

PAKT: Physically-Aligned Kinesthetic Teaching for Reinforcement Learning

Anonymous Author(s)

Affiliation

Address

email

Abstract: Real-world reinforcement learning (RL) systems still struggle with the demands of contact-rich industrial manipulation, including achieving micrometer-level precision, 99+% success rates, and human-level cycle times. Although off-policy algorithms can improve performance by leveraging demonstrations and interventions, a key bottleneck is the lack of an intuitive interface for collecting such guidance while complying with constraints of the physical system and the policy. We propose PAKT, a framework for kinesthetic teaching in real-world RL. As opposed to teleoperation approaches, PAKT relies on kinesthetic guidance, which is widely used in industry. However, a critical weakness of kinesthetic guidance is the possibility for the operator to move the robot along trajectories (e.g., velocities, accelerations, jerk) that the robot and/or policy cannot physically reproduce. Using PAKT, operators guide the robot through admittance control which constrains the human input thus ensuring that collected trajectories remain feasible for the policy and the overall system. To support this teaching interface with an appropriate execution layer, PAKT adds a high-performance control stack that maps low-frequency RL actions to high-frequency torque commands. It consists of a reference generator and subsequent impedance controller, where the reference generator preserves the tracking performance of the impedance controller while improving contact handling and producing smoother policy actions. Across four difficult insertion and industrial assembly benchmarks, including a complex datacenter compute tray, PAKT reduces cycle time by 23%–48% and human intervention by 62%–86%. These cross-task results demonstrate that PAKT is a practical step toward deploying robot learning methodologies in production, supporting intuitive data collection, safe execution, and higher-throughput deployments. Project website: <https://pakt-website.github.io/pakt-website>

Keywords: real-world reinforcement learning, contact-rich manipulation, human-in-the-loop learning, kinesthetic teaching, impedance and admittance control

1 Introduction

High-precision, contact-rich industrial manipulation remains difficult to automate with current robot learning systems. Tasks such as electronics assembly require tight tolerances, high success rates, short cycle times, and reliable behavior under contact. Reinforcement learning (RL) is attractive because it can adapt directly from real-world interaction, but practical deployment is still limited by sample efficiency [1], robustness requirements, and the difficulty of transferring precise contact-rich behavior from simulation to hardware [2, 1, 3, 4, 5]. As a result, real-world demonstrations and human interventions remain central to many successful systems [6, 1].

A key bottleneck is the interface through which this human guidance is collected. Teleoperation devices can provide useful demonstrations, e.g. for pick-and-place-style tasks [7], but they often impose unintuitive mappings, require operator training [8], or add dedicated hardware [9, 10, 11]. Kinesthetic teaching, where the operator physically guides the robot, is more natural and widely used in industry. However, naive kinesthetic teaching introduces a critical mismatch for reinforcement learning: the human may move the robot along trajectories with velocities, accelerations, or jerks

that the learned policy and downstream controller cannot reproduce. Demonstrations that are easy for a human to provide are therefore not necessarily feasible for the policy to imitate [12].

We introduce PAKT, a framework for physically aligned kinesthetic teaching in real-world RL, see Fig. 1. It is a continuation of HIL-SERL, introduced in [6], which showed that human interventions can significantly accelerate RL training which was demonstrated with a teleoperation device. In contrast, PAKT lets operators guide the robot directly through constrained admittance control, while ensuring that the resulting trajectories remain compatible with the same policy parameterization and control stack limits used during autonomous policy execution. This makes kinesthetic demonstrations not only intuitive to collect, but also reproducible by the learned policy.

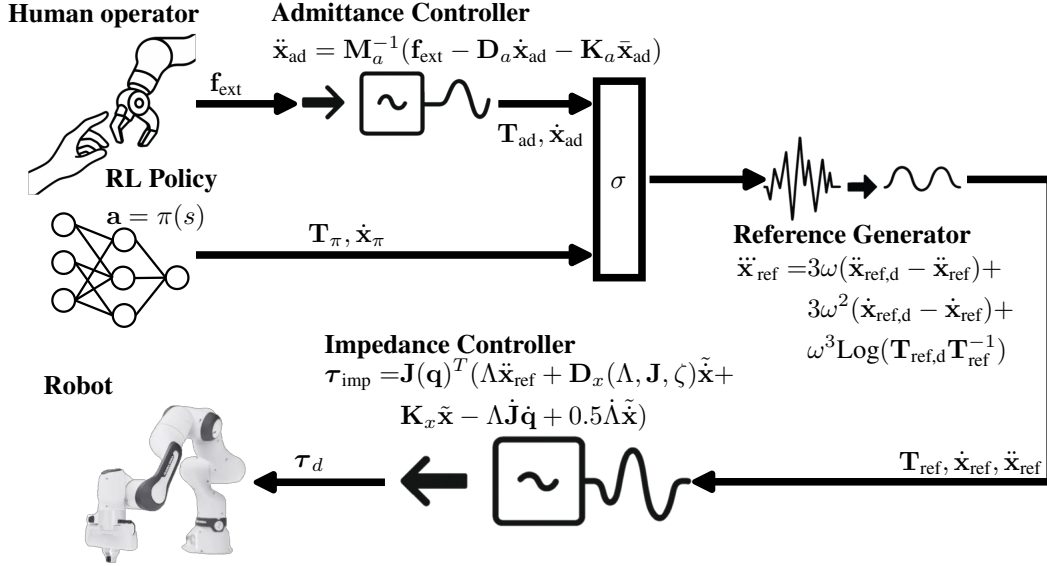


Figure 1: Overview of PAKT. In policy mode, policy actions are sent directly to the reference generator, whereas during guiding mode, human-applied forces first pass through an admittance controller. The reference generator outputs a filtered, limit-compliant command for the impedance controller, which computes the final torque command. The policy runs at 10 Hz and the downstream controller stack at 1000 Hz.

To support this interface, PAKT couples the teaching layer with a high-performance control stack. Low-frequency RL actions are passed through a reference generator that enforces the same limits as the admittance controller before being tracked by a fully compensated Cartesian impedance controller running at high frequency. This design preserves accurate tracking, improves contact handling, and smooths policy execution without changing the high-level learning algorithm.

We evaluate PAKT on four contact-rich insertion and industrial assembly tasks, including RAM insertion and datacenter compute-tray assembly. Compared with the HIL-SERL baseline [6] which uses a SpaceMouse interface [13], PAKT reduces cycle time by 23 to 48 percent, human intervention by 62 to 86 percent, and initial demonstration time by 15 to 34 percent. These results show that demonstration interfaces and downstream control are not peripheral implementation details, but core determinants of real-world RL performance.

2 Related Work

Our framework sits at the intersection of real-world robotic RL, smooth motion generation, and human-in-the-loop learning [14, 1, 15, 16]. The first line of related work focuses on control. Many successful real-world systems are deployed on position-controlled or otherwise simplified low-level interfaces [14, 1, 17], but contact-rich tasks often benefit from torque-controlled platforms and stronger downstream control stacks [18, 19, 20, 21]. For physical interaction, impedance control [22, 23] and admittance control [24] remain standard tools. In RL, however, the downstream controller is often reduced to the simplest approach: pose-error and velocity feedback with manually

70 tuned stiffness and damping. We instead integrate a controller with desired-motion tracking, dynam-
 71 ics compensation, and automatic damping design [25]. These ensure that the robot can accurately
 72 follow a desired motion and is always optimally damped in Cartesian space.

73 A second related line of work addresses the mismatch between policy outputs and smooth robot mo-
 74 tion. RL actions are often jerky [26, 27], motivating remedies such as smoothness-aware exploration
 75 and action regularization [26, 28]. By contrast, classical robotics routinely relies on jerk-bounded
 76 or online trajectory generation to satisfy kinematic limits while preserving execution quality [29].
 77 Recent work has started to combine RL with jerk-bounded trajectory generators explicitly [30]. Our
 78 reference generator follows the same overall motivation, but is intentionally lightweight and inte-
 79 grated directly upstream of the impedance controller.

80 A third line of work studies how human assistance can improve data collection and policy correction.
 81 Human intervention can substantially accelerate real-world RL training [6], and existing interfaces
 82 span 3D mice, modular teleoperation systems, and other operator-facing control devices [31, 7, 13].
 83 Concurrent with our work, CR-DAGger uses compliance control to collect on-policy human correc-
 84 tions and trains a residual policy on top of a pretrained base policy [32]. In contrast, PAKT targets
 85 real-world RL directly and couples policy-aligned kinesthetic teaching with a shared downstream
 86 control stack rather than residual DAGger-style policy correction. An important difference is that
 87 PAKT pauses the policy and allows the operator to perform fine-grained interventions while CR-
 88 DAGger requires them to act concurrently with the active policy. More recent systems also explore
 89 vision-based or low-cost leader-follower teleoperation and demonstration interfaces [33, 11, 10, 12].
 90 Kinesthetic teaching has likewise been studied directly as a user-facing programming interface for
 91 redundant manipulators [34]. We build on this literature by using admittance control [24] to translate
 92 human-applied forces into motion while keeping demonstrations inside the same kinematic limits
 93 and action parameterization seen by the policy and downstream control stack.

94 3 Methods

95 3.1 Reinforcement Learning

96 Each manipulation task is modeled as a continuous-control Markov Decision Process (MDP)
 97 $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \rho, \mathcal{P}, r, \gamma\}$ where $s_t \in \mathcal{S}$ denotes the state, $a_t \in \mathcal{A}$ the action, ρ the initial state
 98 distribution, $\mathcal{P}(s_{t+1}|s_t, a_t)$ the transition dynamics, and $r_t(s_t, a_t)$ the task reward. The parame-
 99 terized policy $\pi_\theta(a_t|s_t)$ induces trajectories $\tau = (s_0, a_0, \dots, s_H, a_H)$ and aims to maximize the
 100 expected discounted return over horizon H as $J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^H \gamma^t r(s_t, a_t) \right]$.

101 The observation space includes two RGB images, Cartesian pose with Euler angle representation
 102 (zyx convention) for orientation. Actions consist of 6-DoF Cartesian delta poses. Similar to [6],
 103 commands are provided to the control stack at 10 Hz. We keep this update rate for direct compara-
 104 bility with the baseline.

105 We use reinforcement learning with prior data (RLPD) [35] to find the optimal policy. RLPD extends
 106 soft-actor critic (SAC) by combining two replay buffers $\mathcal{D}_{\text{demo}}$ and $\mathcal{D}_{\text{rl}}$. $\mathcal{D}_{\text{demo}}$ contains human
 107 demonstration data and $\mathcal{D}_{\text{rl}}$ contains experience collected by the robot during training. At each
 108 iteration, a minibatch $B \subseteq \mathcal{D}_{\text{demo}} \cup \mathcal{D}_{\text{rl}}$ is sampled to update the critic and actor:

$$L_Q = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim B} \left[\left(Q_\phi(s_t, a_t) - \left[r_t + \gamma \left(\min_i Q_{\bar{\phi}_i}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\theta(a_{t+1}|s_{t+1}) \right) \right] \right)^2 \right], \quad (1)$$

$$L_\pi = \mathbb{E}_{\substack{s_t \sim B, \\ a_t \sim \pi_\theta(\cdot|s_t)}} \left[\alpha \log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t) \right], \quad (2)$$

109 where L_Q is the critic loss, L_π the actor loss, Q_ϕ the soft Q-function, $Q_{\bar{\phi}_i}$ the target critic network,
 110 $a_{t+1} \sim \pi_\theta(\cdot|s_{t+1})$, and α is an entropy temperature automatically tuned to maintain a target policy
 111 entropy. Sampling equally from $\mathcal{D}_{\text{demo}}$ and $\mathcal{D}_{\text{rl}}$ ensures stable bootstrapping from human data and
 112 progressive adaptation to self-collected experience.

3.1.1 Human-in-the-Loop Corrections

A human operator supervises execution and can intervene via a 6-DoF input device or through kinesthetic guidance whenever unsafe or suboptimal behavior arises. During intervention, robot control is overridden by human teleoperation, and all state–action–reward tuples are logged as corrective data. Once intervention ends, these trajectories are added to both $\mathcal{D}_{\text{demo}}$ and $\mathcal{D}_{\text{rl}}$ for subsequent training. This mechanism allows continuous refinement without resetting the learning process.

3.2 Reward and Success Classification

We use a binary reward derived from a learned success classifier $f_\psi(s)$ and a sigmoid function so that $r(s, a) = 1$ for $\text{sigmoid}(f_\psi(s)) > 0.85$ and 0 otherwise. More information on this can be found in Appendix C.

3.3 Controller

The input to our control stack is either (1) a desired Cartesian delta pose $\mathbf{T}_a$ from the RL policy or (2) an external wrench $\mathbf{f}_{\text{ext}}$ induced by the human operator if guide mode is activated. In Case 1, the input goes directly to a reference generator, and in Case 2, the input goes to an admittance controller first. The output of the admittance controller is then passed to the reference generator. Next, the output of the reference generator is forwarded to the impedance controller, which calculates torque commands for the robot. Figure 1 provides a visualization of the stack. Note that we omit the dependence on t in the following equation for the sake of clarity.

$\mathbf{T}_a$ is composed with the current pose of the end effector as $\mathbf{T}_\pi = \mathbf{T}_{\text{EE}}\mathbf{T}_a$ where $\mathbf{T}_\pi$ is the new desired pose expressed in world frame, $\mathbf{T}_{\text{EE}}$ is the current end effector pose expressed in world frame, and the policy action $\mathbf{T}_a$ is expressed in body frame. Furthermore, we estimate the desired twist of the policy as $\dot{\mathbf{x}}_\pi = k_v \text{Log}(\mathbf{T}_\pi \mathbf{T}_{\text{EE}}^{-1})$ where Log is the logarithmic map and $k_v = \frac{1}{\Delta t}$ is a proportional factor. Note that we set $\ddot{\mathbf{x}}_\pi = \mathbf{0}$ in our implementation, since its estimation is too noisy.

When guiding mode is active, the admittance controller provides the desired pose, twist, and acceleration to the reference generator instead of the policy. The admittance controller is defined as

$$\ddot{\mathbf{x}}_{\text{ad}} = \mathbf{M}_a^{-1}(\mathbf{f}_{\text{ext}} - \mathbf{D}_a \dot{\mathbf{x}}_{\text{ad}} - \mathbf{K}_a \tilde{\mathbf{x}}_{\text{ad}}) \quad (3)$$

where M_a is the virtual mass matrix, D_a the virtual damping matrix, K_a is the virtual stiffness matrix, and $\tilde{\mathbf{x}}_{\text{ad}} = \text{Log}(\mathbf{T}_{\text{ad}} \mathbf{T}_{\text{EE}}^{-1})$ is the pose error. Note that we set $K_a = \mathbf{0}$, which means that the robot can be freely guided through Cartesian space.

Then, we retrieve the setpoint for the reference generator:

$$\dot{\mathbf{x}}_{\text{ad}} = \int \ddot{\mathbf{x}}_{\text{ad}} dt, \quad \mathbf{T}_{\text{ad}} = \exp(\hat{\mathbf{V}} \Delta t) \mathbf{T}_{\text{ad}}^{\text{prev}}, \quad \hat{\mathbf{V}} = \begin{bmatrix} [\boldsymbol{\omega}]_x & \mathbf{v} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (4)$$

where $\dot{\mathbf{x}}_{\text{ad}}$ is the integrated twist, $\mathbf{T}_{\text{ad}}$ the pose (where the superscript prev indicates the pose from the previous step), $[\boldsymbol{\omega}]_x$ is the skew-symmetric matrix, and $\mathbf{v}$ the linear velocity.

The reference generator is a critically-damped, third-order filter with the general dynamics

$$\ddot{\mathbf{x}}_{\text{ref}} = a_2(\ddot{\mathbf{x}}_{\text{ref,d}} - \ddot{\mathbf{x}}_{\text{ref}}) + a_1(\dot{\mathbf{x}}_{\text{ref,d}} - \dot{\mathbf{x}}_{\text{ref}}) + a_0 \text{Log}(\mathbf{T}_{\text{ref,d}} \mathbf{T}_{\text{ref}}^{-1}) \quad (5)$$

where

$$(\mathbf{T}_{\text{ref,d}}, \dot{\mathbf{x}}_{\text{ref,d}}, \ddot{\mathbf{x}}_{\text{ref,d}}) = \begin{cases} (\mathbf{T}_{\text{ad}}, \dot{\mathbf{x}}_{\text{ad}}, \ddot{\mathbf{x}}_{\text{ad}}), & \sigma = 1, \\ (\mathbf{T}_\pi, \dot{\mathbf{x}}_\pi, \ddot{\mathbf{x}}_\pi), & \sigma = 0, \end{cases} \quad (6)$$

Here, $\mathbf{T}_{\text{ref,d}}$, $\dot{\mathbf{x}}_{\text{ref,d}}$, and $\ddot{\mathbf{x}}_{\text{ref,d}}$ denote the desired pose, twist, and acceleration supplied to the reference generator. Moreover, $\sigma \in \{0, 1\}$ indicates the control mode: $\sigma = 1$ activates guiding mode, whereas $\sigma = 0$ means that the policy drives the robot.

We choose a triple pole at $-\omega$, the bandwidth, which yields the critically-damped system

$$\ddot{\mathbf{x}}_{\text{ref}} = 3\omega(\ddot{\mathbf{x}}_{\text{ref,d}} - \ddot{\mathbf{x}}_{\text{ref}}) + 3\omega^2(\dot{\mathbf{x}}_{\text{ref,d}} - \dot{\mathbf{x}}_{\text{ref}}) + \omega^3 \text{Log}(\mathbf{T}_{\text{ref,d}} \mathbf{T}_{\text{ref}}^{-1}). \quad (7)$$

150 $\ddot{\mathbf{x}}_{\text{ref}}$ is integrated three times to obtain $\mathbf{T}_{\text{ref}}$ while we apply user-provided limits $\ddot{\mathbf{x}}_{\text{lim}}, \dot{\mathbf{x}}_{\text{lim}}, \mathbf{x}_{\text{lim}}$ at
 151 each integration step:

$$\ddot{\mathbf{x}}_{\text{ref}} = \text{clip}(\ddot{\mathbf{x}}_{\text{ref}}, \ddot{\mathbf{x}}_{\text{lim}}), \quad \dot{\mathbf{x}}_{\text{ref}} = \text{clip}(\dot{\mathbf{x}}_{\text{ref}}, \dot{\mathbf{x}}_{\text{lim}}), \quad \mathbf{x}_{\text{ref}} = \text{clip}(\mathbf{x}_{\text{ref}}, \mathbf{x}_{\text{lim}}) \quad (8)$$

152 The reference generator’s output is then passed to the Cartesian impedance controller. Our controller
 153 stack assumes the robot’s dynamics as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_d + \boldsymbol{\tau}_{\text{ext}}, \quad (9)$$

154 where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the Coriolis matrix, $\mathbf{g}$ are the torques caused by gravity, $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ are
 155 joint position, velocity, and acceleration, $\boldsymbol{\tau}_d$ is the desired torque, and $\boldsymbol{\tau}_{\text{ext}}$ is the measured external
 156 torque. Note that for clarity we dropped the dependency on time.

157 We use an impedance controller with the equation

$$\boldsymbol{\tau}_{\text{imp}} = \mathbf{J}(\mathbf{q})^T \left[\Lambda \ddot{\mathbf{x}}_{\text{ref}} + \mathbf{D}_x(\Lambda, \mathbf{K}_x, \zeta) \tilde{\dot{\mathbf{x}}} + \mathbf{K}_x \tilde{\mathbf{x}} - \Lambda \dot{\mathbf{J}}(\dot{\mathbf{q}})\dot{\mathbf{q}} + 0.5\dot{\Lambda} \tilde{\mathbf{x}} \right], \quad (10)$$

158 where Λ is the Cartesian mass matrix, $\mathbf{J}$ is the Jacobian, $\mathbf{K}_x$ is the Cartesian stiffness matrix, $\mathbf{D}_x$
 159 is the Cartesian damping matrix with damping design based on double-diagonalization according
 160 to [25] with damping factors ζ , and $\tilde{\mathbf{x}} = \text{Log}(\mathbf{T}_{\text{ref}}\mathbf{T}_{\text{EE}}^{-1})$ is the Cartesian space pose error. Double
 161 diagonalization transforms the Cartesian mass and stiffness matrices into modal space, where the
 162 damping matrix can be designed as a diagonal matrix and then transformed back into Cartesian
 163 space. Furthermore, $\Lambda \dot{\mathbf{J}}\dot{\mathbf{q}}$ is the geometric acceleration compensation and $0.5\dot{\Lambda}$ ensures passivity.
 164 For a more in-depth discussion of this controller type, the reader may refer to [25].

165 Furthermore, we use a nullspace controller to keep the elbow of our redundant robot upright. It is
 166 defined as

$$\boldsymbol{\tau}_{\text{null}} = \mathbf{N}^T(\mathbf{D}_{\text{null}}\tilde{\dot{\mathbf{q}}} + \mathbf{K}_{\text{null}}\tilde{\mathbf{q}}) \quad (11)$$

167 where $\mathbf{D}_{\text{null}}$ is a damping matrix, $\mathbf{K}_{\text{null}}$ a stiffness matrix, $\tilde{\mathbf{q}} = \mathbf{q}_{\text{d,null}} - \mathbf{q}$ and $\tilde{\dot{\mathbf{q}}} = \dot{\mathbf{q}}_{\text{d,null}} - \dot{\mathbf{q}}$ denote
 168 the nullspace position and velocity errors, respectively, and $\mathbf{N} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{J}^T(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)^{-1}\mathbf{J}$ is the
 169 dynamically consistent nullspace projector [21].

170 Finally, the robot receives the resulting torque including compensation of Coriolis and gravity:

$$\boldsymbol{\tau}_d = \boldsymbol{\tau}_{\text{imp}} + \boldsymbol{\tau}_{\text{null}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \quad (12)$$

171 In Sec. 4.2 we compare this controller stack to variants commonly used in the RL literature. Table 1
 172 gives an overview with the respective equations and a comment that describes the main difference.
 173 C3 closely matches a controller structure that is common in the RL literature.

Table 1: Variations of controllers used in RL. For C5, $\mathbf{e}_{\text{lim}}$ denotes the clipped error. C5 is the controller used in our baseline [6]

ID	Equation	Comment
C1	Eq. 10	Ours
C2	$\mathbf{f}_{\text{imp}} = -\mathbf{D}_x(\mathbf{M}, \mathbf{J})\dot{\mathbf{x}} + \mathbf{K}_x \tilde{\mathbf{x}}$	No inertia compensation or desired velocity
C3	$\mathbf{f}_{\text{imp}} = -\mathbf{D}_x \dot{\mathbf{x}} + \mathbf{K}_x \tilde{\mathbf{x}}$	No inertia compensation or desired velocity, and no damping design
C4	$\mathbf{f}_{\text{imp}} = -\mathbf{D}_x \dot{\mathbf{x}} + \mathbf{K}_x \tilde{\mathbf{x}} + \int k_i \tilde{\mathbf{x}} dt$	No inertia compensation or desired velocity, and no damping design, but with integrator
C5	$\mathbf{f}_{\text{imp}} = -\mathbf{D}_x \dot{\mathbf{x}} + \mathbf{K}_x \text{clip}(\tilde{\mathbf{x}}, \mathbf{e}_{\text{lim}})$	No inertia compensation or desired velocity, and no damping design, but with error clipping

174 4 Experiments

175 4.1 Setup

176 Our experimental setup consists of a Franka Emika robot [18] equipped with a Franka Hand end
 177 effector. The complete controller stack runs on an NVIDIA Jetson Orin AGX with a real-time
 178 kernel, while the policy and success classifier run on a desktop PC with an NVIDIA RTX 4090
 179 GPU. Attached to the robot end effector are two Intel RealSense D405 cameras.

4.2 Controller Stack Comparison

Our first set of experiments compares the proposed control stack and the controller variants listed in Tab. 1 in terms of tracking and contact response. Cartesian pose commands are issued at 10 Hz, matching the RL experiments. For tracking, the robot follows a square in the xy-plane with sinusoidal motion in z. For contact, it moves toward a hard surface in the z-direction until the observed external force exceeds 10 N.

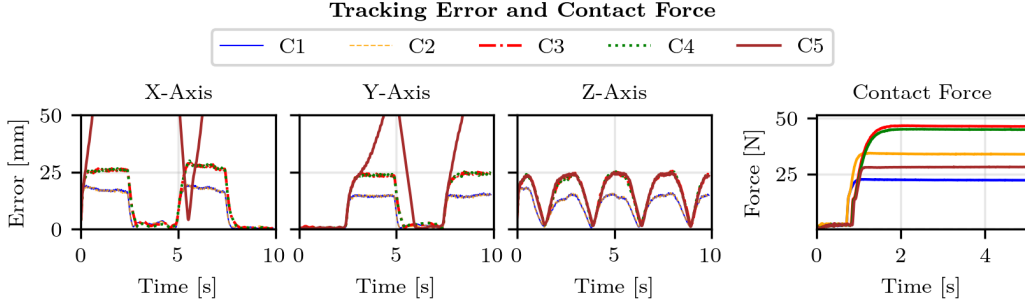


Figure 2: Tracking error for the tested control stacks in the x, y, and z dimensions. Tracking errors are low-pass filtered at 7 Hz for visibility. We truncate C5 to maintain visibility for the other time series, since error clipping makes C5 lag behind the reference trajectory and therefore produce much larger absolute tracking errors than the other stacks. Note that C1 and C2 overlap in the tracking error plots, as well as C3 and C4.

The left three plots of Fig. 2 show that C2, the closest ablation of C1, achieves nearly identical tracking error. This indicates that the upstream reference generator does not degrade the free-space tracking performance of the impedance controller. By contrast, the simpler controller variants C3/C4 have substantially larger tracking error, with the integrator in C4 offering little benefit in this dynamic setting. C5, the clipped-error baseline used in the RL comparison, performs worst because error clipping saturates the corrective action once the pose error exceeds the clipping threshold, causing large lag. The right plot shows contact forces during the hard-contact test. Reaction forces can be much higher than the 10 N stop criterion because the controller reacts at 10 Hz, matching the RL policy frequency. C1 yields the smallest peak force overall. C2 reaches noticeably higher peaks. The simpler controllers C3/C4 reach up to $2\times$ higher peak forces, while C5 reduces force only by clipping error at the cost of severe tracking degradation. Table 2 reports the numeric values. The introduced control stack C1 demonstrates low tracking errors (together with C2) similar to C3, C4, and C5 indicating a significant benefit of automatic damping in this setting. At the same time it reduces impact forces compared to the other control stacks.

Table 2: Average tracking error and contact force for the tested controller stacks.

Controller	Average tracking error [mm]	Contact force [N]
C1 (PAKT)	16.8, 14.1, 9.6	22.5
C2	16.6, 14.1, 9.6	33.9
C3	24.8, 22.5, 16.2	46.5
C4	24.8, 22.4, 16.0	45.0
C5 (HIL-SERL)	100.5, 37.2, 16.8	28.2

4.3 RL Experiments

Our second set of experiments compares PAKT to the HIL-SERL baseline [6] and additionally to a hybrid variant that combines our controller stack with the baseline SpaceMouse interface. This three-way comparison serves as a structured ablation: baseline versus hybrid isolates the controller stack, while hybrid versus PAKT isolates kinesthetic guidance under the same controller.

Task details are given in Appendix A. We consider four tasks: peg insertion from the Functional Manipulation Benchmark (FMB) [36], RAM insertion, limit fixture assembly, and busbar assembly.

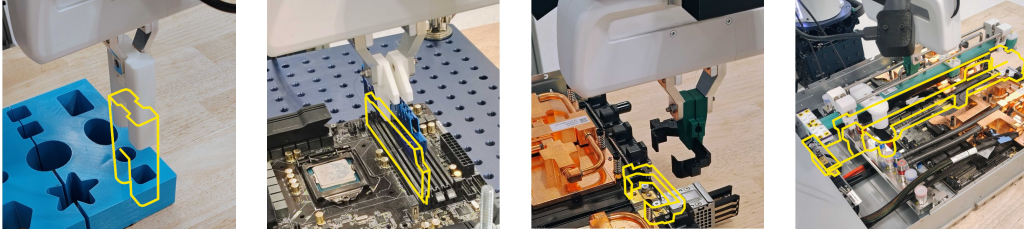


Figure 3: The four insertion tasks used in the RL experiments with outlines indicating the insertion location: a) FMB peg insertion, b) RAM insertion, c) limit fixture assembly, and d) busbar assembly. We show a more detailed overview in Appendix A and in the supplementary video.

All four tasks follow the same protocol. We first train a success classifier (Appendix C), then collect 20 demonstrations with either a SpaceMouse or kinesthetic guidance (Appendix E), train the policy with interventions, and finally evaluate the learned policy in 100 executions from random initial states. Before data collection, the operator was familiarized with both the SpaceMouse and kinesthetic guidance interfaces. Within each task, all compared methods share the same demonstration budget, reset protocol, and training procedure. For statistical verification, we repeated the RAM insertion task, as the most difficult of the four, with five random seeds per method.

The results across the four tasks show consistent benefits of PAKT. Figure 4 shows reward, cycle time, and intervention rate over training, and Table 3 reports demonstration time, final success rate, average cycle time, and cumulative intervention count. For RAM insertion, Fig. 4 shows the mean and standard deviation across five training runs with different random seeds to establish statistical robustness of our approach, while the other tasks are shown as single runs. Relative to the baseline, both PAKT and the hybrid variant reduce average cycle time, with cross-task reductions of 23–48%. Furthermore, PAKT decreases demonstration time by 15–34%, and cumulative intervention count by 62–86% across tasks. This matches our motivation for PAKT: demonstrations become simpler and more intuitive while remaining constrained to trajectories that are reproducible under the policy parameterization.

Table 3: Results on the four tasks for the baseline and PAKT. Cumulative intervention count is the discrete area under the intervention-rate curve over training steps. We show the values for the HIL-SERL baseline (B), the hybrid (H), and PAKT (P).

Task	Total demo time [s] B/H/P	Success rate B/H/P	Average cycle time [s] B/H/P	Cumulative Intervention Count B/H/P
Peg	216/199/183	1.0/1.0/1.0	3.88/2.25/2.17	6276/2690/980
RAM stick	243/219/182	0.88/0.99/0.98	5.2/4.07/4.0	8780/5205/3325
Limit fixture	209/194/177	0.99/0.98/0.99	3.01/1.87/1.92	5937/2026/1342
Busbar	221/187/146	0.85/0.98/0.99	3.91/2.09/2.03	9286/2920/1327

5 Limitations

PAKT has limitations that offer promising directions for future work. First, kinesthetic guidance is not compatible with scene cameras, since the human operator would at least partially occlude them. We focus on settings in which this trade-off is acceptable, but future extensions could address it by post-processing to remove the human and inpaint the scene. Second, our current system cannot detect environment interaction forces while the human operator is guiding the robot, since these are estimated based on joint-torque sensors. Thus, external wrench and torques cannot be used in the observation space. Potential solutions are a dedicated F/T sensor or a bilateral teleoperation setup. Third, our experiments focus exclusively on single-arm insertion tasks. This regime is already a highly demanding testing ground, however, an important direction for future work is to extend the framework to a broader set of manipulation scenarios, including non-insertion tasks and bimanual manipulation. We also note that PAKT is restricted to torque-controlled robots. However, many widely available robots such as the Franka, Flexiv, and UR arms support torque control.

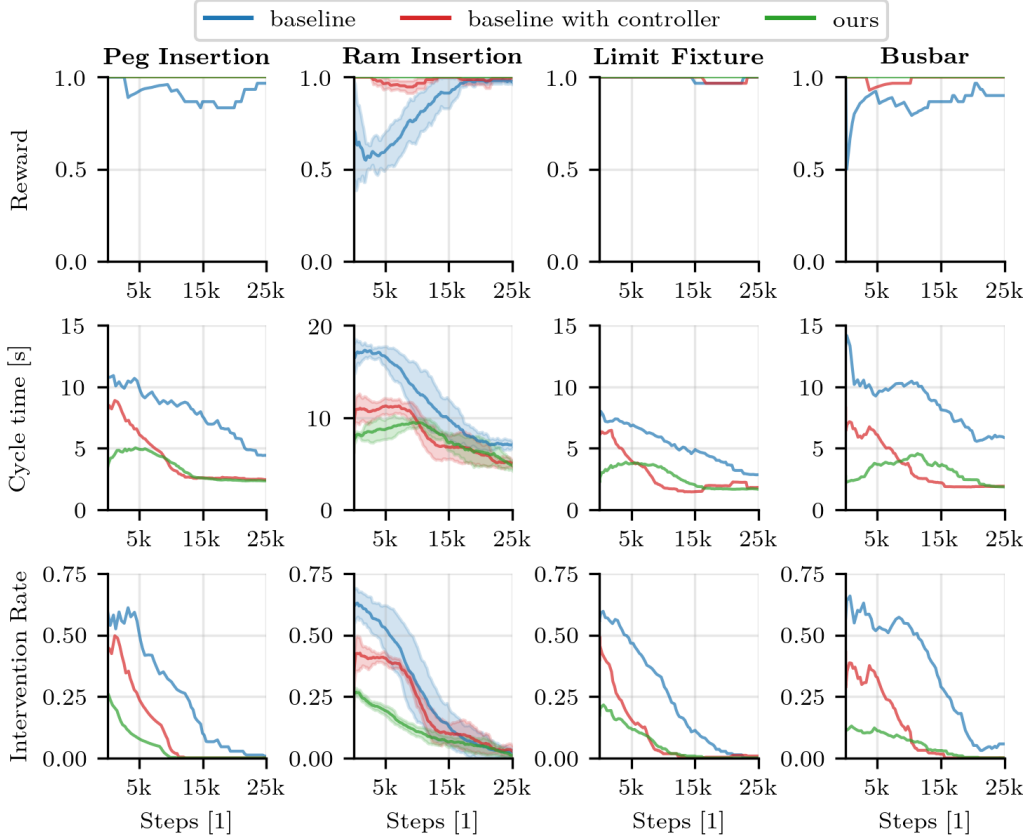


Figure 4: Reward, cycle time, and intervention rate over training for the baseline, the hybrid approach, and PAKT. For RAM insertion, the curves show the mean and standard deviation across five random seeds. The other tasks are shown as single training runs. **Top row:** PAKT and the hybrid have higher success rates throughout training compared to the baseline which can be attributed to the easier handling provided by the superior controller. **Middle row:** Cycle time of the hybrid and PAKT converges to lower values due to the controller, while PAKT has lower cycle times throughout as kinesthetic guidance enables more intuitive motions. **Bottom row:** PAKT has the lowest amount of human interventions due to kinesthetic guidance.

6 Conclusion

PAKT provides an intuitive and safe way to collect kinesthetic demonstrations, while ensuring that all demonstrated trajectories remain physically feasible for the downstream policy and the hardware. We achieve this through a tightly coupled controller stack: an admittance controller enables kinesthetic guidance in a way that is explicitly constrained by the same downstream dynamics, so demonstrations remain reproducible by the learned policy. Then a reference generator shapes policy actions or, depending on the mode, human inputs, to preserve velocity, acceleration, and jerk limits. This is passed to a Cartesian impedance controller which preserves accurate tracking.

Across common controller variants, PAKT delivers substantially better tracking and contact handling than typical RL control stacks while maintaining compliance with kinematic limits. On the evaluated manipulation tasks, it outperforms the baseline by reducing cycle time by 23%–48%, human interventions by 62%–86%, and initial demonstration time by 15%–34%. These gains show that low-level control design and human demonstration interfaces are not peripheral engineering choices, but core determinants of real-world RL performance.

Overall, PAKT advances a practical path from research prototypes to deployable industrial robotic learning systems.

References

- [1] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8(1):153–188, 2025.
- [2] D. Blanco-Mulero, O. Barbany, G. Alcan, A. Colomé, C. Torras, and V. Kyrki. Benchmarking the sim-to-real gap in cloth manipulation. *IEEE Robotics and Automation Letters*, 9(3):2981–2988, 2024.
- [3] E. Aljalbout, J. Xing, A. Romero, I. Akinola, C. R. Garrett, E. Heiden, A. Gupta, T. Hermans, Y. Narang, D. Fox, et al. The reality gap in robotics: Challenges, solutions, and best practices. *Annual Review of Control, Robotics, and Autonomous Systems*, 9, 2025.
- [4] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *Transactions on Automation Science and Engineering (T-ASE)*, 18(2):398–400, 2021.
- [5] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.
- [6] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.
- [7] S. Dass, W. Ai, Y. Jiang, S. Singh, J. Hu, R. Zhang, P. Stone, B. Abbatematteo, and R. Martín-Martín. Telemoma: A modular and versatile teleoperation system for mobile manipulation. *arXiv preprint arXiv:2403.07869*, 2024.
- [8] Z. Chen, Y. J. Yoon, R. Bautista-Montesano, Z. Zhao, A. Mandlekar, and J. Liu. TeleopLab: Accessible and intuitive teleoperation of a robotic manipulator for remote labs. *arXiv preprint arXiv:2509.05547*, 2025.
- [9] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.
- [10] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.
- [11] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.
- [12] 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. *arXiv preprint arXiv:2402.10329*, 2024.
- [13] A. Smith and M. Kennedy III. An augmented reality interface for teleoperating robot manipulators. *arXiv preprint arXiv:2409.18394*, 2024.
- [14] D. Han, B. Mulyana, V. Stankovic, and S. Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7):3762, 2023.
- [15] C. O. Retzlaff, S. Das, C. Wayllace, P. Mousavi, M. Afshari, T. Yang, A. Saranti, A. Angerschmid, M. E. Taylor, and A. Holzinger. Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities. *Journal of Artificial Intelligence Research (JAIR)*, 79:359–415, 2024.
- [16] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054, 2023.
- [17] Universal Robots A/S. Ur series – collaborative robots. <https://www.universal-robots.com/products/ur-series/>, 2026. Accessed: 2025-12-15.

- [18] S. Haddadin. The Franka Emika robot: A standard platform in robotics research. *Robotics & Automation Magazine (RAM)*, 2024.
- [19] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. C. L’Ecuyer. Kinova modular robot arms for service robotics applications. In *Rapid Automation: Concepts, Methodologies, Tools, and Applications*, pages 693–719. IGI global, 2019.
- [20] G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl. DLR’s torque-controlled light weight robot iii—are we reaching the technological limits now? In *Proc. International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1710–1716. IEEE, 2002.
- [21] A. Dietrich, C. Ott, and A. Albu-Schäffer. An overview of null space projections for redundant, torque-controlled robots. *International Journal of Robotics Research (IJRR)*, 34(11):1385–1400, 2015.
- [22] N. Hogan. Impedance control: An approach to manipulation: Part ii—implementation. *Journal of dynamic systems, measurement, and control*, 107(1):8–16, 1985.
- [23] C. Ott, A. Albu-Schaffer, A. Kugi, and G. Hirzinger. On the passivity-based impedance control of flexible joint robots. *Transactions on Robotics (T-RO)*, 24(2):416–429, 2008.
- [24] A. Q. Keemink, H. Van der Kooij, and A. H. Stienen. Admittance control for physical human–robot interaction. *The International Journal of Robotics Research (IJRR)*, 37(11):1421–1444, 2018.
- [25] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger. Cartesian impedance control of redundant robots: Recent results with the dlr-light-weight-arms. In *Proc. International Conference on Robotics and Automation (ICRA)*, volume 3, pages 3704–3709. IEEE, 2003.
- [26] A. Raffin, J. Kober, and F. Stulp. Smooth exploration for robotic reinforcement learning. In *Proc. Conference on Robot Learning (CoRL)*, pages 1634–1644. PMLR, 2022.
- [27] R. Yang, H. Wei, R. Zhang, Z. Feng, X. Chen, T. Li, C. Zhang, L. Zhao, J. Bian, X. Su, et al. Beyond human demonstrations: Diffusion-based reinforcement learning to generate data for vla training. *arXiv preprint arXiv:2509.19752*, 2025.
- [28] A. Raffin and F. Stulp. Generalized state-dependent exploration for deep reinforcement learning in robotics. *arXiv*, 2020.
- [29] J.-E. Lee, A. Bylard, R. Sun, and L. Sentis. On the performance of jerk-constrained time-optimal trajectory planning for industrial manipulators. In *Proc. International Conference on Robotics and Automation (ICRA)*, pages 9772–9778. IEEE, 2024.
- [30] S. A. A. Kolagar, M. H. Shahna, and J. Mattila. Combining deep reinforcement learning with a jerk-bounded trajectory generator for kinematically constrained motion planning. In *Proc. European Control Conference (ECC)*, pages 2507–2514. IEEE, 2025.
- [31] V. Dhat, N. Walker, and M. Cakmak. Using 3d mice to control robot manipulators. In *Proc. International Conference on Human-Robot Interaction*, pages 896–900. ACM/IEEE, 2024.
- [32] X. Xu, Y. Hou, C. Xin, Z. Liu, and S. Song. Compliant residual dagger: Improving real-world contact-rich manipulation with human corrections. *arXiv preprint arXiv:2506.16685*, 2025.
- [33] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.
- [34] S. Wrede, C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil. A user study on kinesthetic teaching of redundant robots in task and configuration space. *Journal of Human-Robot Interaction (JHRI)*, 2(1):56–81, 2013.

- 346 [35] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with
347 offline data. In *Proc. International Conference on Machine Learning (ICML)*, pages 1577–
348 1594. PMLR, 2023.
- 349 [36] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. FMB: a functional ma-
350 nipulation benchmark for generalizable robotic learning. *The International Journal of Robotics*
351 *Research (IJRR)*, 44(4):592–606, 2025.

A Experimental Setup

This appendix summarizes the exact task configuration, learning setup, reward implementation, and data-collection protocol used in the real-world experiments. Unless otherwise noted, all four tasks share the same hardware platform, policy-to-controller interface, and high-level training workflow. Where task-specific settings differ, we report them explicitly in the tables below.

Our experimental setup consists of a Franka Emika Robot [18] equipped with a Franka Hand end effector. The complete controller stack runs on an NVIDIA Jetson Orin AGX with a real-time kernel, while the policy and success classifier run on a desktop PC with an NVIDIA RTX 4090 GPU. Policy commands are sent at 10 Hz to the controller stack, while the admittance controller, reference generator, and impedance controller execute in a 1 kHz real-time loop on the Jetson. Attached to the robot end effector are two Intel RealSense D405 cameras with the same custom mount as in [6]. The Franka Hand is used only to establish and maintain the grasp for each episode; the studied tasks do not require regrasping or any form of in-hand manipulation. Figure 5 shows the full setup and Fig. 6 shows closeups of the respective tasks.

A.1 Shared Setup

Table 4 lists the setup details shared across all tasks. The policy observation and action interfaces are also shared across tasks: the policy observes two RGB images from wrist cameras Cartesian pose with Euler angle representation (zyx convention) for orientation, and it outputs 6-DoF Cartesian delta poses. The success-classifier inputs and environment settings remain task-dependent and are therefore reported separately in Table 5.

Table 4: Shared hardware and runtime details used across all tasks.

Parameter	Value
Robot platform	Franka Emika Robot with Franka Hand
Control compute	NVIDIA Jetson Orin AGX with L4T and PREEMPT_RT kernel
Policy / classifier compute	Desktop PC with Ubuntu 22 and NVIDIA RTX 4090
Policy command rate	10 Hz
Cameras	Two Intel RealSense D405 wrist cameras
Observation space	Two RGB wrist-camera images, Cartesian pose (zyx Euler convention), Cartesian velocity
Action space	6-DoF Cartesian delta pose
Raw camera resolution	1280×720
Network input resolution	128×128
Camera exposure	$40,000 \mu\text{s}$
Gripper usage	Objects are pre-grasped; regrasping is not required in the reported tasks

A.2 Peg Insertion

This task is taken from the Functional Manipulation Benchmark (FMB) [36]. The peg and its counterpart are 3D-printed with white material. The clearance is ≈ 2 mm on the diameter. The peg is initially grasped by the robot and retained during the experiment. The robot fingers are designed such that the peg cannot drop out of its grip, but it can move slightly in the end effector. The objective is to insert the peg into a matching hole on the benchmark board. See also Fig. 6.

A.3 RAM Insertion

This task follows the RAM insertion setting used in the original HIL-SERL work [6]. At the beginning of each episode, the robot has already grasped the RAM stick. The objective is to align the module with the correct socket and insert it until it is properly seated. Compared with peg insertion, this task is visually and mechanically more demanding because the insertion tolerances are tighter and partial insertion states are common. We consider an episode successful once the RAM stick is fully inserted into the designated slot. Note that we modified the clips at the sides of the slot on the mainboard to enable the robot to remove the RAM stick from its slot by itself.

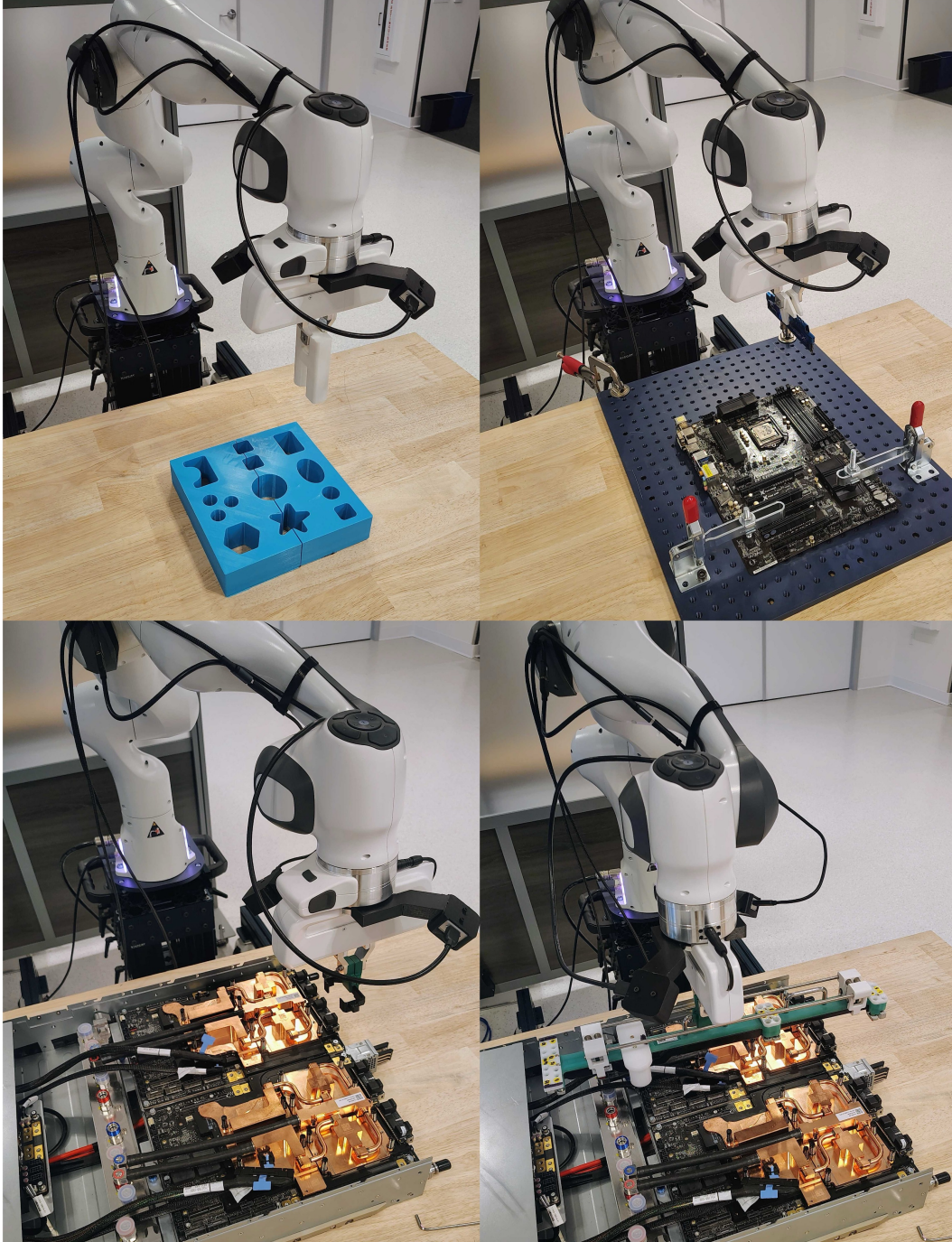


Figure 5: Experimental setup used for all real-world tasks.

386 **A.4 Limit Fixture**

387 The limit fixture task is taken from an existing high-value, real-world electronics manufacturing
 388 workflow. In this problem, the robot must insert a fixture into a tray that is used in a downstream
 389 testing step. The fixture is necessary for the subsequent step, in which a busbar is inserted. The
 390 task is representative of industrial contact-rich assembly, as successful completion requires accurate
 391 alignment and reliable insertion under limited clearances. Because the benchmark originates from a
 392 proprietary production process, we cannot disclose the full geometry or all process-specific details.

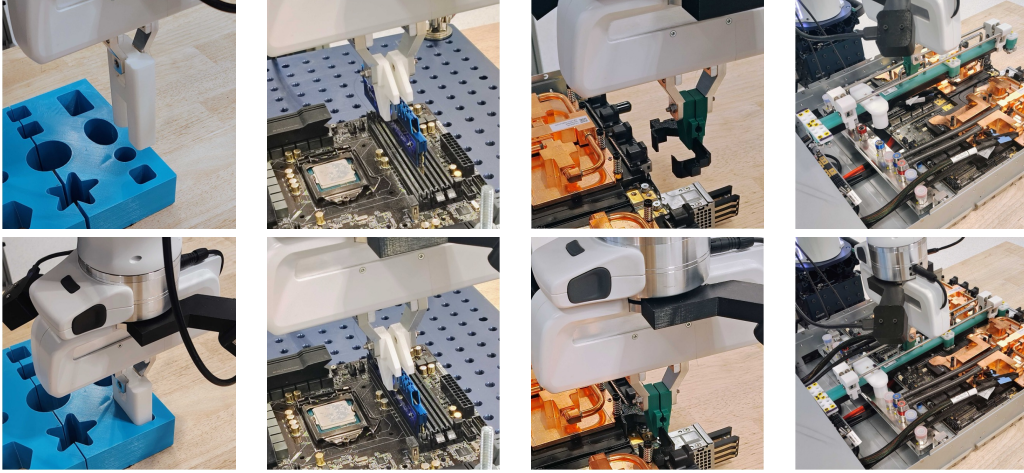


Figure 6: The four insertion tasks used in the RL experiments: a) FMB peg insertion, b) RAM insertion, c) limit fixture assembly, and d) busbar assembly.

A.5 Busbar

The busbar task is drawn from the same production workflow as the limit fixture assembly and is the direct follow-up step. Here, the robot inserts a busbar into its target location after the preceding fixture-related preparation step. The purpose of the busbar is to conduct electrical power across the tray. This benchmark is again contact-rich and requires accurate positioning during the final insertion phase. As with the limit fixture task, some setup details cannot be reported in full due to confidentiality constraints.

A.6 Task Configuration

Table 5 summarizes the episode length, action scales, and reset randomization for the four tasks studied in the paper.

Table 5: Task-specific configuration for the four RL tasks in the paper.

Task	Episode length	Action scale (translation, rotation)	xy reset	rz reset
Peg insertion	150	(0.01, 0.02)	± 0.02 m	± 0.05 rad
RAM insertion	200	(0.005, 0.02)	± 0.02 m	± 0.05 rad
Limit fixture	150	(0.01, 0.02)	± 0.02 m	± 0.05 rad
Busbar	150	(0.01, 0.02)	± 0.02 m	± 0.05 rad

B Policy and Training Details

This section records the reinforcement-learning configuration used across tasks. The defaults in this section are shared across tasks.

B.1 Training Defaults

B.2 Architecture Details

Table 7 summarizes the actor, critic, and optimization settings used for policy learning. Table 8 reports the vision encoder and image augmentation details.

Table 6: Shared policy and training details.

Parameter	Value
Learning algorithm	RLPD with SAC-style pixel agent
Batch size	256 total = 128 demo + 128 online RL
Critic-to-actor ratio	2
Discount factor γ	0.97
Maximum training steps	1,000,000 gradient steps
Replay buffer capacity	200,000 transitions
Random exploration steps	0
Training starts after	100 online transitions
Policy publication / update interval	Every 50 environment steps
Evaluation period	Every 2,000 training steps
Evaluation rollouts during training	5 trajectories
Final evaluation protocol	100 trials per task
Demonstration budget	20 demonstrations per method and task

Table 7: Actor, critic, and optimization details.

Parameter	Value
Policy hidden dimensions	[256, 256]
Policy activation	tanh
Policy output distribution	Gaussian with tanh squashing
Standard-deviation parameterization	exp
Standard-deviation range	$[10^{-5}, 5]$
Policy layer normalization	Enabled
Critic hidden dimensions	[256, 256]
Critic activation	tanh
Critic layer normalization	Enabled
Critic ensemble size	2 (clipped double- Q)
Temperature initialization	10^{-2}
Target entropy	negative half of action dimension (automatic)
Learning rate	3×10^{-4} for all networks
Polyak averaging coefficient τ	0.005
Reward bias	0.0

Table 8: Vision encoder and data augmentation details.

Parameter	Value
Encoder type	Pretrained frozen ResNet-10
Image pretraining	ImageNet
Pre-pooling	Enabled
Pooling method	Spatial learned embeddings
Number of spatial blocks	8
Bottleneck dimension	256
Network image size	128×128
Data augmentation	Batched random crop
Crop padding	4 pixels

C Reward and Success Classification

The main text describes a sparse binary reward signal, and the implementation matches that description across all four tasks. Each task uses a task-specific learned visual success classifier whose output is thresholded online to obtain the binary reward used for reinforcement learning. The classifier input views differ by task, but the reward construction itself is shared: for all four tasks, we use the binary reward $r(s, a) = 1$ if $\text{sigmoid}(f_\psi(s)) > 0.85$ and $r(s, a) = 0$ otherwise.

C.1 Classifier Details

Following the reward-design recipe of [6], we use a separate learned binary success classifier for each task. The classifier maps the current visual observation to a scalar success score $f_\psi(s)$, which is then thresholded to obtain the binary reward used during reinforcement learning. We keep one classifier per task rather than sharing a single model across all tasks, since the visual signatures of success differ substantially between peg insertion, RAM insertion, limit fixture assembly, and busbar assembly.

Table 9: Success-classifier architecture and optimization details.

Parameter	Value
Classifier inputs	Task-relevant wrist-camera images
Backbone	Pretrained frozen ResNet-10
Pooling	Spatial learned embeddings
Bottleneck dimension	256
Classification head	Two-layer MLP
Loss	Cross-entropy
Optimizer	Adam
Learning rate	3×10^{-4}
Optimization iterations	100
Reward threshold	$\text{sigmoid}(f_\psi(s)) > 0.85$

C.2 Classifier Dataset and Hard Negatives

The classifier dataset is collected before policy training by recording short teleoperated rollouts and storing the corresponding images together with binary success labels. During rollout, while the user is pressing a button, all frames are labelled as positives, while all remaining frames are treated as negatives. To make the classifier robust, we deliberately include difficult negative examples that are visually similar to success, for example states with incomplete insertion, asymmetric seating, contact at the wrong slot (in case of RAM insertion), or other near-goal misalignments. These hard negatives are important because they reduce false positives during online reward evaluation and make the sparse reward considerably more reliable. After training, the classifier is frozen and used only for reward computation; it is not optimized jointly with the policy during RL.

D Controller Stack Parameters

This section records the controller, guide-mode, and reference generator parameters used in the experimental comparison. To distinguish the HIL-SERL baseline from our controller stack, Table 10 contrasts the two impedance-layer parameterizations: the baseline uses a manually specified Cartesian damping matrix together with asymmetric error clipping, whereas our stack uses damping factors and no clipping. The damping factors are used during double diagonalization to calculate a damping matrix in modal space which is then transformed back into Cartesian space. The velocity/acceleration limits of our stack depend on the task-specific action scales listed in Table 5.

E Demonstration and Intervention Protocol

This section provides the operator-side data-collection protocol used for demonstrations and corrective interventions. We report it separately because these workflow details strongly affect real-world training time and usability, even though they are not part of the core control or learning equations. Unless otherwise stated, the same demonstration and intervention logic is used across all four tasks.

E.1 Demonstration Collection

Demonstrations are collected either with a SpaceMouse or through kinesthetic guidance. In both cases, the object is initially held in the gripper, and the reported tasks do not require regrasping. For each method and task, we collect 20 demonstrations before RL training. The operator then moves the

Table 10: Baseline-versus-ours comparison of the controller parameterization.

Parameter	Baseline	Ours
Cartesian translational stiffness	[3000, 3000, 3000] N/m	[3000, 3000, 3000] N/m
Cartesian rotational stiffness	[300, 300, 300] N m/rad	[300, 300, 300] N m/rad
Cartesian translational damping	[89, 89, 89] N s/m	N/A
Cartesian rotational damping	[7, 7, 7] N m s/rad	N/A
Cartesian translational damping factors	N/A	[0.7, 0.7, 0.7]
Cartesian rotational damping factors	N/A	[0.7, 0.7, 0.7]
Translational clip $(+x, +y, +z)$	$(7.5, 1.6, 5.5) \times 10^{-3}$	N/A
Translational clip $(-x, -y, -z)$	$(2.0, 1.6, 5.0) \times 10^{-3}$	N/A
Rotational clip $(+x, +y, +z)$	$(10, 25, 5) \times 10^{-3}$	N/A
Rotational clip $(-x, -y, -z)$	$(10, 25, 5) \times 10^{-3}$	N/A
Transl./rot. integral gains	0 / 0	0 / 0
Admittance mass translation M_a	N/A	[1, 1, 1] kg
Admittance mass rotation M_a	N/A	[0.1, 0.1, 0.1] kg m ²
Admittance damping translation D_a	N/A	[100, 100, 100] N s/m
Admittance damping rotation D_a	N/A	[20, 20, 20] N m s/rad
Admittance stiffness K_a	N/A	$\mathbf{0}_{6 \times 6}$
Reference generator frequency ω_c	N/A	30 Hz

Table 11: Task-specific reference-generator limits of our stack derived from the action scales in Table 5.

Task	Trans. vel. [min, max] [m/s]	Trans. acc. [min, max] [m/s ²]	Rot. vel. [min, max] [rad/s]	Rot. acc. [min, max] [rad/s ²]	Trans. jerk max [m/s ³]	Rot. jerk max [rad/s ³]
Peg insertion	[0.09, 0.10]	[0.90, 1.00]	[0.18, 0.20]	[1.8, 2.0]	10000.0	10000.0
RAM insertion	[0.045, 0.05]	[0.45, 0.50]	[0.18, 0.20]	[1.8, 2.0]	10000.0	10000.0
Limit fixture	[0.09, 0.10]	[0.90, 1.00]	[0.18, 0.20]	[1.8, 2.0]	10000.0	10000.0
Busbar	[0.09, 0.10]	[0.90, 1.00]	[0.18, 0.20]	[1.8, 2.0]	10000.0	10000.0

robot to the goal pose, either by using the SpaceMouse or, in the kinesthetic setting, by guiding the robot by hand while guide mode remains active throughout the demonstration. Once a demonstration is successful and the allowed episode length has been exceeded, the robot first moves upward and then returns to its reset pose. The reset pose is randomized in the x and y directions and around the z axis before the next episode, with task-specific magnitudes listed in Table 5. During kinesthetic demonstrations, the operator must additionally take care not to occlude the wrist cameras. Before the reported experiments, the operator practiced with both demonstration interfaces until they could reliably complete the tasks with each method. Practice rollouts were not included in the reported demonstration times or replay buffers. This familiarization was used to reduce first-use effects when comparing SpaceMouse demonstrations and kinesthetic guidance.

E.2 Human Interventions During Training

A human operator supervises execution during RL training and can intervene whenever the policy becomes unsafe, loses alignment, or otherwise behaves suboptimally. During intervention, autonomous control is overridden by human input through either the SpaceMouse or kinesthetic guidance, depending on the interface condition being evaluated. All state-action-reward tuples encountered during the intervention window are logged as corrective data rather than discarded. After an intervention ends, the resulting corrective trajectory is inserted into both the demonstration replay buffer and the online RL replay buffer. This dual insertion preserves the original RLPD bootstrapping logic while also allowing the newly collected correction to influence subsequent online updates immediately.