The Missing Integrator: Executing Frozen UMI Policies on Imperfect Arms

Yutong Wang[1*] **Narek Harutyunyan**[1,2*] **Sebastian Scherer**[1] **Guanya Shi**[1]
[1]Carnegie Mellon University [2]UC Berkeley
[*]Equal contribution

Abstract: Policies trained on UMI data assume that the deploying arm executes their commands faithfully, but low-cost and custom arms sag, lag and have deadband. On a low-cost 6-DoF arm and its simulated twin, changing only the controller drops simulated success from 53% to none and real button presses from 9 of 10 to none, yet the size of the tracking error does not determine success: in simulation, simple command-side fixes cut it without consistently raising success, while changing the controller can. Because UMI data record a single pose stream, the policy acts like a controller without integral action, and the execution anchor, on which each new chunk is composed, decides how much integral action execution adds. The measured anchor adds none and re-injects a steady error at every replan, while the commanded anchor adds full integral action and winds up when the arm cannot follow; both failure modes appear on the real arm. In simulation the best amount depends on the controller and the task: the measured anchor does best when the controller already removes steady error, and a blend toward the command when a steady offset remains.

1 Introduction

UMI data decouple demonstrations from any particular robot and have become one of the most scalable sources of manipulation data, with policies deployed on arms, hands, and legged, mobile, humanoid and aerial platforms [1, 2, 3, 4, 5, 6, 7, 8]. Unlike teleoperated data, which reflect the dynamics of the robot that collected them, UMI data are easier to scale but record only the gripper's motion, without commands or actuator dynamics [2, 9]. Deploying policies trained on them therefore assumes that the arm can execute this motion faithfully [1, 8]. The same pursuit of scale pushes deployment toward low-cost and custom-built arms, such as ALOHA [10, 11], whose hardware and controllers limit how precisely a policy's commands are executed [12].

Low-cost and custom arms are imperfect executors in systematic ways. The dynamics models supplied with such arms can be substantially inaccurate [5], and friction, backlash and gravity-compensation error persist as sources of discrepancy between commanded and executed motion [9, 12]. Feedback gains are typically selected heuristically and seldom reported [13]; for teleoperated data they are frequently retained from data collection [14], whereas UMI data come with no controller, leaving the choice entirely to the practitioner who deploys the policy. The resulting error is structured, appearing as sag, deadband and lag, and UMI-style deployments report its effects, from overshoot near contact to lagging motion and drift from the plan [15, 6, 7]. Practitioners patch these errors ad hoc by slowing down, blending or averaging chunks [1, 5, 10, 6], rarely comparing the patches on one arm and policy.

Existing fixes differ in what they require and what they give up (Fig. 1). **Retraining the policy (L1)** [16, 17, 18, 12] needs data from each arm and gives up policy reuse. **Model-free execution shaping (L2)**, such as the execution anchor, chunk blending, the replan interval or a time lead [1, 19, 10, 20, 15], can only change when and from which pose each chunk is executed. **Model-based correction outside the controller (L3)**, by feeding back tracking error or inverting the arm's response [21, 22, 23], relies on a model of the arm's response, fitted offline or adapted online, which for fixed error feedback is implicit in a tuned gain. **Changing the controller (L4)** acts on the source of the error [14, 24] but requires controller access and per-arm effort, substantial on legged, aerial

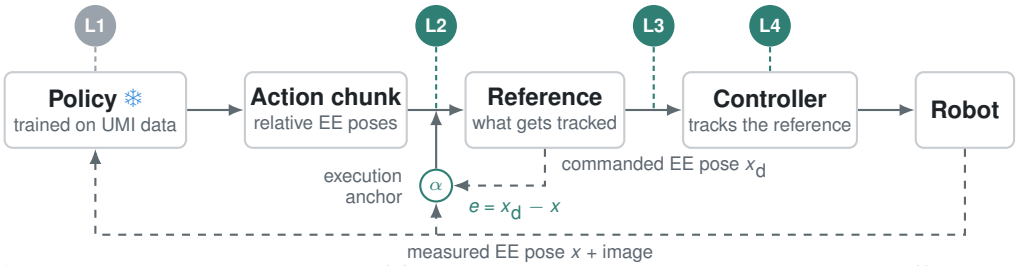


Figure 1: L1–L4: where each class of fix acts. x_d, x: commanded and measured end-effector pose; $e = x_d - x$: tracking error; chunks start at the execution anchor $x_d - \alpha e$ (α=0 commanded, 1 measured).

and humanoid platforms [5, 8, 7, 25]. We keep the policy frozen, since policy reuse is the main appeal of UMI data, and test fixes in L2–L4.

Our findings are threefold. First, the size of the tracking error lowers success but does not determine it: at similar error, success depends on the error's type and direction and on the task. Second, in simulation the simple command-side fixes we test cut open-loop tracking error but do not consistently raise success, whereas changing the controller can. Third, UMI policies leave integral action to execution: since UMI data record a single pose stream, the policy never sees the gap between command and arm, and the execution anchor decides how much integral action execution adds. The measured anchor that UMI uses [1] adds none and re-injects a steady error, while the commanded anchor, a policy-level action integrator [26], adds full integral action and winds up; we observe both on the real arm and in simulation, where the best choice depends on the controller and the task.

2 Methods and Results

2.1 Setup

We deploy Diffusion Policies [27], trained on UMI-style data with no arm or controller, on a low-cost 6-DoF arm and its simulated twin fitted to the real arm's gravity and friction (App. A–B). The real-world arm runs a ladder of three controllers of increasing accuracy. The most accurate, the full-model PD, adds feedforward torques from a full dynamics model of the arm. The simulation counterpart adds three flawed variants with sag, deadband, or lag, mirroring real controllers' dominant errors (Table 1). We characterize each controller by the open-loop tracking error on reference motions, and we evaluate task performance over four simulated tasks, Push Slider, Cube Stack, Mug Hook and Peg-in-Hole (App. B), and over a real-world button press (App. A). Simulation lets us add one error type at a time to an accurate arm and compare against an ideal arm on identical scenes; the real arm, whose errors come bundled, tests whether the mechanisms hold on hardware.

Table 1: Controller variants tested, with each flaw's change from the full-model PD in bold. Refit: factors refit to the real arm; nominal: unit factors; RMS: open-loop tracking error (mm).

Controller	Gains	Gravity	Friction	Inertia, Coriolis	Integrator	Ref. filter	RMS sim	RMS real
Factory PD	factory	factory					42.5	44.9
Hand-tuned PD	hand-tuned	nominal	✓				7.1	10.1
Full-model PD	hand-tuned	refit	✓	✓	✓	60 rad/s	4.1	4.5
Sag flaw	hand-tuned	**0.87 × refit**	✓	✓	**off**	60 rad/s	9.5	–
Deadband flaw	hand-tuned	refit	**off**	✓	**off**	60 rad/s	9.4	–
Lag flaw	hand-tuned	refit	✓	✓	✓	**7 rad/s**	8.5	–

2.2 Tracking error and success

Error size lowers success but does not determine it. Success falls as open-loop error grows, as Fig. 2a shows, but controllers with similar error succeed at very different rates: the three flaws of Table 1 track within 1 mm of one another, yet succeed between 24% and 48% of the time. Even the best controller stays short of an ideal arm that executes every command exactly (App. B).

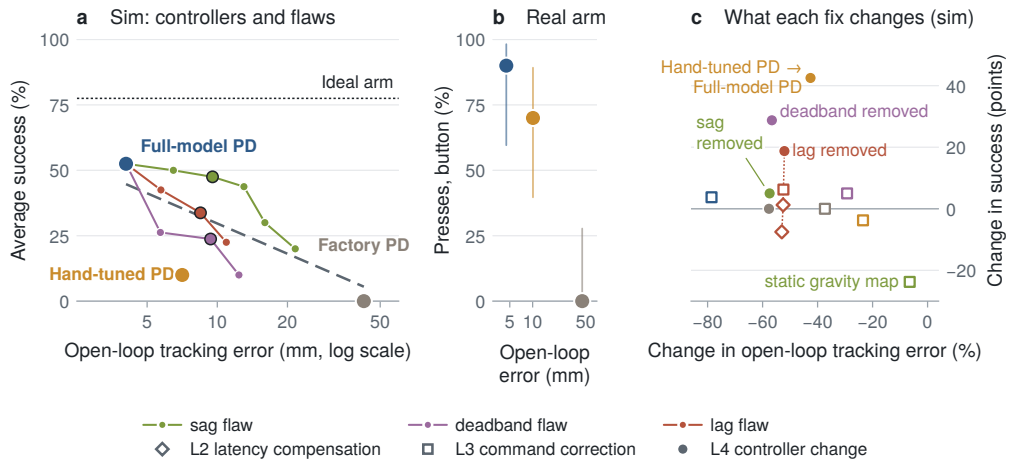


Figure 2: (a) Simulated average success against open-loop tracking error; dashed: log-linear fit; lines: each flaw at growing sizes, starting from the full-model PD; outlined: the size used elsewhere. (b) Real presses of 10 against real error. (c) Change in success against change in open-loop error per fix, colours as (a); hollow: L2, L3 fixes; filled: L4 changes; dotted: lag-flaw fixes and removal.

Different types of error affect success differently. Sag, from imperfect gravity compensation, leaves the arm resting below its target, and deadband, due to motor friction, leaves small motions unexecuted; both are spatial errors, which leave the arm in the wrong place even when it holds still, as it does at a grasp or a placement. Lag, from low gains or heavy filtering, is a temporal error: the arm reaches the right place too late, trailing a moving command and catching up once it stops. Under the UMI default, which starts each new chunk at the arm, a spatial error returns at every replan while a lag is dropped, so averaged over Push Slider, Cube Stack and Mug Hook the policy suffers more from spatial than from temporal error, as Sec. 2.3 details. Whether an error hurts also depends on the task: on Peg-in-Hole, a downward insertion task, the sag flaw's downward bias carries the peg into the hole, so it succeeds 85% of the time, as often as an ideal arm.

Fixes that cut tracking error do not consistently improve success. Fig. 2c plots each fix's change in success against its change in open-loop error. The command-side fixes at L2 and L3, from UMI-style latency compensation and error feedback to inverse models and calibration maps, cut the open-loop error by up to 79%, yet do not consistently improve success (App. F.2). Timing fixes such as latency compensation and RTC [19] correct pipeline delays such as inference latency, but they do not address lag inside the controller, whereas removing that lag does. Across episodes, the offset carried into each replan separates success from failure on every task, yet the fixes shrink it without raising success; its size alone does not decide success, and Sec. 2.3 turns to how the execution anchor feeds it into the next plan.

2.3 The execution anchor

The execution anchor decides where each new chunk starts. Each action chunk $\{\Delta_k\}$ is predicted relative to a starting pose and composed onto the execution anchor,

$$x_t^{\text{anc}} = x_{\text{d},t} \oplus \alpha\,(x_t \ominus x_{\text{d},t}), \qquad x_{\text{d},t+k} = x_t^{\text{anc}} \oplus \Delta_k, \tag{1}$$

where $x_{\text{d},t}$ and x_t are the commanded and measured poses at observation time, and $\oplus$ and $\ominus$ compose and difference poses. With $\alpha{=}1$, the measured anchor used by UMI [1], each chunk starts at the arm's measured pose. With $\alpha{=}0$, the commanded anchor, it starts from the pose commanded by the previous chunk at that time, as in IndustReal's action integrator [26]. Intermediate values blend the two, as Fig. 3a shows. Training cannot settle this choice: UMI data record a single pose stream, so the commanded and the measured pose coincide.

The measured anchor re-injects a steady error at every replan. Because each new chunk is composed on the measured pose, a steady offset between command and arm shifts the plan at every replan, and only the policy's own corrections, based on what it observes, can counteract it. In the

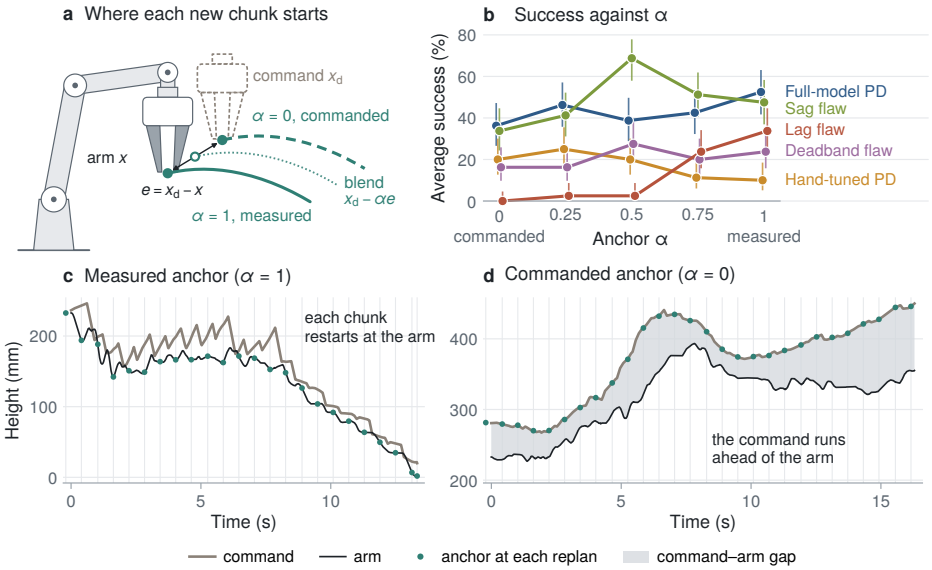


Figure 3: (a) One replan, schematic: the same next chunk starts at the commanded pose (α=0), at the measured pose (α=1) or at a blend between them. (b) Simulated average success against the anchor weight α per controller; bars: 95% intervals. (c, d) Live real factory-PD runs under the measured (c) and commanded (d) anchor: height of the command, arm and anchor at each replan.

run of Fig. 3c, through the real factory PD, these corrections roughly balance the shift while the arm approaches the button. Once they cease at the button, the plan descends in steps. The jump at each replan, the offset of the new chunk from the running plan, grows toward the measured anchor (App. F.4). A lag, in contrast, vanishes once the arm comes to rest, so the measured anchor discards the motion the arm has not yet executed. The same failure mode occurs in simulation for every controller we evaluated (App. F.4).

The commanded anchor keeps the arm's error out of the plan but winds up. Because each new chunk is composed on the pose commanded by the previous chunk rather than on the arm, every step the arm fails to execute remains in the plan and accumulates. Through the real factory PD the plan climbs away from the arm, as Fig. 3d shows. A lagging arm's unexecuted motion accumulates in the same way, so the lag flaw never succeeds with the commanded anchor, and Push Slider, which requires a sustained push, suffers most from windup. Both failure modes diminish with a more accurate controller.

The policy acts like a feedback controller without integral action. It corrects what it observes but has no record of what it commanded, so a steady error persists unless execution integrates its corrections. In these terms, the commanded anchor adds full integral action, the measured anchor adds none, and a blend acts as a leaky integrator, the standard fix for windup [28]. IndustReal [26] introduced this integrator view, with a clamp against windup, for single-step RL policies in contact-rich assembly. For chunked UMI policies, the best amount of integration depends on the controller and the task (Fig. 3b). The measured anchor does best where the controller already integrates, as the full-model PD does, or where the error is lag. A blend does best where a steady offset remains: at α=0.5 the sag flaw rises from 48% to 69%. IndustReal's clamped integrator removes the windup but is best only for the hand-tuned PD (App. F.3). The commanded anchor is never best on average.

3 Discussion

A frozen UMI policy leaves integral action to execution. As a rule of thumb, keep the measured anchor when the controller leaves no steady-state error, and blend toward the command, for example at α=0.5, when a steady offset remains. Future directions include cross-embodiment training and error-aware deployment that adapts the anchor to each arm.

Acknowledgments

We thank Professor S. Shankar Sastry and the Berkeley Ember Center at UC Berkeley for generously providing the laboratory resources that made it possible to complete this work.

References

[1] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Robotics: Science and Systems (RSS)*, 2024.

[2] Y. Ye, Y. Fu, Y. Lv, B. Hou, J. Cen, L. Kong, et al. Data pyramid for embodied manipulation: A survey, 2026. Preprint, arXiv:2607.24744.

[3] Zhaxizhuoma, K. Liu, C. Guan, Z. Jia, Z. Wu, X. Liu, T. Wang, S. Liang, P. Chen, P. Zhang, H. Song, D. Qu, D. Wang, Z. Wang, N. Cao, Y. Ding, B. Zhao, and X. Li. FastUMI: A scalable and hardware-independent universal manipulation interface with dataset. In *Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pages 3069–3093, 2025.

[4] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song. DexUMI: Using human hand as the universal manipulation interface for dexterous manipulation. In *Conference on Robot Learning (CoRL)*, 2025.

[5] H. Ha, Y. Gao, Z. Fu, J. Tan, and S. Song. UMI-on-Legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. In *Conference on Robot Learning (CoRL)*, volume 270 of *PMLR*, pages 5254–5270, 2024.

[6] H. Huang, H. Dong, and H. Dong. Mobile UMI: Cross-view diffusion policy with decoupled kinematics for mobile manipulation, 2026. Preprint, arXiv:2605.20894.

[7] Y. Nai, L. Chang, L. Yang, S. Yang, and Z. Li. Whole-body UMI: Transferring UMI manipulation skills to humanoid whole-body manipulation via real-time motion generation, 2026. Preprint, arXiv:2609.22829.

[8] H. Gupta, X. Guo, H. Ha, C. Pan, M. Cao, D. Lee, S. Scherer, S. Song, and G. Shi. UMI-on-Air: Embodiment-aware guidance for embodiment-agnostic visuomotor policies. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[9] C. Xu, S. Zheng, Z. Su, Z. Jian, Y. Xu, T. Yi, X. Zhang, J. Wang, Y. Wang, and J. Yu. Mind the gap: Rethinking I/O design for contact-rich visuomotor policy learning. In *Conference on Robot Learning (CoRL)*, 2026. To appear.

[10] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023. doi:10.15607/RSS.2023.XIX.016.

[11] Z. Fu, T. Z. Zhao, and C. Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[12] S. Lee, Y. Baek, J. Won, J. Kim, K. Hyunwoo, S.-H. Baek, I. Laptev, and S. Kwak. Taming VLAs under robot execution errors: Self-compensation and stress testing, 2026. URL https://arxiv.org/abs/2609.37334. Preprint, arXiv:2609.37334.

[13] A. Bronars, Y. Park, and P. Agrawal. Tune to learn: How controller gains shape robot policy learning, 2026. Preprint, arXiv:2604.02523.

[14] Y. Yang, F. Wu, C. Allen-Blanchette, and A. Chakraborty. Demonstration calibrated port-hamiltonian retuning for manipulation policies, 2026. URL https://arxiv.org/abs/2610.05755. Preprint, arXiv:2610.05755.

[15] M. Shirasaka, T. Ohkawa, T. Okubo, N. Scianca, T. Matsushima, and K. Ota. RoboPace: Contact-aware time-optimal retiming for action-chunk policies, 2026. Preprint, arXiv:2610.09696.

[16] S. Ross, G. J. Gordon, and J. A. 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 (AISTATS)*, volume 15 of *PMLR*, pages 627–635, 2011.

[17] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. DART: Noise injection for robust imitation learning. In *Conference on Robot Learning (CoRL)*, volume 78 of *PMLR*, pages 143–156, 2017.

[18] X. Xu, Y. Hou, C. Xin, Z. Liu, and S. Song. Compliant residual DAgger: Improving real-world contact-rich manipulation with human corrections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[19] K. Black, M. Y. Galliker, and S. Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

[21] M. Zhao, C. Jiang, T. An, J. Jia, and J. Yang. APEX: Adaptive policy execution for precise manipulation, 2026. Preprint, arXiv:2606.16504.

[22] Y. Huang, S. A. Moore, and B. Chen. Online sim-to-real adaptation via closed-loop system modeling, 2026. Preprint, arXiv:2609.28878.

[23] S. Sagar and R. Senanayake. Test-time adaptation of manipulation policies under actuator degradation, 2026. Preprint, arXiv:2609.36182.

[24] D. Son, F. Shkurti, J. Lee, N. Shah, B. Kim, and D. Fox. TAM: Torque adaptation module for robust motion transfer in manipulation. In *Conference on Robot Learning (CoRL)*, 2026. To appear.

[25] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, et al. ASAP: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. In *Robotics: Science and Systems (RSS)*, 2025.

[26] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang. IndustReal: Transferring contact-rich assembly tasks from simulation to reality. In *Robotics: Science and Systems (RSS)*, 2023. doi:10.15607/RSS.2023.XIX.039.

[27] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[28] K. J. Åström and T. Hägglund. *Advanced PID Control*. ISA – The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC, 2006.

[29] B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7): 1083–1138, 1994. doi:10.1016/0005-1098(94)90209-7.

Appendix overview

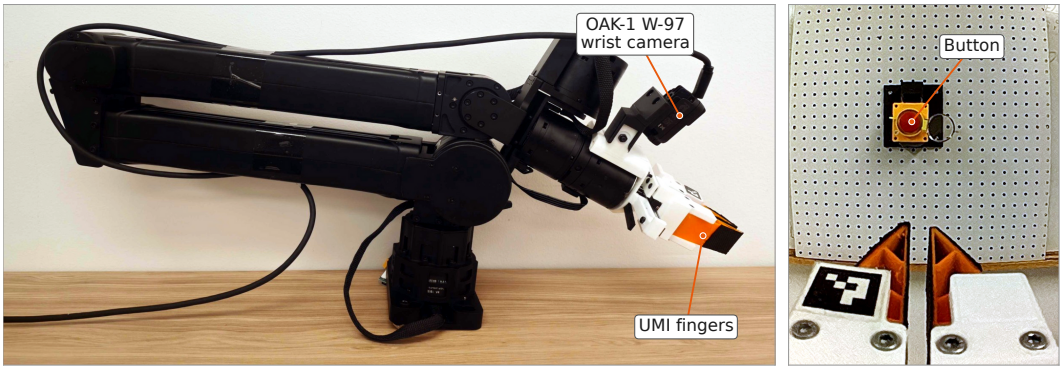
- A **Hardware and the real task**: the Big YAM, gripper and the button task.
- B **Simulated twin and tasks**: gravity and friction fit of the twin; the four simulated tasks.
- C **Policies and data**: the frozen Diffusion Policies and their training data.
- D **Controllers**: how tracking error is measured, the three standard variants and their control laws, and how each flawed variant is injected and sized.
- E **Execution and the anchor**: the execution recipe and the anchor equation.
- F **Additional results**: per-task success and the scripted plan, every command-side fix, full anchor curves, and the anchor mechanism on the real arm and in simulation.

A Hardware and the real task

Arm. We use the i2rt Big YAM arm for real-world experiments, a 6-DoF arm with about 1.1 m reach. Its joint motors run in MIT mode, a motor-side impedance law that evaluates $\tau = K_p(q_{\mathrm{des}} - q) + K_d(\dot{q}_{\mathrm{des}} - \dot{q}) + \tau_{\mathrm{ff}}$ on the motor; command packets are quantised to 12 bits, and the damping gain is capped at $K_d \leq 5$, which limits how stiff any controller on this arm can be. The policy drives a UMI-style gripper. The executor resamples each chunk into 10 ms command rows and applies the recipe of App. E.

Fig. 4 shows the real arm and the button task.

Button task. The real policy presses a button on the table within 45 s.



(a) Big YAM arm **(b)** Press-button task

Figure 4: Real hardware. (a) The Big YAM arm with the OAK-1 W-97 wrist camera and the UMI fingers. (b) The button task, seen from the wrist camera with the fingers above the button.

B Simulated twin and tasks

Twin. The simulated twin is an Isaac Lab model of the Big YAM. Its gravity is refit to the real arm: the simulated gravity on joints 2–4 is scaled by 1.115, 1.105 and 0.724, the least-squares fit to a refit model of the arm with its tool at 389 static poses held on the real arm. Its joint friction follows a stick-slip law fitted to real breakaway ramps and constant-speed sweeps: a joint slower than 0.005 rad/s is held up to its breakaway torque, and a moving joint feels Coulomb sliding plus viscous friction (Eq. 2, Table 2) [29]. For joint j, the friction torque is

$$\tau_{\mathrm{f},j} = \tau_{\mathrm{c},j} - b_j\,\dot{q}_j, \qquad -F_j^{+} \leq \tau_{\mathrm{c},j} \leq F_j^{-}, \tag{2}$$

$$F_j^{\pm} = \begin{cases} F_{\mathrm{b},j}^{\pm} + \kappa_j\,|\tau_{g,j}|, & |\dot{q}_j| < v_{\mathrm{s}}, \\ F_{\mathrm{s},j}^{\pm} + \kappa_j\,|\tau_{g,j}|, & \text{otherwise}, \end{cases} \tag{3}$$

where F_{s} is the sliding (Coulomb) amplitude, $F_{\mathrm{b}} \geq F_{\mathrm{s}}$ the breakaway amplitude, b_j the viscous coefficient, κ_j a load slope on the plant's gravity torque $\tau_{g,j}$, $v_{\mathrm{s}} = 0.005$ rad/s the stick velocity, and $\pm$ the direction of motion. The Coulomb term $\tau_{\mathrm{c},j}$ is solved implicitly at every physics step

as the maximum-dissipation torque within these bounds: a sliding joint feels the full bound against its motion, and a joint at rest is held as long as the other torques on it stay within the bounds. The viscous term uses the end-of-step velocity. Actuator delay, torque lag and rotor (armature) inertia are not modelled.

Table 2: Stick-slip friction of the simulated twin. τ_g: gravity torque on joint 2.

Joint	Sliding (N m)	Breakaway (N m)	Viscous (N m s/rad)
J1	1.6	2.25	2.2
J2	$1.07 + 0.055\,\lvert\tau_g\rvert$	as sliding	1.3
J3	2.1	$+2.3, -3.0$	0
J4	0.45	as sliding	0.05
J5	0.0408	as sliding	0.1524
J6	0.2	as sliding	0

Ideal arm. The ideal arm sets the gripper pose kinematically to the command, with no arm dynamics.

Tasks. The four simulated tasks are shown in Fig. 5. They span sustained contact, placement and insertion, and differ in which execution error they punish. Each task is evaluated on 20 paired seeds, and success must hold for 1 s.

- *Push Slider*: pinch the post and push it along a rail, from one end to within 3 cm of the other; a sustained push, which punishes windup.
- *Cube Stack*: place one cube on another, released within 2 cm of its centre and 10° of upright; a fine placement, which punishes a steady offset.
- *Mug Hook*: hang a mug by its handle on a hook branch, threading a 4 by 4 cm handle opening onto a 1.2 cm rod and releasing the mug.
- *Peg-in-Hole*: drop a cylindrical 4 cm peg into a 6 cm entry, slide it under a plate along a slot with 3.5 mm clearance per side, and tip it at least 4.5 cm deep into the hole, within 25° of its axis.



(a) Push Slider **(b)** Cube Stack **(c)** Mug Hook **(d)** Peg-in-Hole

Figure 5: Simulated tasks: (a) Push Slider, (b) Cube Stack, (c) Mug Hook, and (d) Peg-in-Hole.

C Policies and data

Policies. Every task uses its own Diffusion Policy [27] in the UMI configuration [1]. Each policy takes the wrist-camera image and the proprioceptive state and predicts a chunk of end-effector poses and gripper widths, expressed relative to the end-effector pose at observation time. We use a one-dimensional UNet denoiser conditioned on a fine-tuned CLIP ViT-B/16 image encoder. Additional parameters are shown in Table 3.

Data. The simulated demonstrations, 100 successful episodes per task, come from a demonstration generator with human pacing, recorded at 100 Hz on a free-floating gripper with no arm or controller. A few design choices in the scripted policies shape these data for the Diffusion Policy. Each motion starts slowly, so the first frames show the scene, and poses are chosen so that the object and its target stay in the wrist camera's view. Until the task is done, the gripper never stays still for a whole action chunk, since a still window becomes a plan the policy can repeat forever. Releases are quick and happen while the gripper is still moving, and recording continues for about one second afterwards,

Table 3: Diffusion Policy parameters. Where the real policy differs, its value follows the slash.

Parameter	Value
Image encoder	CLIP ViT-B/16, fine-tuned
Observation horizon (image, proprioception)	2
Image size	224×224
Action horizon (chunk)	16
Waypoint spacing	60 ms / 100 ms
Diffusion steps, training / inference (DDIM)	50 / 16
Training epochs	40 / 50
Validation split	5%

so the policy learns to let go. Fig. 6 shows how a scripted trajectory runs in Isaac Sim, for one Mug Hook demonstration. The real demonstrations, about 100 episodes of the button press, were recorded with a handheld UMI-style gripper whose pose is tracked by a phone, with a wrist camera at 30 fps. Fig. 7 shows the handheld rig.

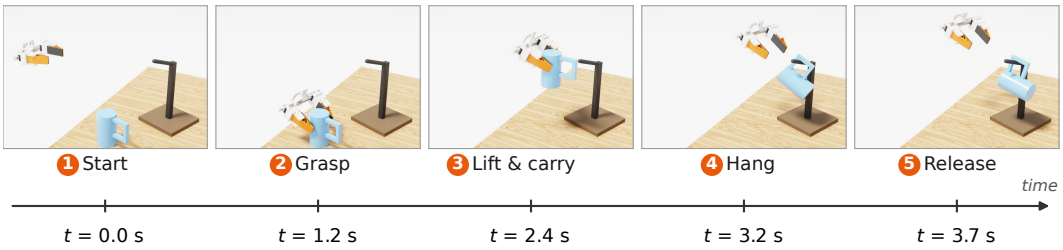


Figure 6: A scripted Mug Hook demonstration in simulation, recorded on a free-floating UMI-style gripper with no arm or controller: start, grasp, lift and carry, hang, and release.

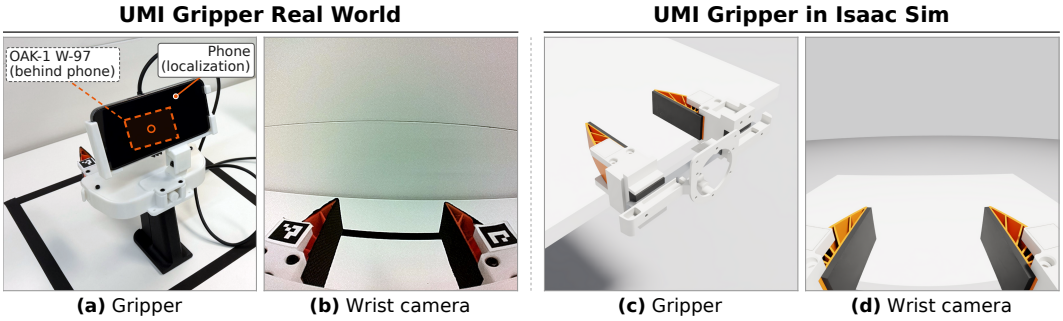


Figure 7: The UMI-style gripper, real and simulated. (a) The handheld gripper used to record the real demonstrations, with a phone for localization and the wrist camera behind it; (b) its wrist camera's view. (c) The same gripper simulated in Isaac Sim; (d) its simulated wrist camera's view.

D Controllers

All controllers command joint targets q_{des}, $\dot{q}_{\text{des}}$ and a feedforward torque τ_{ff} to the motor law of App. A. Table 1 lists the components each controller uses; here we give only the gains and control laws. We test three standard variants, the controller ladder, and three flawed variants built from the best of them.

Open-loop tracking error. Table 1 characterizes each controller by its open-loop tracking error. The 12 standard takes are recorded references replayed through each controller without the policy: eight calibration motions from the real arm, one motion near the button, one at the start pose, and a rate ramp and a rate step. After the first 3 s, in which the first command is held, we take the RMS of the 3-D tool position error $\|x_{\text{d}} - x\|$ and average over the takes. Rotation is not included. On the real arm the same RMS is computed on a held-out block of 14 calibration motions.

D.1 Standard variants

Factory PD. The vendor's controller: $K_p = (80, 80, 80, 40, 40, 10)$, $K_d = (5, 5, 5, 3, 1.5, 1.5)$, $\dot{q}_{\text{des}} = 0$ and $\tau_{\text{ff}} = f_g \odot g(q)$ with $f_g = (1, 1.2, 1, 1.1, 1, 1)$.

Hand-tuned PD. The same law with $K_p = (120, 245, 175, 70, 40, 20)$ and $K_d = (5, 5, 5, 3.75, 1.5, 1.5)$; its gravity model uses a refit tool mass, and it applies half Coulomb compensation on the measured velocity:

$$\tau_{\mathrm{ff}} = g(q) + c_{\mathrm{h}} \odot s(\dot{q}), \qquad s(\dot{q}) = \mathrm{sign}(\dot{q}) \tanh\big(\max(|\dot{q}| - 0.03, 0)/0.03\big), \tag{4}$$

with $c_{\mathrm{h}} = (0.95, 0.9, 1.3, 0.125, 0.03, 0.03)$ N m.

Full-model PD. With $K_p = (120, 245, 175, 80, 40, 20)$ and the effective $K_d = (5, 5, 5, 4, 1.5, 1.5)$, the target $q^\star$ from inverse kinematics first passes through a critically damped reference filter of bandwidth ω,

$$\ddot{r} = \mathrm{clip}\big(\omega^2(q^\star - r) - 2\omega\dot{r}\big), \qquad q_{\mathrm{des}} = r, \quad \dot{q}_{\mathrm{des}} = \dot{r}, \tag{5}$$

with acceleration and velocity limits of 80 rad/s^2 and 5 rad/s. The feedforward, with Coulomb compensation on the reference velocity and a slow integrator I, is

$$\tau_{\mathrm{ff}} = f_g \odot g(q) + M(q)\,\ddot{r} + h(q, \dot{q}) + c_{\pm} \odot \tanh(\dot{r}/0.02) + I, \tag{6}$$

where M, g and the Coriolis term h come from a model of the arm with its tool, $f_g = (1, 1.539, 1.542, 1.385, 1, 1)$, and $c_{\pm}$ are per-joint Coulomb torques of up to 3 N m. The integrator accumulates $I \leftarrow \mathrm{clip}\big(I + 0.5\,K_p(r - q)\,\Delta t\big)$ within per-joint caps of 0.3–4.5 N m and pauses when the torque saturates in the same direction.

D.2 Flawed variants

Each flaw is the full-model PD with one error type added back, as marked in Table 1. The lag flaw's filter acts on the six arm joints only, while the gripper is commanded directly, so the arm lags the gripper, as on a real arm whose gripper is commanded separately. The integrator is switched off for sag and deadband because it would otherwise absorb the injected error.

We size the sag and lag flaws by the error they inject; the deadband flaw has no size. The sag size is the arm's mean downward offset after a 3 s settle over the 12 standard takes; the lag size is the ramp lag, the pure time shift that best aligns the arm's motion with the reference on a ramp take. For the flaw-size points of Fig. 2, the sag flaw scales gravity by 0.935, 0.87 and 0.805 (4.9, 8.5 and 12.2 mm offset); the lag flaw uses 13, 7 and 5 rad/s (90, 210 and 290 ms); and the deadband flaw uses half friction compensation, none, or none with K_p scaled by 0.7. Elsewhere the sag flaw is the 8.5 mm size and the lag flaw the 210 ms size.

E Execution and the anchor

Recipe. The executor follows the real one: the policy sees the measured pose, replans every 0.6 s, and blends each new chunk in over 0.1 s from the unexecuted remainder of the previous one, linearly in position and by slerp in rotation. In simulation the physics keeps running on the previous reference during the 0.19 s inference latency, and the waypoints of the new chunk that are already in the past are dropped.

Execution anchor. A pose is $x = (p, R)$ and an increment $\delta = (v, \omega)$, with v a translation in the pose's own frame and ω a rotation vector. We write $y \oplus \delta = \big(p_y + R_y v,\ R_y\,\mathrm{Exp}(\omega)\big)$ for composing a pose with an increment and $x \ominus y = \big(R_y^\top(p_x - p_y),\ \mathrm{Log}(R_y^\top R_x)\big)$ for the increment from y to x. Each chunk of relative poses $\{\Delta_k\}$ is composed onto an anchor pose, $x_{\mathrm{d},t+k} = x_t^{\mathrm{anc}} \oplus \Delta_k$, with

$$x_t^{\mathrm{anc}} = x_{\mathrm{d},t} \oplus \alpha\,(x_t \ominus x_{\mathrm{d},t}), \tag{7}$$

where x_t is the measured pose at observation time and $x_{\mathrm{d},t}$ the pose commanded for that time. The policy always sees the measured pose and the image; only where each new chunk starts depends on α.

F Additional results

F.1 Controllers per task and the scripted plan

The controller ladder holds on every simulated task (Table 4) and on the real arm, where the factory PD, the hand-tuned PD and the full-model PD press the button in 0, 7 and 9 of 10 runs. On Peg-in-

Hole, a flaw's direction can outweigh its size: the sag flaw succeeds more often than the full-model PD (Sec. 2.2). Two extra sag runs, with gravity compensation scaled down further, fill the range between the flaws and the factory PD in Fig. 2a. Two controller changes show that success does not follow open-loop error. A refit gravity model cuts the factory PD's open-loop error by more than half but leaves its success at 0%, as the grey filled point in Fig. 2c shows, whereas the gravity and friction compensation of the full-model PD raises the hand-tuned PD's success from 10% to 20% at about the same open-loop error.

Policy against a scripted plan. The scripted plan is planned once from the true initial state and executed on a timer, gripper included, with no replanning. It is privileged, since it sees the true state, so it is a diagnostic of how each error affects a timed execution, not a ceiling on the policy. On Push Slider, Cube Stack and Mug Hook the same error hurts the two executors differently (Table 4): steady offsets cost the replanning policy more than the timed plan, while lag costs the timed plan more, because its gripper fires on schedule while the arm still trails. An error's cost therefore depends on how the plan is executed, which is what the execution anchor controls.

Table 4: Success (%) of the policy and the scripted plan.

Controller	Executor	Push Slider	Cube Stack	Mug Hook	Peg-in-Hole	Average
Ideal arm	policy	90	80	55	85	78
Full-model PD	policy	100	55	45	10	53
	scripted plan	100	95	60	95	88
Hand-tuned PD	policy	10	10	20	0	10
	scripted plan	65	75	70	55	66
Factory PD	policy	0	0	0	0	0
	scripted plan	60	60	0	0	30
Sag flaw	policy	55	45	5	85	48
	scripted plan	100	95	15	100	78
Deadband flaw	policy	50	35	10	0	24
	scripted plan	60	80	55	15	53
Lag flaw	policy	85	50	0	0	34
	scripted plan	100	5	0	0	26

F.2 Command-side fixes

Table 5 gives every command-side fix of Fig. 2c. Each fix is compared with its own baseline on the same 20 paired seeds per task, 80 episodes per average, so changes of a few points, such as 34% to 40%, are within sampling noise. A time lead sends each command early by a fixed time; position-only leads shift the position, full-pose leads also the orientation and gripper. Error feedback adds a gain times the current tracking error to the command. The inverse-model feedforward, which exists only for the lag flaw, inverts its reference filter. Calibration maps subtract a fitted pose-dependent offset, and the inverse dead zone pre-compensates small commanded steps. An RTC-style prefix freezes the already committed part of each chunk [19]. Error feedback runs at gain 2.0 on the full-model PD and 0.5 on the hand-tuned PD, where 2.0 is unstable. On the lag flaw, fixes that target the right error and match the open-loop error of removing it still do not recover its success; only removing the lag inside the controller does. An RTC-style prefix leaves the lag flaw's success unchanged. The static gravity map's drop on the sag flaw comes mostly from Peg-in-Hole (85% to 15%), where the sag flaw's low bias had carried the peg into the hole (App. F.3).

F.3 Full anchor curves

Lag is transient: the arm trails the command while moving and catches up when it stops, so the measured anchor drops the unexecuted motion and the lag never accumulates, whereas the commanded anchor stacks new motion on it and winds up. For a steady offset such as the sag flaw, the measured anchor re-injects the offset at every replan, and the commanded anchor still winds up wherever the arm cannot follow, so a blend keeps both modes smaller (Fig. 8). Per task (Table 6), the best anchor moves with the controller's error, and blending often trades Push Slider, which punishes windup, for Mug Hook, which punishes a re-injected offset. The best weights named in Sec. 2.3 and Table 6 are the highest points of this sweep on the same seeds, not settings chosen beforehand and evaluated

Table 5: Command-side fixes: open-loop tracking error and average success, each before and after the fix.

Fix	Applied to	Open-loop RMS (mm)	Success (%)
Time lead 0.21 s, position only	Lag flaw	8.5 → 4.0	34 → 35
Full-pose time lead 0.21 s	Lag flaw	8.5 → 4.0	34 → 26
Inverse-model feedforward	Lag flaw	8.5 → 4.0	34 → 40
Error feedback	Full-model PD	4.1 → 0.9	53 → 56
Error feedback	Hand-tuned PD	7.1 → 5.4	10 → 6
Inverse dead zone	Deadband flaw	9.4 → 6.6	24 → 29
Static gravity map	Sag flaw	9.5 → 8.9	48 → 24
Calibrated sag map	Factory PD	42.5 → 26.6	0 → 0

separately; with intervals of about ± 10 points (Fig. 3b), only the larger gaps, such as the sag flaw at α=0.5, are reliable.

Leaky PLAI. IndustReal's leaky policy-level action integrator [26] anchors each chunk on the commanded pose but clamps its gap to the measured pose per axis. It removes the commanded anchor's windup and beats the commanded anchor for every controller, but it is the best anchor only for the hand-tuned PD (30% against 25% on average): its best clamp, 10 mm, is the smallest of the 40, 20 and 10 mm tested on the same seeds, which brings it close to the measured anchor.

Table 6: Success (%) per task against the anchor weight α.

Controller	Task	$\alpha = 0$	0.25	0.5	0.75	1
Sag flaw	Push Slider	15	35	65	65	55
	Cube Stack	35	40	50	50	45
	Mug Hook	55	45	75	15	5
	Peg-in-Hole	30	45	85	75	85
	Average	34	41	69	51	48
Hand-tuned PD	Push Slider	5	15	15	15	10
	Cube Stack	30	25	15	0	10
	Mug Hook	25	50	45	30	20
	Peg-in-Hole	20	10	5	0	0
	Average	20	25	20	11	10
Full-model PD	Push Slider	40	60	70	65	100
	Cube Stack	20	45	35	45	55
	Mug Hook	40	30	45	55	45
	Peg-in-Hole	45	50	5	5	10
	Average	36	46	39	43	53
Lag flaw	Push Slider	0	10	10	50	85
	Cube Stack	0	0	0	40	50
	Mug Hook	0	0	0	0	0
	Peg-in-Hole	0	0	0	5	0
	Average	0	3	3	24	34
Deadband flaw	Push Slider	10	5	20	45	50
	Cube Stack	0	25	35	35	35
	Mug Hook	15	30	55	0	10
	Peg-in-Hole	40	5	0	0	0
	Average	16	16	28	20	24

F.4 The anchor mechanism on the real arm and in simulation

Across all real runs the jump at replan grows toward the measured anchor and the command–arm gap toward the commanded anchor, and both stay small for the accurate controllers (Table 7). Both modes appear in simulation for every controller we swept (Fig. 8 shows three), and under the measured anchor the flaws re-inject a larger offset than the full-model PD, as the less accurate controllers do on the real arm. In simulation the policy's corrections balance this offset in the same way: in a Peg-in-Hole episode with the sag flaw, while the arm hovers, each chunk climbs about 5 mm and each replan lowers the plan by about 5 mm, so the height holds.

Table 7: Real arm: median jump at replan and median largest command–arm gap per run against the anchor weight α. –: not run.

	Jump at replan (mm)			Command–arm gap (mm)		
Controller	α=0	0.5	1	α=0	0.5	1
Factory PD	7.1	18.2	45.0	105.0	70.4	50.0
Hand-tuned PD	6.4	7.5	9.1	19.0	19.7	18.2
Full-model PD	5.7	–	7.7	15.1	–	12.8

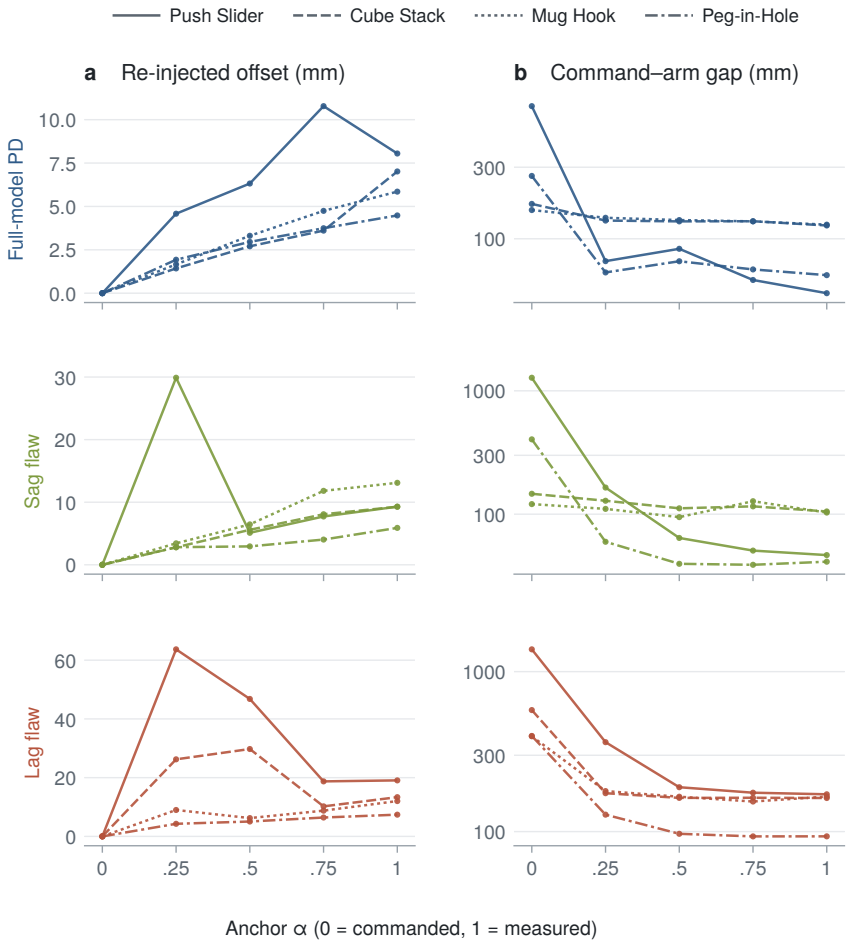


Figure 8: Both modes in simulation, for the full-model PD, the sag flaw and the lag flaw: (a) the offset re-injected at each replan, α times the command–arm offset, which grows toward the measured anchor; (b) the largest command–arm gap in an episode, which grows toward the commanded anchor. Medians against α; Push Slider solid, Cube Stack dashed, Mug Hook dotted, Peg-in-Hole dash-dotted.