Efficient and Reliable Teleoperation through Real-to-Sim-to-Real Shared Autonomy

Shuo Sha[1], Yixuan Wang[1], Binghao Huang[1], Antonio Loquercio[2], Yunzhu Li[1]
[1]Columbia University [2]University of Pennsylvania

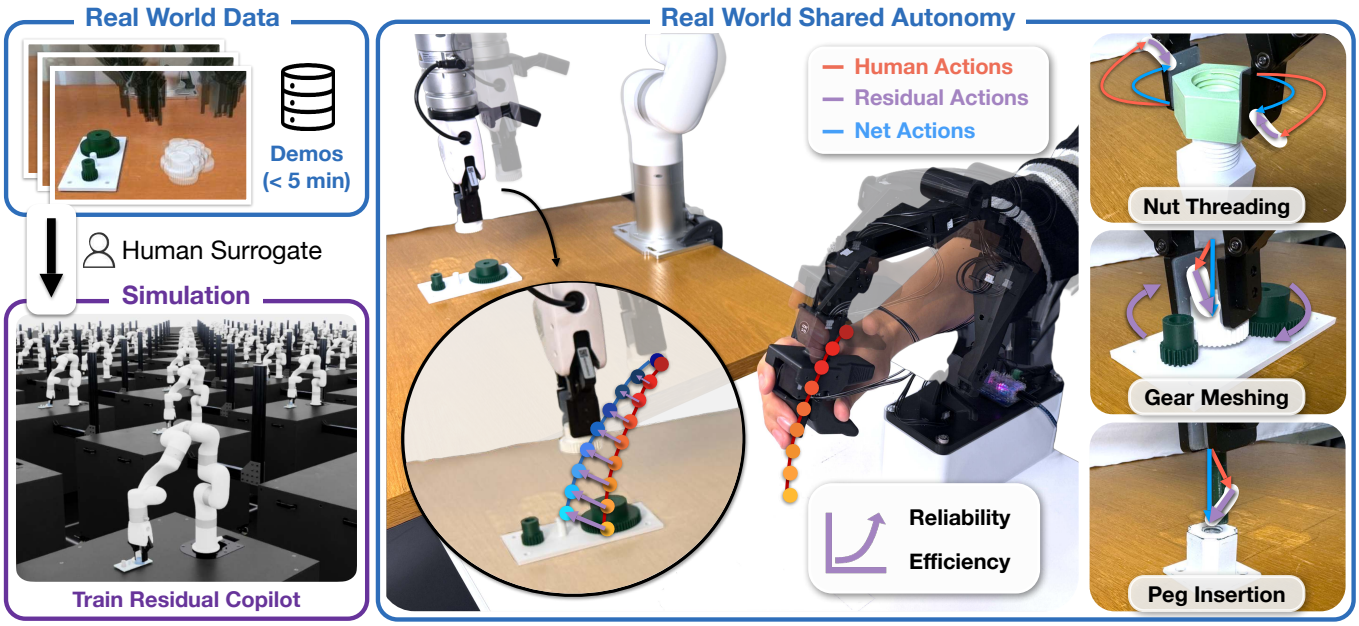

Fig. 1: **Overview of our real-to-sim-to-real shared autonomy framework.** A small amount of real teleoperation data (< 5 minutes) is used to construct a lightweight human surrogate, which drives simulation-based training of a residual copilot policy. At deployment, the copilot provides low-level corrective actions that combine with human commands to produce reliable and efficient shared autonomy for fine-grained, contact-rich manipulation tasks, where direct teleoperation struggles with precise alignment, axis-constrained rotation, and contact regulation, including nut threading, gear meshing, and peg insertion.

***Abstract*—Fine-grained, contact-rich teleoperation remains slow, error-prone, and unreliable in real-world manipulation tasks, even for experienced operators. Shared autonomy offers a promising way to improve performance by combining human intent with automated assistance, but learning effective assistance in simulation requires a faithful model of human behavior, which is difficult to obtain in practice. We propose a real-to-sim-to-real shared autonomy framework that augments human teleoperation with learned corrective behaviors, using a simple yet effective k-nearest-neighbor (kNN) human surrogate to model operator actions in simulation. The surrogate is fit from less than five minutes of real-world teleoperation data and enables stable training of a residual copilot policy with model-free reinforcement learning. The resulting copilot is deployed to assist human operators in real-world fine-grained manipulation tasks. Through simulation experiments and a user study with sixteen participants on industry-relevant tasks, including nut threading, gear meshing, and peg insertion, we show that our system improves task success for novice operators and execution efficiency for experienced operators compared to direct teleoperation and shared-autonomy baselines that rely on expert priors or behavioral-cloning pilots. In addition, copilot-assisted teleoperation produces higher-quality demonstrations for downstream imitation learning. Website: https://residual-copilot.github.io/**

I. Introduction

Teleoperation is essential for executing complex robotic manipulation tasks and for collecting demonstrations to train visuomotor policies. However, for fine-grained, contact-rich, high-precision manipulation, teleoperation is often slow, error-prone, and unreliable. These limitations make teleoperation a practical bottleneck both for deploying robotic systems in the real world and for scaling high-quality demonstration data.

A key reason for this difficulty lies in a mismatch between human capabilities and the role assigned to them in conventional teleoperation. Humans excel at high-level intent specification and task reasoning, but struggle to provide continuous, low-level robustness through a teleoperation interface due to limited viewpoints, latency, and the embodiment gap. As a result, millimeter-scale alignment and contact regulation (crucial for precision manipulation) are poorly suited to direct human control and are better handled by an automated copilot.

Shared autonomy aims to address this mismatch by assisting human operators during task execution. Prior work [6, 15, 41, 46, 71, 72] has shown that shared autonomy can improve

teleoperation by mapping human commands toward a stronger action prior, such as an expert policy, planner, or structured objective. A complementary line of work [5, 52, 54, 60] instead learns assistance directly as a copilot conditioned on the pilot command and system state, typically using model-free reinforcement learning or residual formulations that minimally adjust human input.

Despite their success, existing approaches face fundamental limitations. Methods that rely on expert priors shift the core challenge to obtaining the prior itself; once a competent autonomous policy exists, teleoperation is often no longer the limiting factor. Existing copilot-learning approaches avoid this assumption but either depend on extensive human-in-the-loop training, require large-scale data to learn an accurate human surrogate, or remain challenging to deploy in complex real-world manipulation settings.

In this work, we focus on copilot learning without assuming an expert prior. Rather than addressing teleoperation by first solving full autonomy, we treat autonomy as the harder problem and instead learn low-level corrective behaviors that preserve human intent. We propose a real-to-sim-to-real shared autonomy pipeline that trains a residual copilot policy using model-free reinforcement learning in simulation, driven by a lightweight k-nearest-neighbor (kNN) human surrogate fit from a small amount of real teleoperation data. By remaining within empirical data support, the kNN surrogate enables stable copilot training in simulation and effective transfer to the real world. We evaluate our approach on a suite of fine-grained, contact-rich manipulation tasks, including nut threading, gear meshing, and peg insertion. Through simulation experiments and a user study with both novice and experienced operators, we show that our system improves task success for novice operators and execution efficiency for experienced operators compared to direct teleoperation and shared-autonomy baselines that rely on expert priors or behavioral-cloning pilots. In addition, our approach produces higher-quality demonstrations for downstream imitation learning.

In summary, our contributions are threefold: (1) we propose a real-to-sim-to-real shared autonomy pipeline that assists teleoperation for a diverse suite of high-precision, contact-rich manipulation tasks; (2) we show that a lightweight kNN model, constructed from less than five minutes of human demonstrations, can serve as an effective surrogate for a human during model-free policy training in simulation; and (3) we demonstrate that our system increases the *effectiveness* and *efficiency* for both novice and experienced operators, enabling higher-quality demonstrations for imitation learning as well as practical assistive and remote teleoperation.

II. Related Works

A. Shared Autonomy

Shared autonomy combines human input with robot assistance during task execution and has been widely studied in teleoperation, surgical robotics, and assistive manipulation. To simplify terminology, we refer to the human user as the *pilot* and the assistive agent as the *copilot*.

Early work primarily focused on improving *efficiency*, such as reducing time-on-task [13, 14, 19, 25, 27, 33, 35, 38, 42, 48, 74, 76], lowering input dimensionality [9, 16, 22, 36, 49, 50, 57, 61, 75], or enabling a single pilot to supervise multiple robots [11, 47, 59, 70]. We focus instead on shared autonomy for high-precision, contact-rich manipulation, where improving the *quality* of pilot input is essential.

Expert-prior shared autonomy. One class of approaches assists teleoperation by blending pilot commands with an autonomous prior, such as a policy, planner, or action distribution. Assistance is often goal-conditioned, with pilot intent inferred explicitly or implicitly [7, 29, 44, 45, 53, 62, 73]. Pilot inputs are guided toward high-value or high-likelihood actions using Bayesian inference [2, 28, 30], inverse reinforcement learning [3, 8, 23, 39, 51, 78], or optimization-based formulations [6, 20, 41, 65, 77]. Recent diffusion-based methods guide denoising with similarity gradients between pilot commands and policy outputs [15, 37, 46, 58, 66, 71, 72].

A key limitation of this class is its reliance on a competent autonomous prior. Acquiring expert data, training strong policies, or specifying accurate dynamics and objectives is often the dominant challenge. Moreover, prior work shows that blending pilot input with an autonomous policy does not guarantee the resulting behavior remains on the original motion manifold [58, 64, 66]. These issues limit applicability where expert priors are unavailable or insufficiently robust.

Copilot learning. A second class learns assistance directly as a copilot conditioned on the pilot command and system state. Representative approaches use reinforcement learning or POMDP formulations to trade off task return with deviation from pilot input [5, 21, 26, 31, 52, 60], while residual formulations improve sample efficiency by learning corrective actions on top of a nominal pilot policy [54]. These methods avoid assuming an expert autonomous prior and are well suited to preserving human intent.

A central design challenge in copilot learning is the choice of pilot model during training. Prior work studies abstract pilots, such as noisy or laggy experts [5, 21, 31, 52, 60], and demonstrates that pilot choice strongly affects generalization [52, 54, 60]. More realistic pilots can be obtained via behavioral cloning from interaction data [10, 54], but parametric BC pilots are data-hungry and brittle under copilot-induced distribution shift in low-data regimes.

Our work builds on residual copilot learning and addresses this pilot-model bottleneck. Instead of fitting a parametric BC pilot, we use a lightweight k-nearest-neighbor (kNN) surrogate constructed from limited teleoperation data. By remaining within empirical data support, the kNN pilot enables stable copilot training in simulation and effective transfer to the real world. Unlike prior residual shared autonomy work [54], which primarily evaluates in simulation, we demonstrate real-world performance on fine-grained manipulation tasks.

B. Residual Policy Learning

Residual policy learning [4, 24, 56] decomposes control into a nominal action and a learned correction. By learning only the

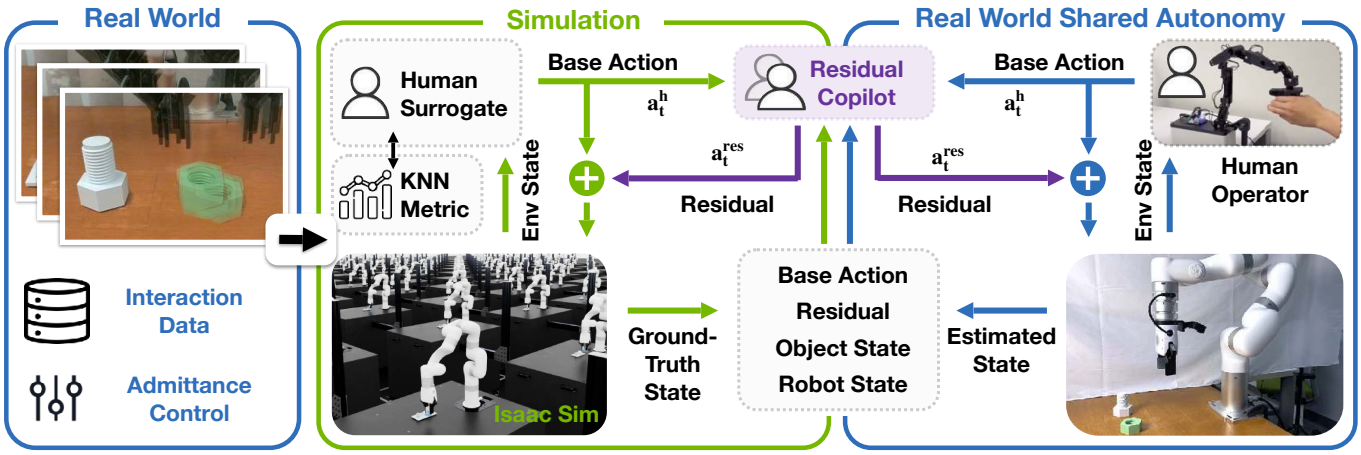

Fig. 2: **Method overview of residual copilot learning with a real-to-sim-to-real pipeline. Left:** less than five minutes of real-world teleoperation interaction data is collected under admittance control. **Center:** in simulation, a lightweight kNN-based human surrogate constructed from real-world data generates base actions conditioned on the environment state, while a residual copilot policy is trained with reinforcement learning to produce corrective residual actions. **Right:** at deployment, the residual copilot assists the human operator in zero-shot, conditioning on the estimated environment state and human base actions. The residual formulation preserves human intent while improving alignment, contact regulation, and execution robustness.

residual, the policy is constrained to local corrections, which improves exploration, sample efficiency, and training stability compared to end-to-end learning [56].

For shared autonomy, residual formulations [52] naturally separate roles: the pilot provides high-level motion commands, while the residual copilot applies low-level corrections for alignment, contact regulation, and robustness, which preserves human intent and improves performance.

We adopt this abstraction and train a residual copilot with model-free reinforcement learning to correct teleoperation commands from sparse task rewards, without requiring explicit intent modeling or expert priors.

C. Real-to-Sim-to-Real Learning

Real-to-sim-to-real pipelines use real-world data to construct task-relevant simulations for scalable and transferrable policy learning, followed by deployment back to the physical system. Prior work [17, 32, 63, 69] typically digitalizes geometry, dynamics, and sensing, often with domain randomization to improve sim-to-real robustness.

In shared autonomy, the simulator must also model the human pilot, since the copilot conditions on the pilot's actions. As a result, the quality of learned assistance depends on how faithfully human behavior is represented during training.

We therefore use real teleoperation data to instantiate a human surrogate in simulation, then train residual copilot policies entirely in simulation with model-free reinforcement learning, and deploy them directly in the real world to assist human operators. Beyond human modeling, the real-to-sim stage also digitalizes task and dynamics details, including environment recreation and system identification of controller and physics parameters, improving simulation fidelity for contact-rich interaction.

III. Method

A. Problem Definition

We model the shared autonomy problem as a partially observable Markov decision process (POMDP) [21]:

$$\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, O, \gamma \rangle.$$

At each timestep, the full state $x_t \in \mathcal{X}$ is defined as $x_t = (s_t, g)$, where $s_t \in \mathcal{S}$ denotes the environment state and $g \in \mathcal{G}$ represents the human pilot's latent goal, which is unobservable to the copilot (assistive agent). The environment state s_t includes the robot end-effector pose and velocity in task space, the gripper state, and the object poses expressed in the end-effector frame.

Observing the environment, the pilot provides a goal-implicit control command $a_t^h = [p, q, u] \in \mathcal{A}$ via teleoperation, consisting of a target task-space end-effector pose $[p, q]$ and a gripper command u. Rather than explicitly inferring the pilot's latent goal g, we treat intent as being expressed implicitly through teleoperation commands.

The copilot observes both the environment state and the pilot input, i.e., $o_t = (s_t, a_t^h) \in \Omega$. We parameterize the copilot as a residual policy, which we refer to as the `Residual Copilot`, treating the pilot command a_t^h as a base action and learning a corrective delta. At each step, the copilot predicts a normalized residual action $a_t^{\text{res}} \sim \pi_r(\cdot \mid o_t)$, and the final action command is

$$a_t = a_t^h \oplus \alpha a_t^{\text{res}},$$

where $\oplus$ composes translational and gripper increments additively and applies rotational increments multiplicatively in task space (see Appendix A-B1). The residual scale α maps normalized residual action to controllable physical magnitudes.

This action induces a state transition $x_{t+1} \sim \mathcal{T}(\cdot \mid x_t, a_t)$ and yields a reward $r_t = \mathcal{R}(x_t, a_t)$.

Our objective is to learn the `Residual Copilot` policy $\pi_r(a_t^{\text{res}} \mid o_t)$ that maximizes the expected discounted return, without access to the environment dynamics $\mathcal{T}$, the pilot goal space $\mathcal{G}$, or the pilot policy $\pi_h(a_t^h \mid x_t)$. The `Residual Copilot` is parameterized as a Gaussian policy with an MLP backbone and optimized using model-free reinforcement learning with PPO [55].

We decompose the reward as $\mathcal{R} = \mathcal{R}_{\text{general}} + \mathcal{R}_{\text{success}}$, where $\mathcal{R}_{\text{general}}$ captures goal-agnostic objectives shared across a set of tasks, and $\mathcal{R}_{\text{success}}$ provides a task-specific sparse success signal. In our fine-grained assembly experiments, $\mathcal{R}_{\text{general}}$ includes termination and contact-force penalties, as well as shaping terms for assembly-axis alignment, upright end-effector orientation, and action regularization (Table V).

B. Human Surrogate Model

Training the copilot directly with human pilots in the loop is costly and difficult to scale. Following prior work [52, 54], we train in simulation using a surrogate pilot in place of the human pilot.

A suitable human surrogate must satisfy two requirements: it must remain coherent under the distribution shift induced by copilot exploration (*mutual dependency*); and it must be constructible from limited data, as otherwise the problem effectively reduces to training a fully autonomous policy from a large dataset.

We therefore adopt a non-parametric k-nearest-neighbor (kNN) surrogate π_h^{kNN} (the `kNN Pilot`) built from a small demonstration set that may include unsuccessful demonstrations. By retrieving actions directly from the empirical demonstration manifold, the `kNN Pilot` avoids extrapolation beyond distributional support; assuming the human pilot acts along a consistent action manifold with or without assistance, this inductive bias promotes stable behavior under moderate distribution shift while remaining inherently data-efficient.

For neighbor retrieval, we define a weighted distance over translation p, rotation q, and gripper u components using only proprioceptive end-effector commands:

$$d(a, a') = \alpha_1 \|p - p'\|_2 + \alpha_2 d_{\text{SO(3)}}(q, q') + \alpha_3 |u - u'|, \quad (1)$$

where $d_{\text{SO(3)}}$ denotes quaternion geodesic distance and $\{\alpha_i\}$ are tuned by minimizing nearest-neighbor prediction error on the demonstration set.

To improve robustness during copilot training, we augment `kNN Pilot` with two mechanisms: (1) action chunking to preserve short-horizon structure and avoid myopic switching between neighbors, and (2) local stochastic perturbations to expand coverage under distribution shift.

Action chunking. At runtime, $\pi_h^{\text{kNN}} : \mathcal{S} \to \mathcal{A}$ retrieves the k nearest demonstrated commands under Eq. (1) and samples one via a temperature-scaled softmax over negative distances. Instead of returning a single command, the surrogate outputs a short *action chunk* of consecutive demonstrated commands from the selected neighbor, preserving temporal consistency.

Smooth local perturbations. Under residual assistance, the copilot may drive the system outside the empirical support of the demonstrations, making pure retrieval brittle. To locally expand coverage, we inject i.i.d. local perturbations composed with the current command using the same operator $\oplus$. At each step, we sample $\varepsilon_t \sim \mathcal{U}([l, h]^d)$ and form $\delta a_t = m_t \, \varepsilon_t$, where the gate $m_t \in [0, 1]$ evolves as $m_t = \beta m_{t-1} + (1 - \beta) b_t$ with $b_t \sim \text{Bernoulli}(p_{\text{on}})$. The perturbation is applied via $a_t^h \leftarrow a_t^h \oplus \delta a_t$. The Bernoulli parameter p_{on} controls activation frequency, while β governs the smooth rise and decay of noisy phases.

C. Admittance Control

To safely complete contact-rich manipulation tasks, we execute commands with task-space compliance via admittance control, allowing the end-effector to yield under contact while tracking goal commands. We model compliance with a virtual spring–mass–damper relationship [18, 34] between task-space motion and the measured external wrench:

$$f = M\ddot{\xi} + D\dot{\xi} + K(\xi - \xi_{\text{ref}}),$$

where $\xi \in \mathbb{R}^N$ denotes task-space pose coordinates, $f \in \mathbb{R}^N$ is the measured wrench, and (M, D, K) are the virtual inertia, damping, and stiffness; ξ_{ref} is the reference pose at rest.

D. Real-to-Sim-to-Real

To improve human surrogate fidelity, reduce simulation bias, and learn reliable assistance behaviors, we adopt a real-to-sim-to-real pipeline. In the real-to-sim stage, we digitalize both the human surrogate and task environment. A small set of real teleoperation demonstrations is used to fit the `kNN Pilot` and to calibrate the simulator by tuning low-level control and physical parameters to match real trajectories. We optimize joint-space PID gains (K_p, K_i, K_d) for the arm and gripper; task-space admittance gains, including stiffness (K_x, K_r), damping (D_x, D_r), and virtual mass/inertia (M_x, M_r); and physical parameters affecting contact dynamics, including friction coefficient μ and object center of mass m_{CoM}.

In the sim-to-real stage, we apply domain randomization over controller parameters and object pose observations to account for controller discrepancies and real-world pose estimation error (Appendix A-B3).

IV. Experiments

In this section, we evaluate our `Residual Copilot` in a suite of fine-grained, contact-rich manipulation tasks. We aim to address the following questions: (**Q1**) Can our `Residual Copilot` improve a human operator's performance in fine-grained, contact-rich manipulation tasks? (**Q2**) Does our choice of human surrogate `kNN Pilot` outperform prior choices of human surrogates and guided diffusion baselines? (**Q3**) Does data collected with the `Residual Copilot` lead to improved downstream imitation policy performance?

Experiment Setup. To address these questions, we evaluate the `Residual Copilot` through a real-world user study

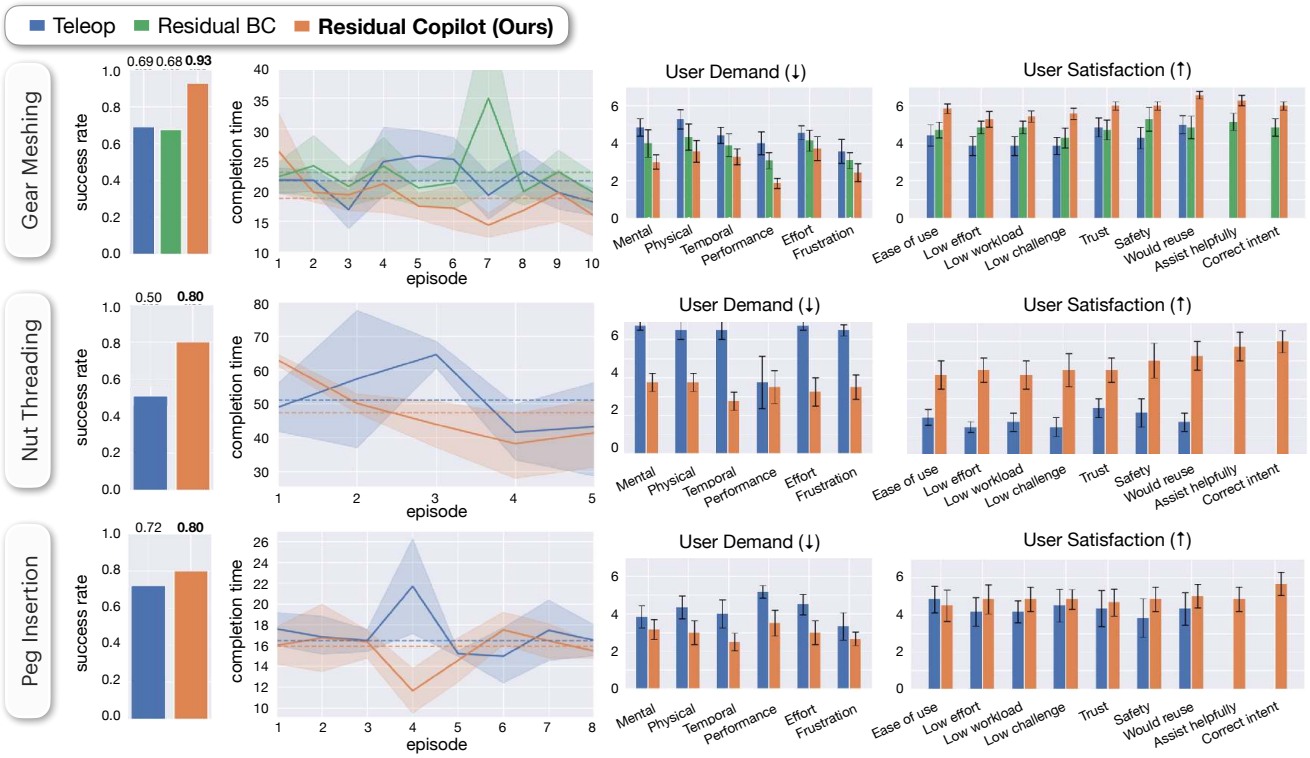

Fig. 3: **Human experiment results across three contact-rich assembly tasks.** Rows correspond to Gear Meshing, Nut Threading, and Peg Insertion. **Left:** success rate and mean completion time across successful episodes (solid line; shaded region denotes standard deviation; dashed line indicates per-method mean). **Right:** subjective workload (NASA-TLX subscales; lower is better) and user satisfaction (higher is better). Across tasks, the `Residual Copilot` improves success rate and can effectively reduce completion time, while lowering reported workload and increasing user satisfaction compared to direct teleoperation and a residual baseline.

across three contact-rich assembly tasks (Sec. IV-A), a real-world and simulation analysis of pilot model choice and robustness (Sec. IV-B), and a downstream imitation learning comparison (Sec. IV-C).

We consider three representative fine-grained, contact-rich assembly tasks from the NIST board #1 [1] (radial clearance $< 1\,\mathrm{mm}$):

- **Gear Meshing:** pick up the medium gear, insert it onto its shaft, and mesh with the other gears. A trial succeeds only if the gear remains grasped until insertion completes.
- **Nut Threading:** pick up an M32 nut and tighten it onto the bolt by at least $180°$. Due to hardware limits of GELLO, participants perform three $60°$ turns.
- **Peg Insertion:** pick up an $8\,\mathrm{mm}$-long peg and insert it into the base. A trial succeeds only if the peg reaches the bottom of the receptacle.

For all tasks, failure is defined as dropping the object prior to success or exceeding the xArm safety force threshold. To ensure a fair comparison, we fix the robot to a common initial pose and randomize the initial object placement within the workspace at the start of each trial.

In the real-world setup, participants teleoperate a UFactory xArm7 using GELLO [68] in 7-DoF task space (end-effector pose and gripper command). We attach a wrist-mounted force-torque sensor to implement admittance control. We use an external Intel RealSense D455 camera and estimate object poses with FoundationPose [67]. In simulation, we use the NVIDIA Isaac Lab [40] gear-meshing and peg-insertion environments from Factory [43], and implement the M32 nut-and-bolt task similarly with SDF-based collision. We read ground-truth contact forces at the end-effector for admittance control.

Subject Allocation. We recruited 12 novice teleoperators (fewer than 20 prior teleoperation trajectories) for gear meshing and peg insertion, and 4 experienced teleoperators (at least 50 prior teleoperation trajectories) for nut threading, totaling over 20 hours of teleoperation trials across the three tasks. Participants received task objectives and a high-level description of each method (see Appendix for details). Each participant completed a 5-minute warm-up to familiarize with the interface and system dynamics. Following prior work [52, 54], we interleave direct teleoperation and copilot-assisted teleoperation to control for learning effects. After completing all trials, participants filled out NASA-TLX and user-satisfaction questionnaires.

Baselines. We implement the following pilot baselines to isolate the effect of pilot modeling on copilot learning and to systematically evaluate robustness under varying assumptions about human behavior:

- `BC Pilot`: Following Schaff and Walter [54], we learn a behavioral surrogate by imitation. For each task, we

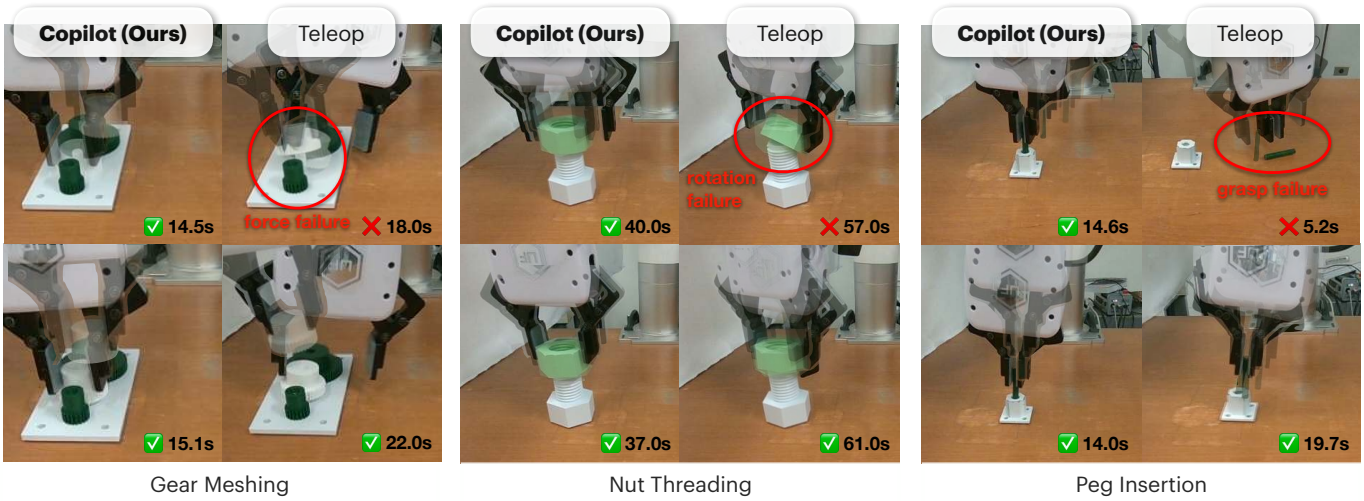

Fig. 4: **User study qualitative results.** Representative trajectory temporal overlays comparing unassisted teleoperation and our `Residual Copilot` on three high-precision tasks. **Top row (effectiveness):** our copilot enables reliable task completion in cases where teleoperation fails due to force, rotation, or grasp errors. **Bottom row (efficiency):** when both succeed, our copilot consistently reduces completion time and exemplifies smooth motions. Together, these results demonstrate that our method provides both reliable and efficient assistance in challenging manipulation scenarios.

collect fewer than 5 minutes of real teleoperation data, apply geometric data augmentation, and train a Diffusion Policy (DP) [12] to model the pilot's action distribution.

- `Expert-based Pilots`: Following Reddy et al. [52], we construct a near-optimal `Expert Pilot` by training a DP on 2000 successful rollouts (200–400k transitions) generated by our `Residual Copilot`. We additionally instantiate a `Laggy Pilot` (previous action repeated with probability 0.8) and a `Noisy Pilot` (smooth gated noise injected with probability 0.5; see Section III-B), following the abstract pilot perturbations proposed in Reddy et al. [52] to model delayed reaction and execution errors, respectively.

Using these pilots, we train the following copilot baselines:

- `GD Copilots`: Following prior guided diffusion (GD) shared autonomy formulations [71], we train a diffusion copilot that denoises actions conditioned on pilot commands. We consider two variants differing in training data optimality: `GD BC`, trained on the augmented teleoperation dataset used for the `BC Pilot`; and `GD Expert`, trained on the expert rollouts used for the `Expert Pilot` (an impractical upper-bound baseline included to characterize the limits of GD methods).
- `Residual BC`: A residual RL copilot trained to assist the `BC Pilot` using the same RL setup as our method, differing only in the choice of pilot model.

A. Copilot Performance

To address **Q1**, we conduct the user study to compare direct and copilot-assisted teleoperation, evaluating both objective task performance and subjective user experience across three fine-grained, contact-rich assembly tasks. Fig. 3 summarizes success rate and completion time, along with NASA-TLX workload scores and user satisfaction. Overall, residual assistance consistently outperforms direct teleoperation, yielding higher success rates, faster execution, and improved subjective experience across tasks.

The benefits of residual assistance manifest differently depending on task structure and failure modes. For **Nut Threading**, which requires sustained axis-constrained rotation and precise orientation control under contact, residual rotational stabilization and acceleration are critical for maintaining thread engagement. Without assistance, operators frequently lose alignment or apply insufficient rotational motion, leading to premature disengagement, as shown in Fig. 4. The copilot mitigates these issues by stabilizing orientation and amplifying effective rotational commands, improving success by up to 30% while reducing the need for repeated corrective motions. For **Gear Meshing**, success depends on achieving precise insertion depth followed by small corrective rotations to align gear teeth. Direct teleoperation often fails due to slight misalignment that is difficult for operators to perceive or correct through the interface. The `Residual Copilot` learns to reliably complete insertion and apply subtle rotational corrections at contact, reducing the operator's low-level alignment burden while preserving their high-level intent. As a result, operators achieve higher success with fewer failed insertion attempts. For **Peg Insertion**, failures arise not only during insertion but also from unreliable grasping. In this setting, the copilot stabilizes the end-effector at an effective pre-grasp pose, centered above the peg, while leaving the timing of grasp initiation entirely to the user. This division of roles preserves operator control over task sequencing while improving grasp reliability, leading to higher overall success rates.

Completion-time trends further verify these improvements. Across tasks, the `Residual Copilot` generally reduces

Copilot	Eval. Pilot														
	Laggy Pilot			Noisy Pilot			Expert Pilot			BC Pilot			kNN Pilot		
	Gear	**Peg**	**Nut**	**Gear**	**Peg**	**Nut**	**Gear**	**Peg**	**Nut**	**Gear**	**Peg**	**Nut**	**Gear**	**Peg**	**Nut**
No Copilot	0.97	**0.94**	0.49	0.92	0.91	0.24	**0.99**	**0.99**	0.61	0.84	0.71	0.03	0.87	0.85	0.16
GD Expert	0.96	0.89	0.24	0.95	0.91	0.25	0.94	0.90	0.28	0.94	**0.89**	**0.34**	0.95	0.90	0.27
GD BC	0.63	0.56	0.00	0.65	0.56	0.00	0.62	0.53	0.01	0.63	0.52	0.01	0.66	0.55	0.00
Residual BC	0.96	0.44	0.00	0.91	0.44	0.00	0.98	0.44	0.00	0.70	0.40	0.00	0.86	0.46	0.02
Residual Copilot (Ours)	**0.99**	0.92	**0.65**	**0.96**	**0.92**	**0.69**	0.98	0.93	**0.74**	**0.97**	0.83	0.21	**1.00**	**0.99**	**0.81**

TABLE I: **Cross-pilot generalization and robustness in simulation.** Rows denote copilots trained with different training pilots; columns evaluate each trained copilot under distinct *evaluation* pilots across three tasks. Each cell reports task progression (over 1000 simulation trials); see Appendix for metric details. Gray and blue diagonal entries indicate in-distribution evaluation, i.e., a copilot evaluated with the pilot it was trained on; off-diagonal entries measure generalization to unseen pilots. The Residual Copilot maintains consistently high performance across evaluation pilots, demonstrating robustness to pilot mismatch and perturbations such as lag and action noise.

completion time by decreasing the number of failed attempts, corrective motions, and recovery behaviors required to complete each trial. These objective gains align with subjective reports (Fig. 3, right), where participants report lower workload across multiple NASA-TLX subscales and higher overall satisfaction. Together, these results indicate that residual assistance not only improves task outcomes, but also reduces cognitive and physical burden while maintaining a clear and intuitive division of control between the human and the copilot.

B. Human Surrogate Analysis

To address **Q2**, we compare our Residual Copilot against Residual BC with novice teleoperators on gear meshing (Fig. 3, top row). Our copilot improves objective performance (success rate and completion time) and receives higher subjective ratings. In particular, participants report higher *correct intent*, suggesting that training with the kNN Pilot promotes more stable exploration and better pilot generalization, yielding more reliable local corrections that preserve human intent.

We further evaluate surrogate quality in simulation by training copilots under different pilot models and testing them across evaluation pilots (Table I). Across tasks and evaluation pilots, our Residual Copilot achieves the highest in-distribution performance and exhibits the smallest degradation under pilot mismatch, indicating improved robustness to distribution shift.

Under suboptimal pilots, Residual BC achieves consistently lower task progression, suggesting that it fails to explore success-relevant regions when distribution shifts arise from copilot-induced exploration under sparse rewards, leading to premature convergence to local minima where only generalized rewards are obtained.

Finally, GD Copilots are highly sensitive to the pilot model. In particular, guiding GD Expert with the Expert Pilot distribution reduces performance relative to No Copilot under the Expert Pilot. For contact-sensitive tasks such as Nut Threading, which require consistent axis-constrained rotation and precise contact regulation, GD Expert does not exceed 34% progression under any pilot model, despite achieving 61% progression autonomously. This indicates that test-time guidance can steer actions away from task-optimal regions when the pilot distribution is suboptimal. Conversely, steering GD BC with stronger priors does not yield meaningful improvements. Overall, this asymmetry suggests that test-time guidance alone is insufficient to reconcile action optimality with preservation of human intent.

C. Data Quality Comparison

To address **Q3**, we compare the quality of demonstrations collected under Residual Copilot assistance versus direct teleoperation for downstream imitation learning. We train a vision-based Diffusion Policy (DP) under two conditions:

(i) **Matched attempts.** We randomly sample five participants from the gear-meshing study and train a DP using all successful demonstrations collected from an equal number of teleoperation attempts for each method.

(ii) **Matched successes.** We randomly sample five participants, compute the minimum number of successful trials across the two methods, and train a DP using the same number of successful demonstrations for both.

Under *matched attempts*, where both methods contribute the same number of teleoperation trials, the DP trained on copilot-assisted data achieves substantially higher task progression than the DP trained on teleoperation data. As shown in Table II, the DP trained on copilot-collected data reaches 18/20 grasp successes and 11/20 insertion successes, compared to 7/20 grasps and 1/20 insertion for teleoperation. This gap indicates that copilot-assisted demonstrations provide more learnable supervision for the downstream DP under the same data-collection attempt budget.

More importantly, under *matched successes*, this advantage persists even when both datasets contain the same number of successful demonstrations. The DP trained on copilot-assisted data achieves 19/20 grasp success and 9/20 insertion success, whereas the DP trained on teleoperation data achieves 6/20 grasps and 0/20 insertions. Because the number of successful demonstrations and the model capacity are controlled, this comparison isolates demonstration quality rather than quantity.

We attribute this difference to qualitative properties of the collected trajectories, illustrated in Fig. 4. Residual copilot assistance produces demonstrations that are more consistent in

	Matched attempts		Matched successes	
Method	**Grasp**	**Insert**	**Grasp**	**Insert**
Teleop	7/20	1/20	6/20	0/20
Residual (Ours)	**18/20**	**11/20**	**19/20**	**9/20**

TABLE II: **Downstream policy progression from copilot vs. teleoperation data (Gear Meshing).** Diffusion policies are trained on demonstrations collected under `Residual Copilot` assistance or direct teleoperation, using matched attempts and matched successes to control for data budget. Results show copilot assistance improves learnability and consistency of the collected demonstrations.

alignment, contact timing, and approach strategy, while still reflecting human intent. In contrast, successful teleoperation demonstrations often include larger corrective motions, abrupt recoveries, or incidental contacts, increasing trajectory variability. Such variability makes the action distribution harder for the diffusion policy to model and degrades generalization.

Overall, these results indicate that `Residual Copilot` assistance improves the structure and consistency of successful demonstrations. This leads to downstream policies that generalize more reliably, even when trained on the same number of successes, highlighting the value of shared autonomy as a tool for scalable, high-quality demonstration collection.

V. Conclusion

We presented a real-to-sim-to-real shared autonomy framework for fine-grained, contact-rich teleoperation. Our approach trains a residual copilot using model-free reinforcement learning in simulation, driven by a lightweight kNN human surrogate fit from fewer than five minutes of real teleoperation data. By learning only low-level corrective behaviors on top of human commands, the copilot preserves operator intent while avoiding reliance on brittle parametric behavioral-cloning pilots or expert autonomous priors.

Across both simulation and a real-world user study on nut threading, gear meshing, and peg insertion, our system improves task success for novice operators and execution efficiency for experienced operators relative to direct teleoperation and prior shared-autonomy baselines. Beyond execution performance, copilot-assisted teleoperation produces demonstrations more efficiently and with greater consistency and structure, leading to improved downstream imitation learning policy performance.

A key strength of the proposed approach is its residual formulation, learned through a real-to-sim-to-real pipeline. Because the copilot operates as a correction on top of user commands, it is largely agnostic to teleoperator skill level and can be integrated with existing teleoperation interfaces without modifying user control strategies. This makes shared autonomy a practical tool not only for assistive execution, but also for scalable, high-quality data collection in fine-grained, contact-rich manipulation.

One limitation of our work is the assumption that user behavior during data collection matches behavior under assistance. In practice, users may adapt and exhibit emergent strategies once a copilot is present, introducing additional distribution shift. An important direction for future work is to explicitly model this co-adaptation, for example by updating the human surrogate online or learning interaction policies that jointly account for human behavior and copilot assistance.

Acknowledgment

This work was partially supported by the DARPA TIAMAT program (HR0011-24-9-0430), the Toyota Research Institute, NSF Award #2409661, Samsung Research America, and an Amazon Research Award (Fall 2024). This article solely reflects the opinions and conclusions of its authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

We would like to thank Kaifeng Zhang, Hanxiao Jiang, Fangyu Wu, Shivansh Patel, Jaisel Singh, and all other members of the RoboPIL Lab for helpful discussions during the project. We would also like to thank each participant of the user study for their time and feedback.

References

[1] Assembly performance metrics and test methods. https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly, April 2022.

[2] D. Aarno, S. Ekvall, and D. Kragic. Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1139–1144, 2005. doi: 10.1109/ROBOT.2005.1570269.

[3] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015430. URL https://doi.org/10.1145/1015330.1015430.

[4] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual rl for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

[5] Kal Backman, Dana Kulić, and Hoam Chung. Reinforcement learning for shared autonomy drone landings. *Auton. Robots*, 47(8):1419–1438, October 2023. ISSN 0929-5593. doi: 10.1007/s10514-023-10143-3. URL https://doi.org/10.1007/s10514-023-10143-3.

[6] Alexander Broad, Todd Murphey, and Brenna Argall. Learning models for shared control of human-machine systems with unknown dynamics, 2018. URL https://arxiv.org/abs/1808.08268.

[7] Alexander Broad, Todd Murphey, and Brenna Argall. Highly parallelized data-driven mpc for minimal intervention shared control, 2019. URL https://arxiv.org/abs/1906.02318.

[8] Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demon-

strations via inverse reinforcement learning from observations, 2019. URL https://arxiv.org/abs/1904.06387.

[9] Samuel Bustamante, Ismael Rodríguez, Gabriel Quere, Peter Lehner, Maged Iskandar, Daniel Leidner, Andreas Dömel, Alin Albu-Schäffer, Jörn Vogel, and Freek Stulp. Feasibility checking and constraint refinement for shared control in assistive robotics. *IEEE Robotics and Automation Letters*, 9(9):8019–8026, 2024. doi: 10.1109/LRA.2024.3430710.

[10] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination, 2020. URL https://arxiv.org/abs/1910.05789.

[11] Kishan Chandan, Vidisha Kudalkar, Xiang Li, and Shiqi Zhang. Arroch: Augmented reality for robots collaborating with a human, 2022. URL https://arxiv.org/abs/2109.10400.

[12] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

[13] Wenqiang Chi, Giulio Dagnino, Trevor M. Y. Kwok, Anh Nguyen, Dennis Kundrat, Mohamed E. M. K. Abdelaziz, Celia Riga, Colin Bicknell, and Guang-Zhong Yang. Collaborative robot-assisted endovascular catheterization with generative adversarial imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2414–2420, 2020. doi: 10.1109/ICRA40945.2020.9196912.

[14] Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. No, to the right: Online language corrections for robotic manipulation via shared autonomy. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '23, page 93–101. ACM, March 2023. doi: 10.1145/3568162.3578623. URL http://dx.doi.org/10.1145/3568162.3578623.

[15] Yunxin Fan and Monroe Kennedy III. Diffusion-safe: Shared autonomy framework with diffusion for safe human-to-robot driving handover, 2025. URL https://arxiv.org/abs/2505.09889.

[16] Michael Hagenow, Emmanuel Senft, Robert Radwin, Michael Gleicher, Bilge Mutlu, and Michael Zinn. Informing real-time corrections in corrective shared autonomy through expert demonstrations. *IEEE Robotics and Automation Letters*, 6(4):6442–6449, 2021. doi: 10.1109/LRA.2021.3094480.

[17] Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobanbab, Chaoyi Pan, Zeji Yi, Guannan Qu, Kris Kitani, Jessica Hodgins, Linxi "Jim" Fan, Yuke Zhu, Changliu Liu, and Guanya Shi. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills, 2025. URL https://arxiv.org/abs/2502.01143.

[18] Yifan Hou, Zeyi Liu, Cheng Chi, Eric Cousineau, Naveen Kuppuswamy, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control, 2025. URL https://arxiv.org/abs/2410.09309.

[19] Xin Huang, Stephen G. McGill, Jonathan A. DeCastro, Luke Fletcher, John J. Leonard, Brian C. Williams, and Guy Rosman. Carpal: Confidence-aware intent recognition for parallel autonomy, 2021. URL https://arxiv.org/abs/2003.08003.

[20] Lukas Huber, Aude Billard, and Jean-Jacques Slotine. Fast obstacle avoidance based on real-time sensing, 2022. URL https://arxiv.org/abs/2205.04928.

[21] Shervin Javdani, Siddhartha S. Srinivasa, and J. Andrew Bagnell. Shared autonomy via hindsight optimization, 2015. URL https://arxiv.org/abs/1503.07619.

[22] Hong Jun Jeon, Dylan P. Losey, and Dorsa Sadigh. Shared autonomy with learned latent actions, 2020. URL https://arxiv.org/abs/2005.03210.

[23] Hong Jun Jeon, Smitha Milli, and Anca D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning, 2020. URL https://arxiv.org/abs/2002.04833.

[24] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control, 2018. URL https://arxiv.org/abs/1812.03201.

[25] Burak Kizilkaya, Changyang She, Guodong Zhao, and Muhammad Ali Imran. Intelligent mode-switching framework for teleoperation, 2024. URL https://arxiv.org/abs/2402.06047.

[26] Sangjin Ko and Reza Langari. Reinforcement learning for shared driving. *IFAC-PapersOnLine*, 56(3):247–252, 2023. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2023.12.032. URL https://www.sciencedirect.com/science/article/pii/S2405896323023650. 3rd Modeling, Estimation and Control Conference MECC 2023.

[27] David Kortenkamp, R. Bonasso, Dan Ryan, and D. Schreckenghost. Traded control with autonomous robots as mixed initiative interaction. 01 2009.

[28] Danica Kragic, Panadda Marayong, Ming Li, {Allison M.} Okamura, and {Gregory D.} Hager. Human-machine collaborative systems for microsurgical applications. *Springer Tracts in Advanced Robotics*, 15:162–171, 2005. ISSN 1610-7438. doi: 10.1007/11008941_18.

[29] Zhen Lei, Bang Yi Tan, Neha P. Garg, Lei Li, Ananda Sidarta, and Wei Tech Ang. An intention prediction based shared control system for point-to-point navigation of a robotic wheelchair. *IEEE Robotics and Automation Letters*, 7(4):8893–8900, 2022. doi: 10.1109/LRA.2022.3189151.

[30] Ming Li and A.M. Okamura. Recognition of operator motions for real-time assistance using virtual fixtures. In *11th Symposium on Haptic Interfaces for Virtual*

Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings., pages 125–131, 2003. doi: 10.1109/HAPTIC.2003.1191253.

[31] Ming Li, Yu Kang, Yun-Bo Zhao, Jin Zhu, and Shiyi You. Shared autonomy based on human-in-the-loop reinforcement learning with policy constraints. In *2022 41st Chinese Control Conference (CCC)*, pages 7349–7354, 2022. doi: 10.23919/CCC55666.2022.9902295.

[32] Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real robotic gaussian splatting simulator, 2025. URL https://arxiv.org/abs/2411.11839.

[33] Yinglin Li, Rongxin Cui, Weisheng Yan, Shi Zhang, and Chenguang Yang. Reconciling conflicting intents: Bidirectional trust-based variable autonomy for mobile robots. *IEEE Robotics and Automation Letters*, 9(6): 5615–5622, 2024. doi: 10.1109/LRA.2024.3396100.

[34] Cunqiu Liu, Junyao Gao, Yi Liu, Xuanyang Shi, Fangzhou Zhao, Jingchao Zhao, and Chuzhao Liu. Admittance control of manipulators in unknown environment. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, page 2157–2162. IEEE Press, 2018. ISBN 978-1-5386-6074-4. doi: 10.1109/ICMA.2018.8484383. URL https://doi.org/10.1109/ICMA.2018.8484383.

[35] Huihan Liu, Rutav Shah, Shuijing Liu, Jack Pittenger, Mingyo Seo, Yuchen Cui, Yonatan Bisk, Roberto Martín-Martín, and Yuke Zhu. Casper: Inferring diverse intents for assistive teleoperation with vision language models, 2025. URL https://arxiv.org/abs/2506.14727.

[36] Xin Ma, Chengzhi Song, Philip Waiyan Chiu, and Zheng Li. Visual servo of a 6-dof robotic stereo flexible endoscope based on da vinci research kit (dvrk) system. *IEEE Robotics and Automation Letters*, 5(2):820–827, 2020. doi: 10.1109/LRA.2020.2965863.

[37] Brandon J. McMahan, Zhenghao Peng, Bolei Zhou, and Jonathan C. Kao. Shared autonomy with ida: Interventional diffusion assistance, 2024. URL https://arxiv.org/abs/2409.15317.

[38] Elle Miller, Maximilian Durner, Matthias Humt, Gabriel Quere, Wout Boerdijk, Ashok M. Sundaram, Freek Stulp, and Jorn Vogel. Unknown object grasping for assistive robotics, 2024. URL https://arxiv.org/abs/2404.15001.

[39] Mukund Mitra, Gyanig Kumar, Partha Pratim Chakrabarti, and Pradipta Biswas. Enhanced human-robot collaboration with intent prediction using deep inverse reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7880–7887, 2024. doi: 10.1109/ICRA57147.2024.10610595.

[40] Mayank Mittal, Pascal Roth, James Tigue, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[41] Rocco Moccia, Cristina Iacono, Bruno Siciliano, and Fanny Ficuciello. Vision-based dynamic virtual fixtures for tools collision avoidance in robotic surgery. *IEEE Robotics and Automation Letters*, 5(2):1650–1655, 2020. doi: 10.1109/LRA.2020.2969941.

[42] Vivek Myers, Erdem Bıyık, and Dorsa Sadigh. Active reward learning from online preferences, 2023. URL https://arxiv.org/abs/2302.13507.

[43] Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, Ankur Handa, and Dieter Fox. Factory: Fast contact for robotic assembly, 2022. URL https://arxiv.org/abs/2205.03532.

[44] Patrick Naughton, James Seungbum Nam, Andrew Stratton, and Kris Hauser. Integrating open-world shared control in immersive avatars, 2024. URL https://arxiv.org/abs/2401.03079.

[45] Heramb Nemlekar, Jignesh Modi, Satyandra K. Gupta, and Stefanos Nikolaidis. Two-stage clustering of human preferences for action prediction in assembly tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3487–3494, 2021. doi: 10.1109/ICRA48506.2021.9561649.

[46] Eley Ng, Ziang Liu, and Monroe Kennedy III. Diffusion co-policy for synergistic human-robot collaborative tasks, 2023. URL https://arxiv.org/abs/2305.12171.

[47] Idil Ozdamar, Marco Laghi, Giorgio Grioli, Arash Ajoudani, Manuel G. Catalano, and Antonio Bicchi. A shared autonomy reconfigurable control framework for telemanipulation of multi-arm systems, 2022. URL https://arxiv.org/abs/2207.09813.

[48] Sangbeom Park, Yoonbyung Chai, Sunghyun Park, Jeongeun Park, Kyungjae Lee, and Sungjoon Choi. Semi-autonomous teleoperation via learning non-prehensile manipulation skills, 2022. URL https://arxiv.org/abs/2109.13081.

[49] Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante, Daniel Leidner, Freek Stulp, and Jörn Vogel. Shared control templates for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1956–1962, 2020. doi: 10.1109/ICRA40945.2020.9197041.

[50] Gabriel Quere, Freek Stulp, David Filliat, and João Silvério. A probabilistic approach for learning and adapting shared control skills with the human in the loop. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15728–15734, 2024. doi: 10.1109/ICRA57147.2024.10610956.

[51] Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. Maximum margin planning. In *Proceedings of (ICML) International Conference on Machine Learning*, pages 729 – 736, July 2006.

[52] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. Shared autonomy via deep reinforcement learning, 2018. URL https://arxiv.org/abs/1802.01744.

[53] Siddharth Reddy, Anca D. Dragan, and Sergey Levine.

Where do you think you're going?: Inferring beliefs about dynamics from behavior, 2019. URL https://arxiv.org/abs/1805.08010.

[54] Charles Schaff and Matthew R. Walter. Residual policy learning for shared autonomy. In *Robotics: Science and Systems (RSS)*, 2020. URL https://arxiv.org/abs/2004.05097.

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

[56] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *CoRR*, abs/1812.06298, 2018. URL https://arxiv.org/abs/1812.06298.

[57] Nicholas A. Strohmeyer, Ji Hwan Park, Braden P. Murphy, and Farshid Alambeigi. A semi-autonomous data-driven shared control framework for robotic manipulation and cutting of an unknown deformable tissue. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9881–9886, 2024. doi: 10.1109/ICRA57147.2024.10610931.

[58] Luzhe Sun, Jingtian Ji, Xiangshan Tan, and Matthew R. Walter. Flashback: Consistency model-accelerated shared autonomy, 2025. URL https://arxiv.org/abs/2505.16892.

[59] Gokul Swamy, Siddharth Reddy, Sergey Levine, and Anca D. Dragan. Scaled autonomy: Enabling human operators to control robot fleets, 2020. URL https://arxiv.org/abs/1910.02910.

[60] Weihao Tan, David Koleczek, Siddhant Pradhan, Nicholas Perello, Vivek Chettiar, Vishal Rohra, Aaslesha Rajaram, Soundararajan Srinivasan, H M Sajjad Hossain, and Yash Chandak. On optimizing interventions in shared autonomy, 2022. URL https://arxiv.org/abs/2112.09169.

[61] Yoshihiro Tanaka, Takumi Katagiri, Hikari Yukawa, Takumi Nishimura, Ryohei Tanada, Itsuki Ogura, Takayoshi Hagiwara, and Kouta Minamizawa. Sensorimotor control sharing with vibrotactile feedback for body integration through avatar robot. *IEEE Robotics and Automation Letters*, 7(4):9509–9516, 2022. doi: 10.1109/LRA.2022.3191191.

[62] Ran Tian, Nan Li, Anouck Girard, Ilya Kolmanovsky, and Masayoshi Tomizuka. Cost-effective sensing for goal inference: A model predictive approach. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6451–6457, 2022. doi: 10.1109/ICRA46639.2022.9811974.

[63] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *Arxiv*, 2024.

[64] Pete Trautman. Assistive planning in complex, dynamic environments: A probabilistic approach. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3072–3078, 2015. doi: 10.1109/SMC.2015.534.

[65] Linda F. van der Spaa, Giovanni Franzese, Jens Kober, and Michael Gienger. Disagreement-aware variable impedance control for online learning of physical human-robot cooperation tasks. 2022. URL https://api.semanticscholar.org/CorpusID:252276977.

[66] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions, 2025. URL https://arxiv.org/abs/2411.16627.

[67] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects, 2024. URL https://arxiv.org/abs/2312.08344.

[68] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2024. URL https://arxiv.org/abs/2309.13037.

[69] Yuxuan Wu, Lei Pan, Wenhua Wu, Guangming Wang, Yanzi Miao, Fan Xu, and Hesheng Wang. Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning, 2025. URL https://arxiv.org/abs/2409.20291.

[70] Yuan Yang, Daniela Constantinescu, and Yang Shi. Proportional and reachable cluster teleoperation of a distributed multi-robot system. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8984–8990, 2021. doi: 10.1109/ICRA48506.2021.9561156.

[71] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, Mike Lambeta, Tingfan Wu, Pieter Abbeel, and Mustafa Mukadam. Dexteritygen: Foundation controller for unprecedented dexterity, 2025. URL https://arxiv.org/abs/2502.04307.

[72] Takuma Yoneda, Luzhe Sun, , Ge Yang, Bradly Stadie, and Matthew Walter. To the noise and back: Diffusion for shared autonomy, 2023. URL https://arxiv.org/abs/2302.12244.

[73] Ehsan Yousefi, Dylan P. Losey, and Inna Sharf. Assisting operators of articulated machinery with optimal planning and goal inference. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2832–2838, 2022. doi: 10.1109/ICRA46639.2022.9811864.

[74] Mohammad Kassem Zein, Abbas Sidaoui, Daniel Asmar, and Imad H. Elhajj. Enhanced teleoperation using autocomplete. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9178–9184, 2020. doi: 10.1109/ICRA40945.2020.9197140.

[75] Lihan Zha, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections, 2024. URL https://arxiv.org/abs/2311.10678.

[76] Dandan Zhang, Zicong Wu, Junhong Chen, Ruiqi Zhu, Adnan Munawar, Bo Xiao, Yuan Guan, Hang Su,

Wuzhou Hong, Yao Guo, Gregory S. Fischer, Benny Lo, and Guang-Zhong Yang. Human-robot shared control for surgical robot based on context-aware sim-to-real adaptation, 2022. URL https://arxiv.org/abs/2204.11116.

[77] Heng Zhang, Lifeng Zhu, Jiangwei Shen, and Aiguo Song. Implicit neural field guidance for teleoperated robot-assisted surgery. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6866–6872, 2023. doi: 10.1109/ICRA48891.2023.10160475.

[78] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1433–1438. AAAI Press, 2008. ISBN 9781577353683.

APPENDIX

Contents

APPENDIX A
ADDITIONAL TECHNICAL DETAILS

A. Hardware System

1) Robot Setup: We use a UFactory xArm 7 mounted on a tabletop, shown in Fig. 5. The robot has 7 degrees of freedom. A wrist-mounted force–torque sensor is attached at the end-effector (link 7), with a standard xArm parallel gripper mounted to the sensor. An Intel RealSense D455 RGB-D camera is placed in front of the robot to provide an overhead view of the workspace.

2) Residual Implementation: We use GELLO [68] as the teacher arm for teleoperation. For direct teleoperation, GELLO joint angles are streamed as joint-position commands directly to the xArm at 15 Hz. For residual-assisted teleoperation, we convert GELLO joints to a task-space end-effector pose via forward kinematics and treat it as a Cartesian target.

We additionally condition the residual copilot on the current observation and this Cartesian target; it predicts a corrective residual that is composed with the target before solving inverse kinematics to obtain robot joint targets, which are then streamed as joint-position commands to the xArm at 15 Hz. For safety, both methods clip the step-wise change in joint targets with an ℓ_2 bound consistent with a joint-velocity limit.

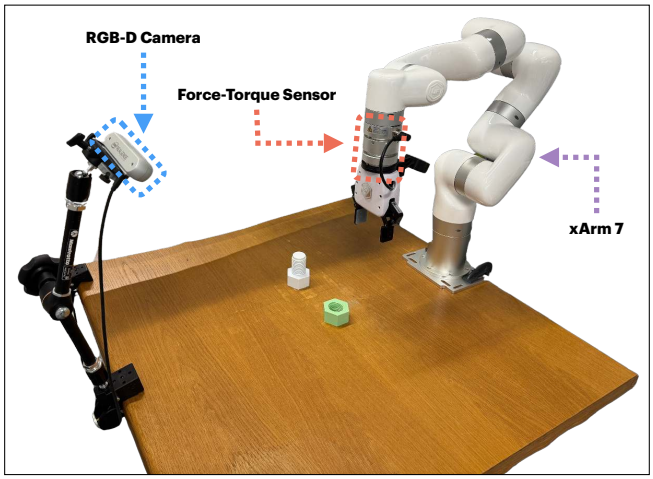

Fig. 5: **xArm 7 Setup.**

Controller	Parameter	Real	Sim	Unit
Task-space admittance controller	K_x	1000	200	N/m
	K_r	10	100	Nm/rad
	M_x	0.1	0.125	kg
	M_r	1×10^{-3}	0.015	kg·m^2
	D_x	0	5	N·s/m
	D_r	0	1.2	N·m·s/rad
	adm. axes	6	6	DoF
Joint-space PID controller	K_p	–	200	N·m/rad
	K_i	–	20	N·m/(rad·s)
	K_d	–	0	N·m·s/rad

TABLE III: **Controller parameters.** Real-world and simulation settings for the task-space admittance and joint-space PID controllers.

3) Admittance Control: We enable the xArm 7's built-in FT-sensor admittance mode at the end-effector, with 6-DoF compliance enabled (i.e., compliance along all translational and rotational axes) in the robot base frame. Damping is set to zero and we rely on the robot's internal low-level servo for stability. Admittance control parameters can be found in Table III.

4) State Estimation: We estimate object states using FoundationPose [67] in a separate thread running at 15 Hz. The environment asynchronously retrieves the latest object pose estimates at its own timestep. To improve sim-to-real alignment, we estimate the robot's linear and angular velocities in both simulation and the real world using finite differences followed by low-pass filtering.

B. Residual Policy Training

1) Representation Spaces: We define the following representations for the human surrogate (base) and residual action and observation spaces.

Base action space. We use an 8D action comprising an end-effector target pose in the robot base frame and a gripper command. The first three dimensions specify the target fingertip position, the next four specify the target fingertip orientation (quaternion), and the final dimension is a continuous gripper openness command in $[-1, 1]$.

Group	Symbol(s)	Distribution / value
Ctrl.	K_x, K_r, M_x, M_r	$\mathcal{U}(0.95,\ 1.05) \odot (\cdot)$
Pose	$\Delta p_{\text{fix}}, \Delta p_{\text{held}}$	$\mathcal{N}(0,\ 0.002^2 I)$
Init.	$\Delta p_0, \Delta\theta_0$	$\mathcal{N}(0,\ 0.02^2 I),\ \mathcal{N}(0,\ 0.035^2 I)$
Base	L	$L \sim \mathcal{U}\{5, \ldots, 15\}$
	ε_t	$\mathcal{U}([-0.6, 0.6]^d)$
	β, p_{on}	$\beta = 0.8,\ p_{\text{on}} = 0.5$

TABLE IV: **Domain randomization (DMR).** Per-episode sampling unless noted; base-action DMR is sampled per timestep. For base-action randomization, L denotes the length of kNN-retrieved action chunks, ε_t is i.i.d. residual noise, and (β, p_{on}) parameterize a smooth stochastic gate controlling the activation frequency and temporal correlation of the noise.

Category	Name	Term	Scale
$\mathcal{R}_{\text{general}}$	Regularization	$-\lVert a_{\text{res}} \rVert$	0.1
	Tilt Penalty	$-\theta_{\text{tilt}}$	1.0
	Force Penalty	$-\phi(F)$	0.2
	Axis Align	$\mathbf{1}[\lVert p_{xy} - p^*_{xy} \rVert < \epsilon]$	0.05
	Smoothness	$-\lVert a^{prev}_{\text{res}} - a_{\text{res}} \rVert$	0.1
	Termination	$-\mathbf{1}[\text{fail / timeout}]$	50.0
$\mathcal{R}_{\text{success}}$	Success	$\mathbf{1}[\text{first success}]$	30.0

TABLE V: **Reward terms.** a_{res} and a^{prev}_{res} denote current and previous normalized residual actions, θ_{tilt} the end-effector tilt angle from upright, $\phi(F)$ a clipped contact-force penalty, and p_{xy} the held-object planar position. The success reward is issued once per episode upon first satisfying the task-specific success condition. The termination condition is dropping the held object early on.

Residual action space. We use a 7D residual that is applied relative to the base action. The first three dimensions are a fingertip position delta, the next three are an axis–angle rotation delta, and the final dimension is a gripper openness delta.

Residual observation space. The residual policy receives a 35D vector consisting of fingertip position (3), fingertip quaternion (4), gripper openness (1), fingertip position relative to the fixed object (3), fingertip position relative to the held object (3), end-effector linear velocity (3), end-effector angular velocity (3), base action (8), and previous residual action (7).

Residual composition. Let the base orientation be represented as a quaternion q and the rotational residual as an axis–angle vector $\delta\theta \in \mathbb{R}^3$. We convert $\delta\theta$ to a quaternion δq and update the orientation via $q' = \delta q \otimes q$, where $\otimes$ denotes quaternion multiplication. Translational and gripper residuals are applied additively.

Normalization. All action and observation dimensions are normalized to $[-1, 1]$ using running mean and standard deviation statistics updated online during training and frozen during inference.

2) Reward Terms: Table V lists the complete reward terms. For Gear Meshing and Peg Insertion, we use a sparse success reward that triggers when the held object reaches a predefined relative pose corresponding to a successful insertion (position error $<$ 2.5 mm). For NUT THREADING, success requires both reaching the predefined assembly pose and achieving a cumulative 180° rotation of the nut about its local yaw axis in the tightening direction while maintaining the prescribed screwing pose.

3) Domain Randomization: To improve robustness and sim-to-real transfer, we apply domain randomization (DMR) at multiple levels, summarized in Table IV. We use controller randomization to account for residual controller mismatch between simulation and hardware, object pose randomization to model real-world pose estimation errors, and initial configuration randomization to promote generalization. In addition, we randomize the human surrogate by injecting temporally correlated residual noise to improve robustness of both the surrogate and the residual copilot.

4) RL Hyperparameters: We train policies with PPO using an actor–critic architecture with a shared recurrent backbone (LSTM) followed by an MLP, and a Gaussian policy head.

Hyperparameter	**Value**
Architecture	
Backbone	LSTM (2 layers, 1024 units)
MLP	[512, 128, 64], ELU
Actor–critic	shared trunk
Policy head	Gaussian (learnable $\log\sigma$)
PPO / Optimization	
γ / λ	0.995 / 0.95
LR schedule	adaptive (base lr 10^{-4})
KL threshold	0.008
Clip ϵ	0.2
Entropy coef.	0.0
Critic coef.	2.0
Grad clip	1.0
Update epochs	4
Rollout horizon / actors	128 / 128
Minibatch size	512

TABLE VI: **RL hyperparameters.** Actor–critic architecture and PPO settings used for training.

Category	**State DP**	**Vision DP**
Inputs / Outputs		
Observation	$s_t \in \mathbb{R}^{20}$	RGB (200×200) + $s_t \in \mathbb{R}^8$
Action	$a_t \in \mathbb{R}^8$	$a_t \in \mathbb{R}^8$
Horizon (H)	16	16
Action steps	8	8
Diffusion model		
Noise scheduler	DDPM	DDPM
Train timesteps	100	100
Inference steps	10	10
β schedule	Squared cosine	Squared cosine
Prediction target	ϵ (noise)	ϵ (noise)
FiLM modulation	Enabled	Enabled
Architecture		
Encoder	None	ResNet-18
U-Net dims	[512, 1024, 2048]	[512, 1024, 2048]
Group norm groups	8	8
Optimization		
Batch size	256	512
Optimizer	Adam (lr $= 10^{-4}$)	Adam (lr $= 10^{-4}$)
LR schedule	Cosine	Cosine
Weight decay	10^{-6}	10^{-6}
Training steps	40k	10k

TABLE VII: **Diffusion policy hyperparameters.** State- and vision-based variants share the diffusion process and training setup, differing only in input modality, encoder, and training scale.

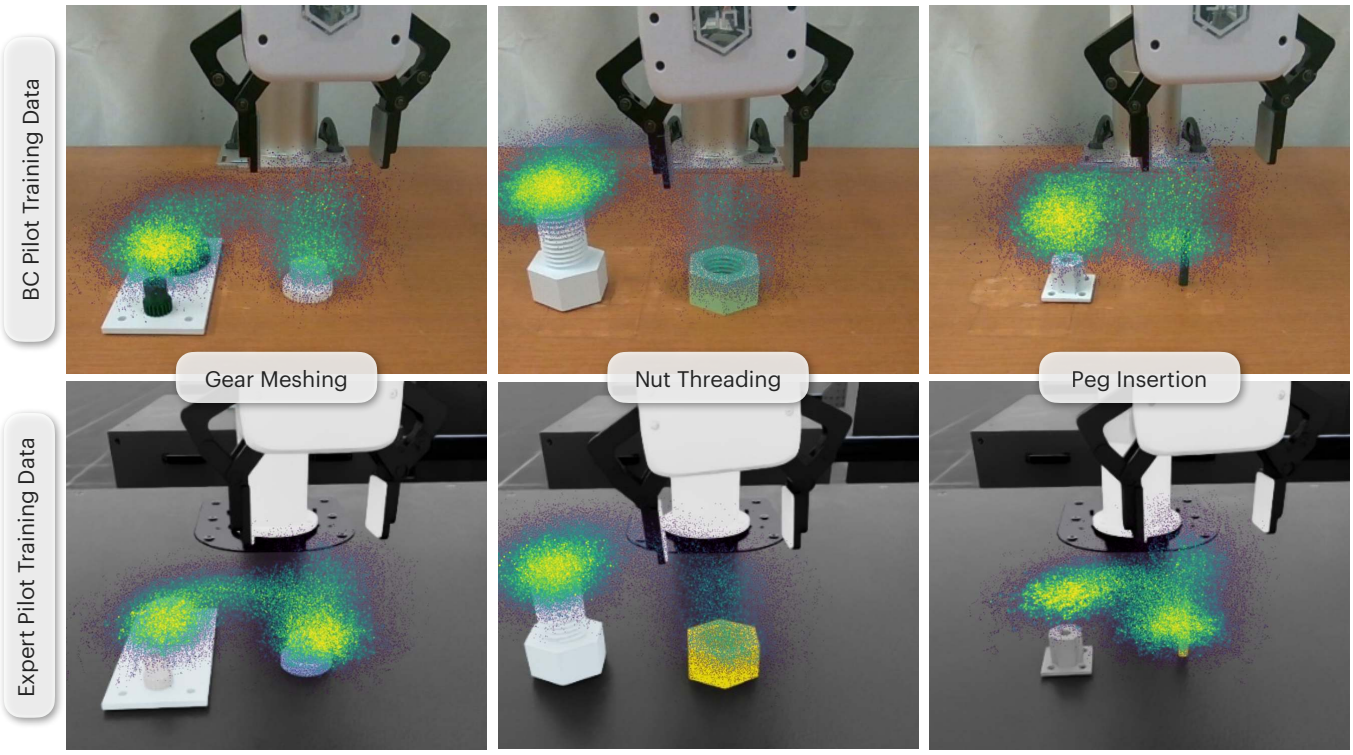

Fig. 6: **State-based DP training data distributions.** Spatial coverage of 3D fingertip positions, downsampled to 100k points for visualization. The top row shows training data for `BC Pilot`, obtained by augmenting real-world teleoperation demonstrations; the bottom row shows training data for `Expert Pilot`, collected from successful rollouts of the residual copilot policies. Columns correspond to the three tasks: Gear Meshing, Nut Threading, and Peg Insertion. The top row (left to right) contains 442k, 786k, and 427k transitions, while the bottom row contains 220k, 758k, and 195k transitions.

We apply normalization to observations, value targets, and advantages. Key architecture and PPO hyperparameters are summarized in Table VI.

C. Simulation Setup

1) Admittance Control: We deploy the same task-space admittance model in simulation as in Sec. III-C. At each control step, we represent the end-effector pose as $\xi \in \mathbb{R}^6$ (position and axis–angle orientation) with reference ξ_{ref}, and compute the pose error $e = \xi - \xi_{\text{ref}}$. We maintain the task-space velocity $\dot{\xi} \in \mathbb{R}^6$ and use diagonal virtual parameters (M, D, K) to form the spring–damper wrench

$$f_{\text{sd}} = D\dot{\xi} + Ke.$$

Given the measured external wrench f, we integrate the admittance dynamics with forward Euler:

$$\ddot{\xi} = M^{-1}(f - f_{\text{sd}}), \quad \dot{\xi} \leftarrow \dot{\xi} + \Delta t\, \ddot{\xi}, \quad \xi \leftarrow \xi + \Delta t\, \dot{\xi}.$$

We update orientation by converting the angular component of $\Delta t\, \dot{\xi}$ to an incremental axis–angle rotation and applying it via quaternion composition.

The task-space admittance control and joint-space PD parameters can be found in Table III.

2) Task Progression Metric: We define task progression as $1 - e/e_{\max}$, where error e is clipped to $[0, e_{\max}]$ so that the metric lies in $[0, 1]$. A value of 1 indicates task completion (zero error), while 0 indicates no meaningful progress. For Gear Meshing and Peg Insertion, error is the Euclidean distance between the end-effector pose and a pre-defined success pose, with $\text{e}_{\max} = 15\,\text{cm}$. For Nut Threading, error is the remaining yaw rotation needed after insertion, with $\text{e}_{\max} = 90$.

3) Physics Engine: We use the NVIDIA PhysX engine for rigid-body simulation with a configuration tailored for stable contact-rich manipulation. We employ the Temporal Gauss–Seidel (TGS) solver and set a high number of position iterations (192) to reduce interpenetration and improve contact resolution, while keeping a single velocity iteration for efficiency. To stabilize frictional contact, we use conservative friction thresholds and correlation distances, and set a low bounce threshold velocity (0.2) to suppress spurious rebounds. The GPU solver is provisioned with large contact and collision buffers, and we restrict partitioning to a single partition to ensure deterministic and stable contact dynamics.

D. Behavior Cloning Policy Training

1) State-based Diffusion Policy: We train state-based Diffusion Policies for `BC Pilot` and `Expert Pilot` in IV-A using a standard DDPM formulation with a fixed action

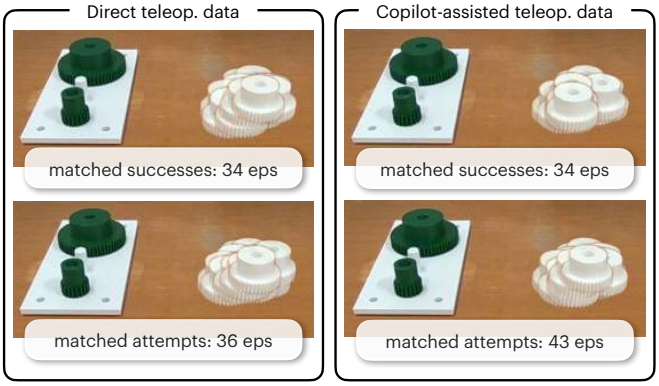

Fig. 7: **Vision-based DP training data distributions.** Spatial coverage visualization of initial states in the training set used to train vision-based diffusion policies. Columns show episodes collected via direct vs. copilot-assisted teleoperation. Rows correspond to the two controlled IV-C settings (same number of successful episodes and same number of teleoperation attempts), isolating how data quality and collection efficiency affect downstream policy learning.

horizon. The policy operates on a low-dimensional state representation s_t that closely mirrors the residual observation space but excludes action-dependent terms. Specifically, s_t includes proprioceptive and task-relevant variables (end-effector pose, gripper state, and relative object poses), while omitting the base action and previous residual action, yielding a 20-dimensional state vector.

For `BC Pilot`, we expand the teleoperation dataset to 2000 episodes by applying global geometric augmentations—random translations and rotations—to entire trajectories. Translations are sampled from $\mathcal{N}(0, 0.02)$ and rotations from $\mathcal{N}(0, 0.087)$. For `Expert Pilot`, we collect 2000 successful episodes by rolling out the corresponding `Residual Copilot` policy. The resulting training data distributions for these state-based diffusion policies are shown in Fig. 6.

The diffusion process, noise schedule, and optimization settings are shared across the state- and vision-based variants. The two policies differ only in their input modalities and encoder architectures, while the diffusion decoder and training procedure remain identical. Key diffusion and optimization hyperparameters are summarized in Table VII.

2) Vision-based Diffusion Policy: We additionally train several vision-based Diffusion Policies for IV-C, where action-sequence predictions are conditioned on visual observations. In this setting, the state input s_t is restricted to the robot's proprioceptive state (i.e., end-effector position, orientation, and gripper openness), yielding an 8-dimensional vector. Visual information is provided separately via a single 200×200 RGB image from the front D455 camera. Fig. 7 visualizes the spatial coverage of the initial workspace for each dataset.

RGB observations are encoded using a ResNet-18 backbone with a spatial softmax keypoint layer, whose output is fused with the low-dimensional state before diffusion-based action decoding. Aside from the visual encoder and the reduced state definition, the diffusion model, horizon, and training hyperparameters are shared with the state-based policy. Standard image augmentations are applied during training to improve robustness. All architectural and diffusion hyperparameters common to both variants are reported in Table VII.

E. User Study Details

1) Participant Information: We recruited 16 non-author participants (ages 20–30), all male students. Novice teleoperators had limited prior experience, having collected fewer than 20 teleoperation trajectories (often their first exposure to teleoperation). Experienced teleoperators had substantially more prior experience, including prior data collection for imitation learning, and had collected at least 50 teleoperation trajectories.

2) Participant Questionnaire: All subjective metrics were collected using 7-point Likert-scale questions. Each question followed the same format: *"How much X did you feel when using method Y?"* where *X* denotes the criterion (e.g., mental demand, effort, trust, satisfaction) and *Y* denotes the teleoperation method. Responses ranged from 1 (least *X*) to 7 (most *X*). Criteria labeled as *User Demand* are better when lower, while criteria labeled as *User Satisfaction* are better when higher.

3) Participant Instructions: The full instruction sheet provided to participants prior to the experiment is included on the final page of the appendix (Sec. B).

APPENDIX B
PARTICIPANT INSTRUCTION SHEET

Method Overview. In this study, you will perform teleoperation using two different methods:

1) **Direct Teleoperation.** The robot directly mirrors the motion of the teacher arm by matching its end-effector position and orientation. You have full control of the robot and are responsible for completing the entire task from start to finish.
2) **Copilot-Assisted Teleoperation.** In this mode, a copilot agent assists your teleoperation. Your commands from the teacher arm are combined with corrective actions from the copilot before being executed by the robot. You share control with the copilot: you should focus on high-level, coarse motions, while the copilot corrects local position and orientation errors. For insertion tasks, the copilot will complete the insertion once the robot is near the insertion region. For screwing tasks, the copilot will amplify your rotations and regulate contact force.

Teleoperation Objective. Your teleoperation objective has three priorities:

1) **Safety and contact regulation.** Always prioritize safe interaction. Noticeable shakiness during contact indicates excessive force and should be avoided.
2) **Task success.** Try to successfully complete the task to the best of your ability.
3) **Input smoothness.** Aim for smooth motions. Avoid sudden accelerations and excessively fast movements.

Task Descriptions. You will perform one of the following tasks:

- **Gear Meshing:** Pick up the medium gear, insert it onto its shaft, and mesh it with the other gears. A trial is successful only if the gear remains grasped until insertion is fully completed.
- **Nut Threading:** Pick up an M32 nut and tighten it onto the bolt by at least 180°. Due to hardware limits, this is performed as three consecutive 60° turns.
- **Peg Insertion:** Pick up an $8\,\mathrm{mm}$-long peg and insert it into the base. A trial is successful only if the peg reaches the bottom of the receptacle.

For all tasks, failure is defined as either dropping the object before success or exceeding the xArm safety force threshold.

Experiment Procedure. Before the experiment begins, you will have 5 minutes to familiarize yourself with each teleoperation method. During the experiment, you will alternate between direct teleoperation and copilot-assisted teleoperation. At the start of each trial, you will be told which method to use.

At the beginning of every trial, place the teacher arm on the mount to ensure a consistent initial pose. After the experimenter says "start," you may move the arm off the mount and begin teleoperating. The experimenter will say "stop" when the trial ends due to either success or failure, and then the next trial will begin.

You will perform only one task in the study. The number of trials per method is:

- **Gear meshing**: 10 trials per method.
- **Peg insertion**: 8 trials per method.
- **Nut threading**: 5 trials per method.

After completing all trials, you will fill out a questionnaire about your experience with each method, including workload and satisfaction.