POLICYIK: HIGH-PRECISION INVERSE KINEMATICS FOR ROBOT ARMS WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Inverse kinematics (IK) converts end-effector pose commands into joint motion, and many vision-language-action (VLA) policies rely on it to act on a given arm and to transfer across embodiments. Precise IK remains laborious: redundancy and singularities complicate the inverse mapping, execution depends on hand-tuned controllers, and training data on physical robots are costly to collect. We present PolicyIK, a goal-conditioned reinforcement learning policy that updates joint-position references from pose feedback and observes its accumulated reference relative to the measured joints, so the arm can rest on the goal rather than millimeters away. Two-scale pose errors with an error-dependent action gain let it correct millimeter residuals without slowing large motions, and rewards for progress and for staying within tolerance teach it to both reach and retain the target. In simulation, a single attempt keeps the mean errors over the final second within 1 mm and 0.2° on 93.5% of held-out tasks, and a deployment protocol raises success to 98.9%. On physical hardware, PolicyIK achieves comparable or higher holding success than the evaluated classical IK controllers under matched execution budgets, with higher success rates under longer holding requirements. In simulated pick-and-place, it executes the end-effector commands of the $\pi_{0.5}$ VLA policy with 95.7% success. These results show that reinforcement learning can provide high-precision inverse kinematics for robot arms.

1 INTRODUCTION

Many learned manipulation policies act on a robot arm through inverse kinematics (IK), which turns end-effector pose commands into joint motion. Vision-language-action (VLA) and other generalist policies often predict end-effector poses or pose increments and leave the conversion to joint commands to an IK or operational-space controller (Open X-Embodiment Collaboration et al., 2024; Octo Model Team et al., 2024; Kim et al., 2024). Because this interface is shared across arms, a policy trained on one robot can also run on another (Chi et al., 2024; Chen et al., 2024; Lepert et al., 2024; Ha et al., 2024). Such transfer delegates the feasibility and accuracy of every command to the target robot's IK. Transfer failures occur at joint limits and on motions that the target arm cannot follow (Chi et al., 2024; Yang et al., 2023), and transfer methods assume controllers that reach commanded poses within a few millimeters (Chen et al., 2024). The accuracy of IK thus carries over to every policy that acts through it.

Developing and deploying a precise pose controller raises three challenges. First, the inverse mapping itself is difficult. A redundant arm can reach a pose with many joint configurations, and near singularities small pose corrections demand large joint motions, so numerical solvers trade accuracy for robustness and can fail even on reachable poses (Burdick, 1989; Buss, 2009; Deo & Walker, 1995; Beeson & Ames, 2015). Second, the realized pose depends on robot-specific execution. Finite gains, gravity, and joint limits shift where a position-controlled arm comes to rest, and the controllers that compensate for these effects rely on damping factors, task weights, and gains that experts tune for each robot and task (Lynch & Park, 2017; Charbonneau et al., 2018; Buss & Kim, 2005; Lee et al., 2026). Third, collecting training data on physical robots is costly, as each trial requires robot execution and a reset.

We investigate reinforcement learning (RL) for this setting. An RL policy learns the mapping from pose error to joint commands through interaction with the robot's dynamics, without IK la-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

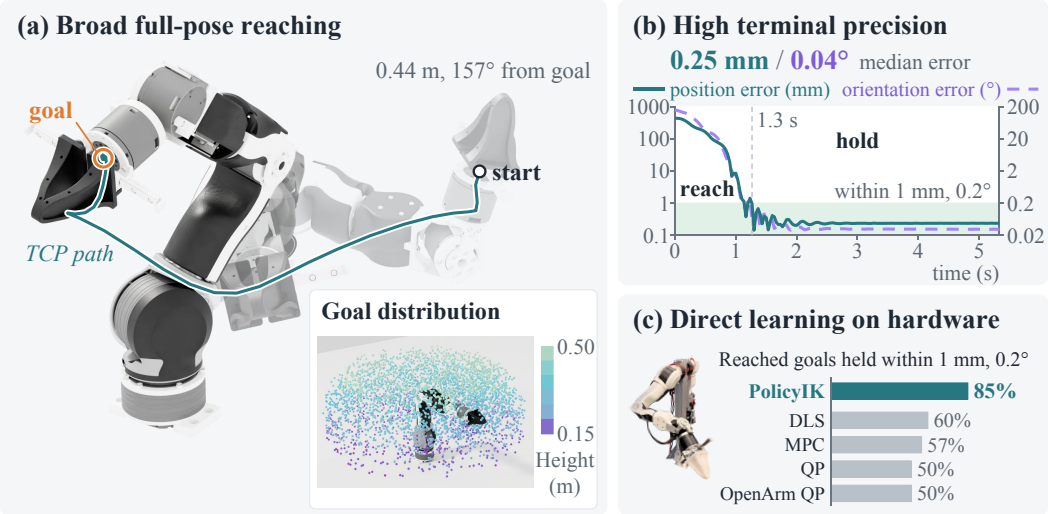

Figure 1: **PolicyIK: from broad reaching to precise regulation.** (a) Full-pose reaching across the simulated workspace. (b) Reaching and holding within 1 mm and 0.2°. (c) Direct learning on hardware improves sustained pose holding relative to classical controllers.

bels or demonstrations. Sampling goals from reachable configurations yields feasible goals across the workspace, the reward can target the required tolerance, and a new mounting calls for retraining rather than manual tuning. Prior RL controllers have reached position goals with fixed-base arms and tracked full poses with legged manipulators, with errors from several millimeters to a few centimeters (Kumar et al., 2021; Aljalbout et al., 2024; Portela et al., 2025; He et al., 2026), and a hybrid reaches sub-millimeter accuracy in kinematic simulation by pairing the policy with damped least squares, which supplies the final correction (Yu & Tan, 2025). Reaching millimeter-level full-pose accuracy with the policy alone calls for a formulation designed around terminal precision.

We present PolicyIK, a goal-conditioned RL formulation that learns high-precision IK for robot arms through a joint-position interface (Figure 2). As in closed-loop IK (Siciliano, 1990), the policy updates a joint reference from pose feedback, but it learns this update from interaction. The formulation addresses three properties of precise regulation. First, the policy accumulates its actions in the reference and observes the reference relative to the measured joints, so it can retain the offset that the position controller needs at rest. Second, pose errors enter the observation at coarse and fine scales, and an error-dependent gain reduces the action scale near the goal, which resolves small residuals while preserving large motions. Third, rewards for progress beyond the best error so far and for remaining within tolerance complement dense tracking terms, so training rewards both reaching the tolerance and staying within it. We use parallel PPO training in simulation and an off-policy learner with experience replay on hardware to accommodate the different interaction budgets.

We evaluate PolicyIK in simulation, on a physical arm, and in executing the end-effector commands of a VLA policy (Figure 1). On a simulated 7-DoF arm, PolicyIK reaches median errors of 0.25 mm and 0.04° and meets the 1 mm and 0.2° tolerance on 93.5% of held-out tasks in one attempt and on 98.9% with a deployment protocol. Removing any single design component lowers single-attempt success by 1.8 to 9.2 percentage points. Trained directly on a physical arm, PolicyIK achieves comparable or higher holding success than the evaluated classical IK controllers under matched execution budgets, with higher success rates when the pose must remain within 1 mm and 0.2° for longer holding windows. In simulated pick-and-place, it executes the end-effector commands of $\pi_{0.5}$ (Physical Intelligence et al., 2025) with 95.7% success across ten adapted LIBERO-Object tasks (Liu et al., 2023), without any pick-and-place data in its training. PolicyIK is inexpensive: an actor forward pass takes 0.035 ms on CPU, well below the 33 ms control period. Our contributions are as follows:

- We cast inverse kinematics for robot arms as goal-conditioned reinforcement learning and show that a learned policy alone reaches millimeter-level full-pose precision.

- We raise terminal precision by exposing the otherwise hidden joint reference to the policy, resolving pose errors at two scales with an error-dependent gain, and rewarding each accuracy gain once and each step within tolerance.
- We show that PolicyIK learns on hardware through off-policy updates with near-goal hindsight relabeling and a start-state curriculum, keeping the poses it reaches within 1 mm and 0.2° more reliably than classical IK controllers, and executes a VLA policy's commands in simulated pick-and-place with 95.7% success.

2 Related Work

Inverse kinematics and task-space control. Classical approaches map pose commands to joint motion analytically, numerically, or through feedback. Closed-form solutions exist mainly for arms with special geometry (Diankov, 2010), so general and redundant arms rely on damped least squares or constrained optimization (Wampler, 1986; Deo & Walker, 1995; Beeson & Ames, 2015), and closed-loop controllers convert task-space error into joint references under joint limits and secondary objectives (Siciliano, 1990; Haviland & Corke, 2023; Cheng et al., 2021; Xie et al., 2020). These methods require tuning for each robot (Charbonneau et al., 2018; Buss & Kim, 2005) and leave execution of joint motion to the robot's controller. Learned IK models fit the inverse map from sampled configurations or demonstrations (Ames et al., 2022; Habekost et al., 2024; Yang et al., 2026) and rely on numerical refinement for high accuracy (Morgan et al., 2024). In contrast, PolicyIK learns closed-loop IK from interaction with the robot's dynamics and regulates the realized pose.

Reinforcement learning for reaching and pose tracking. Reinforcement learning controllers for end-effector reaching range from benchmark tasks to fixed-base arms and legged manipulators. Common multi-goal benchmarks count a reach as successful within 5 cm of a position goal (Plappert et al., 2018; Yu et al., 2019). Policies for fixed-base arms map task-space errors to joint commands and reach position goals with errors of millimeters to centimeters (Kumar et al., 2021; Aljalbout et al., 2024), while whole-body controllers track full end-effector poses on legged manipulators to within a few centimeters (Ha et al., 2024; Portela et al., 2025; He et al., 2026). When tighter tolerances are needed, learning is paired with a classical solver, such as damped least squares, that performs the final correction (Yu & Tan, 2025). These studies thus favor broad reaching and tracking, mostly for position goals, and rely on classical components when millimeter-level accuracy is needed. PolicyIK reaches this accuracy without a classical solver and holds the pose after arrival.

Learning for execution-error compensation. Residual methods add learned corrections to a conventional controller or its reference trajectory (Johannink et al., 2019; Pavlichenko & Behnke, 2022) and build on its tuning, whereas integrating methods accumulate policy outputs in the commanded state to reduce steady-state error (Tang et al., 2023; Aljalbout et al., 2024), and integral state augmentation exposes accumulated information for reference tracking (Weber et al., 2022). Once commands are integrated, the accumulated reference becomes part of the system state. PolicyIK observes this reference relative to the measured joints and pairs it with multiscale pose errors and error-dependent action scaling, so one policy handles both large motions and millimeter corrections.

3 Problem Formulation

Let $q_t, \dot{q}_t \in \mathbb{R}^7$ denote the measured joint positions and velocities, $T_t = (R_t, p_t) = \mathrm{FK}(q_t)$ the end-effector pose, and $T_g = (R_g, p_g) \in SE(3)$ the goal pose. The policy acts through a joint reference z_t, which a fixed-gain position controller tracks under the rigid-body dynamics and the joint, command, and torque limits. Classical IK computes a configuration q^* with $\mathrm{FK}(q^*) = T_g$; executing it through the controller further requires a closed-loop motion that attains the pose from the start configuration q_0 and then holds it. PolicyIK learns this closed-loop regulation directly.

Goal-conditioned MDP. We use a goal-conditioned Markov decision process whose state contains the joint state, the reference z_{t-1}, the previous action, the goal T_g, and the best error reached for the goal (Section 4.3). At each 30 Hz policy step in simulation, a Gaussian policy $\pi_\theta(a_t \mid o_t)$ outputs a reference increment $a_t \in \mathbb{R}^7$ (Equation 1) from the observation o_t (Equation 2) and is

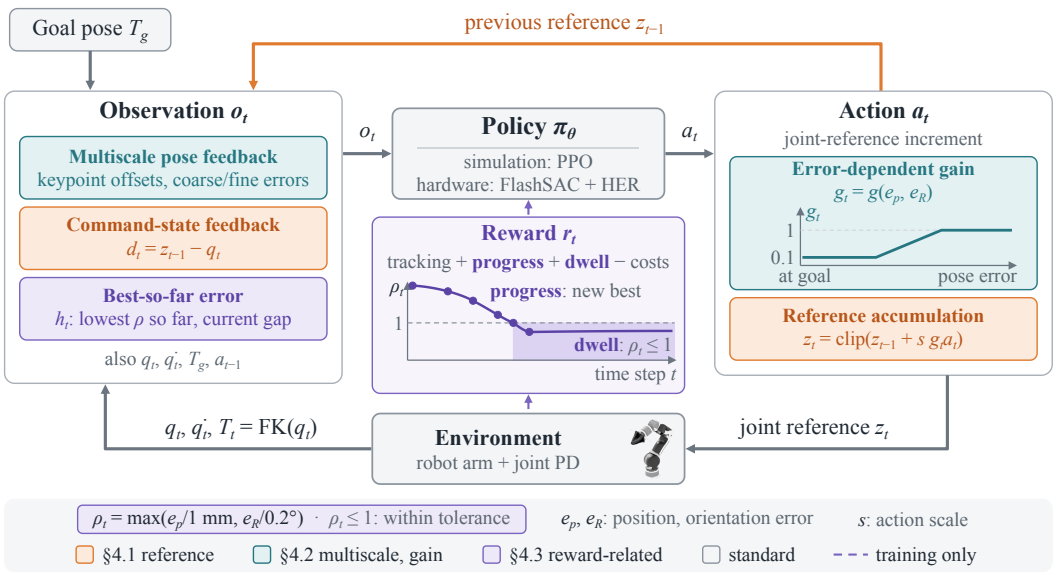

Figure 2: **PolicyIK control formulation.** At each step, the policy maps the observation to a joint-reference increment, which is scaled by the error-dependent gain, accumulated in the joint reference, and tracked by a joint PD controller. Colored blocks are the designs of Sections 4.1 to 4.3.

trained to maximize the expected discounted return with $\gamma = 0.99$. Evaluation executes the mean action.

Success criteria. Position and orientation errors are $e_p(t) = \|p_t - p_g\|_2$ and $e_R(t) = \|\text{Log}(R_g^\top R_t)^\vee\|_2$, and $\bar{e}_p$ and $\bar{e}_R$ denote their means over the final $H = 30$ steps (1 s) of an attempt. The *strict* criterion, $\bar{e}_p < 1\,\text{mm}$ and $\bar{e}_R < 0.2°$, measures terminal precision. The *coarse* criterion, 1 cm and 2°, measures whether the arm reaches the goal region, at an accuracy comparable to prior learned reaching policies; the gap between the two criteria therefore isolates failures of terminal regulation. The window means measure accuracy over the whole final second; Section 5.1.1 also reports containment, which requires both tolerances at every sample.

4 Method

At each step, PolicyIK updates a joint reference that the position controller tracks (Figure 2). Holding a pose to a millimeter through this interface raises three problems that coarse reaching can tolerate: the command that holds a pose is state that the joint measurements do not reveal (Section 4.1), the residuals that decide success lie orders of magnitude below the initial errors (Section 4.2), and dense tracking rewards barely distinguish meeting the tolerance from missing it (Section 4.3).

4.1 Reference Accumulation and Command-State Feedback

Under gravity and finite gains, a position-controlled arm settles away from its reference, so holding a goal requires a reference offset from the goal configuration (Appendix A). If each action is applied relative to the measured joints, the policy must regenerate this offset with a nonzero action at every step, and the arm sags whenever the action relaxes. PolicyIK instead accumulates its actions in the reference,

$$z_t = \text{clip}\big[z_{t-1} + s\, g(e_p, e_R)\, a_t,\ q^{\text{lo}},\ q^{\text{hi}}\big]\,, \tag{1}$$

so that a zero action retains the offset, a multi-step integration in the terminology of Aljalbout et al. (2024). Here s is the action scale, g is the error-dependent gain of Section 4.2, and $q^{\text{lo}}, q^{\text{hi}}$ are the joint limits.

Accumulation makes the reference part of the state: the same joint state, goal, and action lead to different next states under different references. Without observing the reference, the policy would

have to infer it from its action history. PolicyIK therefore feeds back the command state $d_t = z_{t-1} - q_t$, the offset between the reference and the measured joints, within the observation

$$o_t = \mathrm{concat}\big(q_t - \bar{q},\, \dot{q}_t,\, p_g^B,\, \Delta k_t^B,\, v_t^c,\, v_t^f,\, h_t,\, d_t,\, a_{t-1}\big), \tag{2}$$

where $\bar{q}$ is a nominal configuration, the superscript B denotes the base frame, Δk_t^B, v_t^c, and v_t^f are the pose features of Section 4.2, and h_t exposes the running best error of Section 4.3. The actor and critic share this observation. Section 5.1.2 compares the accumulated reference with a measured-joint anchor.

4.2 Multiscale Pose Feedback and Action Scaling

Correcting millimeter residuals requires the policy both to perceive them and to resolve them with its actions. Without error channels scaled to the tolerance, we observed the arm settling at a biased equilibrium a few millimeters from the goal. Input normalization scales each channel by its spread over the workspace, so millimeter differences occupy a negligible fraction of the normalized joint, goal, and keypoint inputs. PolicyIK therefore represents the pose error at two scales and adapts its step size to the error.

Multiscale pose feedback. Following keypoint-based pose representations (Portela et al., 2025), three noncollinear local offsets c_i define

$$\Delta k_{i,t}^B = (p_g^B + R_g^B c_i) - (p_t^B + R_t^B c_i), \tag{3}$$

which express translation and orientation as position errors that the policy can servo. To resolve small residuals, we also express the error in the current end-effector frame, clipped at a coarse and a fine scale,

$$v_t^\ell = \begin{bmatrix} \mathrm{clip}\big(R_t^\top (p_g - p_t)/\delta_\ell, -1, 1\big) \\ \mathrm{clip}\big(\mathrm{Log}(R_t^\top R_g)^\vee/\alpha_\ell, -1, 1\big) \end{bmatrix}, \qquad \ell \in \{c, f\}, \tag{4}$$

with $(\delta_c, \alpha_c) = (1\,\mathrm{cm}, 20^\circ)$ and $(\delta_f, \alpha_f) = (0.5\,\mathrm{mm}, 0.25^\circ)$. The coarse channels resolve errors below 1 cm and 20°, and the fine channels resolve errors at the scale of the tolerance: an orientation error of 0.2° fills 1% of the coarse range but 80% of the fine range.

Error-dependent action scaling. A single action scale sets both the speed of transport and the resolution of the final correction, because the reference moves by s times any error in the action. None of the fixed scales we tested served both: small scales slowed transport, and large ones lowered terminal precision. PolicyIK therefore scales each update by the gain

$$\begin{aligned} g(e_p, e_R) &= \max\{G(e_p; l_p, u_p), G(e_R; l_R, u_R)\}, \\ G(e; l, u) &= g_{\min} + (1 - g_{\min})\,\mathrm{clip}\left(\frac{e - l}{u - l}, 0, 1\right), \end{aligned} \tag{5}$$

which keeps the full step while either error is large and reduces it to $g_{\min} s$ once both errors fall below their respective lower thresholds. Because the gain follows the error rather than time, fine resolution engages whenever the arm nears the goal, regardless of duration. Section 5.1.2 evaluates both components.

4.3 Reward Design

The reward combines terms for the approach and for the terminal phase:

$$r_t = r_t^{\mathrm{track}} + r_t^{\mathrm{progress}} + r_t^{\mathrm{dwell}} - c_t^{\mathrm{motion}} - c_t^{\mathrm{limit}}. \tag{6}$$

Tracking guides the approach, progress and dwell reward reaching and holding the tolerance, and the costs regularize motion. Appendix B lists all terms and weights, and Section 5.1.2 evaluates the progress and dwell rewards.

Tracking. The tracking reward combines a linear position penalty, which acts across the workspace, with kernels $K(x; \sigma) = 1 - \tanh(x/\sigma)$ on the position and keypoint errors. Each kernel is bounded and steepest within its bandwidth of 5 to 20 cm, so it strengthens the pull toward the goal while keeping the reward scale fixed at large errors. Within a millimeter of the goal, however, each kernel changes by at most 2% of its range, a weak signal at the scale of the tolerance.

Tolerance-normalized error. Success is a conjunction: both errors must satisfy their tolerances at once. A reward on summed errors lets progress on one error compensate for the other, because the sum rewards reducing whichever error is easier while the other remains outside its tolerance. The precision terms therefore use the tolerance-normalized error and its running minimum,

$$\rho_t = \max\Big\{\frac{e_p(t)}{1\,\mathrm{mm}}, \frac{e_R(t)}{0.2^\circ}\Big\}, \qquad b_t = \min\{b_{t-1}, \rho_t\}. \tag{7}$$

Thus $\rho_t \leq 1$ exactly when both strict tolerances hold, and ρ_t decreases only when the binding error decreases. The running minimum b_t, reset for each new goal, records the best accuracy reached so far. The observation includes h_t, which contains b_{t-1} and the gap $\rho_t - b_{t-1}$ after clipping and scaling (Appendix B).

Progress. The progress reward drives ρ_t below one with three terms of distinct roles:

$$r_t^{\mathrm{progress}} = w_n\big[U(\rho_t) - U(b_{t-1})\big]_+ + w_\Phi(\gamma\Phi_t - \Phi_{t-1}) - w_b \min\{3, [\rho_t - b_{t-1}]_+\}, \tag{8}$$

where $[x]_+ = \max(x, 0)$, $U(\rho) = \frac{1}{2}[1 + \tanh((1-\rho)/\beta)]$ rises from 0 to 1 as ρ crosses one, and $\Phi_t = -\min(50, \rho_t)$. The improvement term pays only when the running best improves. Since b_t never increases, its payments telescope: each accuracy level is paid once and staying earns nothing, so the term can carry a large weight without rewarding oscillation or parking short of the tolerance, and U concentrates its payment on crossing into the tolerance. The potential-based shaping term (Ng et al., 1999) adds a dense signal within 50 tolerance units, which also rewards returning toward a best that the improvement term no longer pays for. The backslide term charges losses relative to the best, discouraging drift away from the accuracy already reached; capped at three tolerance units, it grows with small drifts but stays constant for the larger excursions that reconfiguration requires.

Dwell. The dwell reward $r_t^{\mathrm{dwell}} = w_d \mathbf{1}[\rho_t \leq 1]$ pays for every step inside the tolerance. The improvement term pays nothing for staying, and without the dwell reward, policies often touched the tolerance and then settled just outside it. Dwell makes retaining accuracy accrue return, while the improvement term rewards acquiring it. For the same reason, episodes end only at time-out, since ending a goal at success would forfeit the dwell rewards that follow.

Costs. Squared action-rate and joint-velocity costs regularize command changes and motion; their weights increase after an initial training phase, following the Isaac Lab reach schedule. A joint-margin cost keeps the worst joint away from its limits, since failing policies often ended with a joint at a limit. It applies only while $\rho_t > 10$, so it does not compete with the final correction near the goal.

4.4 Training and Deployment

We train with PPO (Schulman et al., 2017) and generalized advantage estimation (Schulman et al., 2016) in 4,096 parallel environments. Each 12 s training episode starts from a feasible configuration and samples goals every 4 s, so later goals start wherever the previous goal left the arm, and the policy practices both the approach and the terminal correction. Appendix B gives the architecture and training settings.

Deployment protocol. When further attempts are allowed, a fixed protocol retries a strict failure within 1 cm of the goal with an error-corrected goal and gives remaining failures up to eight attempts that first command a transient goal, judging every attempt against the original goal (Appendix C).

4.5 Real-Robot Implementation

We train a separate policy directly from physical interaction on a 7-DoF OpenArm, without initialization from simulation weights or replay data. The hardware implementation retains the central structure of the simulation formulation: accumulated joint-reference updates, command-state feedback, multiscale pose-error observations, and the reward terms of Section 4.3. The actor updates the reference by Equation 1 with $s = 0.03$ at a nominal 10 Hz.

Because each physical transition is costly, we train with FlashSAC (Kim et al., 2026), a soft actor-critic variant (Haarnoja et al., 2018) with a distributional critic (Bellemare et al., 2017), and hindsight

Table 1: Simulation results. Deployment rows report the delivered attempt, the first strict success or otherwise the last attempt, and $E[N]$ is the mean number of attempts.

Controller	Strict (%) ↑	Coarse (%) ↑	Median error		$E[N]$
			Pos. (mm) ↓	Ori. (°) ↓	
UMI-RL (adapted)	0.00 ± 0.00	0.04 ± 0.01	19.45 ± 3.80	74.69 ± 2.62	1.000
+ deployment	0.02 ± 0.01	0.05 ± 0.02	20.51 ± 4.90	79.35 ± 5.40	9.196
PolicyIK	93.49 ± 0.20	96.56 ± 0.16	$\mathbf{0.25 \pm 0.02}$	$\mathbf{0.04 \pm 0.00}$	1.000
+ deployment	$\mathbf{98.92 \pm 0.23}$	$\mathbf{99.43 \pm 0.09}$	$\mathbf{0.25 \pm 0.02}$	$\mathbf{0.04 \pm 0.00}$	1.212

experience replay (Andrychowicz et al., 2017), which relabels near-goal outcomes to add supervision for terminal regulation. A staged start-state curriculum widens the initial perturbation around each goal while keeping near-goal experience frequent. Details are in Appendix G.

5 Experiments

We evaluate simulated pose regulation, the contributions of the control formulation, and its use in simulated manipulation and physical execution.

5.1 Simulation Experiments

Setup and metrics. We evaluate a fixed-base 7-DoF OpenArm in Isaac Sim on 4,120 start–goal pairs from collision-free configurations disjoint from the training pools, with goals inside the deployment workspace. Each attempt lasts 160 policy steps (5.3 s). We report strict and coarse success (Section 3) and median terminal-window errors over all tasks, including failures, as mean ± standard deviation across three training seeds. Appendices A and C give the split and aggregation details.

Baseline. We adapt the full-pose tracking recipe of UMI on Legs (Ha et al., 2024) to our fixed-base arm (UMI-RL), retaining its reward, orientation curriculum, network, and PPO settings with a comparable training budget. Additional tuning of the discount horizon and orientation kernel is reported in Appendix D.

5.1.1 Main Results

Coverage and precision. In a single attempt, PolicyIK achieves 93.5% strict success and 96.6% coarse success, with median position and orientation errors of 0.25 mm and $0.04°$, respectively (Table 1). The deployment protocol further increases strict success to 98.9% through corrected-goal and transient-goal retries (Appendix C). By comparison, UMI-RL achieves only 0.04% coarse success, and none of its tuned variants meets the strict criterion.

Holding. PolicyIK stays within both strict thresholds at every sample of the final second on 92.4% of tasks, showing that its terminal accuracy extends beyond the mean-error criterion. On successful tasks, median net motion over that second is 14 μm and $0.002°$ (Appendix C).

Efficiency. PolicyIK is inexpensive to execute: a forward pass of its two-layer actor takes 0.035 ms at batch size one with PyTorch on one AMD EPYC 9454 CPU core, three orders of magnitude below the 33 ms control period, and the deployment protocol adds only 0.21 attempts per task on average.

5.1.2 Component Ablations

Command-state feedback and reference anchor. Terminal accuracy is more sensitive to the reference interface than coarse reaching (Figure 3). Removing command-state feedback reduces strict success by 9.2 percentage points. Re-anchoring at the measured joints, compared with the same 47-dimensional variant without feedback, retains 95.9% coarse success but reduces strict success from 84.3% to 30.8%. These comparisons support retaining the accumulated reference state.

Pose feedback and action scaling. Fine pose channels and error-dependent scaling support complementary aspects of terminal correction. Removing the fine channels increases median position

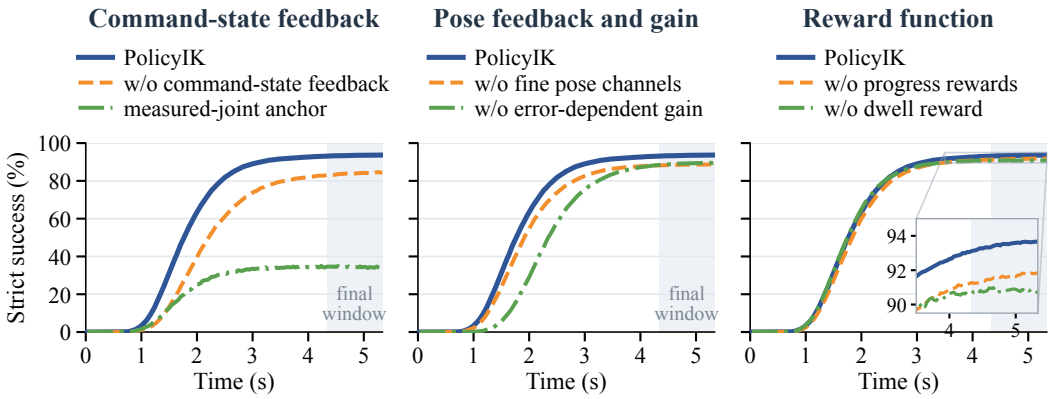

Figure 3: Ablations on command-state feedback, pose feedback and gain, and rewards.

error from 0.25 to 0.36 mm; replacing the error-dependent gain with a fixed gain increases median entry time from 1.73 to 2.28 s. Either change reduces strict success by 4–5 percentage points. Appendix E reports the combined removal and gain-selection analysis.

Reward function. The reward ablations distinguish entering the tolerance from final-window containment. Removing dwell leaves entry time similar but increases the share of entered tasks not contained in the final second from 2.4% to 7.8%. Removing the progress rewards reduces strict success by 1.8 percentage points. These results support rewarding both progress and sustained accuracy.

5.1.3 Pick-and-Place with a VLA Policy

Setup. We evaluate $\pi_{0.5}$-LIBERO (Physical Intelligence et al., 2025) on ten adapted LIBERO-Object tasks (Liu et al., 2023) with a side-mounted OpenArm in Isaac Sim. PolicyIK is trained for this mount without pick-and-place data. A fixed model of the Panda operational-space controller converts the VLA commands into pose references shared by all evaluated controllers. We compare PolicyIK with default DLS (Wampler, 1986), which clamps joint targets, and hand-tuned DLS and QP with joint-limit constraints, using 50 official initial states per task and three runs per state. Appendix F details the adapter and controller settings.

Results. PolicyIK achieves 95.7% success, compared with 96.7% for hand-tuned DLS and 90.8% for both default DLS and hand-tuned QP (Table 2). Differences are concentrated on targets near the mount, where grasping brings a shoulder joint to its limit (Figure 4; Appendix F). These results demonstrate the utility of a controller trained only for pose regulation in VLA-driven manipulation. Here, its low-level policy requires no Jacobian, online optimization, or gravity feedforward.

Table 2: Pick-and-place success over the ten LIBERO-Object tasks in simulation, each run three times from its 50 official initial states.

Low-level controller	Success (%)
DLS (default)	90.8
DLS (hand-tuned)	96.7
QP (hand-tuned)	90.8
PolicyIK	95.7



Figure 4: PolicyIK successfully grasping the butter at a shoulder joint limit (left) and placing it in the basket (right).

5.2 Real-Robot Experiments

Setup. We evaluate PolicyIK trained directly on a physical 7-DoF OpenArm, without sim-to-real transfer (Section 4.5). Joint-position references are tracked by PD control with gravity feedforward. Hardware pose errors are computed from joint encoders using forward kinematics in the modeled

gripper frame. The reported precision therefore measures regulation in this model-defined frame. We evaluate 60 full-pose tasks at 15 s and 25 s horizons. Holding success requires both errors to remain within 1 mm and 0.2° at every sample of the final 1 s or 5 s. Trials that fail to meet this criterion, including safety stops, count as failures. Appendices G and H specify the evaluation protocol, task-bank composition, and execution profiles.

Baselines. We compare four classical controllers that output joint-position references: DLS, resolved-rate QP, kinematic MPC, and adapted OpenArm QP. Their hardware-tuned settings and execution constraints are given in Appendix H.

Results. PolicyIK achieves comparable or higher holding success under the evaluated time budgets, with a larger advantage for longer holding windows (Table 3). With a 1 s hold, it matches the best classical controller at 25 s and exceeds all four at 15 s. Extending the hold to 5 s leaves 26 PolicyIK successes versus at most 17 for the classical controllers at 25 s, and 20 versus at most 6 at 15 s (Figure 5). Its 25 s median errors, 0.76 mm and 0.19°, fall within the classical controllers' range. The main difference is therefore sustained satisfaction of both tolerances. The classical controllers keep correcting the residual error inside the tolerance, so the pose leaves and re-enters it (Figure 6). PolicyIK is instead rewarded for staying within the tolerance, and a zero action retains the reference that brought it to the goal, in line with the simulation ablations (Section 5.1.2). Appendix G provides the 25 s holding curve (Figure 7) and additional temporal analysis.

Table 3: Tasks held (of 60) for the final 1 s or 5 s of each budget, and median errors at 25 s.

	15 s		25 s		Median	
Controller	1 s	5 s	1 s	5 s	mm	°
DLS	21	6	**29**	17	0.65	**0.13**
Resolved-rate QP	21	6	24	16	**0.57**	0.14
Kinematic MPC	10	3	26	15	0.81	0.18
Adapted OpenArm QP	8	1	16	11	0.99	0.23
PolicyIK	**25**	**20**	**29**	**26**	0.76	0.19

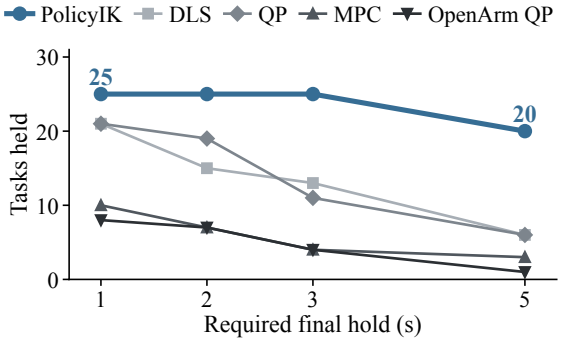

Figure 5: Tasks held within 15 s.

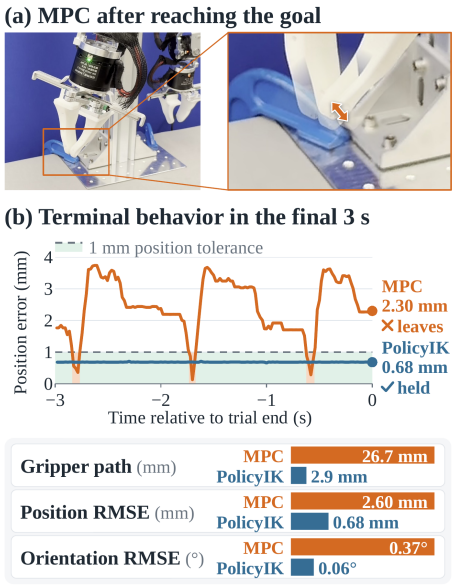

Figure 6: A case in which MPC jitters around the goal after reaching it, while PolicyIK holds the pose.

6 Conclusion

We presented PolicyIK, a goal-conditioned RL formulation for high-precision inverse kinematics through a joint-position interface. Controlled ablations show that command-state feedback, error resolution, and reward design affect broad reaching and terminal regulation differently, highlighting the importance of training for both reaching and retaining a target pose. In simulation, PolicyIK achieves 93.5% single-attempt success under a joint 1 mm and 0.2° terminal-window mean-error criterion, rising to 98.9% with a deployment protocol averaging 1.21 attempts. A separately trained hardware policy achieves comparable or higher holding success than the evaluated classical controllers, with larger advantages for longer holds. In simulated manipulation, it executes $\pi_{0.5}$ commands without pick-and-place training of the low-level policy. These results show that reinforcement learning can provide high-precision inverse kinematics for robot arms.

AI USE STATEMENT

In this work, we used generative AI tools for refining the text and figures of the paper and for checking the literature. We have not used generative AI tools to generate data sets, and formulating mathematical claims and writing proofs are not applicable to this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work consists of simulation experiments and experiments on a physical robot arm in a laboratory. It involves no human subjects and uses no sensitive or personal data.

REPRODUCIBILITY STATEMENT

Appendix A gives the controller, observation, and action settings and the data splits, Appendix B the PPO settings, checkpoint averaging, and reward parameters, and Appendix C the evaluation metrics and deployment protocol. Appendices D–H give the corresponding details for the simulation baseline, the ablations, the pick-and-place evaluation, and the hardware experiments. The supplementary material provides the simulation code of PolicyIK, covering the task, training, checkpoint averaging, evaluation, and the deployment protocol.

REFERENCES

Elie Aljalbout, Felix Frank, Maximilian Karl, and Patrick van der Smagt. On the role of the action space in robot manipulation learning and sim-to-real transfer, 2024. URL https://arxiv.org/abs/2312.03673.

Barrett Ames, Jeremy Morgan, and George Konidaris. Ikflow: Generating diverse inverse kinematics solutions, 2022. URL https://arxiv.org/html/2111.08933v3.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017.

Patrick Beeson and Barrett Ames. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 928–935, 2015. doi: 10.1109/HUMANOIDS.2015.7363472.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458, 2017.

Joel W. Burdick. On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 264–270, 1989. doi: 10.1109/ROBOT.1989.99999.

Samuel R. Buss. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. Technical report, University of California, San Diego, 2009.

Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005. doi: 10.1080/2151237X.2005.10129202.

Marie Charbonneau, Valerio Modugno, Francesco Nori, et al. Learning robust task priorities of QP-based whole-body torque-controllers. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2018. doi: 10.1109/HUMANOIDS.2018.8624995.

Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, et al. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. In *Robotics: Science and Systems (RSS)*, 2024.

Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. Rmpflow: A geometric framework for generation of multi-task motion policies, 2021. URL https://arxiv.org/abs/2007.14256.

Cheng Chi, Zhenjia Xu, Chuer Pan, et al. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Robotics: Science and Systems (RSS)*, 2024.

Arati S. Deo and Ian D. Walker. Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, 14(1):43–68, 1995. doi: 10.1007/BF01254007.

Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, 2010.

Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers, 2024. URL https://arxiv.org/abs/2407.10353.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870, 2018.

Jan-Gerrit Habekost, Connor Gäde, Philipp Allgeuer, and Stefan Wermter. Inverse kinematics for neuro-robotic grasping with humanoid embodied agents. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7315–7322, 2024. doi: 10.1109/IROS58592.2024.10802010.

Jesse Haviland and Peter Corke. Manipulator differential kinematics: Part 2: Acceleration and advanced applications. *IEEE Robotics and Automation Magazine*, 2023. doi: 10.1109/MRA.2023.3270221. URL https://arxiv.org/abs/2207.01794.

Zhengmao He, Moonkyu Jung, Hyeongjun Kim, Jiseong Lee, Hui Zhang, Jemin Hwangbo, and Jie Song. Pake: Learning whole-body loco-manipulation with partial kinematic embeddings, 2026. URL https://arxiv.org/html/2607.11041.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control, 2019. URL https://arxiv.org/abs/1812.03201.

Donghu Kim, Youngdo Lee, Minho Park, Kinam Kim, I Made Aswin Nahendra, Takuma Seno, Sehee Min, Daniel Palenicek, Florian Vogt, Danica Kragic, Jan Peters, Jaegul Choo, and Hojoon Lee. FlashSAC: Fast and stable off-policy reinforcement learning for high-dimensional robot control, 2026. URL https://arxiv.org/abs/2604.04539.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, pp. 2679–2713, 2024.

Visak Kumar, David Hoeller, Balakumar Sundaralingam, Jonathan Tremblay, and Stan Birchfield. Joint space control via deep reinforcement learning, 2021. URL https://arxiv.org/html/2011.06332v2.

Haeone Lee, Byeongguk Jeon, Suchae Jeong, et al. SPACE: Enabling learning from cross-robot data toward generalist policies, 2026.

Marion Lepert, Ria Doshi, and Jeannette Bohg. Shadow: Leveraging segmentation masks for cross-embodiment policy transfer. In *Conference on Robot Learning (CoRL)*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023.

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

Mayank Mittal et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 2023.

Jeremy Morgan, David Millard, and Gaurav S. Sukhatme. CppFlow: Generative inverse kinematics for efficient and robust Cartesian path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi: 10.1109/ICRA57147.2024.10611724.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pp. 278–287, 1999.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

Dmytro Pavlichenko and Sven Behnke. Real-robot deep reinforcement learning: Improving trajectory tracking of flexible-joint manipulator with reference correction, 2022. URL https://arxiv.org/html/2203.07051.

Physical Intelligence et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018.

Tifanny Portela, Andrei Cramariuc, Mayank Mittal, and Marco Hutter. Whole-body end-effector pose tracking, 2025. URL https://arxiv.org/html/2409.16048v2.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Bruno Siciliano. A closed-loop inverse kinematic scheme for on-line joint-based robot control. *Robotica*, 8(3):231–243, 1990. doi: 10.1017/S0263574700000096.

Bingjie Tang, Michael A. Lin, Iretiayo Akinola, Ankur Handa, Gaurav S. Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality, 2023. URL https://arxiv.org/html/2305.17110v1.

Charles W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1): 93–101, 1986.

Daniel Weber, Maximilian Schenke, and Oliver Wallscheid. Steady-state error compensation in reference tracking and disturbance rejection problems for reinforcement learning-based control, 2022. URL https://arxiv.org/abs/2201.13331.

Mandy Xie, Karl Van Wyk, Anqi Li, Muhammad Asif Rana, Qian Wan, Dieter Fox, Byron Boots, and Nathan Ratliff. Geometric fabrics for the acceleration-based design of robotic motion, 2020. URL https://arxiv.org/abs/2010.14750.

J. Yang, S. Yan, F. Feng, et al. MimicIK: Real-time generative inverse kinematics from teleoperation with FK consistency, 2026.

Jonathan Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training one policy across robots while embracing variability. In *Conference on Robot Learning (CoRL)*, 2023.

Shusheng Yu and Gongquan Tan. Inverse kinematics of a 7-degree-of-freedom robotic arm based on deep reinforcement learning and damped least squares. *IEEE Access*, 13:4857–4868, 2025. doi: 10.1109/ACCESS.2024.3521539.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, et al. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A Simulation Setup

Controller, observation, and action. Physics runs at 60 Hz. Between policy steps, an implicit PD controller with stiffness 80 and damping 4 on every arm joint tracks the reference under gravity and finite effort limits, without feedforward compensation, and the solver uses 32 position iterations. The reference starts at the measured joint positions. The action scale is $s = 0.0625$. In Equation 5, $g_{\min} = 0.1$, $(l_p, u_p) = (0.5, 9.31)$ mm, and $(l_R, u_R) = (0.415, 2.328)^\circ$, with angular computations in radians.

In Equation 3, the observation uses the offsets $c_i = 0.3\, e_i$ m along the end-effector axes e_i, $i = 1, 2, 3$. The observation groups in Equation 2 have widths $7 + 7 + 3 + 9 + 6 + 6 + 2 + 7 + 7 = 54$. During training, uniform noise in $[-0.01, 0.01]$ perturbs the joint-position and joint-velocity channels; evaluation uses noise-free observations.

Holding offset. For an unsaturated PD controller without feedforward, static balance gives

$$K_p(z^* - q^*) = \tau_g(q^*), \qquad z^* = q^* + K_p^{-1}\tau_g(q^*). \tag{9}$$

An accumulated reference retains this offset with zero action increments.

Data splits. The source set contains 200,000 joint configurations sampled uniformly within the joint limits, together with their forward-kinematics poses. After floor-contact and self-collision filtering and the removal of all evaluation rows, training starts are sampled uniformly over 157,504 rows and goals over the 104,599 rows whose end effector lies in the deployment workspace. This workspace is a cylinder about the base axis with a radius of 0.49 m and heights from 0.15 to 0.50 m, excluding a keep-out region around the base links within 0.128 m of the axis and below 0.222 m. Evaluation uses 4,120 fixed start–goal pairs sampled without replacement under the same rules. Training and evaluation share no configuration, and no evaluation goal lies within both 1 mm and 0.2° of a training goal.

B Reward and Training

PPO and checkpoint averaging. We report training progress in rl_games epochs. One epoch is one PPO iteration: it collects 24 steps from each of 4,096 parallel environments (98,304 transitions) and then makes 5 passes over this batch in minibatches of 24,576. Each of three training seeds runs 13,000 epochs, approximately 1.28×10^9 transitions. The actor and critic share a two-layer 256-unit ELU network, the Gaussian policy has a learned state-independent log standard deviation, and the critic receives the policy observation. PPO uses $\gamma = 0.99$, GAE $\lambda = 0.95$, clip ratio 0.2 with value clipping, entropy coefficient 0.01, value-loss coefficient 2, bounds-loss coefficient 10^{-4}, gradient-norm clipping at 1.0, and an adaptive learning rate initialized at 10^{-3} with KL target 0.01. Inputs, values, and advantages are normalized, and value targets bootstrap at time-outs. The deployed policy for each seed is an exponentially weighted average of the last 20 saved checkpoints (epochs 11,100 to 13,000 in steps of 100), with weight proportional to $0.5^{\ell/400}$ for the checkpoint ℓ epochs before the last.

Error history. The reward at state t corresponds to action a_{t-1} and uses the previous best b_{t-1} before the update $b_t = \min(b_{t-1}, \rho_t)$. The observation exposes the same quantities through

$$h_t = \left[\mathrm{clip}(b_{t-1}/20, 0, 1),\ \mathrm{clip}((\rho_t - b_{t-1})/20, 0, 1)\right]. \tag{10}$$

On each new goal, the running best is initialized to the current ρ, and the improvement, backslide, potential-shaping, and dwell terms are zeroed for that step.

Reward parameters. Equation 6 groups the eleven implemented terms. The reward keypoints use centered offsets $c_i = \kappa(e_i - \frac{1}{3}\mathbf{1})$ with $\kappa = 0.45$ m, and ε_t denotes their mean distance between the current and goal poses. With $K(x; \sigma) = 1 - \tanh(x/\sigma)$, the full tracking reward is

$$r_t^{\mathrm{track}} = -w_p e_p(t) + w_f K(e_p(t); \sigma_p) + w_w K(\varepsilon_t; \sigma_w) + w_s K(\varepsilon_t; \sigma_s). \tag{11}$$

Table 4: Strict success as checkpoint averaging and the two deployment steps are added. $E[N]$ is the mean number of attempts per task.

Setting	Strict (%) ↑	$E[N]$
Final checkpoint, single attempt	92.30 ± 0.13	1.000
Checkpoint average, single attempt	93.49 ± 0.20	1.000
+ error-corrected goal	94.80 ± 0.12	1.036
+ transient-goal attempts	$\mathbf{98.92 \pm 0.23}$	1.212

The regularization costs are

$$
\begin{aligned}
c_t^{\text{motion}} &= w_a(t)\|a_{t-1} - a_{t-2}\|_2^2 + w_v(t)\|\dot{q}_t\|_2^2, \\
c_t^{\text{limit}} &= w_m \,\text{clip}\left(1 - \frac{\min_j m_j(q_t)}{0.05}, 0, 1\right)\mathbf{1}[\rho_t > 10], \\
m_j(q) &= \frac{\min(q_j - q_j^{\text{lo}}, q_j^{\text{hi}} - q_j)}{q_j^{\text{hi}} - q_j^{\text{lo}}}.
\end{aligned}
\tag{12}
$$

The motion-cost weights follow the Isaac Lab reward-weight schedule: $(w_a, w_v) = (5 \times 10^{-4}, 5 \times 10^{-4})$ until the environment step counter exceeds 4,500 policy steps, and $(0.025, 0.005)$ afterwards. The counter restarts in each of the three training segments (epochs 0 to 3,500, 3,500 to 8,500, and 8,500 to 13,000), so the larger weights apply for about 95% of training. The configured weights are $(w_p, w_f, w_w, w_s) = (0.2, 0.1, 1, 1)$, $(w_n, w_\Phi, w_b, w_d) = (26.831, 1, 0.2, 1)$, and $w_m = 1$. The distance bandwidths are $(\sigma_p, \sigma_w, \sigma_s) = (0.1, 0.2, 0.05)$ m, and the shaping term uses the PPO discount $\gamma = 0.99$. The improvement bandwidth is $\beta = 0.5$. The reward manager multiplies every term by the policy period of 1/30 s, which scales all weights equally.

C Metrics and Deployment Protocol

Metrics. Beyond the success criteria of Section 3, three measures describe the terminal phase. A task *enters* the strict region when one sample meets both strict thresholds, and its entry time is the time of the first such sample. A task is *contained* when every sample of the final window meets both thresholds. Among the tasks that enter, we report the share that is not contained. The final-second net motion is the distance between the end-effector positions at the start and end of the final second, together with the rotation angle between the corresponding orientations.

Median errors are taken over the terminal-window means of all tasks, including failures. Each statistic is computed separately for each training seed and summarized by its mean and sample standard deviation across the three seeds.

Deployment protocol. Each attempt is a 160-step episode from the task's start configuration. When a failed first attempt ends with a terminal-window position error within 1 cm, the error-corrected goal shifts the requested position by the negative final-step position error of that attempt and rotates the requested orientation by the inverse of its final-step rotation error. A transient goal displaces the requested position by 0.30 m along a random unit direction and rotates the requested orientation by 90° about a random unit axis. It is commanded for the first 40 steps, and the original goal for the remaining 120 steps. Each of the eight transient-goal attempts draws its directions and axes per task from its own fixed seed.

Table 4 separates the contributions of checkpoint averaging and the two protocol steps. Per seed, a single attempt leaves on average 127 tasks that meet only the coarse criterion and 142 that miss it. The error-corrected goal recovers 54 tasks, nearly all from the first group, and transient goals recover 170 more, including 117 from the second. The protocol thus acts on the tail of the error distribution: it lowers the mean position error by a factor of five, while the median stays at 0.25 mm. The 45 tasks per seed that still fail have goal configurations closer to joint limits: the worst joint of the goal configuration lies a median of 1.3% of its range from a limit, against 5.7% for tasks solved in the first attempt.

D SIMULATION BASELINE

UMI-RL converts the per-step constants of the source recipe from its 50 Hz policy rate to our 30 Hz, so that the discount horizon and rollout span stay fixed in seconds. It trains for 20,000 epochs of 14 steps in 4,096 environments, 1.15×10^9 transitions against 1.28×10^9 for PolicyIK.

Tuning. Table 5 lists the tuned variants. A longer discount horizon of 0.54 s lowers both median errors of the source recipe. Its orientation curriculum stays on the coarsest rung on this task, which promotes at a mean orientation error below 1.0 rad ($57.3°$), while our bank has a median orientation demand of $116.4°$. Scaling the whole published ladder by k sharpens the orientation kernel: at $k = 0.25$, orientation improves by an order of magnitude and the curriculum advances by one rung, while position error grows. Strict success stays at zero in every setting.

Table 5: Tuned variants of UMI-RL with three seeds each. The factor k scales the published orientation ladder, and strict success is zero in every setting.

		Median error ↓	
Setting	Coarse (%) ↑	Pos. (mm)	Ori. (°)
0.20 s horizon, $k = 1$ (source)	0.04 ± 0.01	19.45 ± 3.80	74.69 ± 2.62
0.54 s horizon, $k = 1$	0.02 ± 0.02	6.27 ± 0.43	54.87 ± 3.85
0.54 s horizon, $k = 0.5$	0.35 ± 0.09	11.17 ± 2.27	37.45 ± 0.61
0.54 s horizon, $k = 0.25$	16.38 ± 3.10	16.65 ± 0.80	4.25 ± 0.77

Deployment protocol. The deployment protocol converts near misses into successes, so it acts on controllers that already approach the tolerance region. Applied in the same order to the adapted recipe at the source horizon, with a corrected-goal bank generated from its own terminal errors, it raises strict success only to 0.02% after 9.196 attempts per task on average (Table 1). Only 19.7% of its tasks end within 1 cm, the gate for the error-corrected goal, and the eight transient-goal attempts add no strict success.

E ABLATION DETAILS

Table 6 lists the full results of the component ablations in Section 5.1.2. Removing command-state feedback deletes its seven channels from the 54-dimensional observation. The measured-joint anchor uses the same 47 dimensions and replaces the anchor z_{t-1} in Equation 1 with q_t, so the comparison between the two isolates the anchor. Removing the fine pose channels leaves 48 dimensions. Removing channels shrinks the input layer, while hidden widths and training budgets stay fixed. Removing the progress group disables the improvement, potential-shaping, and backslide rewards but keeps the running-best observation, and removing dwell sets only its reward weight to zero.

Fixed-gain selection. Fixed gains are selected on 2,084 development tasks with one seed per candidate. With fine channels, gains of 0.1, 0.25, 0.5, and 1.0 reach 87.24%, 85.70%, 82.63%, and 82.77% strict success. Without them, the same gains reach 0.00%, 79.41%, 76.20%, and 74.90%. Each fixed-gain variant therefore uses its best gain, 0.1 with fine channels and 0.25 without. At $g = 0.1$, the variant without fine channels never enters the strict region, and its median errors are 11.57 mm and $5.55°$.

F PICK-AND-PLACE EVALUATION DETAILS

Tasks and protocol. Each LIBERO-Object task asks the robot to pick one grocery item and place it in a basket. We use each task's 50 official initial states and language instruction, the agent-view and wrist cameras at LIBERO's poses, a budget of 280 actions, and the evaluation settings of $\pi_{0.5}$-LIBERO: action chunks are re-planned every five actions and images are resized to 224×224. Objects start from the official states after LIBERO's settling interval. An episode succeeds the first time LIBERO's success predicate holds at an action boundary, and it fails if the fingers pass into the

Table 6: Component ablations with matched training budgets and checkpoint averaging, one attempt on each of 4,120 tasks. Entry is the median time of the first entry into the strict region among the tasks that enter it, and Uncont. is the share of these tasks that are outside the strict region at some sample of the final window. The measured-joint anchor re-anchors the reference at the measured joints, $z_t = q_t + s\, g\, a_t$, without command-state feedback, and the variant without the error-dependent gain uses the fixed gain $g = 0.1$. Values are mean $\pm$ standard deviation over three training seeds.

	Success (%) ↑		Terminal phase ↓		Median error ↓	
Variant	Coarse	Strict	Entry (s)	Uncont. (%)	Pos. (mm)	Ori. (°)
PolicyIK	96.56±0.16	93.49±0.20	1.73±0.01	2.43±0.23	0.25±0.02	0.04±0.00
Command-state feedback and reference anchor						
w/o command-state feedback	92.63±0.20	84.26±0.28	2.04±0.02	10.32±0.32	0.30±0.00	0.06±0.00
measured-joint anchor	95.90±0.17	30.77±7.94	1.80±0.03	82.46±6.62	1.27±0.05	0.16±0.01
Pose feedback and action scaling						
w/o fine pose channels	95.39±0.25	88.41±0.34	1.83±0.03	4.38±0.11	0.36±0.01	0.06±0.00
w/o error-dependent gain	94.89±0.14	89.13±1.64	2.28±0.07	5.69±2.62	0.26±0.03	0.05±0.01
w/o both ($g = 0.25$)	93.01±0.59	85.27±0.39	2.11±0.02	4.85±0.33	0.34±0.00	0.06±0.00
Reward function						
w/o progress rewards	96.47±0.49	91.66±0.71	1.79±0.05	3.41±0.65	0.26±0.00	0.04±0.00
w/o dwell reward	96.87±0.29	91.54±1.33	1.69±0.02	7.83±1.18	0.34±0.00	0.07±0.00

table. Each controller runs three times per initial state. $\pi_{0.5}$ samples its own noise in every run, and the three runs of PolicyIK use its three training seeds.

Pose reference. $\pi_{0.5}$ outputs relative end-effector commands for robosuite's operational-space controller (Zhu et al., 2020) at 20 Hz. A model of that controller with robosuite's gains (stiffness 150, critical damping) integrates them into an end-effector pose that is re-anchored at every action, and the low-level controller tracks this pose at 30 Hz. No parameter of the model is fitted. The commanded tool-center point is kept at least 5 mm above the table, and the gripper opens or closes with the sign of the gripper command.

Robot and scene. One planar translation and yaw of the robot and its table, chosen from source-robot trajectories, places the objects within reach, and the arm starts at the Panda's initial end-effector pose. The cameras render the LIBERO scene appearance, object collision geometry, mass and inertia follow the LIBERO assets, and the physics runs at 60 Hz.

Controllers. PolicyIK is trained from random initialization with the recipe of Section 4 for the side-mounted arm, whose base sees gravity along a different axis than in Section 5.1, with three seeds and checkpoint averaging. Its goals are sampled from reachable configurations of the side-mounted arm, and no pick-and-place data or VLA is used in training. The default DLS is the differential inverse-kinematics controller of Isaac Lab (Mittal et al., 2023) with damping 0.01, whose joint targets are clamped at the soft limits. The hand-tuned DLS and QP use the hardware settings of Appendix H (task weighting W, damping 0.02 and 0.005, joint limits as bounds of a bounded least-squares step) and take one full step per control cycle without the hardware speed and command-lead bounds. All three classical controllers add gravity-compensation torques and share the arm's joint drives.

Statistics. The difference between PolicyIK and the hand-tuned DLS is -0.9 percentage points, with a 95% bootstrap interval of $[-2.3, 0.4]$ over the 500 initial states.

Behavior near joint limits. In the evaluated trajectories, the default DLS discards the bound-violating part of its step and stops short of the commanded pose, while the lightly damped QP can stall at a joint limit during lifting. PolicyIK completes these grasps through a reference clipped at the soft limits.

Results by target location. Table 7 groups the ten tasks by where their targets lie. The targets of alphabet soup, butter, chocolate pudding, ketchup, and milk lie closest to the mount, 0.28 m in front of the shoulder and 0.28–0.34 m below it, where the grasp holds a shoulder joint at its limit. The other five lie a further 0.21 m to the side.

Table 7: Pick-and-place success (%) by target location, three runs from each of the 50 official initial states per task (750 episodes per group).

Targets	DLS (default)	DLS (hand-tuned)	QP (hand-tuned)	PolicyIK
Closest to the mount	85.5	96.8	85.9	95.1
Farther to the side	96.1	96.5	95.7	96.4

G Physical Policy Training and Evaluation Details

Relationship to the simulation formulation. The hardware policy keeps the accumulated joint reference, the command-state and multiscale pose-error observations, and the reward terms of Section 4.3 with the same tracking, progress, and dwell weights. It departs from the simulation recipe where physical training requires it: an off-policy learner with hindsight relabeling replaces PPO because each transition is costly, the actor updates the reference at 10 Hz with $s = 0.03$, the motion costs and the potential cap are retuned on the arm, and a start-state curriculum replaces uniform task sampling.

Hardware observation. The hardware policy uses a 72-dimensional observation containing the measured joint state, end-effector and goal poses, command-state feedback $z_{t-1} - q_t$, the previous action, keypoint offsets, coarse and fine pose-error channels, and running-best error features. The command-state and multiscale pose-error channels retain the structure of the simulation formulation.

Learning algorithm. We train the hardware policy using FlashSAC (Kim et al., 2026), a soft actor-critic variant with a distributional critic, together with hindsight experience replay (HER). The actor has two hidden layers of 128 units, and each critic has two hidden layers of 256 units. The distributional critic uses 201 return bins. Training uses a replay capacity of 32,000 transitions, a batch size of 1,024, an update-to-data ratio of three, discount factor $\gamma = 0.98$, and target update coefficient $\tau = 0.005$. Gradient updates begin after 4,096 physical transitions have entered the replay buffer.

HER relabels collected transitions with achieved goals, including near-goal outcomes that provide additional training signal for terminal regulation.

Training curriculum. Training uses a staged start-state curriculum around sampled goal configurations. The first stage collects 40,000 physical transitions with per-joint Gaussian start perturbations whose standard deviation is at most 10°. The second stage raises this bound to 20° and continues to a total budget of 100,000 physical transitions. This progressively increases the required motion range while retaining frequent near-goal experience for terminal regulation. The training distribution is generated independently from the frozen physical evaluation bank.

Hardware command and safety bounds. The learned policy uses the same joint-position, reference-speed, and command-to-measurement lead bounds as the hardware baselines in Appendix H. Measured-state guards stop execution on stale feedback, excessive tracking error, motor faults, or excessive joint speed.

Learned-policy evaluation protocol. Frozen hardware-policy checkpoints are evaluated on the same 60-task full-pose bank used for the classical-controller campaign, with 20 medium-, 20 long-, and 20 extended-displacement tasks. The requested joint-space starts and world-frame goals therefore match across controller families. Evaluation tasks are excluded from the training replay buffer.

Each trial starts from the specified joint configuration and uses the same modeled gripper-base frame and encoder-FK scoring convention as the classical-controller campaign.

A goal counts as reached when the position and orientation errors remain within 1 mm and 0.2° continuously for at least 0.5 s. Holding success (Section 5.2) additionally requires both errors to stay within these thresholds throughout the final 1 s or 5 s of the trial. Figure 7 repeats the hold-window comparison of Figure 5 for the 25 s budget.

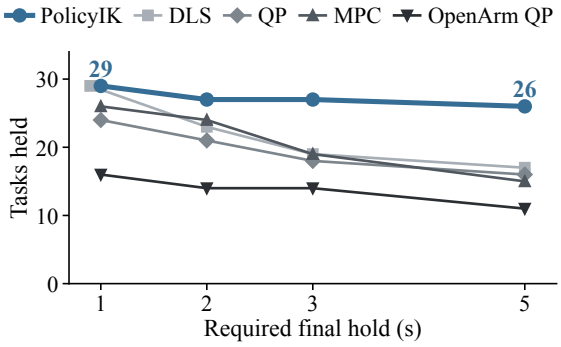

Figure 7: Tasks held within 1 mm and 0.2° in 25 s trials as the required final hold grows from 1 s to 5 s.

Additional holding analysis. At the 25 s horizon, extending the final hold from 1 s to 5 s retains 26 of 29 PolicyIK successes (90%), compared with 58–69% for the classical controllers. At 15 s, the corresponding proportions are 80% and 13–30%.

H Hardware Baseline Implementations

We compare four Cartesian controllers on the physical OpenArm: damped least-squares differential inverse kinematics (DLS), bounded resolved-rate quadratic programming (QP), locally linearized kinematic model-predictive control (MPC), and the QP controller distributed with the OpenArm control library (adapted OpenArm QP). All four controllers operate through the same supervised execution stack. They receive the measured seven-dimensional joint state, the desired world-frame end-effector pose, and the previous accepted joint reference, and return the next joint-position reference. None uses an external pose measurement or inverse dynamics in its Cartesian optimization. The shared motor layer tracks accepted joint references using PD control with model-based gravity feedforward.

Shared pose, command, and safety interface. Let q denote the measured joint positions, r the previous accepted joint reference, $e = [e_p^\top, e_R^\top]^\top$ the world-frame position and rotation-vector error, and $J(q) \in \mathbb{R}^{6\times 7}$ the geometric Jacobian. For DLS, resolved-rate QP, and kinematic MPC, forward kinematics and J are evaluated from the same MuJoCo model at the measured q. Translation and rotation are balanced using

$$W = \mathrm{diag}(1, 1, 1, 0.2, 0.2, 0.2). \tag{13}$$

The adapted OpenArm QP uses its separately pinned OpenArm model internally, while the common model is retained for evaluation. The evaluated end-effector is a gripper-base frame translated by 102.5 mm along the source hand-TCP z axis with no rotational offset; this transform is used consistently for goal generation, control, and scoring.

Accepted commands are constrained by the intersection of the robot-model and physical-rig joint limits, a per-joint reference-speed bound of 0.20 rad/s, and a command-to-measurement lead bound. The final baseline profile maintains a 5 mrad inward command margin from the hard joint limits.

Damped least-squares controller. The DLS controller first computes

$$\dot{q}_{\mathrm{dls}} = (WJ)^\top \left[(WJ)(WJ)^\top + \lambda^2 I\right]^{-1} kWe, \qquad k = 2, \quad \lambda = 0.02. \tag{14}$$

It uniformly rescales this velocity when its infinity norm exceeds the joint- speed bound and integrates from r. When that candidate would violate a joint-position, per-cycle speed, or reference-lead

Table 8: Classical hardware baselines. All four map the measured joint state and the goal pose to joint-position references at 50 Hz within the joint limits and the 0.20 rad/s reference-speed bound. DLS, QP, and MPC weight rotation errors by 0.2 relative to translation.

Controller	Formulation	Parameters
DLS	Damped least squares with global speed scaling	$k = 2$, $\lambda = 0.02$
Resolved-rate QP	Damped least squares within the joint bounds	$k = 3$, $\lambda = 0.005$
Kinematic MPC	Seven-step receding horizon (0.14 s)	$k = 3$, $\lambda = 0.005$
Adapted OpenArm QP	OpenArm QP with measured-pose settling	Costs 12/1.5, damping 0.1

bound, the implementation does not clip joints independently. Instead, it resolves the same globally scaled damped task as a bounded least-squares problem over the feasible joint- velocity box.

Bounded resolved-rate QP. The resolved-rate controller directly solves

$$\min_{\dot{q}} \ \frac{1}{2}\|WJ\dot{q} - kWe\|_2^2 + \frac{\lambda^2}{2}\|\dot{q}\|_2^2, \qquad k = 3, \quad \lambda = 0.005, \tag{15}$$

subject to the joint-position, per-cycle speed, and command-to-measurement lead bounds before setting $r^+ = r + \Delta t\dot{q}$. We solve the dense bounded least-squares form with SciPy's BVLS implementation (exact least-squares subsolver, tolerance 10^{-10}, at most 100 iterations). DLS and QP therefore share the same local task family but differ under saturation: DLS first applies a single global speed scale and only then redistributes a bound-violating step, whereas QP optimizes the unscaled task directly inside the feasible box.

Kinematic MPC. The MPC controller replans at every feedback cycle and applies only the first control of a seven-step horizon (0.14 s at the nominal rate). It freezes the measured-state Jacobian within each solve and optimizes a sequence of joint- reference velocities using the same $k = 3$, $\lambda = 0.005$, and W as QP. The objective combines predicted resolved-rate residuals, velocity damping, a 0.001 velocity-smoothness weight, and a unit terminal pose-error weight. Constraints cover reference and predicted joint positions, the 0.20 rad/s speed bound, a 0.052 rad reference-lead bound, a 0.10 rad trust region, and 0.5 rad/s^2 reference acceleration. A piecewise-linear braking envelope reserves stopping distance near position and lead limits. The convex QP is solved with DAQP 0.7.2 using primal and dual tolerances of 10^{-11} and 10^{-12} and an iteration limit of 20,000.

Adapted OpenArm QP. The OpenArm baseline uses `openarm_control` 0.4.0 with its DAQP backend. At 50 Hz, the controller performs five internal iterations with an explicit right-arm velocity limit of 0.20 rad/s. The configured task parameters use position and orientation costs of 12 and 1.5, damping 0.1, Levenberg–Marquardt damping 0.01, joint and singularity braking, and zero posture and null-space costs.

Our adapter supplies measured joint positions to the library's state-dependent constraints, synchronizes the internal task state to the previous accepted reference, and transforms the world-frame goal into the library representation. The returned candidate is then projected through the common hard and operational position bounds, reference-speed bound, and 0.05 rad command-to-measurement lead bound. No unconstrained retry or fallback controller is used.

The evaluated implementation additionally includes a bounded measured-pose settling loop. Once the commanded pose remains within 0.1 mm/0.001 rad and the measured pose lies within 10 mm/0.05 rad for 0.2 s, the loop adjusts the internal pose target using the measured residual error. Translational and rotational correction rates are limited to 1 mm/s and 0.01 rad/s, with target-bias limits of 10 mm and 0.05 rad, respectively. This modification compensates residual command–measurement bias while retaining the upstream reaching controller and shared command governor. We therefore refer to the evaluated controller as the *adapted OpenArm QP*, rather than an unmodified reproduction of the upstream controller.

Parameter selection and development exposure. Controller parameters and implementation profiles were selected during hardware development rather than under a fixed tuning budget. DLS development tested four gain/damping/speed profiles on a 30-task bank; the final implementation retains

$k = 2$ and $\lambda = 0.02$ while adopting the common 0.20 rad/s speed limit and bounded redistribution at joint limits. Resolved-rate QP retains $k = 3$ and $\lambda = 0.005$ across three documented implementation revisions. MPC passed through five development profiles, primarily varying the prediction horizon and reference-lead handling, before selecting the seven-step, 0.052 rad lead profile. The adapted OpenArm QP passed through four adapter revisions before fixing the `openarm_control` 0.4.0 implementation and measured-pose settling mode described above.

Original hardware evaluation protocol. The frozen bank contains 60 full-pose start–goal tasks, with 20 tasks in each of three translational-displacement bands: medium (> 2 to ≤ 5 cm), long (> 5 to ≤ 10 cm), and extended (> 10 cm). The realized displacement ranges are 2.13–4.81 cm, 5.25–9.71 cm, and 10.79–32.93 cm, respectively.

The original classical-controller campaign uses the modeled gripper-base frame, a nominal 50 Hz controller rate, and a 40 s end-effector regulation budget. The synchronized transition to the specified start configuration and the return motion occur outside this budget. Start and return motions are limited to 0.10 rad/s and 0.20 rad/s^2.

The motor profile used during the main campaign applies temporary K_p/K_d settings of $100/2.75$ for J1, $110/2.5$ for J2, and $90/2.0$ for J4, with gravity-feedforward gain one.

The primary comparison in Section 5.2 rescores the archived trajectories using the 15 s and 25 s horizons and the holding criterion.