 shows its MAP interpretation for a fixed linearization. We then roll the corrected increments forward to obtain a new nominal trajectory and repeat the relinearization for $N_{\mathrm{relin}} = 2$ iterations. The resulting procedure is an iterative approximate MAP scheme for the nonlinear observation model, while each RTS pass is exact for its fixed linearization. We extract the refined joint increments as $\tilde{\Delta q}_{t+k} = E\tilde{x}_{t+k}$, $E = [I \quad 0]$, roll out $\tilde{q}_t = q_t$, $\tilde{q}_{t+k+1} = \tilde{q}_{t+k} + \tilde{\Delta q}_{t+k}$, and project back to operational-space:

$$\bar{u}_{t+k} \triangleq J_{t+k} \tilde{\Delta q}_{t+k}. \tag{15}$$

Trust-region correction. Since the learned Jacobian can be inaccurate or ill-conditioned, we bound the final correction around the decoder proposal:

$$\rho^{\mathrm{eff}}_{t+k} \triangleq \lambda_{t+k} \rho_{t+k}, \qquad u_{t+k} = \hat{u}_{t+k} + \mathrm{clip}_{\rho^{\mathrm{eff}}_{t+k}}(\bar{u}_{t+k} - \hat{u}_{t+k}), \tag{16}$$

where $\mathrm{clip}_\rho(\delta) \triangleq \delta \cdot \min\left\{1, \frac{\rho}{\|\delta\|_2 + \epsilon}\right\}$. Here λ_{t+k} is a continuous innovation-dependent modulation factor and ρ_{t+k} is a kinematic-degeneracy-dependent radius. We do not interpret λ_{t+k} as a binary detector of "bad" actions; rather, it continuously adjusts the allowable correction while the trust region bounds the deviation from the base proposal. Appendix B.3 shows that this is the Euclidean projection onto an ℓ_2 ball around the original proposal.

3.4 KINEMATICS-INFORMED PREDICTOR-CORRECTOR DECODING

We combine the flow-matching predictor with the kinematics-informed corrector. At decoding step s, the Euler predictor gives

$$\widehat{\mathbf{A}}_t^{(s)} = \mathcal{P}_\psi^{\mathrm{FM}}(\mathbf{A}_t^{(s-1)}, o_t, \tau_s) = \mathbf{A}_t^{(s-1)} + \delta\, \mathbf{v}_\psi(\mathbf{A}_t^{(s-1)}, o_t, \tau_s), \qquad \delta = \frac{1}{N}, \tag{17}$$

where $\mathbf{v}_\psi$ is the learned flow vector field. We then refine the proposal as

$$\mathbf{A}_t^{(s)} \leftarrow \mathcal{C}_{\theta,\phi}\Big(q_t, \widehat{\mathbf{A}}_t^{(s)}, \xi_s\Big). \tag{18}$$

Starting from $\mathbf{A}_t^{(0)} \sim \mathcal{N}(0, I)$, this yields a predictor-corrector decoding trajectory over $s = 1, \ldots, N$. Applying the corrector during decoding repeatedly regularizes intermediate samples, whereas a final-only variant applies the same correction machinery once after the last flow update. Our current experiments show that repeated decoding-time correction is more effective than a single final correction; however, placement and correction frequency are not fully disentangled by that comparison. Appendix B.4 gives an operator-splitting interpretation.

Sparse correction schedules. To trade compute for robustness, we apply the corrector only at selected decoding steps: $\textsc{ApplyCorr}(s, \xi_s; M, \xi_0) = \mathbf{1}\{s \bmod M = 0\} \cdot \mathbf{1}\{\xi_s \geq \xi_0\}$. The number of corrector calls is $N_{\mathrm{corr}} = \sum_{s=1}^{N} \textsc{ApplyCorr}(s, \xi_s; M, \xi_0)$. Setting $M = N + 1$ disables correction, $(M, \xi_0) = (N, 1)$ gives final-step final correction, and $(M, \xi_0) = (1, 0)$ gives full interleaving.

3.5 TRAINING

Our training pipeline separates the base policy from the corrector. The base flow-matching action decoder is trained with its standard objective on action chunks; details are in Appendix D.1. We then pretrain J_θ using Eq. (7) from pose-derived supervision. This stage does not require analytic inverse kinematics; for real-robot experiments without calibrated online end-effector poses, pose supervision is obtained offline from logged joint states using the available forward-kinematics model.

The uncertainty module h_ϕ is trained by unrolling the corrector on demonstration chunks and minimizing behavior-cloning loss on the corrected actions while keeping the base flow policy fixed. Full training objectives are provided in Appendix D.3.

4 EXPERIMENTS

We evaluate KI-VLA along three primary axes: (i) controlled RoboTwin simulation, (ii) real-robot execution, and (iii) component and schedule ablations. We additionally analyze NFE sensitivity, measurement-covariance and hyperparameter sensitivity, and correction behavior in Appendix F, which delineates when the method helps and where its assumptions become limiting.

4.1 EXPERIMENTAL SETUP

Controlled baseline. Unless otherwise stated, KI-VLA wraps a flow-matching VLA that predicts end-pose / operational-space delta chunks. We denote the frozen controlled baseline by π_0^{EP}. The baseline and KI-VLA use identical observations, training data, backbone, action horizon, and action representation; only decoding-time correction differs. Table 1 additionally includes joint-action methods as benchmark references, but these are not treated as strict head-to-head comparisons because they use a different action space.

Metrics. Task success rate (SR, %) is the primary metric. Hardware experiments also report correction magnitude, peak end-effector velocity/acceleration, joint smoothness and peak joint acceleration. We report the number of trials explicitly and use Wilson 95% intervals or task-level consistency tests where appropriate. These statistics quantify rollout-level evaluation variability for the evaluated checkpoints rather than variability across independently trained policy seeds.

Table 1: Success rates (%) on RoboTwin 2.0 across eight manipulation tasks. Superscript ^{J} denotes joint-angle actions, while ^{EP} denotes end-pose / operational-space delta actions. Joint-action columns reproduce RoboTwin reference results (Chen et al., 2026) and are shown for benchmark context. The primary controlled comparison is between π_0^{EP} and KI-VLA^{EP}, which share the same end-pose action representation and base policy. Bold indicates the better method within the end-pose setting.

	Easy						**Hard**					
	Joint-angle actions				**End-pose actions**		**Joint-angle actions**				**End-pose actions**	
Task	DP^{J}	ACT^{J}	RDT^{J}	π_0^{J}	π_0^{EP}	**KI-VLA^{EP}**	DP^{J}	ACT^{J}	RDT^{J}	π_0^{J}	π_0^{EP}	**KI-VLA^{EP}**
Move Playingcard	47	36	43	53	53	**61**	0	0	11	22	23	**30**
Turn Switch	36	5	35	27	41	**51**	1	2	15	23	30	**35**
Click Bell	54	58	80	44	**35**	33	0	3	9	3	9	**17**
Hang Cup	8	7	23	11	12	**17**	0	0	16	3	5	**13**
Beat Block	42	56	77	43	65	**74**	0	3	37	21	47	**55**
Lift Pot	39	88	72	84	85	**96**	0	0	9	36	60	**65**
Place Shoes	23	5	35	28	56	**64**	0	0	7	6	28	**30**
Diverse Bottle	6	7	2	27	33	**41**	0	0	0	6	12	**16**
Average	31.9	32.8	45.9	39.6	47.5	**54.6**	0.1	1.0	13.0	15.0	26.8	**32.6**

4.2 RoboTwin 2.0 Simulation

We evaluate eight RoboTwin 2.0 tasks under Easy and Hard settings (Chen et al., 2026), using 100 rollouts per task and split. Table 1 reports the full benchmark context: joint-action reference results are shown alongside the end-pose policies, while the primary controlled comparison is π_0^{EP} versus KI-VLA^{EP}. Because the joint-action methods use a different control representation, we use them only to contextualize task difficulty rather than to support head-to-head claims.

Under the controlled end-pose comparison, average SR increases from 47.5% to 54.6% on Easy and from 26.8% to 32.6% on Hard. KI-VLA improves 7/8 Easy tasks and all 8/8 Hard tasks. Click Bell is the only Easy task that degrades ($35\% \rightarrow 33\%$), consistent with a limitation of smoothing-based refinement when a task benefits from brief, sharp motion.

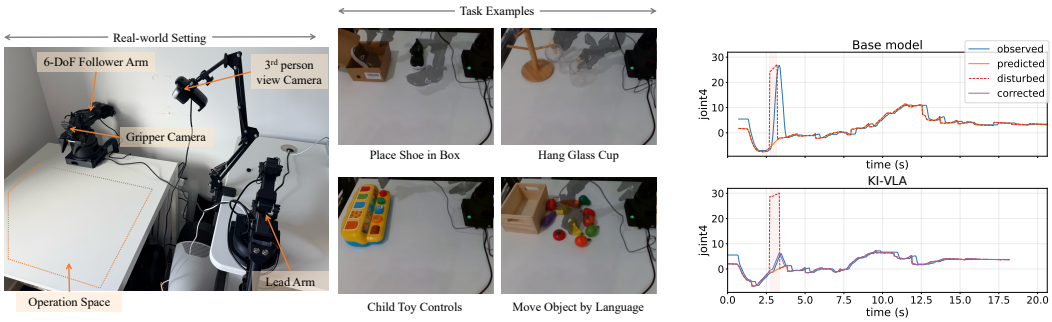
Pooling the 800 rollouts per split gives descriptive two-proportion tests of $p \approx 0.004$ (Easy) and $p \approx 0.010$ (Hard). At the task level, all 8/8 Hard tasks improve (one-sided sign test $p = 0.0039$). We therefore treat the clean RoboTwin benchmark, rather than the injected-disturbance study, as the primary evidence for task-level efficacy.

4.3 Real-Robot Evaluation

We use a 6-DoF Synria Alicia-D arm with a parallel gripper, a front-view RealSense D435i, and a wrist camera. The hardware setup and four evaluation tasks are shown in Fig. 2. Each task is evaluated with 50 trials per condition. App. G gives the four task definitions, their success criteria and per-task variations, and the stress-test probe design. The stress test injects a $+30^\circ$ joint-4 pulse for approximately 0.5 s into the proposal before correction, shortly before target contact. The corrector receives no metadata about the pulse timing, magnitude, or existence, and injected disturbances are not used to train J_θ or h_ϕ.

Table 2 summarizes the task-level results and the variation axis of each task. In clean execution, the average difference is small (68.5% to 71.0%), with gains of up to 4 percentage points across tasks. Under the controlled proposal disturbance, the average gain is much larger (38.5% to 57.0%), with the largest improvements on Place Shoe in Box ($+28$ pp) and Child Toy Controls ($+22$ pp).

Under the clean hardware setting the difference between KI-VLA and the base VLA is small and not statistically significant, so we draw no conclusion about clean-condition performance in either direction; under disturbance, the pooled difference is significant ($p \approx 0.0002$), with Wilson 95% intervals $[32.0, 45.4]$ and $[50.1, 63.7]$ for the base and KI-VLA, respectively. We interpret this disturbance experiment as a controlled robustness probe, not as a model of actuator faults or a substitute for clean-condition evaluation.



(a) Hardware setup and tasks.

(b) Injected-pulse response.

Figure 2: Real-world evaluation. The disturbance is applied to the proposal before KI-VLA correction; the right plot illustrates attenuation of the transient joint-4 deviation.

Table 2: Real-world task success rates (%) with and without an injected joint disturbance. Δ denotes the percentage-point improvement of KI-VLA over the base VLA. Each task is evaluated over 50 trials per condition.

Task	**Primary variation axis**	**Clean rollout**			**With joint-4 disturbance**		
		Base VLA	**KI-VLA**	**Δ**	**Base VLA**	**KI-VLA**	**Δ**
Place Shoe in Box	Left/right shoe	82	**86**	+4	44	**72**	+28
Hang Glass Cup	Handle orientation and height	38	**42**	+4	12	**28**	+16
Child Toy Controls	Button press vs. handle switch	66	**68**	+2	28	**50**	+22
Move Object by Language	Object identity and spatial relation	**88**	**88**	+0	70	**78**	+8
Average	–	68.5	**71.0**	+2.5	38.5	**57.0**	+18.5

Command-level diagnostics in Appendix E.6 show that the corrected command stream is less impulsive under the disturbance: across tasks, peak Cartesian acceleration decreases by 12.8-63.3% and peak joint acceleration by 13.5-42.9%.

4.4 Ablations and What Drives the Gain

Table 3 separates two questions on Turn Switch (100 rollouts/split): whether generic inference-time smoothing can explain the gain, and which internal components of the corrector matter. In Table 3a, fixed smoothing is inconsistent across Easy and Hard, while ACG-style guidance (Park et al., 2025) improves over the base policy but remains below KI-VLA. This suggests that sampling coherence is useful, but generic action-space smoothing or guidance alone does not reproduce the full gain.

Table 3b shows that fixed measurement covariance removes most of the Hard-split improvement, while KF-only inference, removing the trust region, or replacing the kinematic observation with direct action-space smoothing each reduces performance. The analytic URDF Jacobian reaches $50/35$, essentially matching the learned surrogate at $51/35$. We therefore do not attribute the gain to learning J_θ itself; the learned Jacobian is primarily a practical drop-in observation model, while the central mechanism is kinematics-aware latent inference with uncertainty-weighted, bounded correction.

The final-only result is statistically indistinguishable from the no-corrector baseline, so we do not interpret the lower point estimate as evidence that final correction is harmful. Instrumentation instead shows that the final-only pass requests almost no correction; detailed requested-correction and clipping are provided in Appendix F. Since final-only correction uses one call while full interleaving uses multiple calls, this comparison also does not isolate correction placement from its frequency.

4.5 Correction Schedule and Runtime

Table 4 evaluates the compute–performance trade-off obtained by varying the correction schedule while keeping the base policy and corrector components fixed. Moving from no correction to

Table 3: Ablation studies on RoboTwin 2.0 TURN SWITCH (100 rollouts/split). All methods use the same end-pose base VLA. Left: inference-time smoothing and post-processing baselines. Right: internal ablations of the KI-VLA corrector under full interleaving, unless otherwise specified.

(a) Smoothing and post-processing.

Method	**Easy** ↑	**Hard** ↑
Base π_0^{EP} (Black et al., 2025b)	41	30
Moving average	46	29
EMA (Roberts, 1959)	43	31
Jerk-proxy smoothing	46	33
ACG-style guidance (Park et al., 2025)	46	32
Final-only corrector	38	29
KI-VLA (interleaved)	**51**	**35**

(b) Corrector-internal ablations.

Variant	**Easy** ↑	**Hard** ↑
Base π_0^{EP} (Black et al., 2025b)	41	30
Fixed measurement covariance	46	30
Constant $s(\xi) = 1$	47	32
KF only	49	33
No trust region	49	33
Action-space observation	48	31
Analytic URDF Jacobian	50	35
KI-VLA (full)	**51**	**35**

Table 4: Correction-schedule ablation on RoboTwin 2.0 TURN SWITCH. Runtime is corrector compute per decoded chunk, averaged over non-warmup runs. Relative overhead is computed against the measured no-corrector inference and system execution runtime, 28.10 ± 0.23 s/chunk.

Schedule	**Correction schedule**			**Success rate (%)** ↑		**Corrector compute / chunk (ms, mean ± std; overhead)**
	M	ξ_0	N_{corr}	**Easy**	**Hard**	
No corrector	$N+1$	–	0	41	30	0 ± 0 (0.00%)
Final-only corrector	N	1.0	1	38	29	108 ± 16 (0.38%)
Sparse interleaving (2 calls)	5	0.5	2	40	30	183 ± 10 (0.65%)
Sparse interleaving (3 calls)	2	0.5	3	46	33	260 ± 18 (0.93%)
Full interleaving	1	0.0	$N = 10$	**51**	**35**	878 ± 29 (3.12%)

sparse and then full interleaving increases corrector compute from 0 to 878 ± 29 ms/chunk. Sparse interleaving with three calls provides an intermediate operating point ($46/33$ Easy/Hard) at 0.93% relative overhead, while full interleaving achieves the best success rate ($51/35$) at 3.12% overhead in our evaluation stack. The schedule rows vary both correction timing and the number of corrector calls. Accordingly, Table 4 presents a compute-performance trade-off, which shows that repeated decoding-time correction is more effective than a single final pass, but it does not by itself prove that placement alone causes the gain.

Additional robustness and scope analyses. Appendix F evaluates NFE sensitivity, covariance and hyperparameter mis-calibration, and episode-level correction behavior. These analyses sharpen the scope of the main results: the gain is concentrated in few-step decoding and vanishes at $N = 50$ on Turn Switch; severe probabilistic mis-scaling can remove the benefit; and correction statistics do not support interpreting the modulation term as a binary "bad-action" detector. Accordingly, we view KI-VLA as a continuous, kinematics-aware regularizer for efficient flow decoding rather than a universal replacement for high-NFE integration or a discrete error detector.

5 CONCLUSION

We presented KI-VLA, a kinematics-informed predictor-corrector decoder that treats operational-space action chunks as noisy observations of a latent joint-increment trajectory, performs uncertainty-weighted Kalman/RTS inference, and bounds corrections around the frozen policy proposal. Across eight RoboTwin tasks, KI-VLA improves a same-backbone end-pose baseline on both Easy and Hard splits. On real hardware, clean-condition performance remains statistically inconclusive, while success under a controlled proposal-level disturbance increases from 38.5% to 57.0% ($p \approx 0.0002$). Together with the component, schedule, and sensitivity analyses, these results support KI-VLA as a bounded kinematics-aware regularizer for flow-based operational-space decoding. Extending the framework to richer dynamics- or contact-aware structure remains an important direction for future work.

Reproducibility Statement

The main text specifies the controlled baseline, evaluation trial counts, and all reported success rates. Appendix D gives the flow-matching instantiation, decoding algorithm, training objectives, module architectures, and hyperparameters; Appendix E gives baseline definitions, schedules, task protocols, metrics, compute resources, and runtime measurement procedures. The submission is intended to include anonymized code for the corrector, Jacobian training, schedules, ablations, and metric computation.

Ethics Statement

KI-VLA is intended for controlled robot-manipulation research. Improving action robustness can reduce brittle execution failures, but unsafe deployment may cause physical damage or injury. The method does not provide a formal safety guarantee and should be combined with standard safeguards such as joint, velocity, workspace, collision, and emergency-stop constraints as well as human supervision during hardware testing.

AI Use Statement

Generative AI tools were used in two ways. First, for language editing: grammar, spelling, wording, and formatting. Second, as a coding assistant during implementation, experiment scripting, and debugging; all code was read and tested by the authors, and every number reported in this paper comes from runs the authors launched and inspected. The research questions, method design, experimental design, analysis, and all scientific claims are the authors' own. The authors reviewed all AI-assisted text and code and take full responsibility for the content of this work.

References

Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Physics vs distributions: Pareto optimal flow matching with physics constraints. In *International Conference on Learning Representations*, volume 2026, pp. 101215–101243, 2026.

Lars Berscheid and Torsten Kröger. Jerk-limited real-time trajectory generation with arbitrary target states. In *Robotics: Science and Systems XVII*, 2021. doi: 10.15607/RSS.2021.XVII.015.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems XXI*, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, volume 38, 2025c.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023a.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems XIX*, 2023b. doi: 10.15607/RSS.2023.XIX.025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark

with strong domain randomization for robust bimanual robotic manipulation. In *International Conference on Machine Learning*, 2026.

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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488, 2023.

Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience*, 5(7):1688–1703, 1985. doi: 10.1523/JNEUROSCI.05-07-01688.1985.

Chongkai Gao, Haozhuo Zhang, Zhixuan Xu, Zhehao Cai, and Lin Shao. FLIP: Flow-centric generative planning for general-purpose manipulation tasks. In *International Conference on Learning Representations*, 2025.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems XX*, 2024. doi: 10.15607/RSS.2024.XX.090.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318, 2023.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL https://doi.org/10.1115/1.3662552.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

Jian-wei Ma, Song Gao, Hui-teng Yan, Qi Lv, and Guo-qing Hu. A new approach to time-optimal trajectory planning with torque and jerk limits for robot. *Robotics and Autonomous Systems*, 140:103744, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Minho Park, Kinam Kim, Junha Hyung, Hyojin Jang, Hoiyeong Jin, Jooyeol Yun, Hojoon Lee, and Jaegul Choo. Acg: Action coherence guidance for flow-based vla models. *arXiv preprint arXiv:2510.22201*, 2025.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems XXI*, 2025. doi: 10.15607/RSS.2025.XXI.012.

Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Herbert E Rauch, Fong Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

S. W. Roberts. Control chart tests based on geometric moving averages. *Technometrics*, 1(3):239–250, 1959. doi: 10.1080/00401706.1959.10489860.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Dongwoo Son and Suhan Park. LiPo: A lightweight post-optimization framework for smoothing action chunks generated by learned policies. *International Journal of Control, Automation and Systems*, 23(10):3284–3292, 2025. doi: 10.1007/s12555-025-0537-0.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

Yuchen Wang, Hongjue Zhao, Haohong Lin, Enze Xu, Lifang He, and Huajie Shao. A generalizable physics-enhanced state space model for long-term dynamics forecasting in complex environments. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 65708–65726, 2025.

Yuan Yin, Vincent Le Guen, Jerémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021. doi: 10.1088/1742-5468/ac3ae5.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

APPENDIX CONTENTS

A Limitations and Broader Impact **14**

B Interpretation of the Kinematics-Informed Corrector **14**

B.1 Kalman Measurement Update as Regularized Least Squares 14

B.2 Uncertainty Proxies and Measurement Trust 15

B.3 Trust-Region Clip as Euclidean Projection 16

B.4 Why Repeated Decoding-Time Correction Can Help: An Operator-Splitting View 17

C Theoretical Justification **18**

C.1 Justification of the Learned Jacobian Objective 18

C.2 RTS Smoothing as a MAP Estimator 20

C.3 Finite-Horizon Stability of the Smoother 20

D Implementation Details and Reproducibility **21**

D.1 Flow-Matching Decoder Instantiation 21

D.2 Algorithm and Complexity 22

D.3 Training Details 23

D.4 Corrector Hyperparameters 23

D.5 Extensions Beyond Flow Matching 24

E Additional Experimental Details **24**

E.1 Simulation Benchmark Details 24

E.2 Baseline Implementations 25

E.3 Correction Schedule Details 26

E.4 Corrector Ablation Details 26

E.5 Real-Robot Protocol Details 27

E.6 Command-Level Metrics 28

E.7 Compute Resources 29

E.8 Runtime Measurement Details 29

F Additional Robustness and Diagnostic Analyses **29**

F.1 Per-Task Confidence Intervals and Significance Tests 29

F.2 Uncertainty and Hyperparameter Sensitivity 30

F.3 NFE Generalization 30

F.4 Mechanistic Analysis and Interpretation 31

F.5 Observed versus Unobserved Disturbance Channel 31

F.6 Qualitative Rollouts Under Injected Disturbance 31

G Real-Robot Task Definitions and Stress Tests **31**

G.1 Task Design and Success Criteria 31

G.2 Stress-Test Probes: Action Disturbance 33

A Limitations and Broader Impact

Limitations. This work has several limitations. First, the structure enforced by KI-VLA is deliberately limited to differential kinematics and a temporal smoothness prior; it does not model or enforce full rigid-body dynamics, contact, friction, collision avoidance, force/torque limits, or actuator constraints. Second, our empirical validation focuses on flow-matching robot policies. Although the proposed correction principle could be extended to other iterative action decoders, such as diffusion-style denoising policies (sketched in App. D.5), we do not claim broad cross-decoder generality without additional experiments. Moreover, the observed benefit is concentrated in the few-step flow-decoding regime: on Turn Switch, the advantage over the base policy disappears at $N = 50$, suggesting that KI-VLA is most useful when computationally efficient low-NFE decoding leaves greater room for intermediate-sample correction. Third, KI-VLA requires access to joint configurations and pose-derived task-space deltas to train the learned kinematic observation model J_θ. This requirement is satisfied in RoboTwin and in our real-robot logs, but applying the method to datasets that provide only images and end-effector actions may require additional state logging or simulator/model access. The learned Jacobian is also a local surrogate and can become inaccurate under severe calibration error, unmodeled compliance, actuator backlash, or robot configurations that are poorly covered by demonstrations.

Broader Impact. Because KI-VLA is intended for robot control, unsafe deployment could cause physical damage or injury if used without appropriate safeguards. Our experiments are conducted on bounded manipulation tasks with controlled disturbances. In practical deployment, the proposed corrector should be used together with standard robot safety mechanisms, including joint-limit checks, velocity and acceleration limits, workspace constraints, collision monitoring, emergency stops, and human supervision during hardware testing. The method is not designed for safety-critical autonomous operation in unstructured human environments. Rather, it should be viewed as an action-decoding refinement module for improving execution stability of already trained robot policies in controlled manipulation settings.

More broadly, improving the robustness of robot action decoding can have positive impact by reducing brittle execution failures, lowering the need for repeated trials, and making learned robot policies more reliable in contact-sensitive manipulation. At the same time, stronger robot autonomy may also amplify risks if deployed in inappropriate settings or combined with unreliable perception, unsafe task specifications, or insufficient physical safeguards. We therefore emphasize that KI-VLA improves local action consistency and disturbance robustness, but it does not solve high-level task safety, semantic grounding errors, or human-robot interaction safety.

B Interpretation of the Kinematics-Informed Corrector

This section clarifies how the main components of the corrector can be interpreted. In particular, we show that the Kalman update corresponds to a regularized implicit inverse-kinematics step, that the uncertainty proxies have standard statistical interpretations, that trust-region clipping is an exact Euclidean projection, and that interleaving can be viewed through a proximal-splitting lens.

B.1 Kalman Measurement Update as Regularized Least Squares

Proposition 1 (Single-step Kalman update equals regularized least squares)**.** *Consider a Gaussian prior $\Delta q \sim \mathcal{N}(\mu^-, P^-)$ and a linear measurement model*

$$z = J\,\Delta q + \eta, \qquad \eta \sim \mathcal{N}(0, \Sigma),$$

where $P^- \succ 0$ and $\Sigma \succ 0$. Then the posterior mean μ^+ produced by the Kalman measurement update

$$K = P^- J^\top (J P^- J^\top + \Sigma)^{-1}, \qquad \mu^+ = \mu^- + K(z - J\mu^-)$$

is the unique minimizer of the strictly convex quadratic

$$\Delta q^+ = \arg\min_{\Delta q} \; \|z - J\Delta q\|^2_{\Sigma^{-1}} + \|\Delta q - \mu^-\|^2_{(P^-)^{-1}}. \tag{19}$$

Proof. Consider the objective

$$f(\Delta q) = (z - J\Delta q)^\top \Sigma^{-1}(z - J\Delta q) + (\Delta q - \mu^-)^\top (P^-)^{-1}(\Delta q - \mu^-).$$

Because $P^- \succ 0$ and $\Sigma \succ 0$, both quadratic forms are strictly convex, so f is strictly convex and has a unique minimizer.

Taking the gradient with respect to Δq yields

$$\nabla_{\Delta q} f = -2J^\top \Sigma^{-1}(z - J\Delta q) + 2(P^-)^{-1}(\Delta q - \mu^-).$$

Setting the gradient to zero gives

$$J^\top \Sigma^{-1} J\, \Delta q + (P^-)^{-1}\Delta q = J^\top \Sigma^{-1} z + (P^-)^{-1}\mu^-,$$

or equivalently

$$\big(J^\top \Sigma^{-1} J + (P^-)^{-1}\big)\Delta q = J^\top \Sigma^{-1} z + (P^-)^{-1}\mu^-. \tag{20}$$

Since the matrix on the left is positive definite, the solution is unique.

Now use the Woodbury identity:

$$\big((P^-)^{-1} + J^\top \Sigma^{-1} J\big)^{-1} J^\top \Sigma^{-1} = P^- J^\top (JP^- J^\top + \Sigma)^{-1}.$$

Applying this identity to Eq. (20) yields

$$\Delta q^+ = \mu^- + P^- J^\top (JP^- J^\top + \Sigma)^{-1}(z - J\mu^-) = \mu^- + K(z - J\mu^-),$$

which is exactly the Kalman posterior mean. □

B.2 Uncertainty Proxies and Measurement Trust

The corrector uses two low-cost quantities to modulate measurement trust: a Jacobian log-volume proxy and an innovation magnitude. The following propositions formalize why these are reasonable reliability indicators. We also recall that larger measurement covariance makes the filter trust the measurement less.

Proposition 2 (Measurement noise controls trust in the measurement)**.** *For the Kalman update with innovation covariance*

$$S(\Sigma^u) = HPH^\top + \Sigma^u$$

and posterior covariance

$$P^+(\Sigma^u) = P - PH^\top S(\Sigma^u)^{-1} HP,$$

if $\Sigma_1^u \succeq \Sigma_2^u$ in PSD order, then

$$P^+(\Sigma_1^u) \succeq P^+(\Sigma_2^u).$$

Equivalently, increasing the measurement covariance decreases the covariance reduction contributed by the measurement update.

Proof. Let

$$S_i = HPH^\top + \Sigma_i^u, \qquad i \in \{1, 2\}.$$

If $\Sigma_1^u \succeq \Sigma_2^u$, then $S_1 \succeq S_2 \succ 0$. For positive definite matrices, inversion reverses Loewner order, so

$$S_1^{-1} \preceq S_2^{-1}.$$

Therefore,

$$P^+(\Sigma_1^u) - P^+(\Sigma_2^u) = PH^\top \big(S_2^{-1} - S_1^{-1}\big) HP \succeq 0.$$

Thus the posterior covariance increases with measurement covariance. Intuitively, larger measurement covariance means that the filter treats the observation as less reliable and therefore trusts it less. □

Proposition 3 (Log-volume proxy reflects local kinematic degeneracy)**.** *Let $J \in \mathbb{R}^{\alpha \times n}$ have singular values $\{\sigma_i\}_{i=1}^{\alpha}$. Then*

$$\kappa(J) \triangleq -\log\det(JJ^\top + \epsilon I) = -\sum_{i=1}^{\alpha} \log(\sigma_i^2 + \epsilon).$$

In particular, if any $\sigma_i \downarrow 0$ then $\kappa(J)$ increases, so a vanishing singular value drives $\kappa(J)$ up.

Proof. Let $J = U\Sigma V^\top$ be a singular value decomposition, so that

$$JJ^\top = U(\Sigma\Sigma^\top)U^\top, \qquad \Sigma\Sigma^\top = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_\alpha^2).$$

Then

$$JJ^\top + \epsilon I = U(\Sigma\Sigma^\top + \epsilon I)U^\top.$$

Because determinants are invariant under orthogonal conjugation,

$$\det(JJ^\top + \epsilon I) = \det(\Sigma\Sigma^\top + \epsilon I) = \prod_{i=1}^{\alpha}(\sigma_i^2 + \epsilon).$$

Taking $-\log$ of both sides yields

$$-\log\det(JJ^\top + \epsilon I) = -\sum_{i=1}^{\alpha}\log(\sigma_i^2 + \epsilon),$$

which increases when any singular value becomes small. □

Proposition 4 (Innovation magnitude is a mismatch indicator)**.** *Consider the predicted step*

$$z = Hx + \eta, \qquad \eta \sim \mathcal{N}(0, \Sigma),$$

and suppose the prediction error satisfies

$$x - \mu^- \sim \mathcal{N}(0, P^-).$$

Then the innovation

$$\nu \triangleq z - H\mu^-$$

satisfies

$$\nu \sim \mathcal{N}(0, S), \qquad S = HP^-H^\top + \Sigma.$$

Hence

$$\mathbb{E}\|\nu\|_2^2 = \mathrm{tr}(S),$$

and unusually large innovation norm indicates mismatch between the predicted latent state and the observed action.

Proof. Write the innovation as

$$\nu = z - H\mu^- = H(x - \mu^-) + \eta.$$

Since $x - \mu^-$ and η are independent Gaussians, ν is Gaussian with mean zero and covariance

$$\mathrm{Cov}(\nu) = H\,\mathrm{Cov}(x - \mu^-)\,H^\top + \mathrm{Cov}(\eta) = HP^-H^\top + \Sigma.$$

Thus $\nu \sim \mathcal{N}(0, S)$. For any zero-mean random vector with covariance S,

$$\mathbb{E}\|\nu\|_2^2 = \mathbb{E}[\nu^\top\nu] = \mathrm{tr}(\mathbb{E}[\nu\nu^\top]) = \mathrm{tr}(S).$$

Therefore, if $\|\nu\|_2$ is unusually large relative to its typical scale, it indicates that the predicted state and measurement are poorly aligned. □

B.3 Trust-Region Clip as Euclidean Projection

Proposition 5 (Trust-region clip equals Euclidean projection)**.** *Let $\hat{u}, \bar{u} \in \mathbb{R}^m$ and $\rho^{\mathit{eff}} \geq 0$. Define the closed ℓ_2 ball*

$$\mathcal{B} = \{u : \|u - \hat{u}\|_2 \leq \rho^{\mathit{eff}}\}.$$

Then the trust-region update in the main text is the Euclidean projection

$$u^\star = \arg\min_{u \in \mathcal{B}} \|u - \bar{u}\|_2^2, \tag{21}$$

and therefore satisfies $\|u^\star - \hat{u}\|_2 \leq \rho^{\mathit{eff}}$.

Proof. Let $\delta = \bar{u} - \hat{u}$ and define $y = u - \hat{u}$. Then minimizing over $u \in \mathcal{B}$ is equivalent to solving

$$\min_{\|y\|_2 \le \rho^{\text{eff}}} \|y - \delta\|_2^2.$$

If $\|\delta\|_2 \le \rho^{\text{eff}}$, then the unconstrained minimizer $y^\star = \delta$ is already feasible, so $u^\star = \bar{u}$.

Now suppose $\|\delta\|_2 > \rho^{\text{eff}}$. Then the constraint is active, so we introduce a Lagrange multiplier $\lambda \ge 0$ and consider

$$\mathcal{L}(y, \lambda) = \frac{1}{2}\|y - \delta\|_2^2 + \lambda(\|y\|_2^2 - (\rho^{\text{eff}})^2).$$

Stationarity gives

$$\nabla_y \mathcal{L} = y - \delta + 2\lambda y = 0,$$

hence

$$(1 + 2\lambda)y = \delta.$$

Therefore the optimizer is collinear with δ. Because the constraint is active, $\|y\|_2 = \rho^{\text{eff}}$, so

$$y^\star = \rho^{\text{eff}} \frac{\delta}{\|\delta\|_2}.$$

Substituting back,

$$u^\star = \hat{u} + \rho^{\text{eff}} \frac{\bar{u} - \hat{u}}{\|\bar{u} - \hat{u}\|_2}.$$

Combining the feasible and active-constraint cases yields

$$u^\star = \hat{u} + (\bar{u} - \hat{u}) \cdot \min\left\{1, \frac{\rho^{\text{eff}}}{\|\bar{u} - \hat{u}\|_2}\right\},$$

which is exactly the clipping form used in the main text, up to the small numerical ϵ added in implementation. □

Continuous innovation modulation and radius. In our implementation, the effective radius is

$$\rho^{\text{eff}}_{t+k} = \lambda_{t+k}\rho_{t+k},$$

with

$$\lambda_{t+k} = \sigma\left(w_\lambda^\top \left[e^{\text{innov}}_{t+k}, \|\hat{u}_{t+k} - \hat{u}_{t+k-1}\|_2, 1\right]\right), \tag{22}$$

and

$$\rho_{t+k} = \rho_{\max} \cdot \sigma\left(w_\rho^\top \left[-\kappa_{t+k}, 1\right]\right). \tag{23}$$

The sigmoid factors continuously modulate the allowable correction. We do not treat them as a thresholded detector: the episode-level analysis in Appendix F does not support a binary "selective trigger" interpretation, so the appropriate interpretation is continuous uncertainty-weighted regularization.

B.4 Why Repeated Decoding-Time Correction Can Help: An Operator-Splitting View

This interpretation motivates repeated correction between predictor updates, but the original schedule comparison changes both placement and the number of corrector calls. We therefore use it as an interpretation, not as proof that placement alone causes the gain.

Proposition 6 (Predictor-corrector decoding as proximal splitting)**.** *Consider an iterative predictor update*

$$\widehat{\mathbf{A}}^{(s)} = \mathcal{P}(\mathbf{A}^{(s-1)})$$

and a kinematic-consistency energy $\mathcal{E}_{\text{phys}}(\mathbf{A}; q_t)$. Suppose the corrector approximately implements the proximal map

$$\mathcal{C}(\widehat{\mathbf{A}}) \approx \arg\min_{\mathbf{A}} \frac{1}{2\eta}\|\mathbf{A} - \widehat{\mathbf{A}}\|_2^2 + \mathcal{E}_{\text{phys}}(\mathbf{A}; q_t).$$

Then the interleaved update

$$\mathbf{A}^{(s)} \leftarrow \mathcal{C}\Big(\mathcal{P}(\mathbf{A}^{(s-1)})\Big)$$

can be interpreted as a proximal-splitting step that combines the decoder dynamics with a kinematic prior at every iteration.

Proof sketch. Our corrector consists of two parts. First, it computes a latent MAP estimate using Kalman filtering and RTS smoothing, which solves a structured quadratic inference problem over the latent trajectory. Second, it applies a trust-region projection in action space, which by Proposition 5 is itself the minimizer of a Euclidean projection problem.

Together, these two steps can be viewed as approximately minimizing a kinematic-consistency objective regularized by distance to the decoder proposal. Therefore, applying the corrector after every predictor step corresponds to repeatedly pulling the intermediate iterate back toward a kinematically consistent set. This is the same high-level principle underlying proximal splitting for composite objectives: one step moves according to the base dynamics, and the next step regularizes with respect to an auxiliary structure. By contrast, post-hoc correction applies this regularization only once at the very end. □

C Theoretical Justification

This section provides formal support for the learned Jacobian objective and the RTS-based smoothing formulation used in the main method. The goal is not to claim an independent theorem-driven contribution, but to justify why the chosen objectives and latent smoother have the intended properties.

C.1 Justification of the Learned Jacobian Objective

Proposition 7 (Why $\mathcal{L}_{\text{kin}}$ works)**.** *Assume a local discrete-time kinematic model*

$$u = J_\star(q)\Delta q + \varepsilon, \qquad \mathbb{E}[\varepsilon\, \Delta q^\top \mid q] = 0,$$

and define

$$\Sigma_{\Delta q}(q) \triangleq \mathbb{E}[\Delta q \Delta q^\top \mid q].$$

If $\Sigma_{\Delta q}(q) \succ 0$, then the minimizer of the conditional risk

$$\mathbb{E}\big[\|u - J(q)\Delta q\|_2^2 \mid q\big]$$

is unique and given by

$$J^\star(q) = \mathbb{E}[u\, \Delta q^\top \mid q]\, \Sigma_{\Delta q}(q)^{-1}.$$

Moreover, under the assumed model, $J^\star(q) = J_\star(q)$.

Proof. Fix q and define

$$F(J) = \mathbb{E}\big[\|u - J\Delta q\|_2^2 \mid q\big].$$

Expanding the square gives

$$F(J) = \mathbb{E}[u^\top u \mid q] - 2\,\mathrm{tr}\big(J\,\mathbb{E}[\Delta q\, u^\top \mid q]\big) + \mathrm{tr}\big(J\,\mathbb{E}[\Delta q \Delta q^\top \mid q]\, J^\top\big).$$

Differentiating with respect to J and setting the derivative to zero yields the normal equation

$$J\,\mathbb{E}[\Delta q \Delta q^\top \mid q] = \mathbb{E}[u \Delta q^\top \mid q].$$

Since $\Sigma_{\Delta q}(q) \succ 0$, the solution is unique and equals

$$J^\star(q) = \mathbb{E}[u \Delta q^\top \mid q]\, \Sigma_{\Delta q}(q)^{-1}.$$

Now substitute the model

$$u = J_\star(q)\Delta q + \varepsilon.$$

Then

$$\mathbb{E}[u \Delta q^\top \mid q] = J_\star(q)\,\mathbb{E}[\Delta q \Delta q^\top \mid q] + \mathbb{E}[\varepsilon \Delta q^\top \mid q].$$

By assumption, the second term is zero, so

$$\mathbb{E}[u \Delta q^\top \mid q] = J_\star(q) \Sigma_{\Delta q}(q).$$

Multiplying on the right by $\Sigma_{\Delta q}(q)^{-1}$ yields $J^\star(q) = J_\star(q)$. □

Proposition 8 (Why $\mathcal{L}_{\text{smooth}}$ helps)**.** *Let J_θ be differentiable. For any q, q',*

$$\|J_\theta(q') - J_\theta(q)\|_F \le \|q' - q\|_2 \left(\int_0^1 \|\nabla_q J_\theta(q + \tau(q' - q))\|_F^2 \, d\tau \right)^{1/2}.$$

Consequently, penalizing $\|\nabla_q J_\theta\|_F^2$ reduces variation of the learned Jacobian along nearby states and stabilizes local linearization.

Proof. Let $\gamma(\tau) = q + \tau(q' - q)$ for $\tau \in [0, 1]$. By the fundamental theorem of calculus,

$$J_\theta(q') - J_\theta(q) = \int_0^1 \frac{d}{d\tau} J_\theta(\gamma(\tau)) \, d\tau = \int_0^1 \nabla_q J_\theta(\gamma(\tau))(q' - q) \, d\tau.$$

Taking Frobenius norms gives

$$\|J_\theta(q') - J_\theta(q)\|_F \le \int_0^1 \|\nabla_q J_\theta(\gamma(\tau))(q' - q)\|_F \, d\tau \le \|q' - q\|_2 \int_0^1 \|\nabla_q J_\theta(\gamma(\tau))\|_F \, d\tau.$$

Applying Cauchy-Schwarz to the integral,

$$\int_0^1 \|\nabla_q J_\theta(\gamma(\tau))\|_F \, d\tau \le \left(\int_0^1 \|\nabla_q J_\theta(\gamma(\tau))\|_F^2 \, d\tau \right)^{1/2},$$

which proves the claim. □

Proposition 9 (Why $\mathcal{L}_{\text{cond}}$ helps)**.** *Let $J \in \mathbb{R}^{\alpha \times n}$ have singular values $\{\sigma_i\}_{i=1}^{\alpha}$. Then*

$$-\log\det(JJ^\top + \epsilon I) = -\sum_{i=1}^{\alpha} \log(\sigma_i^2 + \epsilon).$$

Moreover, suppose the spectrum is bounded above, $\sigma_i \le S$ for all i, and

$$-\log\det(JJ^\top + \epsilon I) \le c.$$

Then

$$\sigma_{\min}(J)^2 + \epsilon \;\ge\; e^{-c} \, (S^2 + \epsilon)^{-(\alpha - 1)}.$$

Thus the conditioning penalty discourages near-singular Jacobians and lower-bounds the smallest singular value up to the ϵ floor, provided the largest singular values are controlled.

Proof. Using the singular value decomposition $J = U\Sigma V^\top$,

$$JJ^\top + \epsilon I = U(\Sigma\Sigma^\top + \epsilon I)U^\top.$$

Hence

$$\det(JJ^\top + \epsilon I) = \prod_{i=1}^{\alpha} (\sigma_i^2 + \epsilon),$$

and taking $-\log$ gives

$$-\log\det(JJ^\top + \epsilon I) = -\sum_{i=1}^{\alpha} \log(\sigma_i^2 + \epsilon).$$

For the lower bound, isolate the smallest factor and bound the remaining $\alpha - 1$ factors *above* using $\sigma_i \le S$:

$$e^{-c} \;\le\; \det(JJ^\top + \epsilon I) \;=\; \prod_{i=1}^{\alpha} (\sigma_i^2 + \epsilon) \;\le\; (\sigma_{\min}(J)^2 + \epsilon) \, (S^2 + \epsilon)^{\alpha - 1},$$

where the first inequality restates $-\log\det(JJ^\top + \epsilon I) \le c$. Rearranging gives

$$\sigma_{\min}(J)^2 + \epsilon \;\ge\; e^{-c} \, (S^2 + \epsilon)^{-(\alpha - 1)}.$$

□

C.2 RTS SMOOTHING AS A MAP ESTIMATOR

Proposition 10 (RTS smoothing equals the unique MAP minimizer)**.** *Assume a finite-horizon linear-Gaussian state-space model with $Q \succ 0$, $\Sigma^u_{t+k} \succ 0$, and initial prior covariance $P_0 \succ 0$, with $\{H_k\}$ and $\{\Sigma_k\}$ fixed, as holds within one relinearization iteration of* KI-VLA*. Then RTS smoothing computes the unique MAP estimate, equivalently the unique minimizer of*

$$\min_{x_{t:t+H-1}} \sum_{k=0}^{H-1} \|z_{t+k} - H_{t+k}x_{t+k}\|^2_{(\Sigma^u_{t+k})^{-1}} + \sum_{k=0}^{H-2} \|x_{t+k+1} - Ax_{t+k}\|^2_{Q^{-1}} + \|x_t - \mu_t\|^2_{P_0^{-1}}. \quad (24)$$

Proof. Consider the linear-Gaussian model

$$x_{k+1} = Ax_k + w_k, \qquad w_k \sim \mathcal{N}(0, Q),$$
$$z_k = H_k x_k + v_k, \qquad v_k \sim \mathcal{N}(0, \Sigma_k),$$

with prior

$$x_0 \sim \mathcal{N}(\mu_0, P_0).$$

By conditional independence,

$$p(x_{0:T} \mid z_{0:T}) \propto p(x_0) \prod_{k=0}^{T-1} p(x_{k+1} \mid x_k) \prod_{k=0}^{T} p(z_k \mid x_k).$$

Taking negative logarithms and dropping constants yields

$$\|x_0 - \mu_0\|^2_{P_0^{-1}} + \sum_{k=0}^{T-1} \|x_{k+1} - Ax_k\|^2_{Q^{-1}} + \sum_{k=0}^{T} \|z_k - H_k x_k\|^2_{\Sigma_k^{-1}}.$$

Since $P_0^{-1} \succ 0$, $Q^{-1} \succ 0$, and $\Sigma_k^{-1} \succ 0$, this is a strictly convex quadratic in the concatenated trajectory $(x_0, \ldots, x_T)$. Therefore it has a unique global minimizer.

The optimality conditions form a block-tridiagonal linear system because each state x_k appears only in the local measurement term and the two adjacent dynamics terms. Kalman filtering performs forward elimination on this structured system, while the RTS backward pass performs back-substitution to recover the full posterior mean trajectory. Since the posterior is Gaussian, the posterior mean and MAP estimate coincide. Hence RTS smoothing computes the unique MAP minimizer of the stated objective. □

C.3 FINITE-HORIZON STABILITY OF THE SMOOTHER

The previous proposition establishes that RTS smoothing solves a well-posed finite-horizon inference problem. We next record two simple consequences: uniqueness of the finite-horizon solution and Lipschitz dependence on the measurements. These results are stated for a simplified linear time-invariant model to make the argument transparent; the main purpose is to support the robustness intuition behind the smoother.

1D model. Let $x_t = [u_t, \dot{u}_t]^\top \in \mathbb{R}^2$, with dynamics

$$x_{t+1} = Ax_t + w_t, \qquad A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \qquad w_t \sim \mathcal{N}(0, Q),$$

and measurements

$$\hat{u}_t = Cx_t + v_t, \qquad C = [1 \quad 0], \qquad v_t \sim \mathcal{N}(0, R_t).$$

Lifted model. For $u_t \in \mathbb{R}^{d_u}$, define

$$x_t = \begin{bmatrix} u_t \\ \dot{u}_t \end{bmatrix} \in \mathbb{R}^{2d_u}, \qquad A = \begin{bmatrix} I & \Delta t\, I \\ 0 & I \end{bmatrix}, \qquad C = [I \quad 0].$$

All arguments below extend by replacing scalar blocks with the corresponding vector blocks.

Proposition 11 (Unique MAP solution / well-posed smoothing)**.** *Fix horizon H. Assume $Q \succ 0$, $R_t \succ 0$ for all t, and initial prior covariance $P_1 \succ 0$. Then the posterior over $x_{1:H}$ is Gaussian with a unique MAP estimate; equivalently, the finite-horizon smoothing problem has a unique minimizer.*

Proof. Up to constants, the negative log posterior is

$$\min_{x_{1:H}} \sum_{t=1}^{H-1} \|x_{t+1} - Ax_t\|^2_{Q^{-1}} + \sum_{t=1}^{H} \|\hat{u}_t - Cx_t\|^2_{R_t^{-1}} + \|x_1 - \mu_1\|^2_{P_1^{-1}}. \tag{25}$$

Each term is convex, and the initial prior plus positive definite process and measurement weights make the total Hessian positive definite. Hence the objective is strictly convex and admits a unique minimizer. Because the posterior is Gaussian, this minimizer is also the unique MAP estimate. □

Proposition 12 (Finite-horizon Lipschitz stability)**.** *Under the assumptions of Proposition 11, the smoothed trajectory is a globally Lipschitz function of the measurement sequence. In particular, there exists $L > 0$ such that for any two measurement sequences $\hat{u}_{1:H}$ and $\hat{u}'_{1:H}$,*

$$\|\tilde{u}_{1:H} - \tilde{u}'_{1:H}\|_2 \le L \, \|\hat{u}_{1:H} - \hat{u}'_{1:H}\|_2.$$

Proof. Let $x_{1:H}$ denote the concatenated state vector. The objective in Eq. (25) is a strictly convex quadratic of the form

$$\frac{1}{2} x^\top H_{\mathrm{MAP}} x - b(\hat{u})^\top x + c(\hat{u}),$$

where $H_{\mathrm{MAP}} \succ 0$ is the Hessian and $b(\hat{u})$ depends linearly on the measurements. The unique minimizer is therefore

$$x^\star = H_{\mathrm{MAP}}^{-1} b(\hat{u}),$$

which is an affine function of $\hat{u}$. Hence, for two measurement sequences,

$$x^\star(\hat{u}) - x^\star(\hat{u}') = H_{\mathrm{MAP}}^{-1}\big(b(\hat{u}) - b(\hat{u}')\big).$$

Taking norms gives

$$\|x^\star(\hat{u}) - x^\star(\hat{u}')\|_2 \le \|H_{\mathrm{MAP}}^{-1}\|_2 \, \|b(\hat{u}) - b(\hat{u}')\|_2.$$

Since $b(\hat{u})$ is linear in $\hat{u}$, there exists a constant c_b such that

$$\|b(\hat{u}) - b(\hat{u}')\|_2 \le c_b \|\hat{u} - \hat{u}'\|_2.$$

Therefore

$$\|x^\star(\hat{u}) - x^\star(\hat{u}')\|_2 \le \|H_{\mathrm{MAP}}^{-1}\|_2 \, c_b \, \|\hat{u} - \hat{u}'\|_2.$$

Finally, the output trajectory $\tilde{u}_{1:H}$ is obtained by projecting the state trajectory through the fixed linear observation map C, so Lipschitz continuity is preserved. □

D Implementation Details and Reproducibility

This section provides the concrete flow-matching instantiation, decoding algorithm, complexity, and training details needed to implement and reproduce the proposed method. Our main paper focuses on the flow-matching setting, so we present that instantiation first and reserve broader decoder extensions for Sec. D.5. We also submit anonymized supplementary code, including the KI-VLA corrector implementation, learned Jacobian training code, correction schedules, smoothing and post-processing baselines, ablation variants, and scripts for computing the command-level diagnostics reported in the paper.

D.1 Flow-Matching Decoder Instantiation

Our main paper instantiates the proposed predictor-corrector decoder for flow-matching robot policies. Let $\mathbf{A}_t$ denote the target action chunk and let $\tau \in [0, 1]$ denote the flow time. We use the π_0-style convention where $\tau = 1$ corresponds to noise and $\tau = 0$ corresponds to data. Given $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, we define the linear interpolation path

$$\mathbf{A}_t^\tau = (1 - \tau)\mathbf{A}_t + \tau\epsilon.$$

Thus, $\mathbf{A}_t^1 = \epsilon$ and $\mathbf{A}_t^0 = \mathbf{A}_t$. The flow-matching policy learns a vector field $\mathbf{v}_\psi(\mathbf{A}_t^\tau, o_t, \tau)$ that transports samples from noise toward the data action chunk. We train it with

$$\mathcal{L}_{\mathrm{FM}}(\psi) = \mathbb{E}_{\mathbf{A}_t,\epsilon,\tau}\left[\left\|\mathbf{v}_\psi(\mathbf{A}_t^\tau, o_t, \tau) - (\mathbf{A}_t - \epsilon)\right\|_2^2\right], \tag{26}$$

where the target direction $\mathbf{A}_t - \epsilon$ points from the noisy endpoint to the data endpoint.

At inference time, we start from $\mathbf{A}_t^{(0)} \sim \mathcal{N}(0, \mathbf{I})$ and integrate from noise to data using N Euler steps. Let $\tau_s = 1 - s/N$ and let $\xi_s = 1 - \tau_s$ denote the normalized decoding progress, so $\xi_s = 0$ is early/noisy and $\xi_s = 1$ is late/action-like. The flow predictor is

$$\mathcal{P}_\psi^{\mathrm{FM}}\Big(\mathbf{A}_t^{(s-1)}, o_t, \tau_s\Big) = \mathbf{A}_t^{(s-1)} + \delta\, \mathbf{v}_\psi(\mathbf{A}_t^{(s-1)}, o_t, \tau_s), \qquad \delta = \frac{1}{N}. \tag{27}$$

The proposed method augments this predictor with the kinematics-informed corrector:

$$\widehat{\mathbf{A}}_t^{(s)} \leftarrow \mathcal{P}_\psi^{\mathrm{FM}}\Big(\mathbf{A}_t^{(s-1)}, o_t, \tau_s\Big), \qquad \mathbf{A}_t^{(s)} \leftarrow \mathcal{C}_{\theta,\phi}\Big(q_t, \widehat{\mathbf{A}}_t^{(s)}, \xi_s\Big). \tag{28}$$

Thus, the base flow predictor proposes an updated action chunk according to the flow time τ_s, while the corrector uses the progress variable ξ_s to modulate correction strength as the sample becomes more action-like.

D.2 Algorithm and Complexity

Algorithm 1 summarizes the flow-matching predictor-corrector decoder used in the main paper.

Algorithm 1: Kinematics-Informed Predictor-Corrector Decoding for Flow Matching

Input: Observation o_t, joint state q_t, number of flow steps N, schedule (M, ξ_0), flow predictor $\mathcal{P}_\psi^{\mathrm{FM}}$, corrector $\mathcal{C}_{\theta,\phi}$.
Output: Corrected action chunk $\mathbf{A}_t = [u_{t:t+H-1} \;\; g_{t:t+H-1}]$.

```
Initialize A_t^(0) ~ N(0, I);
for s = 1, 2, ..., N do
    τ_s ← 1 − s/N;                          // flow time: noise → data
    ξ_s ← 1 − τ_s;                          // decoding progress: early → late
    Â_t^(s) ← P_ψ^FM(A_t^(s−1), o_t, τ_s);
    if ApplyCorr(s, ξ_s; M, ξ_0) then
        A_t^(s) ← C_{θ,φ}(q_t, Â_t^(s), ξ_s);
    else
        A_t^(s) ← Â_t^(s);
return A_t^(N);
```

Sparse schedules. The function ApplyCorr(s) determines whether the corrector is invoked at decoding step s. In the full predictor-corrector setting, the corrector is applied at every flow step. In sparse settings, it is applied only to a subset of steps, typically later steps where the sampled action chunk is already semantically meaningful but still locally noisy. This provides a practical compute-quality trade-off.

Complexity. Let $N_{\mathrm{pred}} = N$ denote the number of flow predictor steps and $N_{\mathrm{corr}} \le N$ the number of corrector calls. The base decoder cost is unchanged from the underlying flow policy; KI-VLA adds the correction cost described below. Each corrector call runs Kalman filtering and RTS smoothing over horizon H, so the total correction cost scales as

$$\mathcal{O}(N_{\mathrm{corr}} \cdot H)$$

up to the cost of the small dense matrix solves inside each filtering/smoothing step. A more explicit bound is

$$\mathcal{O}\big(N_{\mathrm{corr}} \cdot H \cdot (\alpha^3 + (2n)^3)\big),$$

where α is the operational-space motion dimension and n is the number of arm joints. Since in our setting $\alpha = 6$ and n is small, the practical runtime is close to linear in $N_{\mathrm{corr}} \cdot H$. Empirical runtime overhead is reported in Sec. 4.4 and Appendix E.8.

D.3 Training Details

Our training pipeline separates the base flow policy from the kinematics-informed corrector.

Base flow policy. We first train the flow-matching VLA using the objective in Eq. (26). This stage is identical for the base policy and KI-VLA. The KI-VLA corrector is then trained and evaluated on top of the same frozen base policy, so improvements are attributable to the decoding-time correction rather than changes in the VLA backbone.

Learned Jacobian surrogate. We train J_θ from logged joint states and end-effector poses. The pose-derived operational-space target is

$$u_t^{\text{data}} = \begin{bmatrix} p_{t+1} - p_t \\ \text{Log}(R_t^\top R_{t+1}) \end{bmatrix},$$

and the joint increment is $\Delta q_t = q_{t+1} - q_t$. The learned Jacobian is optimized with

$$\mathcal{L}_{\text{kin}}(\theta) = \mathbb{E}_t \left[\left\| u_t^{\text{data}} - J_\theta(q_t)\Delta q_t \right\|_2^2 \right], \tag{29}$$

$$\mathcal{L}_{\text{smooth}}(\theta) = \mathbb{E}_t \left[\|\nabla_q J_\theta(q_t)\|_F^2 \right], \tag{30}$$

$$\mathcal{L}_{\text{cond}}(\theta) = -\mathbb{E}_t \left[\log\det(J_\theta(q_t)J_\theta(q_t)^\top + \epsilon I) \right], \tag{31}$$

$$\mathcal{L}_J(\theta) = \mathcal{L}_{\text{kin}}(\theta) + \lambda_{\text{reg}} \left(\mathcal{L}_{\text{smooth}}(\theta) + \beta \mathcal{L}_{\text{cond}}(\theta) \right). \tag{32}$$

The analytic Jacobian is not used by the online corrector. When calibrated online end-effector poses are unavailable in the real-robot setup, pose supervision is obtained offline from logged joint states using the available forward-kinematics model.

Uncertainty module. The uncertainty module h_ϕ predicts a diagonal measurement covariance for each chunk element. Its input is

$$\varphi_{t+k} = [\bar{q}_{t+k},\ \|\hat{u}_{t+k}\|_2,\ \|\hat{u}_{t+k} - \hat{u}_{t+k-1}\|_2,\ \kappa_{t+k},\ ],$$

where κ_{t+k} is the Jacobian log-volume proxy. The covariance is

$$\Sigma_{t+k}^u = s(\xi_s) \cdot \text{diag}\left(\text{softplus}(h_\phi(\varphi_{t+k})) + \epsilon\right).$$

We train h_ϕ by unrolling the corrector on demonstration chunks and minimizing behavior-cloning loss on the corrected actions while keeping the base flow policy fixed:

$$\mathcal{L}_{\text{BC}} = \frac{1}{H}\sum_{k=0}^{H-1} \|u_{t+k} - u_{t+k}^*\|_2^2 + \lambda_g \frac{1}{H}\sum_{k=0}^{H-1} \|\hat{g}_{t+k} - g_{t+k}^*\|_2^2. \tag{33}$$

In practice, we set $\lambda_g = 0$, so the corrector is trained and evaluated on the motion component u only and gripper commands are passed through unchanged.

D.4 Corrector Hyperparameters

Table 5 lists the main hyperparameters used by KI-VLA. Unless otherwise stated, we use the same hyperparameters across tasks. Schedule-specific values are reported in Table 4. The values were selected once during early development on Turn Switch and then frozen across all remaining tasks and splits. Selection used 10 validation episodes generated with seeds disjoint from those of the 100-rollout evaluation sets, so no tuning episode appears in any reported result.

For the single-arm real-robot experiments, we use the corresponding 6-DoF instantiation of the same modules and latent-state covariance structure.

Implementation note. The schedule ablation in the main paper varies only (M, ξ_0) while keeping the learned Jacobian, uncertainty module, process noise, trust-region rule, and base flow policy fixed. This isolates the schedule family from other corrector design choices, although correction timing and call count remain coupled across several rows.

Table 5: Main KI-VLA hyperparameters.

Hyperparameter	Meaning	Value
H	Action chunk horizon (per replan)	50
N	Number of flow-matching Euler steps	10
α	Operational-space motion dimension	6 per arm (12 total twist dims in RoboTwin)
n	Arm joint dimension	6 per arm (12 total joint dims in RoboTwin)
Δt	Latent smoother discretization step	1.0 (per action-index step in the chunk)
M	Correction interval	Schedule-dependent (default 1)
ξ_0	Minimum progress for correction	Schedule-dependent (default $\xi_0 = 0$)
λ_{reg}	Weight for Jacobian regularization	10^{-3}
β	Weight for conditioning term	1.0
ϵ	Stability constant in $\log\det(JJ^\top + \epsilon I)$	10^{-4}
Q	Process noise covariance in latent dynamics	$\text{diag}(10^{-3}I_{12},\ 10^{-2}I_{12})$
$\rho_{\max}$	Trust-region radius (max update norm)	0.03
N_{relin}	Relinearization iterations per corrector call	2
λ_g	Gripper loss/correction weight	0 (grippers are passed through unchanged)
$s(\xi)$	Progress-dependent measurement scaling	$\begin{cases} 1 & \text{constant} \\ 0.10 + 0.90(1-\xi) & \text{progress (default)} \end{cases}$
J_θ architecture	MLP for learned Jacobian	per-arm: $6 \to 128 \to 128 \to 36$ (ReLU); block-diagonal $\Rightarrow$ 12×12 for RoboTwin
h_ϕ architecture	MLP for uncertainty (diag variance)	$15 \to 128 \to 128 \to 12$ (ReLU) + Softplus for RoboTwin

D.5 Extensions Beyond Flow Matching

The main paper focuses on flow-matching robot policies, where predictor-corrector decoding is especially natural. For completeness, we briefly note that the same correction principle could also be instantiated with other decoders by replacing the predictor while keeping the latent smoothing-based corrector unchanged.

Diffusion-style iterative decoding. For an iterative denoising decoder, one may define

$$\mathcal{P}_\psi^{\text{Diff}}\left(\mathbf{A}_t^{(s-1)}, o_t, \xi_s\right) \triangleq \textsc{DenoiseStep}(\mathbf{A}_t^{(s-1)}, o_t, s), \tag{34}$$

and apply the same corrector after selected denoising steps:

$$\mathbf{A}_t^{(s)} \leftarrow \mathcal{C}_{\theta,\phi}(q_t, \widehat{\mathbf{A}}_t^{(s)}, \xi_s).$$

One-shot chunk regression. For a deterministic chunk regressor $f_\psi(o_t)$, one may form

$$\widehat{\mathbf{A}}_t = f_\psi(o_t)$$

and apply a single final correction

$$\mathbf{A}_t \leftarrow \mathcal{C}_{\theta,\phi}(q_t, \widehat{\mathbf{A}}_t, 1).$$

These extensions are conceptually straightforward because the corrector operates on candidate action chunks rather than on the internal architecture of the generator. However, our main empirical focus is the flow-matching setting, so we do not make strong cross-decoder claims in the main text.

E Additional Experimental Details

This section provides additional details for the experimental protocol, baseline implementations, correction schedules, corrector ablations, real-robot stress tests, and runtime measurements. The goal is to make the controlled comparisons in the main paper reproducible and to clarify which comparisons should be interpreted as head-to-head evaluations.

E.1 Simulation Benchmark Details

We evaluate on eight RoboTwin 2.0 manipulation tasks covering object relocation, contact-rich manipulation, switch or button interaction, container interaction, and placement behaviors. Each

task is evaluated under Easy and Hard settings. The Easy setting uses simpler initial configurations and lower task variation, while the Hard setting introduces larger variation in object pose, scene configuration, or contact geometry. We report binary task success rate as the primary metric.

Table 1 in the main paper reports both joint-action reference results and the controlled end-pose comparison. The joint-action methods provide benchmark context only; all efficacy claims use the controlled π_0^{EP} versus KI-VLA comparison.

A central experimental issue is the action representation. Some reference baselines use joint-angle actions, whereas our method operates on end-pose / operational-space delta actions. Therefore, joint-action baselines are included as reference points but are not used as strict head-to-head comparisons. The primary controlled baseline is the end-pose flow-matching policy, denoted as π_0^{EP}, which uses the same observation inputs, training data, action horizon, flow-matching policy backbone, and operational-space action interface as KI-VLA, but does not use the kinematics-informed corrector.

E.2 Baseline Implementations

All baselines are applied at inference time in the same end-pose action space as KI-VLA and do not retrain the base policy. Some baselines post-process the final decoded chunk (smoothing), while others modify the sampling procedure itself (guidance). Unless otherwise stated, we smooth only the arm pose components (translation and orientation) and keep gripper commands unchanged to avoid altering discrete contact events. For quaternion smoothing, we enforce sign consistency across timesteps (to avoid antipodal flips) and renormalize after smoothing.

Moving-average smoothing. We apply a causal sliding-window average over the decoded pose sequence within each chunk:

$$\tilde{a}_{t+k} = \frac{1}{|\mathcal{N}(k)|} \sum_{j \in \mathcal{N}(k)} \hat{a}_{t+j},$$

where $\mathcal{N}(k)$ is a fixed causal window. This is related to, but distinct from, the temporal ensembling of Zhao et al. (2023): that method averages overlapping predictions of the *same* timestep produced by different chunks, whereas this baseline averages *adjacent* actions within a single chunk.

Exponential smoothing (EMA). We apply a first-order exponential smoothing filter:

$$\tilde{a}_{t+k} = \alpha_{\mathrm{lp}} \hat{a}_{t+k} + (1 - \alpha_{\mathrm{lp}}) \tilde{a}_{t+k-1},$$

with $\alpha_{\mathrm{lp}} \in [0, 1]$. This provides a fixed-gain low-pass baseline (Roberts, 1959).

Second-difference Jerk-proxy post-smoothing. We smooth the decoded chunk by solving a penalized least-squares problem independently per dimension:

$$\tilde{\mathbf{a}} = \arg\min_{\mathbf{a}} \sum_{k=0}^{H-1} \|a_{t+k} - \hat{a}_{t+k}\|_2^2 + \lambda_{\mathrm{sd}} \sum_{k=1}^{H-2} \|a_{t+k+1} - 2a_{t+k} + a_{t+k-1}\|_2^2.$$

This imposes a generic within-chunk smoothness prior without kinematic latent inference or uncertainty-aware measurement weighting; the minimum-jerk principle it approximates originates with Flash & Hogan (1985). Because u_t is an operational-space *delta* (velocity-like) action, penalizing the second difference $\|u_{k+1} - 2u_k + u_{k-1}\|^2$ acts as a minimum-jerk-inspired regularizer on the *integrated* pose trajectory; we therefore refer to this baseline as a second-difference (jerk-proxy) post-smoother.

ACG-style coherence guidance. We implement an ACG-style test-time guidance baseline following the high-level idea of Action Coherence Guidance (Park et al., 2025): during flow-matching decoding, we bias intermediate samples toward temporally coherent action chunks. Concretely, at each flow step we augment the Euler update with an additional guidance term derived from a coherence objective $\mathcal{L}_{\mathrm{coh}}$ (we use the same second-difference coherence penalty as the jerk-proxy smoother),

$$\mathcal{L}_{\mathrm{coh}}(\mathbf{u}) = \sum_{k=1}^{H-2} \|u_{t+k+1} - 2u_{t+k} + u_{t+k-1}\|_2^2 .$$

Let $\widehat{\mathbf{A}}_t^{(s)}$ denote the intermediate chunk at decoding step s. We apply a guidance-modified predictor of the form

$$\widehat{\mathbf{A}}_t^{(s)} \leftarrow \widehat{\mathbf{A}}_t^{(s-1)} + \delta\, \mathbf{v}_\psi(\widehat{\mathbf{A}}_t^{(s-1)}, o_t, \xi_s) - \gamma\, \nabla_{\mathbf{u}} \mathcal{L}_{\mathrm{coh}}(\widehat{\mathbf{u}}_t^{(s-1)}),$$

where γ is a guidance weight tuned on a held-out set. This baseline improves coherence by shaping the sampling trajectory, but unlike KI-VLA it does not perform kinematics-aware latent inference through a Jacobian observation model.

Final-only corrector. This baseline uses the same learned Jacobian J_θ, uncertainty model h_ϕ, Kalman/RTS smoother, and trust-region update as KI-VLA, but applies the corrector only once after the final flow-matching sample is generated. It provides a final-only reference, but comparison with full interleaving does not isolate placement because the number of corrector calls also changes.

KI-VLA (interleaved). This is the proposed method: we apply the same corrector during flow-matching decoding according to the correction schedule in Appendix E.3.

E.3 Correction Schedule Details

Let N denote the number of flow-matching decoding steps. We define a normalized decoding progress variable $\xi_s \in [0, 1]$, where $\xi_s = 0$ corresponds to early noisy samples and $\xi_s = 1$ corresponds to late action-like samples. In the π_0-style implementation, flow time τ decreases from noise to data, so we set

$$\xi_s = 1 - \tau_s.$$

The corrector is invoked according to

$$\textsc{ApplyCorr}(s, \xi_s; M, \xi_0) = \mathbf{1}\{s \bmod M = 0\} \cdot \mathbf{1}\{\xi_s \geq \xi_0\}.$$

Equivalently, in the implementation using the flag `correct_after_tau`, the threshold condition is $\tau_s \leq \tau_{\max}$, where $\xi_0 = 1 - \tau_{\max}$.

We evaluate the following schedule families:

- **No correction:** $M = N + 1$, so $N_{\text{corr}} = 0$.
- **Final-only correction:** $M = N$ and the threshold selects only the final flow step. In our implementation with $N = 10$, this is realized by `correct_every=10` and `correct_after_tau=0.05`.
- **Full interleaving:** $(M, \xi_0) = (1, 0)$, so the corrector is applied after every flow step.
- **Sparse interleaving:** $M > 1$ and/or $\xi_0 > 0$, so the corrector is applied only at selected decoding steps, typically later steps where the generated chunk is closer to the action manifold.

These schedules characterize three practical questions: (i) whether kinematics-informed correction helps at all, (ii) how repeated decoding-time correction compares with a single final correction, and (iii) how much performance is retained when the number of corrector calls is reduced. Because the schedules vary both timing and correction frequency, they should be interpreted as a compute–performance study rather than as an isolated causal test of placement.

The schedule results and runtime trade-off are reported in Table 4 in the main paper.

Because these schedules also vary the number of corrector calls, they establish a compute–performance trade-off but do not by themselves identify placement as the sole causal factor.

E.4 Corrector Ablation Details

To understand which components of KI-VLA drive performance, we perform internal ablations by replacing or removing one component at a time while keeping the base flow policy and evaluation protocol fixed.

Action-space smoothing instead of kinematic latent smoothing. In the full method, decoded operational-space actions are treated as measurements of latent joint increments through

$$H_{t+k} = [J_\theta(\bar{q}_{t+k}) \;\; 0].$$

In the action-space smoothing ablation, we remove this kinematic bridge and smooth directly in action space using an identity-style observation model. This tests whether the learned joint-space latent representation is necessary, or whether generic action-space temporal smoothing is sufficient.

Fixed measurement covariance. The full method uses a learned heteroscedastic measurement covariance Σ^u_{t+k} to modulate how strongly the corrector trusts each decoded action. In this ablation, we replace it with a fixed isotropic covariance:

$$\Sigma^u_{t+k} = \sigma^2 I.$$

We set $\sigma^2 = 1 \times 10^{-2}$, matching the released evaluation configuration.

Analytic URDF Jacobian. When a URDF/kinematic model is available, we can compute the geometric Jacobian $J_{\mathrm{URDF}}(q)$ at the current linearization state $\bar{q}_{t+k}$ using the same end-effector frame convention as the action interface. We then replace $J_\theta(\bar{q}_{t+k})$ with $J_{\mathrm{URDF}}(\bar{q}_{t+k})$ inside the observation matrix $H_{t+k} = [J \;\; 0]$, keeping all other corrector components unchanged. This tests whether learning J_θ provides benefit beyond directly using an analytic Jacobian.

To make the comparison fair, we match the URDF Jacobian convention to the action interface: the translational block is expressed in the same Cartesian frame as the end-effector delta translation, and the rotational block is aligned with the axis-angle $\mathrm{Log}(\cdot)$ convention used by the policy. We re-evaluate $J_{\mathrm{URDF}}(\bar{q}_{t+k})$ at the same linearization states used by KI-VLA and keep $(Q, \Sigma^u, s(\xi),$ RTS, trust region$)$ identical to the learned-J_θ setting.

Constant progress scaling. The full method uses a progress-dependent scaling factor $s(\xi_s)$, allowing early noisy samples and late action-like samples to be treated differently. In this ablation, we set

$$s(\xi_s) = 1$$

for all decoding steps. This tests whether decoding-progress-aware correction improves stability.

Forward Kalman filter only. The full method applies Rauch-Tung-Striebel smoothing to obtain a fixed-interval MAP estimate over the full action chunk. In this ablation, we use only the forward Kalman filter estimates and remove the backward RTS pass. This tests whether full-chunk smoothing provides benefits beyond causal filtering.

No trust region. The full method bounds the correction magnitude around the decoder proposal:

$$u_{t+k} = \hat{u}_{t+k} + \mathrm{clip}_{\rho^{\mathrm{eff}}_{t+k}}(\bar{u}_{t+k} - \hat{u}_{t+k}).$$

In this ablation, we directly replace the proposal with the corrected projection:

$$u_{t+k} \leftarrow \bar{u}_{t+k}.$$

This tests whether explicit action-space deviation control is necessary to prevent overly aggressive corrections.

E.5 Real-Robot Protocol Details

We evaluate the real-robot system on four manipulation tasks: Place Shoe in Box, Hang Glass Cup, Child Toy Controls, and Move Object by Language. Each task is designed to stress a different aspect of execution: tight-clearance placement, wrist-orientation alignment, contact-mode switching, smooth terminal approach, and language-conditioned spatial control.

Demonstrations and evaluation. For each task, we collect 50 teleoperated demonstrations per variation using the same camera and robot configuration as evaluation. At test time, each episode randomizes the initial object pose and task-specific nuisance factors. We run 50 evaluation trials per task and report binary success rate.

Use of URDF. The robot provides joint-angle readings but not a calibrated online end-effector pose stream. Therefore, we use the manufacturer-provided URDF for offline pose reconstruction and metric computation from logged joint states. The online KI-VLA corrector does not require an analytic Jacobian or an explicit inverse-kinematics solver. Instead, it uses the learned surrogate J_θ described in the main method.

Injected joint disturbance. To evaluate robustness to transient command corruption, we inject a $+30^\circ$ pulse on joint 4 for approximately 0.5 seconds, corresponding to 15 frames at roughly 30 Hz. The disturbance is applied shortly before the end-effector reaches the target object, where local command errors are most likely to affect success. This stress test is not intended to model a specific

hardware failure mode. Rather, it provides a controlled diagnostic for whether the corrector can attenuate a locally corrupted proposal while preserving the intended task motion.

Clean versus disturbed rollouts. For clean rollouts, no artificial disturbance is applied. For disturbed rollouts, the base VLA executes the disturbed proposal directly, while KI-VLA receives the disturbed proposal and applies the kinematics-informed corrector before sending the final command to the robot. This setup directly measures whether the corrector can reject transient action corruption.

Success criteria. Each task uses a binary success criterion. For Place Shoe in Box, the shoe must be fully inside the box and stable after release. For Hang Glass Cup, the cup handle must be placed on the hook and remain hanging. For Child Toy Controls, the specified button or switch must be correctly actuated without triggering a neighboring control. For Move Object by Language, the specified object must be moved to the requested spatial relation or target region without excessive disturbance to other objects.

E.6 Command-Level Metrics

In addition to task success, we report command-level diagnostics to evaluate whether KI-VLA reduces impulsive or high-energy behavior during disturbed execution. At each control frame k, we log the policy prediction q_k^{pred}, the disturbed proposal q_k^{dist}, the final command sent to the robot q_k^{sent}, and the observed joint state q_k^{obs}. Unless otherwise stated, all safety and smoothness metrics are computed on q^{sent}. For the base VLA, no correction is applied, so $q_k^{\mathrm{sent}} = q_k^{\mathrm{dist}}$. For KI-VLA, q_k^{sent} is the command after kinematics-informed correction.

Disturbance-active window. We define the disturbance-active window as

$$\mathcal{W} = \left\{ k : \left\| q_k^{\mathrm{dist}} - q_k^{\mathrm{pred}} \right\|_\infty > \varepsilon \right\}, \tag{35}$$

where ε is a small numerical tolerance. This window identifies the frames in which the synthetic command pulse is active.

Correction magnitude. We measure how strongly the corrector modifies the disturbed proposal during $\mathcal{W}$:

$$C_{\max} = \max_{k \in \mathcal{W}} \left\| q_k^{\mathrm{sent}} - q_k^{\mathrm{dist}} \right\|_\infty, \qquad C_{\mathrm{mean}} = \frac{1}{|\mathcal{W}|} \sum_{k \in \mathcal{W}} \left\| q_k^{\mathrm{sent}} - q_k^{\mathrm{dist}} \right\|_\infty. \tag{36}$$

These values quantify intervention strength. They are zero for the base VLA because the disturbed command is sent directly, and should therefore be interpreted together with task success and motion-quality metrics rather than as standalone performance measures.

Table 6: Disturbed command-level diagnostics. Lower motion metrics are better.

Task	Method	$C_{\max}$	C_{mean}	Peak EE Vel.	Peak EE Acc.	S_q	$A^q_{\max}$
Place Shoe in Box	Base	0	0	2.11	63.24	1.07	960.64
	KI-VLA	31.02	25.77	1.21	33.89	0.93	831.10
Hang Glass Cup	Base	0	0	2.40	72.02	0.95	854.47
	KI-VLA	29.33	26.51	2.05	62.77	0.72	644.79
Child Toy Controls	Base	0	0	3.41	101.99	1.19	904.61
	KI-VLA	29.43	25.94	1.07	38.48	0.75	516.79
Move Object by Language	Base	0	0	2.20	65.70	1.11	724.50
	KI-VLA	28.25	24.76	1.01	24.09	0.76	445.96

Peak end-effector velocity and acceleration. Using end-effector positions p_k^{ee} reconstructed from logged joint states, we compute

$$V^{\mathrm{ee}}_{\max} = \max_k \left\| \frac{p^{\mathrm{ee}}_{k+1} - p^{\mathrm{ee}}_k}{\Delta t_k} \right\|_2, \tag{37}$$

and

$$A^{\mathrm{ee}}_{\max} = \max_k \left\| \frac{v^{\mathrm{ee}}_{k+1} - v^{\mathrm{ee}}_k}{\Delta t^{\mathrm{acc}}_k} \right\|_2, \qquad v^{\mathrm{ee}}_k = \frac{p^{\mathrm{ee}}_{k+1} - p^{\mathrm{ee}}_k}{\Delta t_k}. \tag{38}$$

Lower values indicate fewer high-energy Cartesian bursts.

Joint smoothness and peak joint acceleration. We measure joint-command smoothness by the average second difference:

$$S_q = \frac{1}{K-2} \sum_{k=0}^{K-3} \left\| q_{k+2}^{\text{sent}} - 2q_{k+1}^{\text{sent}} + q_k^{\text{sent}} \right\|_2 . \tag{39}$$

We also report peak joint acceleration:

$$A_{\max}^{q} = \max_k \left\| \frac{\dot{q}_{k+1}^{\text{sent}} - \dot{q}_k^{\text{sent}}}{\Delta t_k^{\text{acc}}} \right\|_2, \qquad \dot{q}_k^{\text{sent}} = \frac{q_{k+1}^{\text{sent}} - q_k^{\text{sent}}}{\Delta t_k}. \tag{40}$$

Lower S_q and $A_{\max}^{q}$ indicate less abrupt and less bursty joint-level control.

E.7 Compute Resources

All simulation experiments were conducted using 8 A100 GPUs with 40GB memory. The base flow-matching VLA for each RoboTwin task was trained for 30k steps with batch size 32, following the official π_0-based RoboTwin recipe. Training one base policy required approximately 13 wall-clock hours on 8×A100 GPUs.

The auxiliary KI-VLA modules are lightweight. Training the learned Jacobian surrogate J_θ for 100 epochs took 220 s on a single A100, and training the uncertainty module h_ϕ for 100 epochs took 190 s on a single A100 (combined < 0.12 A100 GPU-hours). The combined auxiliary checkpoint footprint is $\approx$ 64K parameters, which is negligible compared to the base VLA.

For real-robot experiments, policy inference was run on Nvidia-RTX 3090. Each task was evaluated over 50 clean rollouts and 50 disturbed rollouts. The wall-clock time for real-robot evaluation was dominated by physical execution rather than model inference.

The inference-time overhead of KI-VLA is reported in Sec. 4.4 and Appendix E.8. Full interleaving adds 878 ± 29 ms of corrector compute per decoded chunk, corresponding to 3.12% relative overhead in our evaluation stack, while sparse interleaving with three corrector calls adds 260 ± 18 ms, corresponding to 0.93% overhead.

E.8 Runtime Measurement Details

Since KI-VLA is an inference-time correction module, we report wall-clock overhead in addition to task performance. For each schedule, we record: (i) the number of corrector calls per decoded chunk N_{corr}, (ii) the accumulated time spent inside the KI-VLA corrector, (iii) the no-corrector wall time per chunk, and (iv) the relative overhead of correction.

All timing numbers in Table 4 exclude warmup runs. The no-corrector baseline has wall time 28.10 ± 0.23 s/chunk in our evaluation stack. We define relative overhead as

$$\text{Overhead} = 100 \cdot \frac{T_{\text{corr}}}{T_{\text{wall}}^{\text{no-corr}}},$$

where T_{corr} is the accumulated corrector compute per chunk and $T_{\text{wall}}^{\text{no-corr}}$ is the no-corrector wall time. This denominator is intentionally conservative: it measures the added cost of the corrector relative to the full base decoding pipeline. In our implementation, base flow-matching inference dominates wall time, so the corrector adds a small relative overhead even under full interleaving.

F Additional Robustness and Diagnostic Analyses

F.1 Per-Task Confidence Intervals and Significance Tests

To characterize uncertainty in the RoboTwin results beyond the aggregate success rates, we report per-task Wilson 95% confidence intervals and two-proportion tests in Table 7. Individual tasks have limited statistical power at 100 rollouts per cell, so we use these tests primarily to quantify uncertainty rather than to make task-by-task significance claims. The overall evidence is therefore interpreted

together with the pooled effect and the consistency of improvement across tasks. These intervals and tests quantify rollout-level evaluation variability for the evaluated checkpoints and do not estimate variability across independently trained policy seeds.

Table 7: Per-task Wilson 95% intervals and two-proportion z-tests, $n = 100$ per cell, comparing the π_0^{EP} baseline with KI-VLA. Only Lift Pot (Easy) and Hang Cup (Hard) reach $p < 0.05$, and neither survives Bonferroni correction across the sixteen tests ($\alpha = 0.0031$). The pooled effect and the consistency of its direction, not these per-task tests, carry the claim.

	Clean (Easy)					**Randomized (Hard)**				
Task	base	95% CI	ours	95% CI	p	base	95% CI	ours	95% CI	p
Move Playingcard	53	[43.2, 62.6]	61	[51.2, 70.0]	0.253	23	[15.8, 32.2]	30	[21.9, 39.6]	0.262
Turn Switch	41	[31.7, 50.9]	51	[41.4, 60.6]	0.156	30	[21.9, 39.6]	35	[26.4, 44.7]	0.450
Click Bell	35	[26.4, 44.7]	33	[24.6, 42.7]	0.765	9	[4.8, 16.2]	17	[10.9, 25.6]	0.093
Hang Cup	12	[7.0, 19.8]	17	[10.9, 25.6]	0.315	5	[2.2, 11.2]	13	[7.7, 21.1]	0.048
Beat Block	65	[55.3, 73.6]	74	[64.6, 81.6]	0.167	47	[37.5, 56.7]	55	[45.2, 64.4]	0.258
Lift Pot	85	[76.7, 90.7]	96	[90.2, 98.4]	0.008	60	[50.2, 69.1]	65	[55.3, 73.6]	0.465
Place Shoes	56	[46.2, 65.3]	64	[54.2, 73.0]	0.248	28	[20.2, 37.6]	30	[21.9, 39.6]	0.755
Diverse Bottle	33	[24.6, 42.7]	41	[31.7, 50.9]	0.241	12	[7.0, 19.8]	16	[10.0, 24.4]	0.415
Pooled ($n = 800$)	**47.5**	[44.1, 51.0]	**54.6**	[51.2, 58.0]	**0.004**	**26.8**	[23.8, 29.9]	**32.6**	[29.5, 36.0]	**0.010**

F.2 Uncertainty and Hyperparameter Sensitivity

All settings use Turn Switch with 100 rollouts per split. We vary one parameter at a time around the default configuration to examine whether the corrector remains effective under moderate mis-calibration and to identify failure regimes under more severe mis-scaling.

Table 8: Sensitivity to corrector hyperparameters on Turn Switch (100 rollouts per split). One parameter is varied at a time while all other settings are held fixed.

Setting	Easy ↑	Hard ↑
Default	51	35
$\Sigma^u \times 0.1$	48	32
$\Sigma^u \times 10$	40	29
$Q \times 0.1$	42	33
$Q \times 10$	30	26
$\rho_{\max} = 0.01$	43	33
$\rho_{\max} = 0.03$	51	35
$\rho_{\max} = 0.10$	44	31

Moderate perturbations retain part of the benefit, whereas severe mis-scaling can remove it. In particular, $Q \times 10$ reduces performance to $30/26$, showing that the latent dynamics prior must be reasonably calibrated. Likewise, excessive measurement covariance ($\Sigma^u \times 10$) reduces performance to $40/29$, indicating that over-discounting the decoded proposal can be detrimental. The trust region bounds individual action updates, but it does not guarantee task success under arbitrary hyperparameter mis-scaling.

F.3 NFE Generalization

We vary the number of flow-matching Euler steps while keeping the remaining evaluation configuration fixed. As shown in Table 9, the advantage of KI-VLA is concentrated in the few-step decoding regime and closes at $N = 50$ on Turn Switch. At the default $N = 10$ operating point, KI-VLA improves success by $+10$ pp on Easy and $+5$ pp on Hard, but it is not a universal replacement for higher-NFE integration.

Table 9: Sensitivity to the number of flow evaluations (NFE) on Turn Switch (100 rollouts per cell). Each entry reports Easy / Hard success rate (%).

N	Base	KI-VLA
5	27 / 30	30 / 40
10	41 / 30	51 / 35
20	33 / 47	44 / 45
50	33 / 43	33 / 43

F.4 Mechanistic Analysis and Interpretation

Across logged Turn Switch episodes (170 clean / 180 Hard episodes with corrector calls), the association between per-call correction magnitude and episode success is weak: clean point-biserial $r \approx +0.08$ and Hard $r \approx -0.04$. A thresholded detector of the injected disturbance window obtains approximately 4.8% precision across tested firing thresholds, recall of 22–60%, and roughly 52% clean false positives. These results do not support a binary selective-detector interpretation. Accordingly, we describe h_ϕ, λ_{t+k}, and the trust region as continuous uncertainty-weighted modulation rather than as a trigger that identifies a sparse set of decisive actions.

F.5 Observed versus Unobserved Disturbance Channel

To clarify scope, we conduct an additional diagnostic on Turn Switch in simulation, comparing a proposal-level pulse that the corrector can observe with an actuator-level pulse applied after correction. At 100 rollouts per cell, the observed proposal-pulse setting changes from 30% to 36%, whereas the actuator-level pulse setting changes from 33% to 34%; both differences are within sampling noise. We therefore treat this experiment as diagnostic rather than efficacy evidence. The result is consistent with KI-VLA correcting proposal corruption that enters before the corrector, while offering little protection against disturbances introduced downstream of it.

F.6 Qualitative Rollouts Under Injected Disturbance

Figure 3 shows what the aggregate success rates of Sec. 4.3 look like frame by frame. The same $+30^\circ$ joint-4 pulse is injected during execution for the base policy and for KI-VLA, and the tracked end-effector path is overlaid so that the response to the pulse is visible rather than inferred from the success counts alone. The full video of this rollout is provided in the supplementary material.

Two patterns recur across the three tasks. First, the base policy's failures are failures of *recovery* rather than of intent: it reaches the correct object and begins the correct motion, then the pulse displaces the end effector and the subsequent commands do not bring it back, leaving the cup unhung and the shoe struck on the way into the box. Second, where the base policy does complete the task it does so through a visibly larger excursion, as on the button press, where it recovers the contact but loses grip stability first. The corrected rollouts stay closer to the pre-disturbance path in all three cases, which is the behaviour the command-level diagnostics in App. E.6 measure in aggregate.

G Real-Robot Task Definitions and Stress Tests

G.1 Task Design and Success Criteria

Place Shoe in Box (Horizontal). The robot must pick up a shoe and place it *horizontally* into a shoebox whose inner dimensions are only slightly larger than the shoe. Success requires the shoe to be fully inside the box, approximately flat, and stable after release. The primary variation is **left vs. right shoe**, which changes mirrored geometry and the desired wrist pose for alignment. We annotate the following substeps: (i) grasp shoe, (ii) lift without dropping, (iii) move above the box opening, (iv) align shoe yaw with the box long axis, (v) lower into the box without scraping or collision, (vi) release stably.

Hang Glass Cup. The robot must grasp a glass cup and hang it on a hook by its handle. To stress kinematic consistency at terminal contact, we vary **handle orientation**, **hook height**, and **hook**

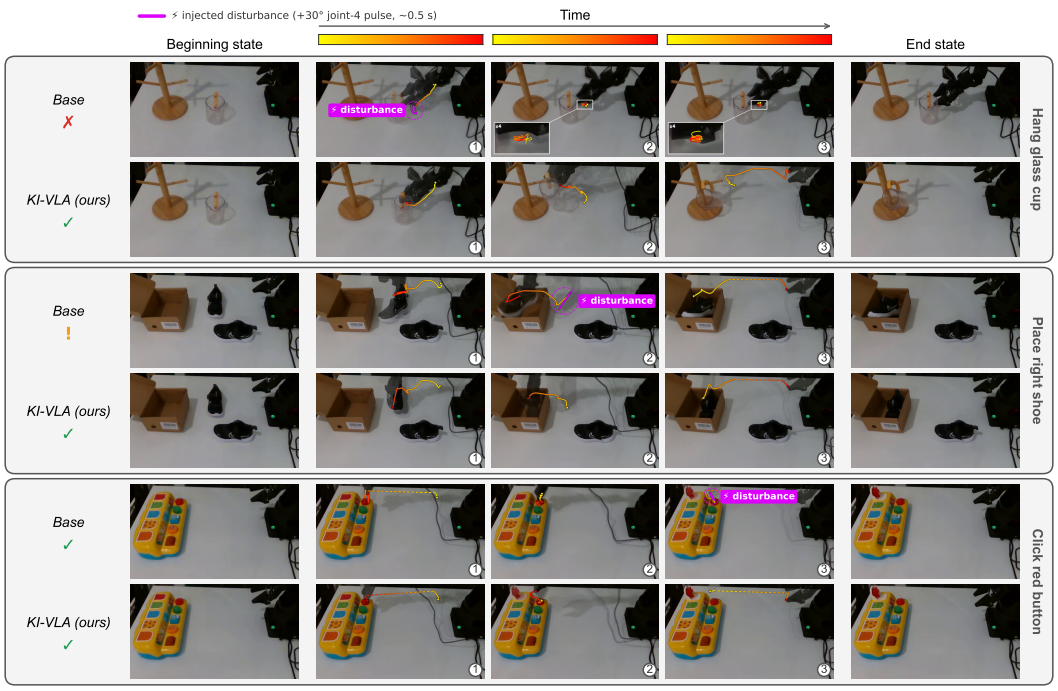


Figure 3: Action-pulse disturbance test (front camera). A $+30°$ pulse is injected on joint 4 for $\sim$0.5 s (15 frames at $\sim$30 Hz) during execution. Purple trajectory segments indicate the interval in which the injected disturbance is active for the base policy. Each panel compares the base policy (top) with KI-VLA (bottom) on hanging a glass cup, placing the right shoe in the box, and clicking the red button. Leftmost column: initial state; columns ①–③: key frames with earlier frames of each segment ghosted and the tracked end-effector path overlaid, coloured yellow→red over time; rightmost column: final frame. Dashed segments mark intervals where the end effector leaves the camera view (linearly interpolated). Insets magnify the stalled base gripper $\times$3.5. Under the disturbance, the base policy fails to hang the cup ($\times$), unintentionally kicks the object while placing the shoe (!), and loses grip stability on the button, yet still completes the press (✓). KI-VLA completes all three tasks with visibly smaller deviations (✓). Single representative rollouts; qualitative illustration only.

clearance. Success requires the cup to remain hanging after release. Substeps: (i) approach cup, (ii) grasp cup/handle, (iii) lift without collision, (iv) rotate handle toward hook, (v) insert handle onto hook (contact-stable approach), (vi) release.

Manipulate Child Toy Controls. The robot interacts with a child activity toy containing multiple controls. The instruction specifies which control to actuate. The primary variation is **contact mode**: pressing a button vs. actuating a handle switch. Button trials require a mostly normal approach and short pressing motion; handle-switch trials require lateral/rotational contact after approach. Success requires the correct control to be actuated and no incorrect control to be triggered. Substeps: (i) approach the correct target, (ii) establish contact at the correct location, (iii) apply the correct motion primitive (press vs. switch), (iv) complete actuation.

Move Object by Language. The robot receives a language instruction specifying both an object and a spatial goal (e.g., left/right/front/behind/near/far relative to a reference). This task tests whether kinematics-aware correction preserves semantic intent while improving smoothness and recoverability. Success requires moving the correct object to the requested spatial relation without disturbing distractors. Substeps: (i) select the correct object, (ii) grasp, (iii) move toward the target spatial region, (iv) release at the requested relation, (v) avoid disturbing other objects.

G.2 Stress-Test Probes: Action Disturbance

Pulse action disturbance (recovery from a bad command). To probe recovery from sudden command corruption, we inject a short pulse disturbance into the executed command stream during evaluation. We use a *time-localized pulse* (rather than a persistent bias) for clear attribution: e.g., add a $+30^\circ$ offset to a chosen arm joint for 0.5 s starting at $t = 2.0$ s. We evaluate two variants:

- **Proposal-level pulse (default).** The pulse is applied to the policy proposal *before* kinematics-aware correction. For the base policy, the disturbed command is executed as-is. For the final-only corrector variant and KI-VLA, the disturbed proposal is passed through the corrector before execution. This directly tests whether the corrector can attenuate a corrupted proposal while preserving intended motion.
- **Actuator-level pulse (optional).** The pulse is applied *after* all policy-side correction, modeling an external execution perturbation. This tests closed-loop recovery rather than proposal rejection.

Applying the pulse across action spaces. If a policy outputs joint-space commands, we apply the pulse directly in joint space. If a policy outputs operational-space actions (end-effector poses or deltas), we inject an equivalent perturbation using a local Jacobian linearization, e.g., $\delta u \approx J(q)\delta q$, or by perturbing the latent joint-increment proposal before projecting back to operational space. All perturbations are clipped to remain within safety limits.

Disturbance metrics. We report task success under disturbance together with command-level motion diagnostics, including correction magnitude, peak end-effector velocity and acceleration, joint smoothness, and peak joint acceleration, as defined in Appendix E.6.