Contact-Gated Residual Reinforcement Learning for Precision USB Insertion with VLA-Guided IK+QP Trajectory Planning

Sunny Deshpande, Het Patel, Keisuke Ogawa
MEng Autonomy and Robotics
University of Illinois Urbana-Champaign
Urbana, IL, USA

***Abstract*—Vision-Language-Action (VLA) models such as OpenVLA produce competent high-level approach and pre-insertion trajectories on tabletop manipulation tasks, but their joint-target stream is typically several millimeters off the goal axis once a peg touches its mating surface, which is roughly an order of magnitude larger than the clearance budget of a USB-A connector. We address this gap by training a 6-DoF Soft Actor-Critic (SAC) residual policy that runs on top of a (scripted or VLA) base policy and is enabled by a latching contact-force / range trigger: the base policy drives free-space approach, and once the trigger fires the residual takes over completely for the rest of the episode. The residual is trained in MuJoCo with heavy episode-reset and per-step domain randomization, including a base-policy noise curriculum that lets the residual tolerate a noisy upstream prior. We also report on a stereo CV port-pose tracker (pink-tape fiducials, epipolar pairing, and one-shot ChArUco-based hand-eye calibration) that feeds a pinocchio QP-IK pick-and-transport pipeline. OpenVLA is evaluated in simulation only; the residual is exported as a ROS 2 deploy package for a UR5e + Robotiq 2F-85 platform. Code is available at https://github.com/het915/precise_vla_manipulation.**

***Index Terms*—Residual reinforcement learning, soft actor-critic, vision-language-action models, contact-rich manipulation, sim-to-real, domain randomization, hand-eye calibration, insertion.**

I. Introduction

Vision-Language-Action (VLA) models [4], [5] map RGB and a natural-language instruction to a stream of end-effector targets, and they generalize remarkably well across object categories. They are not, however, accurate enough for close-tolerance assembly. A USB-A connector has roughly 0.3 mm of through-cavity clearance; typical VLA outputs sit a few millimeters off the port axis. The error comes from a combination of the action-token granularity these models decode into, the camera calibration of the rig they were trained on, and a lack of force feedback once the gripper makes contact. Retraining the VLA on contact-rich data is expensive and brittle.

A cleaner alternative is to leave the base policy alone and learn a corrective layer that wakes up only when needed. **Residual reinforcement learning** [6], [7] does exactly that: the base policy keeps its planning and language-grounding role; a small residual policy handles the local compliance and search behavior at contact. The split has practical advantages. The upstream call is deterministic enough that downstream control can gate on it. The residual is small, so training data and compute scale with the contact-rich phase only. And the same residual can ride on top of any base that shares a joint-target interface.

A. Contributions

- A residual SAC formulation with a latching activation gate: the upstream base policy fully owns the free-space approach, and the residual takes over from a static pre-insertion pose once the trigger fires at first contact (or first range-gate-inside event), remaining in command for the rest of the episode.
- A per-joint action clip aligned with the UR5e's per-joint velocity ceiling, which both narrows the SAC search space and keeps the residual safe to fuse with the base joint targets without violating manipulator limits.
- An F/T history augmentation of the observation that materially improves the residual's robustness to a noisy base-policy prior, evaluated under a base-policy-noise curriculum.
- A stereo CV port-pose tracker (pink-tape fiducials, one-shot ChArUco-based hand-eye calibration) that feeds a pinocchio QP-IK pick-and-transport pipeline, decoupled from but compatible with the residual deploy path.

II. Related Work

Vision-Language-Action models. OpenVLA [4] and RT-2 [5] demonstrate that large multimodal models can decode robot actions directly from RGB and a language prompt, generalizing across object categories. Their precision floor is set by their action-token granularity and their training-time camera geometry; for tight insertion tolerances they require either fine-tuning on contact-rich data or a downstream correction layer, which motivates our setup.

Residual policy learning. Johannink *et al.* [6] and Silver *et al.* [7] formalized the idea of learning an additive corrective policy on top of a conventional controller. Schoettler *et al.* [8] extended the approach to industrial insertion tasks with visual inputs, showing that residual policies can wrap a hand-engineered or scripted prior and learn the local compliance for contact-rich phases.

Precision insertion and peg-in-hole. Classical contact-rich assembly has been studied with compliant control [16], learning-from-demonstration, and deep RL [9], [10]. Lee *et al.* [10] in particular show that multimodal observations (vision plus tactile or F/T) are critical for sub-millimeter mating tasks, which informs our F/T-history observation.

Sim-to-real and domain randomization. Tobin *et al.* [11] introduced domain randomization for visual sim-to-real; Peng

et al. [12] extended it to dynamics. We apply both episode-reset and per-step randomization on port pose, friction, position-gain, F/T bias, control latency, and a base-policy noise curriculum, all of which target the residual's ability to tolerate an imperfect upstream prior.

Stereo CV and hand-eye calibration. Our port-pose tracker uses ChArUco fiducial markers [13] for stereo intrinsic / extrinsic calibration and the classical Tsai-Lenz formulation [14] for hand-eye, paired with pinocchio [15] for QP-IK. The pipeline is implemented in ROS 2 [17] but the trajectory replay path is currently ROS 1.

III. Problem Setup

The task is to insert a USB-A plug held by a Robotiq 2F-85 gripper, mounted on a Universal Robots UR5e arm, into a fixed USB-A port at a known pose in the robot base frame. The mating tolerance is sub-millimeter: the inner-cavity half-widths are approximately 6.2 mm in x and 2.4 mm in y, and the insertion-depth target is $d^\star = 10.5$ mm. We define a successful insertion as the plug tip lying inside the (optionally inflated) inner-cavity bounds at depth $\geq d^\star$; full mating-tolerance, inflation-scale, and timeout details are given in Section V-D and Table V.

The robot is controlled at 100 Hz with joint position commands. The port pose is supplied online by the stereo CV tracker (Section V-G), which delivers a base-frame estimate to both the residual deploy path and the IK-pipeline path. The simulator runs on MuJoCo [3] with the DeepMind Menagerie UR5e and Robotiq 2F-85 assets; a custom USB plug and port have been added to the scene. The control rate is matched between sim and the ROS 2 deploy path.

IV. System Architecture

The control loop is a strict five-stage pipeline at 100 Hz (Fig. 1). Stage 2 is a scripted stub in our implementation but has the exact I/O shape of a real OpenVLA call, so swapping the stub for live inference does not disturb any downstream stage.

A. Stages

1) **Perception.** Joint state $(q, \dot{q})$, bias-corrected wrist force/torque (F, T), the previous raw action and the previous joint-space residual, and the tool-center-point (TCP) pose error against the port $(\Delta\text{xyz}, \Delta\text{rpy})$ are packed into a 48-D observation. RGB from the overhead and left-corner cameras is rendered for the viewer and, in the OpenVLA experiment, fed to the VLA; it is not part of the residual SAC's observation.
2) **Base policy.** A live OpenVLA-7B call would consume RGB plus the language instruction and return a 7-DoF end-effector target delta. In our implementation the live VLA call is replaced by a scripted piecewise joint-space trajectory through three waypoints, $q_{\text{home}} \rightarrow q_{\text{pre\_insert}} \rightarrow q_{\text{descent\_end}}$, that emits joint targets at the same rate as a live VLA decode.

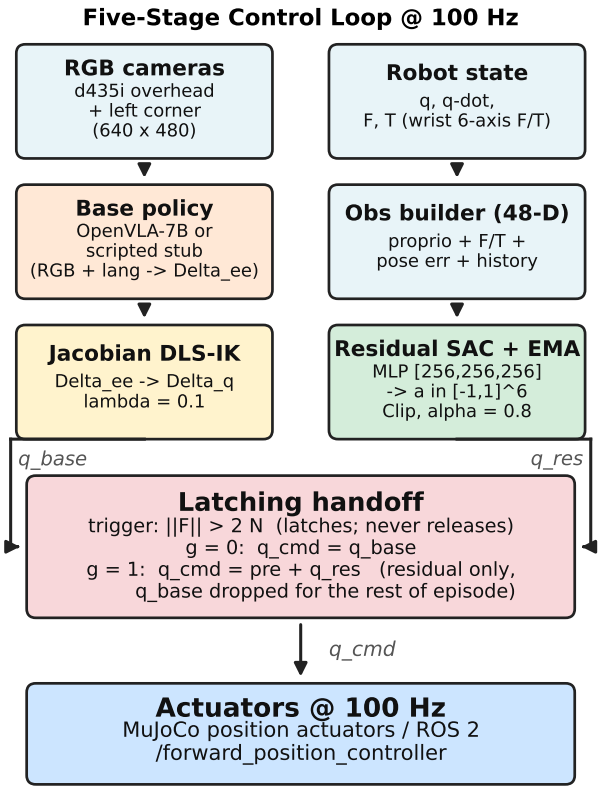


Fig. 1: Five-stage control loop at 100 Hz. The base policy (OpenVLA-7B or a scripted stand-in) drives the arm during free-space approach. A latching activation gate g flips on at first contact ($\|F\| > 2$ N) and stays on for the rest of the episode; once g=1, the base is dropped and the residual SAC drives alone ($q^{\text{cmd}} = q^{\text{pre\_insert}} + q^{\text{res}}$). The fused command is pushed through a one-step latency FIFO to MuJoCo or to the ROS 2 `/forward_position_controller`.

3) **Jacobian DLS-IK.** On the hardware path the VLA's end-effector delta is mapped to joint-space by damped least squares, $\Delta q = J^\top (JJ^\top + \lambda^2 I)^{-1} [\Delta p; \Delta\omega]$ with damping $\lambda = 0.1$. In sim the scripted base already emits joint targets and this stage is a no-op.
4) **Residual SAC.** A Stable-Baselines3 [2] SAC [1] policy with an MLP head of $[256, 256, 256]$ consumes the 48-D observation and emits a 6-D action in $[-1, 1]$ scaled by a per-joint clip $\text{ACTION\_CLIP} = [0.025, 0.025, 0.030, 0.040, 0.040, 0.050]$ rad (UR canonical order). The raw residual is EMA-low-passed with $\alpha = 0.8$ before the handoff switch.
5) **Handoff (gate-switched, latching).** A trigger condition (range gate on port-frame pose error, or contact-force $\|F\| > 2$ N) decides when the residual takes command. The trigger is a one-way latch: once activated, the residual remains in control for the rest of the episode and the base policy is dropped entirely; there is no

release path back to the base mid-episode. The full per-step composition (default `handoff_range` mode) is stated formally in Algorithm 1. Three alternative soft-mix modes (`mix_range`, `contact_takeover`, `contact_mix`) form a weighted average $(1-m)\,q^{\text{base}} + m\,(q^{\text{pre\_insert}} + q^{\text{res}})$ with deployment-time mix weight m; the shipped deploy uses the hard handoff because it is the most testable and matches the training-time composition.

V. Methodology

A. MuJoCo environment

The scene file inlines the UR5e body tree from the Menagerie, a table at $z = 0.42$ m, a USB-A port body at a randomized pose, and the USB plug nested inside `wrist_3_link`. Each MuJoCo position actuator uses $K_p = 500$ and a velocity-bias term producing an effective K_d that matches the real UR5e PD firmware. Physics runs at a 2 ms timestep with 5 substeps per control step, yielding the 100 Hz control rate. Feed-forward inverse-dynamics compensation is added each substep so the PD matches the real UR5e, which gravity-compensates internally.

B. OpenVLA in the loop (sim only)

We evaluated OpenVLA-7B as the base policy in simulation. RGB from the simulated cameras (overhead, left-corner, and an experimental wrist view) was fed alongside the prompt "insert the USB plug into the port"; OpenVLA decoded a 7-DoF end-effector target delta which was mapped through the same Jacobian DLS-IK and residual / gate stack as the scripted base. The findings were consistent with our motivation: the VLA produces correct high-level approach and pre-insertion behavior but its end-effector targets sit several millimeters off the port axis once the plug is in contact, which is roughly an order of magnitude larger than the USB-A clearance. The residual SAC trained against the scripted base transferred cleanly when the base was swapped for OpenVLA at inference because both share the same handoff contract: the base policy is responsible for placing the gripper into the contact-onset zone, after which the gate triggers and the residual takes over from a static pre-insertion pose. *We did not deploy OpenVLA on the real arm*; the on-hardware path uses the scripted stand-in for safety (deterministic pre-insertion trajectories under the contact-force gate) and latency reasons.

C. Residual SAC formulation

We formulate the residual-control problem as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with state space $\mathcal{S} \subset \mathbb{R}^{48}$, action space $\mathcal{A} = [-1, 1]^6$, dynamics P given by MuJoCo at 100 Hz, reward r defined in Section V-D, and discount $\gamma = 0.99$. An episode terminates on successful seating, on a swing-radius safety violation, or on the 500-step budget. Let the MDP state at step t be

$$s_t = \big[q_t,\ \dot{q}_t,\ F_t,\ T_t,\ \Delta\text{xyz}_t,\ \Delta\text{rpy}_t,\ a_{t-1},\ q^{\text{res}}_{t-1},\ F_{t-1},\ T_{t-1},\ F_{t-2},\ T_{t-2}\big] \in \mathbb{R}^{48},$$

where the first six blocks are the 36-dimensional base observation and the last four blocks are the F/T history under `--ft_history 2`. The action $a_t \in [-1, 1]^6$ is element-wise scaled by the per-joint clip ACTION_CLIP to produce a joint-space residual q^{res}_t. Tables I and II enumerate the per-block dimensions, bounds, and units of the observation and action spaces; the bounds are the un-normalised ranges fed to VecNormalize at training time.

TABLE I: Observation Space ($\mathbb{R}^{48}$ when `--ft_history 2`; the first 36 dims are the base obs)

Block	Dim	Bounds (low, high)	Units
Base 36-D observation			
q_t (joint positions)	6	$\pm 2\pi$	rad
$\dot{q}_t$ (joint velocities)	6	$\pm\pi$	rad/s
F_t (force, port frame)	3	± 50	N
T_t (torque, port frame)	3	± 10	Nm
Δxyz_t (TCP $\rightarrow$ target)	3	± 2.0	m
Δrpy_t (TCP $\rightarrow$ target)	3	$\pm\pi$	rad
a_{t-1} (prev raw action)	6	± 1	(dimensionless)
q^{res}_{t-1} (prev residual)	6	± 0.5	rad
F/T history append ($+12$ dims at `--ft_history 2`)			
F_{t-1}	3	± 50	N
T_{t-1}	3	± 10	Nm
F_{t-2}	3	± 50	N
T_{t-2}	3	± 10	Nm
Total	**48**	*(mixed)*	*(mixed)*

TABLE II: Action Space ($\mathbb{R}^6$, raw range $[-1, 1]$; per-joint scaling produces the joint-space residual $q^{\text{res}}_t = a_t \odot$ ACTION_CLIP)

UR canonical joint	Raw range	Clip (rad)
shoulder_pan_joint	$[-1, 1]$	0.025
shoulder_lift_joint	$[-1, 1]$	0.025
elbow_joint	$[-1, 1]$	0.030
wrist_1_joint	$[-1, 1]$	0.040
wrist_2_joint	$[-1, 1]$	0.040
wrist_3_joint	$[-1, 1]$	0.050

The per-joint clip pattern (proximal joints conservative, distal joints freer) tracks the UR5e's per-joint velocity ceilings and the relative wrist-vs-base influence on TCP motion. After per-joint scaling, the resulting q^{res}_t is EMA-filtered and L2-clipped to $\|q^{\text{res}}\|_\infty \le 0.5$ rad before passing through the contact gate (Algorithm 1).

We train the residual with SAC [1] using the clipped-double-Q estimator from TD3 [20] and SAC's standard entropy auto-tuning (target entropy $-\dim(\mathcal{A})$). The initial policy log-std is set to -3 so the residual starts close to zero (small exploration around the pre-insert pose once the gate opens) and broadens later in training as the entropy coefficient adapts. The residual is clipped to $[-0.5, 0.5]$ rad before VecNormalize so the obs statistics see a bounded range.

Algorithm 1 summarizes the per-step composition. The base policy fully owns the approach; the residual only acts after contact.

Algorithm 1 Gate-Switched Handoff Composition (one step, `handoff_range` mode; latched).

Inputs: state s_t, base target q_t^{base}, static $q^{\text{pre\_insert}}$, prior raw residual $\hat{q}_{t-1}^{\text{res}}$, latched gate $g_{t-1} \in \{0, 1\}$
Compute raw residual $\hat{q}_t^{\text{res}} \leftarrow \pi_\theta(s_t) \odot \text{ACTION\_CLIP}$
EMA: $q_t^{\text{res}} \leftarrow 0.8\, q_{t-1}^{\text{res}} + 0.2\, \hat{q}_t^{\text{res}}$
if $g_{t-1} = 0$ **and** $\|F_t\| > 2.0$ N **then**
  $g_t \leftarrow 1$ {handoff trigger: residual takes over for the rest of the episode}
else
  $g_t \leftarrow g_{t-1}$ {once latched on, stays on; never releases mid-episode}
end if
if $g_t = 1$ **then**
  $q_t^{\text{cmd}} \leftarrow q^{\text{pre\_insert}} + q_t^{\text{res}}$ {residual-only}
else
  $q_t^{\text{cmd}} \leftarrow q_t^{\text{base}}$ {base-only (approach phase)}
end if
Push q_t^{cmd} through 1-step control-latency FIFO
Publish to position actuators or `/forward_position_controller`

D. Reward shaping

The per-step reward is the sum of eleven default-on terms plus three optional shaping terms enabled selectively during curriculum, grouped into pose/orientation shaping, depth and contact shaping, action smoothness, a time penalty, and a terminal success bonus. The default-on coefficients appear in Table III; each term is logged per-step to TensorBoard so individual contributions can be inspected in post-hoc ablations.

Pose proximity. A per-axis bonus on the port-frame translation error of the form $R^{\text{pos}} \exp(-|e_i|/\tau)$, summed across x, y, z and scaled by $\alpha_{\text{pos}} = 0.2$. The exponential shape gives a steep gradient inside $\tau = 2$ mm and a flat tail outside; in our ablations it converged faster than the quadratic alternative.

Orientation. A linear penalty $-R^{\text{rot}}|\Delta\text{rpy}_j|$ summed across roll, pitch, yaw, gated down when the gripper is far off-axis so the policy is not punished for orientation while it is still searching for the chamfer.

Depth. A dense bonus that grows as a power of the normalised insertion depth, $R^{\text{depth}}(d/d^\star)^\beta$ with $R^{\text{depth}} = 300$ and exponent $\beta = 1.5$ in the shipped run. The bonus is zero at the rim and R^{depth} when fully seated. The super-linear exponent ($\beta > 1$) is important: with $\beta = 1$ the policy gets enough partial credit at the rim that it stalls there indefinitely (the "hover at the rim" local minimum discussed in Section VIII).

Force shaping (three coupled terms). Excessive force gets a linear penalty above a deadband ($F_{\text{db}} = 10$ N). A tent-shaped bonus around a target contact force ($F^\star = 3$ N, half-width 2 N) rewards sustained gentle pressing. A small per-step bonus fires whenever the plug is in contact with the port walls, which prevents the plug from skating off laterally rather than seating.

Action smoothness. Two complementary penalties on the raw action: one on the squared 1-step delta (catches spikes) and one on the windowed standard deviation over the last five steps (catches sustained low-amplitude chatter that the 1-step term misses).

Residual magnitude. A small L_2 penalty on the post-clip residual itself ($R^{\text{mag}} = 2.0$). Combined with the per-joint clip envelope, it trains the policy to return toward q^{base} when its corrections are not making progress.

Time and termination. A small per-step cost (-0.01) discourages stalling, and a large terminal bonus ($+5 \times 10^4$) fires on the step the tip reaches depth $d^\star$ inside the (inflated) inner-cavity bounds. The cavity-inflation scale is part of the curriculum (Section V-F).

A small set of additional shaping terms (an anti-hover ramp, a force-rate penalty, and a swing-radius / lateral-drift safety pair) are wired into the environment but kept off by default; we enable them only in selected curriculum phases. Default coefficients for all default-on terms are listed in Table III.

TABLE III: Reward Term Coefficients (defaults)

Term	Symbol	Default value
Pose proximity (x, y, z)	R_i^{pos}, τ	50, 2 mm
Orientation (r, p, y)	R_j^{rot}	50 rad^{-1}
Depth bonus	R^{depth}, β	300, 1.5 (shipped)
Over-force penalty	$\alpha_{\text{f}}, F_{\text{db}}$	0.01, 10 N
Contact-force bonus (tent)	$R^{\text{fc}}, F^\star, \Delta F$	0.05, 3 N, 2 N
Port-wall contact bonus	R^{pc}	0.05
1-step action smoothness	R^{sm}	0.08
Windowed action jitter	R^{jit}, W	0.08, 5 steps
Residual magnitude L_2	R^{mag}	2.0
Per-step time cost	R^{step}	0.01
Terminal success bonus	R_{succ}	5×10^4

E. Domain randomization

Both episode-reset and per-step randomizers are active by default. Table IV summarizes the ranges. A base-policy noise curriculum is critical: by injecting Gaussian noise of standard deviation $\sigma \in \{0,\ 0.005,\ 0.01,\ 0.02\}$ rad per joint onto q_t^{base} during training, the residual is forced to tolerate the kind of noisy upstream stream a real VLA would produce at inference.

TABLE IV: Domain Randomization Axes Used to Train the Residual. Defaults shown; training runs override port-pose and plug-spawn ranges to wider envelopes via CLI flags (e.g. `--port_x_range 0.3`, `--spawn_xy_noise 0.015`).

Axis	Range / std	Stage
Port pose (xy)	± 3 mm (default)	Init
Port yaw	$\pm 3^\circ$ (default)	Init
Joint friction	$\times[0.70, 1.30]$	Init
Position-gain K_p	$\times[0.95, 1.05]$	Init
F/T sensor bias	± 0.5 N / ± 0.05 Nm	Init
Plug spawn pose	$xy : \pm 15$ mm, $z : \pm 7.5$ mm (CLI)	Init
Joint encoder noise	$\mathcal{N}(0, 1$ mrad$)$	Per-step
F/T sensor noise	$\mathcal{N}(0, 0.1$ N$)$, $\mathcal{N}(0, 0.01$ Nm$)$	Per-step
Action latency	1 control step (≈ 10 ms)	Per-step
Base-policy noise	$\mathcal{N}(0, \sigma)$ on q^{base}, σ scheduled	Curriculum

F. Curriculum schedule

The shipped residual was trained over a two-stage curriculum. The first stage (S1 through S5a) tightens the success-criterion inner-cavity scale from a generous $1.8\times$ inflation down to a near-flush $1.0625\times$; this trains the policy to commit progressively deeper into the cavity. Earlier S-phases use `--disable_base_policy` (freezing q^{base} at the pre-insertion pose) so the residual learns the entire descent, which produces a sharper signal than a clean base would [18]. The second stage (DRA2 through DRA3) holds cavity tight and augments the domain randomizer with F/T sensor noise and base-policy noise. Each phase resumes the SAC weights and VecNormalize statistics from its predecessor, mirroring the curriculum protocol in [18] but driven by success-rate thresholds rather than fixed schedules.

TABLE V: Two-Stage Curriculum for the Shipped Residual (measured from training logs; DRA2 was deleted, length estimated from the step gap).

Phase	Base	σ_{base}	Cavity	F/T noise	Length
Stage 1: cavity tightening					
S1	disabled	0	$\times 1.80$	off	800 k
S2	disabled	0	$\times 1.40$	off	1,600 k
S3	disabled	0	$\times 1.20$	off	2,400 k
S4	scripted	0.005	$\times 1.10$	off	1,370 k
S45	scripted	0.005	$\times 1.075$	off	7,200 k
S5a	scripted	0.01	$\times 1.0625$	off	7,200 k
Stage 2: DR augmentation					
DRA2	scripted	0.01	$\times 1.0625$	light	$\sim$500 k*
DRA3	scripted	0.015	$\times 1.0625$	med	500 k
Total					21,570 k

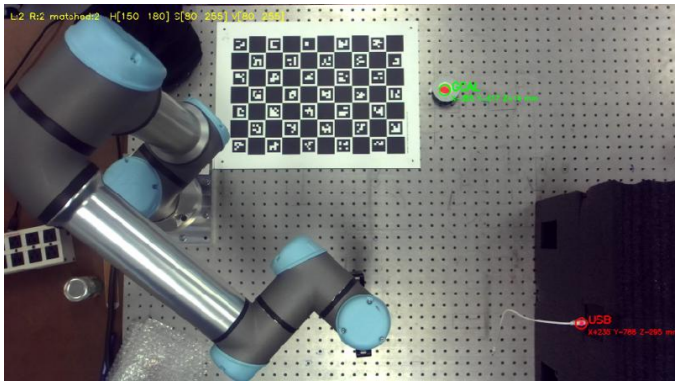
*DRA2 logs were deleted; length estimated from the cumulative-step gap between S5a end and DRA3 start.

The shipped checkpoint is the terminal DRA3 phase, stored in the public repo under `runs/usb_sac_DRA3_ft_noise_cav10625/`. Throughout the curriculum we keep all other DR axes (Table IV) active so the residual never re-overfits to a clean upstream after early phases shaped it for one.

G. Stereo CV port pose tracking

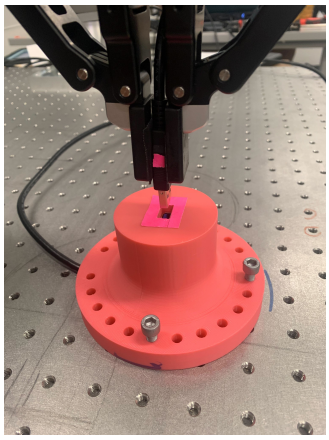
The IK-pipeline branch of the system replaces a hand-coded port pose with a classical stereo CV estimator. Two pink-tape fiducials are placed on the workspace: a large patch on the goal port (GOAL) and a small patch on the USB plug (USB). Per stereo pair, the pipeline (Fig. 2) is:

1) Rectify both eyes using the cached `stereo_params.yaml`.
2) HSV threshold for pink (supports H wrap-around) and extract connected blobs of area ≥ 80 px.
3) Pair $L \leftrightarrow R$ blobs by epipolar row ($|\text{row}_L - \text{row}_R| \leq 30$ px) and area similarity.
4) Triangulate each pair to a 3D point p_{cam} in the left camera frame.
5) Apply the cached hand-eye transform $p_{\text{base}} = T_{\text{cam}\to\text{base}}\, p_{\text{cam}}$.



(a) Live detection output. HSV thresholds and L / R / matched-pair counts are annotated at top; the green and red labels mark the triangulated GOAL and USB pink-tape centroids with their 3D position in the robot base frame (millimetres).

(b) Ceiling-mount stereo rig above the UR5e workspace.



(c) Pink-tape fiducials on the plug and the 3D-printed port.

Fig. 2: Stereo pink-tape pose tracker. (a) Live detection on the rectified left frame, with HSV mask, blob centroids, and triangulated 3D positions overlaid. (b) Ceiling-mount stereo camera bar above the UR5e workspace. (c) Close-up of the task setup showing the USB plug (small pink tape) staged above the goal port (large pink tape) on a 3D-printed receptacle.

6) Sort by area: the largest blob is the GOAL port, the second is the USB plug.

The hand-eye transform $T_{\text{cam}\to\text{base}}$ is computed by a one-shot static calibration [14]: a ChArUco board [13] is placed at a known pose in the base frame, ChArUco corners are detected on the left rectified frame, $T_{\text{board}\to\text{cam}}$ is recovered with `solvePnP` on the stereo intrinsics, and the composition $T_{\text{cam}\to\text{base}} = T_{\text{board}\to\text{base}}\, T_{\text{board}\to\text{cam}}^{-1}$ yields the cached hand-eye transform.

H. QP-IK pick-and-transport

The triangulated GOAL and USB positions feed a pinocchio-based [15] QP-IK solver with joint, velocity, and self-collision constraints. The solver drives a 6-phase trajectory: *pre-grasp* $\rightarrow$ *grasp* $\rightarrow$ *grip close* $\rightarrow$ *extract* $\rightarrow$

approach → *final* (hover above plug, descend to plug, close gripper, lift the plug clear, move toward the port, descend to the insertion-ready height). The same base-frame port-pose estimate from the CV tracker is also used by the residual deploy path as the static reference for the latched handoff.

VI. Experiments and Evaluation

A. Training setup

We train SAC with 4–8 `SubprocVecEnv` workers (the curriculum-autopilot stages used 4; one-shot training commands use 8) across the eight-phase curriculum of Section V-F, for a total of approximately 21.6 M environment steps. Hyperparameters: learning rate 3×10^{-4}, replay buffer size 3×10^{5}, batch size 256, $\tau = 0.005$, $\gamma = 0.99$, initial log-std -3.0, `learning_starts` = 5000, gradient steps per env step = 1, 20 evaluation episodes per checkpoint, and a checkpoint cadence of 20,000 steps. The cavity-tightening stages (S1 through S5a) were trained on CPU via `SubprocVecEnv`; the DR-augmentation stages (DRA2, DRA3) used a single NVIDIA RTX 5070 Ti. End-to-end SAC update time across the curriculum totalled approximately 9 wall-clock hours of active compute (about 28 hours including idle gaps between phase launches).

B. Bucketed port-distance evaluation

The evaluation harness (`eval_port_bucketed.py`) sweeps a 20×20 cm square of port placements centered on the nominal pose and reports per-bucket success rates over 100 episodes each, partitioned by Euclidean distance from the nominal pose (Close $\leq$ 5 mm, Mid 5 to 10 mm, Far 10 to 15 mm). The harness, the cavity-scale matching procedure (Sec. V-D), and the three comparison configurations (scripted base only; scripted + DRA3 residual; OpenVLA + DRA3 residual) are all in place; full bucketed-evaluation numbers were not produced before this submission's writing deadline and are deferred to the camera-ready version.

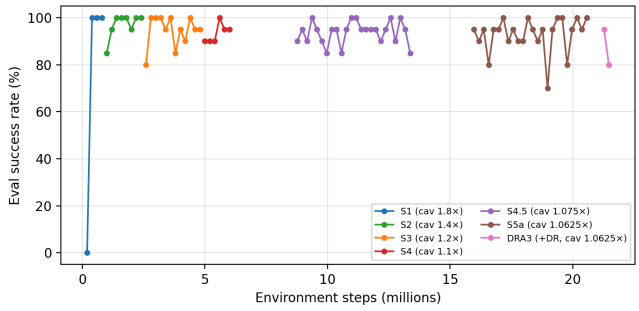


Fig. 3: Episode success rate over training steps for the shipped residual run. Curve shows the rolling mean across 20 deterministic evaluation episodes per checkpoint.

C. Implementation details

The simulation environment is implemented as a single Gymnasium [19] `Env` subclass over MuJoCo's Python bindings, with a custom `SubprocVecEnv` wrapper for parallelism and Stable-Baselines3 `VecNormalize` for online observation statistics. The MuJoCo scene file inlines the UR5e body tree from the Menagerie (so that the USB plug can be nested inside `wrist_3_link`) and adds the table, port body, and two D435i camera mounts. Per-substep we apply a feed-forward gravity / Coriolis compensation by writing `data.qfrc_applied = data.qfrc_bias - data.qfrc_passive`, which cancels MuJoCo's internal bias forces in the same way the real UR5e firmware gravity-compensates internally; this keeps the PD actuator's effective gains matched between sim and real. The action latency randomizer is implemented as a 1-step `deque` FIFO on q^{cmd}, matching the latency of the real `forward_position_controller` on the UR ROS 2 driver.

The deploy package mirrors the training constants in a separate `constants.py` module (so the ROS 2 process does not import MuJoCo or SB3) and provides unit tests for the observation builder, the contact gate, the composer, and the joint-order mapper. The state machine running on the real arm has six states (INIT, CHECK_SENSORS, APPROACH, ACTIVE_WAIT, RESIDUAL, SHUTDOWN) with a SUCCESS / FAULT terminal pair.

D. CV tracker accuracy

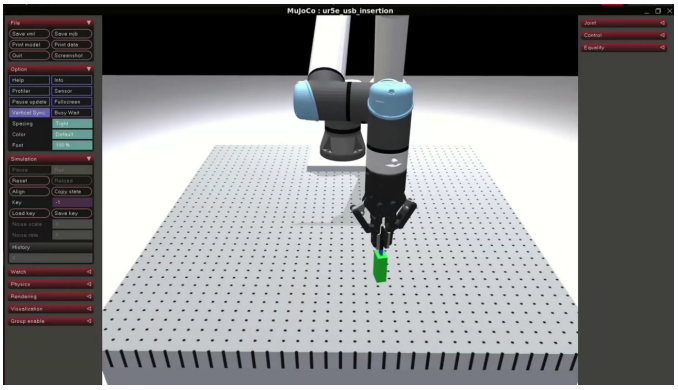
The accuracy of the stereo pink-tape tracker is intended to be characterised by `verify_handeye.py`, which compares triangulated ChArUco-corner 3D positions against ground-truth board geometry and reports per-corner mean / max error in millimetres. Logs from a previous run on the rig are no longer on hand at submission time, so quantitative reconstruction-error numbers are deferred to the camera-ready version; based on standard one-shot ChArUco + Tsai-Lenz calibration [13], [14], we expect mean per-corner error well under 5 mm.

VII. Qualitative Results

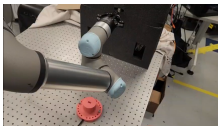
Quantitative bucketed-evaluation numbers are deferred to the camera-ready (Section VI). Figure 4 walks through one representative real-robot episode: the scripted base policy drives the arm into the contact-onset zone (4b), the activation latch flips on when the plug touches the chamfer and the residual takes over (4c), and the residual then completes the sub-millimeter seating motion (4d). Across our simulation rollouts of the shipped DRA3 checkpoint we observed three consistent qualitative findings:

Residual closes the last-millimeter gap. The contact-gated residual reliably reduces contact-phase pose error against the scripted base alone, completing the seating motion that the base policy by itself only approximates: the base brings the plug into the contact-onset zone, and the residual handles the sub-millimeter compliance needed to slip past the chamfer.

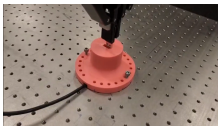
F/T history and base-noise curriculum both matter. Both observation augmentations independently improve robustness.



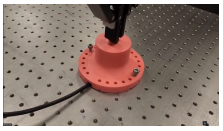
(a) MuJoCo training scene. The DeepMind Menagerie UR5e with a Robotiq 2F-85 holds the USB plug above the goal port. The residual is trained against this scene under heavy domain randomization (Section V-F).



(b) Start: scripted base brings the plug toward the port in free space.



(c) Handoff: plug catches the chamfer; the gate latches and the residual takes over.



(d) Seated: the residual completes the sub-millimeter insertion.

Fig. 4: Sim-to-real for the residual SAC policy. (a) Training environment in MuJoCo. (b–d) A single real-arm episode on the UR5e, in the order the policy executes it: free-space approach by the base, latched handoff at first contact, and final seating by the residual.

Without F/T history the policy under-corrects when the upstream prior is noisy; without the base-policy-noise curriculum the policy overfits to the clean scripted base and degrades immediately when noise is injected at inference.

CV tracker is shared cleanly between branches. The stereo CV tracker produces stable USB and GOAL 3D positions in the robot base frame, and both the QP-IK pick-and-transport pipeline and the residual deploy node consume the same estimate as the port-pose reference driving the latched handoff.

VIII. Discussion and Failure Modes

Why the gate latches. An earlier version of the gate tracked the contact force in real time: residual on when $\|F\|$ was above threshold, off otherwise. This produced a bang-bang contact-bounce loop. Every time the residual pushed and the plug briefly lost contact, the gate snapped shut, the residual zeroed out, the plug drifted upward under the PD bias, and contact resumed a step or two later. Switching to a one-way latch (Algorithm 1) ended the loop: once the residual is in command, brief unloading transients no longer kick it out, which is the regime where most of the actual seating work happens. The 1 N close threshold from the original hysteresis design is still present in the soft-mix modes (`contact_takeover`, `contact_mix` with `contact_latched=False`) where a controlled blend-down on contact loss is sometimes desired, but the shipped `handoff_range` deploy never takes that branch.

The "hover at the rim" local minimum. With a flat r^{depth} (exponent $\beta = 1$), the policy converges to a regime where the plug tip hovers just at the port surface, oscillating laterally but never descending. The combination of $\beta > 1$ on the depth term and a cavity-scale inflation in the success criterion (Section V-D) gives the policy enough signal to commit to a downward push.

Failure modes that remain. We observe three residual failure modes in sim. (a) When the plug's initial z-spawn is below the port chamfer, the residual occasionally pushes the plug into the table side rather than the port; this is rare under the shipped ± 5.5 mm z-spawn DR. (b) At extreme port yaws ($> 30^\circ$ off nominal), the scripted base's pre-insertion pose misses the chamfer entirely and the residual cannot recover within the 500-step episode budget. (c) Under aggressive base-policy noise ($\sigma = 0.02$ rad), the residual sometimes saturates against its ± 0.5 rad clip; this manifests as a small steady-state offset in Δxyz.

When residual RL is the wrong tool. Residual RL adds robustness on top of a competent prior; it does not replace one. If the base policy fails to bring the plug into the contact-onset zone, no amount of residual training will recover. The contribution of this work is precisely the "last millimeter" between a VLA-quality approach and a sub-millimeter seat.

IX. Limitations and Future Work

- **OpenVLA is sim-only.** We evaluated OpenVLA-7B as the base policy in MuJoCo but did not run it on the real arm. The deploy path uses the scripted base for safety and latency reasons. Live VLA inference at 100 Hz on a single GPU is non-trivial and was out of scope.
- **Sim-to-real F/T mismatch.** The wrist F/T noise distribution in MuJoCo is rougher than what the real UR5e reports; the contact gate was hand-tuned on hardware. Calibrating the sim noise from a few minutes of real data would close this gap.
- **Legacy ROS 1 trajectory replay.** The IK-pipeline trajectory is currently replayed on the real arm through a ROS 1 publisher; the residual node is ROS 2.
- **Learned port-pose substitution.** A learned port-pose head trained on synthetic + real RGB would generalize better than pink-tape fiducials in cluttered scenes.

Acknowledgment

We thank Prof. Timothy Bretl, who taught the AE598 Advanced Robotic Planning course at UIUC under which this project was carried out, and the rest of the AE598 teaching staff for their guidance. We also thank the maintainers of the DeepMind MuJoCo Menagerie for the UR5e and Robotiq 2F-85 assets, the Stable-Baselines3 maintainers, the pinocchio team for the QP-IK solver, and the Universal Robots ROS 2

driver maintainers for the open-source driver that made the deploy path possible.

Use of Large Language Models

Claude (Anthropic) was used to draft initial section text and to revise wording for clarity. All technical content, design decisions, and results are the authors' own.

Code and Data Availability

Source code, the trained residual policy checkpoint, the MuJoCo USB-insertion scene, the stereo CV tracker, and the ROS 2 deploy package are publicly available at https://github.com/het915/precise_vla_manipulation. The repository's README documents the full reproduction workflow (training, evaluation, calibration, and the 4-terminal real-robot bring-up).

References

[1] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proc. ICML*, 2018.

[2] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[3] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ IROS*, 2012, pp. 5026–5033.

[4] M. J. Kim *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," *arXiv preprint arXiv:2406.09246*, 2024.

[5] A. Brohan *et al.*, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," in *Proc. CoRL*, 2023.

[6] T. Johannink *et al.*, "Residual Reinforcement Learning for Robot Control," in *Proc. IEEE ICRA*, 2019, pp. 6023–6029.

[7] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual Policy Learning," *arXiv preprint arXiv:1812.06298*, 2018.

[8] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. Aparicio Ojea, E. Solowjow, and S. Levine, "Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards," in *Proc. IEEE/RSJ IROS*, 2020, pp. 5548–5555.

[9] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep Reinforcement Learning for High Precision Assembly Tasks," in *Proc. IEEE/RSJ IROS*, 2017, pp. 819–825.

[10] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks," in *Proc. IEEE ICRA*, 2019, pp. 8943–8950.

[11] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *Proc. IEEE/RSJ IROS*, 2017, pp. 23–30.

[12] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *Proc. IEEE ICRA*, 2018, pp. 3803–3810.

[13] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[14] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.

[15] J. Carpentier *et al.*, "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *Proc. IEEE/SICE SII*, 2019, pp. 614–619.

[16] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Trans. ASME J. Dynamic Systems, Measurement, and Control*, vol. 108, no. 3, pp. 163–171, 1986.

[17] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022.

[18] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. ICML*, 2009, pp. 41–48.

[19] M. Towers *et al.*, "Gymnasium: A Standard Interface for Reinforcement Learning Environments," *arXiv preprint arXiv:2407.17032*, 2024.

[20] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proc. ICML*, 2018, pp. 1587–1596.