OMNITACTUNE: Policy-Agnostic Real-World RL for Tactile Residual Adaptation of Visual Policies

Kelin Yu[1]* **Haode Zhang**[1]* **Harish Ravichandar**[2] **Yunhai Han**[2] **Ruohan Gao**[1]
[1]University of Maryland, College Park [2]Georgia Institute of Technology
*Equal contribution

Project page: https://colinyu1.github.io/omnitactune-site/

Abstract: Visual policies learned from human videos, teleoperation, and robot demonstrations offer scalable motion priors, but often fail in contact-rich manipulation, where success significantly depends on local force and contact geometry. Tactile sensing provides these complementary signals, yet tactile data remain costly to collect and hard to generalize across sensors, robots, and tasks. We introduce OMNITACTUNE, a policy-agnostic real-world RL pipeline that adapts tactile feedback to pretrained visual policies through residual correction. OMNITACTUNE uses a two-stage design: it first warm-starts tactile-aware learning from autonomous base-policy rollouts, then learns a lightweight tactile residual policy through online interaction. Extensive experiments show that OMNITACTUNE generalizes across diverse contact-rich tasks, visual base policies, and tactile representations. Across four real-world contact-rich tasks, it improves visual base policies from 5–40% success to 85–100% within 40–80 minutes, demonstrating an efficient path for adapting tactile feedback to scalable visual robot policies.

Keywords: Visuo-Tactile Policy, Real-World RL, Contact-rich Manipulation

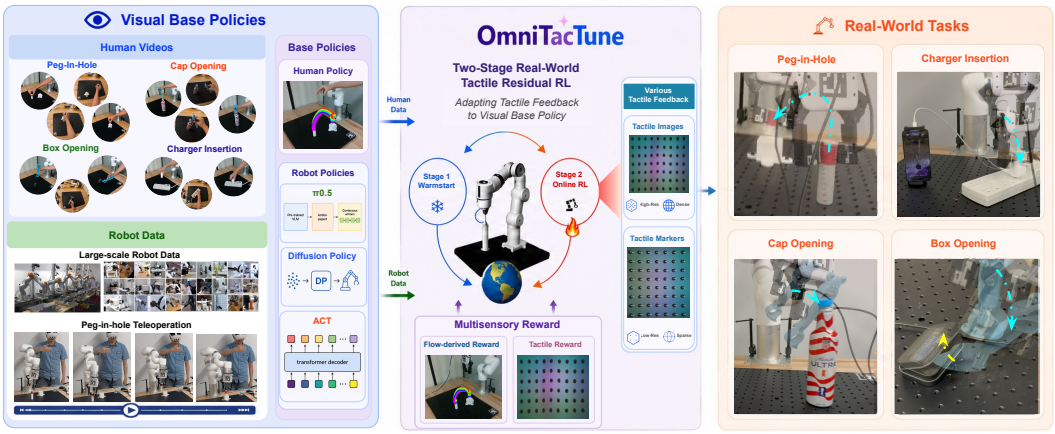

Figure 1: **OMNITACTUNE** adapts tactile feedback to diverse visual base policies trained from human videos or robot data (**left**) through a two-stage real-world tactile residual RL pipeline (**middle**), enabling efficient tactile adaptation across challenging contact-rich manipulation tasks (**right**).

1 Introduction

Visual robot policy learning has made tremendous progress, driven in large part by scaling. The field is moving toward increasingly scalable data sources, from teleoperated robot demonstrations [1, 2] to more recently large-scale human videos [3–5], offering a promising path toward general visual policies across diverse tasks, environments, and embodiments. However, scaling visual data alone does not resolve the challenges of contact-rich manipulation. Cameras provide rich spatial observations for planning, but they cannot directly measure contact or forces during physical interaction, such as insertion and tool use. As a result, many failures occur precisely at contact, where unobserved physical interactions determine whether a task succeeds or fails.

Touch, on the other hand, provides direct access to local interaction signals critical for contact-rich manipulation, including normal force, shear force, and contact geometry [6, 7]. Prior work has

shown that tactile sensing improves tasks requiring precise contact reasoning [7–10], but these methods often rely on task-specific tactile data collected with a particular sensor or setup [11–14], limiting cross-platform transfer. Despite recent efforts to scale tactile data [12, 14, 15], tactile datasets remain orders of magnitude smaller than visual datasets [1, 4, 16]. This creates a fundamental scale gap, as tactile observations are rarely paired with the large-scale visual data from which general robot behaviors are increasingly learned.

Our key idea is to leverage the best of both worlds, rather than scale tactile data to the level of vision. Vision and touch play different roles in manipulation, much like in human learning. By watching others, humans can infer the global structure of a task: what objects are involved, how they move, and what action sequence is likely to succeed. Yet reliable execution often comes only through hands-on practice, where touch reveals whether parts are aligned, resistance is too high, and small adjustments are needed. This distinction suggests a scalable paradigm for contact-rich manipulation: learn general visual policies from abundant visual data, then use tactile feedback to refine and correct these policies during contact. In this view, vision provides broad, scalable supervision for task-level behavior, while touch provides local, residual feedback for the last mile of physical interaction.

Learning such tactile residual corrections is challenging because the necessary supervision is revealed only through physical interaction. Prior real-to-sim-to-real approaches [17–19] are poorly suited to this setting, as they must bridge visual, dynamics, and tactile gaps simultaneously, while tactile signals and dynamic interactions are especially difficult to simulate. Imitation-based correction methods, such as DAgger-style approaches [20], can correct policy failures through expert interventions, but they still depend on repeated human demonstrations rather than autonomous improvement. We therefore turn to *real-world reinforcement learning (RL)* [7, 21–24] as a natural mechanism for learning tactile residuals. Continuing the human-learning analogy, we propose to use visual priors to guide the overall behavior and tactile feedback to refine execution through real-world practice, enabling the robot to improve its contact strategy from experience.

Towards this goal, we introduce **OMNITACTUNE** (Fig. 2), a two-stage real-world RL pipeline for tactile residual adaptation across visual base policies. Firstly, OMNITACTUNE leverages a pre-trained visual base policy, learned from scalable visual data such as human videos or robot demonstrations, as task-level motion priors. In the *first stage*, autonomous rollouts from base policies enable warm-start training, which initializes the replay buffer, bootstraps the critic, and adapts the tactile representation to the target task. In the *second stage*, OMNITACTUNE performs online residual RL to learn a tactile policy that predicts contact-aware corrections on top of the frozen visual policy. To improve sample efficiency and stability, we further introduce object-centric visuo-tactile reward shaping, which provides dense feedback and guides exploration during real-world practice. Together, this framework enables reliable residual adaptation from real-world interaction.

Our main contributions are threefold. *First*, we formulate tactile adaptation as residual correction on top of existing visual policies, enabling tactile-aware practice without training from scratch. *Second*, we introduce OMNITACTUNE, a plug-and-play two-stage real-world RL pipeline for online tactile residual adaptation, together with a multi-sensory reward that combines generated object-centric guidance with tactile contact signals for sample-efficient real-world refinement. *Third*, we demonstrate that OMNITACTUNE generalizes across contact-rich tasks, visual base policies, and tactile representations. On four challenging real-world contact-rich tasks, OMNITACTUNE consistently improves success rates from 5–40% to 85–100% within 40–80 minutes of online practice.

2 Related Work

Visual Policy Learning from Robot and Human Data. Recent robot learning methods increasingly leverage scalable visual data, including robot demonstrations, teleoperation data, and human videos, to learn general manipulation policies [1–4, 25–27]. For robot data, imitation-learning backbones such as ACT [25], Diffusion Policy [26], and recent vision-language-action models [28–30] have shown strong performance when sufficient robot demonstrations are available. For human videos, prior work reduces the embodiment gap through human-robot co-training [3, 31, 32], object-centric representations [27, 33–37], or interaction-based policy refinement [7, 38]. However, these

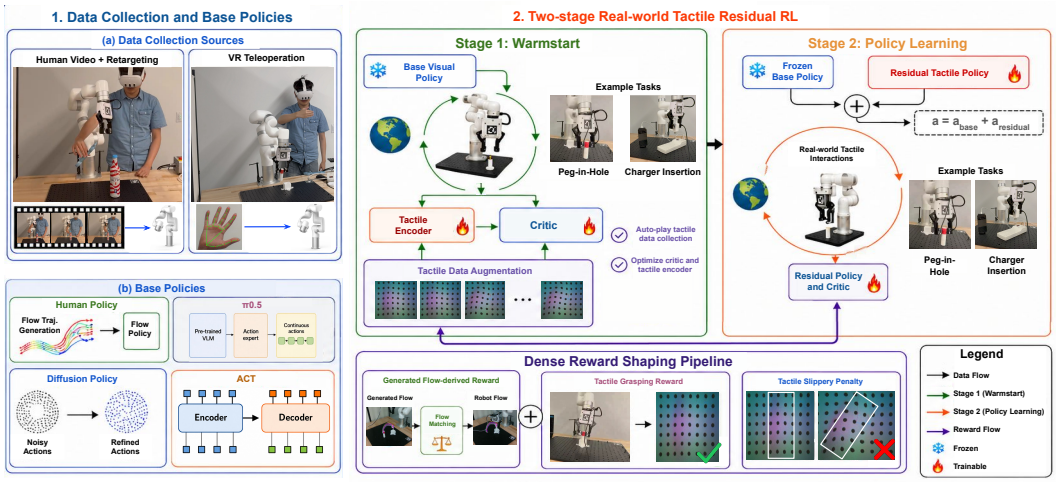

Figure 2: **System overview of OMNITACTUNE.** OMNITACTUNE first collects visual demonstrations via human video retargeting and VR teleoperation, and uses them to train diverse visual base policies (**left**). It then adapts tactile feedback through a two-stage real-world residual RL pipeline: a warm-start stage collects tactile rollouts to optimize the tactile encoder and critic, followed by an online policy learning stage that learns residual tactile corrections (**top right**). A dense reward shaping pipeline combines visuo-tactile feedback to guide efficient real-world learning (**bottom**).

visual policies still lack direct contact feedback, and we use them as frozen base policies while learning tactile residual corrections for contact-rich execution.

Tactile and Multi-Sensory Robot Manipulation. Tactile sensing provides local contact information that is difficult to infer from RGB observations, such as contact geometry and slip [6, 8, 39, 40], and has benefited a wide range of contact manipulation tasks, including grasping, insertion, in-hand manipulation, and dexterous manipulation [9, 10, 41–46]. Prior work incorporates tactile feedback through visuo-tactile sensory fusion [9], slow-fast multimodal architectures [10], tactile world model [43], and reinforcement learning [44, 47]. However, most existing visuo-tactile policy learning methods require paired, task-specific visuo-tactile demos to learn policies. In contrast, OMNITACTUNE adapts tactile feedback online as residual corrections on top of existing visual policies, enabling contact-aware practice without training a visuo-tactile policy from scratch.

Policy Adaptation for Contact-Rich Manipulation. Policy adaptation methods improve pretrained policies through additional interaction. DAgger-style methods [20, 48, 49] reduce compounding errors with expert corrections during execution, and CR-DAgger [49] further adapts force-reactive behaviors, but these methods still rely on repeated human intervention. Real-to-sim-to-real methods [17–19, 50] aim to reduce real-world data collection by adapting policies through simulation, but contact-rich manipulation is especially sensitive to simulation gaps in dynamics, contact, and tactile sensing. Real-world RL avoids these gaps by learning directly on hardware, with prior work improving sample efficiency through offline expert demos, pretrained policies, or warm-start rollouts [7, 21–24, 51–56]. Our method also uses real-world RL, but addresses a different adaptation setting: OMNITACTUNE adapts a visual base policy with newly introduced tactile feedback.

3 Methodology

We aim to build a policy-agnostic real-world RL pipeline for residual tactile adaptation across different visual base policies, tasks, and tactile representations. This setting introduces several bottlenecks that are not addressed by prior real-world RL pipelines [21, 22]. Offline demonstrations often provide only visual feedback, and therefore cannot directly initialize a tactile replay buffer or tactile-aware critic. Warm-start rollouts can collect on-policy data, but usually do not explicitly bootstrap the critic or optimize the tactile encoder. This is particularly limiting because pretrained tactile encoders can exhibit substantial representation shift across tasks and contact conditions.

To address these challenges, we introduce OMNITACTUNE (see Fig. 2). We first describe how we collect human videos and robot data to train different visual base policies (Sec. 3.1 and Sec. 3.2).

We then present our two-stage real-world RL pipeline (Sec. 3.3): during warm-start, the robot autonomously rolls out the frozen base policy to initialize the replay buffer, bootstrap the flow-tactile critic, and fine-tune the tactile encoder; during online training, residual RL learns tactile-informed corrections for the frozen visual policy. Finally, we introduce an object-centric multi-sensory reward that enables more stable and efficient RL training (Sec. 3.4). Details are shown in App. A.

3.1 Data Collection

Human Hand Demonstrations. Similar to HUDOR [57], we collect human demonstrations with a third-view RGB camera and a Meta Quest headset. The camera records in-scene videos of humans expert demonstrations, while the Quest tracks the 6-DoF hand pose under its camera frame ${}^{q}\mathbf{T}_h(t)$. We use tabletop Aruco markers and the third-view camera to retarget it to the robot base frame via the third-view camera, obtaining robot-aligned hand trajectories as ${}^{r}\mathbf{T}_h(t) = {}^{r}\mathbf{T}_w\,{}^{w}\mathbf{T}_q\,{}^{q}\mathbf{T}_h(t)$, where ${}^{w}\mathbf{T}_q$ maps the Quest tracking frame to the world frame, ${}^{r}\mathbf{T}_w$ maps the world frame to the robot base frame, and ${}^{r}\mathbf{T}_h(t)$ is the resulting robot-frame hand pose. The resulting synchronized videos and trajectories are used to extract object-centric motion priors and train the base policies.

Teleoperation Data Collection. In addition to hand demonstrations, we collect robot demonstrations using a Meta Quest headset via OpenTeach [58]. The OpenTeach maps the tracked human hand motion from quest to robot end-effector poses. We record synchronized RGB observations, robot states, and actions, which are used to train robot-data-based visual base policies.

3.2 Learning Visual Base Policies

In this section, we introduce four different base policies, where all of them can be used for residual tactile adaptations with our real-world RL pipeline.

Flow-based Policy. Our flow-based policy builds on top of Im2Flow2Act [27], GenFlowRL [33], and Dex4D [59], where we want to enable cross-embodiment visual policy learning from human videos, aiming to transfer task-level motion priors from videos to robot manipulation. Given a demonstration video, we extract and track a sparse set of object keypoints using DINOv2 [60] and SAM [61], and then using CoTracker 3 [62] to track their motions. Following Im2Flow2Act [27], we fine-tune a pretrained flow generator using our collected task-specific human or robot demonstrations to predict a complete task-level object flow from the initial observation. The generated flow provides two complementary signals for our system: an object-conditioned motion guidance for both base policy and residual policy, and a dense object-centric reward signal during residual RL.

To train the lightwight, embodiment-agnostic base policy, we convert the generated and observed object flows into compact policy inputs. At time t, the flow-guided base policy is formulated as $\mathbf{a}^{\mathrm{b}}_{t:t+K} = \pi^{\mathrm{flow}}_{\theta}(\mathbf{o}^{\mathrm{flow}}_t, \mathbf{q}_t)$, where $\mathbf{q}_t$ is the robot proprioceptive state and $\mathbf{o}_t$ is the flow representation, $\mathbf{o}^{\mathrm{flow}}_t = [\mathbf{c}^{3D}_0, \mathbf{c}_t, \mathbf{T}^{\mathrm{rel}}_{t_0\to t}, \hat{\mathbf{c}}_\ell, \hat{\mathbf{T}}^{\mathrm{rel}}_{t_0\to \ell}, \mathbf{T}^{\mathrm{rel}}_{t\to \ell}]$. Here, $\mathbf{c}^{3D}_0$ is the initial 3D centroid of keypoints for grasping, while $\mathbf{c}_t$ and $\hat{\mathbf{c}}_\ell$ are the current and generated lookahead centroids of keypoints. The relative transformations $\mathbf{T}$ represent the observed object motion, generated subgoal motion, and remaining motion to the subgoal. The predicted actions $\mathbf{a}^{\mathrm{b}}_{t:t+K}$ are a sequence of $6D$ delta poses with binary grasping signal. We keep the generated goal fixed until the observed centroid is sufficiently close to it, where it enables the robot to reach the next subgoal with a closed-loop manner.

Other Visual Base Policies. We also instantiate OMNITACTUNE with Action Chunking Transformer (ACT) [25], Diffusion Policy (DP) [26], and $\pi_{0.5}$ [28]. All three policies are trained or fine-tuned on our teleoperation data with synchronized RGB observations, robot states, and actions.

3.3 Real-World Tactile-Informed Residual RL Adaptation

We introduce our two-stage real-world RL pipeline for adapting tactile feedback into different kinds of frozen visual base policies with high stability and efficiency.

Stage I: warm-start. A key challenge in our setting is enabling tactile residual learning without offline visual-tactile demonstrations. A naive approach [21, 63] is to roll out the base policy and start SAC on the collected transitions. However, this leads to unstable exploration: a randomly initialized

critic produces unreliable Q-value estimates, while a pretrained tactile encoder suffers from representation shift. Our warm-start stage addresses both issues jointly. Rather than only populating the replay buffer with autonomous rollouts [22, 23], we use these transitions to bootstrap a flow-tactile critic $Q_\eta(\mathbf{q}_t, \mathbf{z}_t^f, \mathbf{z}_t^\tau, \mathbf{a}_t)$, where $\mathbf{q}_t$ is proprioception, $\mathbf{z}_t^f$ is flow features, $\mathbf{z}_t^\tau$ is tactile features, and $\mathbf{a}_t$ is final action. At the same time, we adapt a pretrained tactile encoder, such as AnyTouch2 [64], to the specific task. The tactile encoder is optimized with both critic loss and reconstruction regularization [47], $\mathcal{L}_E = \mathcal{L}_Q + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}$. We only use contact transitions for optimizing tactile encoder, where contact is detected by thresholding marker displacement. By optimizing the tactile encoder and the critic, the online tactile residual learning can be more stable and efficient.

To improve data efficiency, inspired by DrqV2 [65], we conduct trajectory-level tactile augmentation after each rollout. For each collected trajectory, we synthesize two new trajectories with temporally consistent tactile images conditioned on contact force while keeping the robot states, actions, and reward labels unchanged. These augmented transitions provide additional contact-rich samples for initializing the tactile representation and flow-tactile critic before online residual learning. We utilize ControlTac [66]-style SOTA augmentation method, with more details shown in Sec. A.4.

Stage II: online tactile residual learning. The key question here is how a single residual actor can correct architecturally diverse policies such as Flow Policy [27] and ACT [25]. We address this through interface-level agnosticism: instead of relying on internal representations, the residual actor uses two policy-independent inputs: the generated keypoints goals, which provide a shared task-level guidance, and the base-policy action chunk, which exposes its short-horizon motion intent. This allows the same residual actor to be attached to different visual policies for closed-loop correction. Specifically, the final action is $\mathbf{a}_t = \mathbf{a}_t^b + \mathbf{s}_t\mathbf{a}_t^r$, where $\mathbf{a}_t^b$ is the base action, a_t^r is the residual correction, and s_t is the scheduler. The residual policy takes as input $\mathbf{a}_t^r = \pi_\theta^r(\mathbf{q}_t, \mathbf{z}_t^f, \mathbf{g}_t \cdot \mathbf{z}_t^\tau, \mathbf{a}_t^b, \mathbf{a}_{t:t+K}^b)$, where $\mathbf{q}_t$ is the proprioception, $\mathbf{z}_t^f$ is the flow representation, $\mathbf{z}_t^\tau$ is the tactile representation, $\mathbf{g}_t$ is a contact gate to add tactile as input, and $\mathbf{a}_{t:t+K}^b$ is the base policy action chunk. For training details, we keep π_{b} frozen, learn a residual tactile policy π_θ^r, and keep optimizing the flow-tactile critic Q_η.

During execution, we keep the current keypoint condition $\hat{\mathbf{c}}_\ell$ fixed until the observed object keypoints are close enough, which enables close-loop corrections with the keypoints as subgoal. Following PLD [22], we add residual scale with a scheduler to ensure safe exploration.

3.4 Multi-Sensory Reward Design

Real-world RL is often limited by the difficulty of designing dense rewards. To guide the robot toward the near-contact states and encourage safe contact, our pipeline uses a normalized multi-sensory reward that combines generated object-centric flow with tactile signal: $r_t = \text{normalize}(w_r r_t^{\text{reach}} + w_g r_t^{\text{grasp}} + w_f r_t^{\text{flow}}) - w_s r_t^{\text{safety}}$. The generated flow provides task-level, object-centric motion guidance, while tactile feedback encourages stable and safe exploration. When the task succeeds, the operator assigns a terminal success reward of 1. Before grasping, the reaching reward r_t^{reach} guides the end-effector toward the initial 3D object centroid $\mathbf{c}_0^{3D}$; after grasping, the flow reward r_t^{flow} tracks sparse subgoals from the generated object flow generated from Sec. 3.2 [33, 59]:

$$r_t^{\text{reach}} = 1 - \text{clip}\left(\frac{\|\mathbf{p}_t^{ee} - \mathbf{c}_0^{3D}\|_2}{d_{\max}^{\text{reach}}}, 0, 1\right), \quad d_t^{\text{flow}} = \left\|\mathbf{T}_{t_0\to t}^{\text{rel}} - \hat{\mathbf{T}}_{t_0\to \ell}^{\text{rel}}\right\|_2,$$
$$r_t^{\text{flow}} = \frac{\ell_t}{L} + \frac{1}{L}\left(1 - \text{clip}\left(\frac{d_t^{\text{flow}}}{d_{\max}^{\text{flow}}}, 0, 1\right)\right).$$

Here, $\mathbf{p}_t^{ee}$ is the end-effector position, $\mathbf{c}_0^{3D}$ is the initial 3D object centroid, $d_{\max}^{\text{reach}}$ and $d_{\max}^{\text{flow}}$ are normalization constants, $\mathbf{T}$ represent the observed object motion and the generated subgoal motion, d_t^{flow} is their transformation error, L is the total number of sparse subgoals, and $\ell_t \in \{0, \ldots, L\}$ is the number of subgoals reached by time t. When $d_t^{\text{flow}} < \epsilon_{\text{flow}}$, the current subgoal is considered reached and the policy switches to the next sparse subgoal.

We further add tactile rewards to encourage stable contact and penalize unsafe interaction:

$$r_t^{\text{grasp}} = \mathbf{1}\left[\frac{1}{|\Omega|}\sum_{u\in\Omega}\mathbf{D}_t^\tau(u) > \epsilon_{\text{depth}}\right], \qquad r_t^{\text{safety}} = \mathbf{1}[m_t > \epsilon_{\text{safety}}].$$

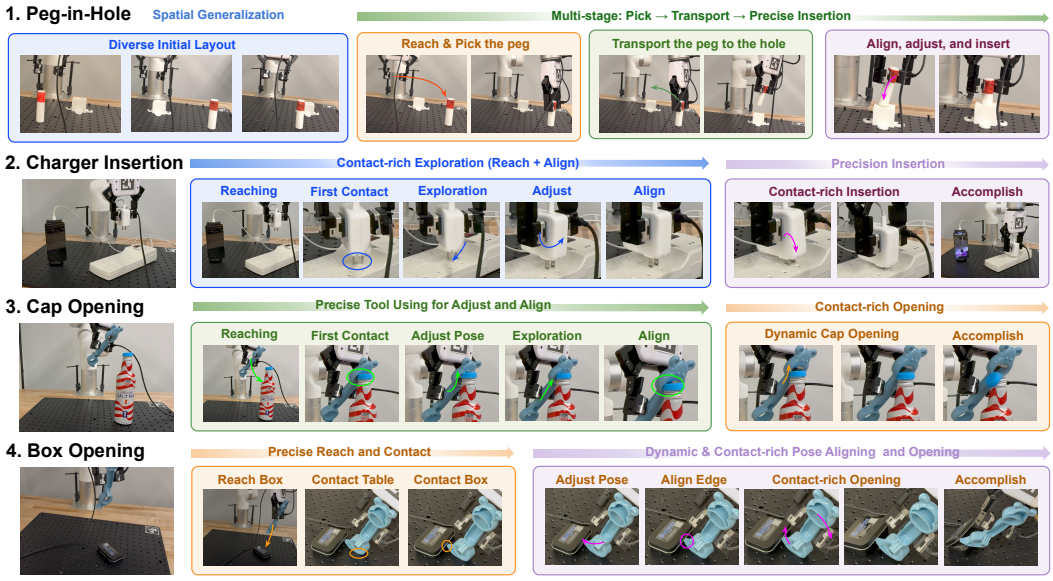

Figure 3: **Real-world contact-rich manipulation tasks.** We evaluate OMNITACTUNE on four tasks: *peg-in-hole* requires spatial generalization and multi-stage insertion; *charger insertion* requires contact exploration and precise insertion; *cap opening* requires dynamic contact reasoning and pose adjustment; and *box opening* requires precise edge alignment and contact-rich opening.

Table 1: Final success rates across four real-world contact-rich manipulation tasks after RL training.

Method	Peg-in-Hole	Charger Insertion	Cap Opening	Box Opening	Average
PLD* [22]	65%	60%	50%	35%	52.5%
PLD [22] (Visual Only)	60%	30%	40%	20%	37.5%
ViTAL [63]	50%	50%	45%	30%	43.75%
Ours	**100%**	**100%**	**90%**	**85%**	**93.75%**

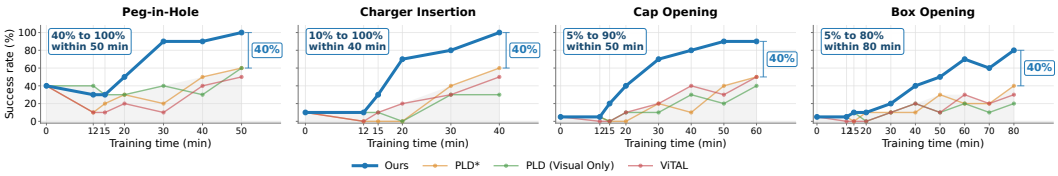

Figure 4: Comparison of RL training results across four different contact-rich manipulation tasks.

Here, $\mathbf{D}_t^\tau$ is the tactile depth image, Ω is the valid tactile region, u indexes tactile pixels, ϵ_{depth} is the contact threshold, m_t is the average tactile marker displacement, and ϵ_{safety} is the safety threshold. If the safety penalty is triggered, we assign a large negative reward and reset the robot.

4 Experiments

In this section, we evaluate OMNITACTUNE along four axes: real-world learning efficiency for adapting tactile feedback (Sec. 4.1), compatibility with various visual base policies (Sec. 4.2), comparison to other visuo-tactile policy learning methods (Sec. 4.3), and generalizability across different tactile representations (Sec. 4.4). Furthermore, we conduct extensive ablation studies on reward shaping, residual policy design, warm-start strategies, and action designs. We also conduct failure case evaluations and analysis of generated flows and base policies. See Appendix C for details.

Tasks. We use an xArm7 with a gripper mounted with GelSight Mini and a calibrated Intel RealSense D435 third-view camera. We evaluate OMNITACTUNE on four challenging contact-rich manipulation tasks, shown in Fig. 3, including: i) **Peg-in-Hole**. Reach and grasp a cylinder and insert it into a target receptacle. The task requires spatial generalizability across different locations, stable grasping during transport, and millimeter contact alignment during insertion; ii) **Charger In-**

sertion. Insert the charger into a power strip. This task is particularly challenging because of tiny charger head and the small port tolerance; iii) **Cap Opening**. A tool using task, where the robot use a cap opener to open a bottle cap. This task is challenging because it requires maintaining a precise opener pose while handling dynamic contact-rich interactions with the cap; and iv) **Box Opening**. A tool using task, where the robot use a lever-type bottle opener to open the GelSight box. This task is challenging because the opener must make precise contact with a very small cap edge and maintain contact-rich interaction while executing a highly dynamic levering motion. All those tasks are contact-rich manipulation tasks with dynamic and precise contact need tactile feedback.

4.1 Adapting Tactile Feedback with Residual RL

In this section, we evaluate OMNITACTUNE across all tasks and compare it with other real-world RL methods, which are adapted into our tactile adaptation setting. For each task, we collect 50 human demos using our data collection system in Sec. 3.1 and train the human flow policy (in Sec. 3.2) as the base policy. Evaluated in 20 trials, the success rates of the base policies are 40%, 10%, 5%, and 5% for **Peg-in-Hole**, **Charger Insertion**, **Cap Opening**, and **Box Opening**, respectively.

We implement three different baselines: i) **PLD*** [22]: adapting PLD to our flow-tactile setting, collecting real-world data with tactile into the replay buffer, but no initialization and bootstrapping for the critic; ii) **PLD (visual only)**: as the original version of PLD, we initialize the critic for the flow policy via offline data, warm-start the replay buffer, and do the residual RL; iii) **ViTAL** [63]: we train the residual policy and critic from scratch without warm-start. For each task, we perform a 12-minute warm-start phase for all methods except ViTAL, followed by online policy learning. The total training time is 50, 40, 60, and 80 minutes across the four tasks, respectively. We evaluate the policies in 10 trials for each checkpoint shown in the figure, and 20 trials for the final model.

As shown in Tab. 1 and Fig. 4, our method improves the weak human-flow base policies from only 5–40% success rate to 100%, 100%, 90%, and 85% final success rate within 40–80 mins, outperforming all baselines with more than 40% success rate. In contrast, **PLD*** improves more slowly and unstably, showing the importance of bootstrapping the tactile encoder and critic. **PLD (visual only)** performs better in the early stage due to its critic initialization, but achieves the worst overall results without adapting tactile feedback. **ViTAL** is less sample-efficient because it learns visuo-tactile critics from scratch. Overall, these results show that replay-buffer warm-start, tactile encoder and critic optimization, and tactile residual learning are all critical for efficient adaptation.

4.2 Adapting Tactile Feedback to Various Base Visual Policies

In this section, we evaluate OMNITACTUNE on the **Peg-in-Hole** task across different base policies trained from different data sources. We evaluate it on five different base policies: i) Human Flow Policy in Sec. 3.2, with initial success rate 40%; ii) Teleoperation-based Flow Policy, with initial success rate 50%; iii) Teleoperation-based `ACT` [25], with initial success rate 20%; iv) Teleoperation-based `DP` [26], with initial success rate 30%; v) fine-tuned $\pi_{0.5}$ [28], with initial success rate 20%. In this section, we only evaluate the performance on the **Peg-in-Hole** task because collecting high-quality teleoperation data for the other tasks is challenging due to the contact-rich interactions, making the demonstrations too noisy to policy training. The result is shown in Fig. 5.

As shown in Fig. 5, OMNITACTUNE improves all base policies from 15–50% to 75–100% after 50 minutes of real-world practice. These results show that tactile residual adaptation is broadly compatible with different visual motion priors, rather than being tied to a specific policy architecture. Among them, the human flow policy achieves the highest performance, suggesting that human demos provide a smoother motion prior for contact-rich refinement. We provide further analysis of data and policy smoothness in the Appendix C, which helps explain why teleoperation data becomes low-quality and impractical for contact-rich manipulations.

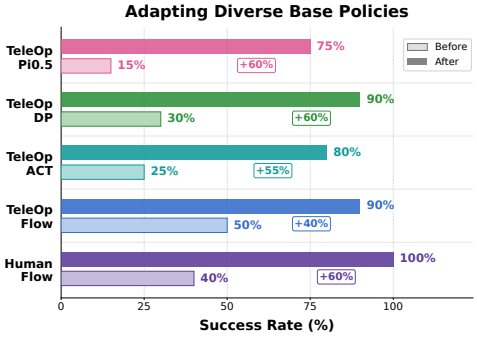

Figure 5: OMNITACTUNE consistently improves diverse base visual policies.

4.3 Comparison with Other Visuo-Tactile Robot Learning Methods

To evaluate the data efficiency of OMNITACTUNE, we compare against visuo-tactile policy learning pipelines built on different policy backbones: `ACT`, `DP`, and $\pi_{0.5}$. For `ACT`, we adopt a straightforward visuo-tactile fusion strategy by concatenating visual and tactile features. For `DP`, we implement `RDP [10]`, a SOTA diffusion-based visuo-tactile learning method. For $\pi_{0.5}$, we incorporate additional tactile tokens and conduct supervised fine-tuning [67]. To ensure a fair comparison, all baselines are trained with additional teleoperation data collected under the same time budget as our online training stage (i.e., 50 minutes), increasing the number of demonstrations from 50 to 90.

Table 2: Success rates of different visuo-tactile robot learning methods on the **Peg-in-Hole** task.

Task	ACT [25]	ACT + Tactile	DP [26]	RDP [10]	$\pi_{0.5}$ [28]	$\pi_{0.5}$ + Tactile	**Ours (ACT)**	**Ours (DP)**	**Ours ($\pi_{0.5}$)**	**Ours**
Peg-in-Hole	25%	60%	30%	65%	15%	45%	**80%**	**90%**	**75%**	**100%**

As shown in Tab. 2, imitation-learning-based visuo-tactile policy learning remains less effective than OMNITACTUNE, which achieves 20%–30% higher success rates for all backbones and reaches 100% success with our human flow policy. This shows online tactile residual refinement is more effective than learning a visuo-tactile policy from extra teleoperated demonstrations, highlighting that tactile correction for contact-rich manipulation is better learned through trial-and-error.

4.4 Adaptation Across Different Tactile Representations

To highlight its compatibility, we further evaluate OMNITACTUNE across different tactile representations in two different contact-rich manipulation tasks, **Peg-in-Hole** and **Charger Insertion**. We compare three pretrained tactile image encoders, `AnyTouch2 [64]`, `Sparsh [68]`, and `T3 [69]`, as well as low-dimensional `Tactile Markers` with two-layer MLP. For pretrained tactile encoders, we freeze the backbone and only optimize the final projection layers and a lightweight adaptive layer that aligns with the flow features. For markers, we optimize their encoder from scratch.

As shown in Fig. 6, our method consistently improves performance across all tactile representations, which demonstrates that our method is not tied to a particular setting. On both **Peg-in-Hole** and **Charger Insertion** tasks, `AnyTouch2` and low-dimensional tactile markers achieve comparable final performance, which highlights that OMNITACTUNE can effectively adapt with both pretrained tactile features and compact marker-based tactile signals. Meanwhile, we observe that `T3` and `Sparsh` perform worse than `AnyTouch2` on Charger Insertion, where the task involves more dynamic contact. The potential reason is that `T3` and `Sparsh` are not pretrained on dynamic manipulation tactile data.

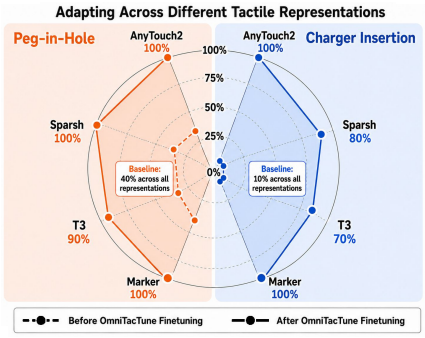

Figure 6: Adapting different tactile representations via OMNITACTUNE.

5 Conclusion and Limitation

We presented OMNITACTUNE, a policy-agnostic real-world RL pipeline that adapts tactile feedback to pretrained visual policies through residual correction. OMNITACTUNE enables efficient tactile adaptation without offline tactile demonstrations by using visual base policies as motion priors, warm-starting tactile-aware critics and encoders from autonomous rollouts, and incorporating multi-sensory reward shaping. Across four challenging real-world contact-rich manipulation tasks, our method improves visual base policies from 5–40% success to 85–100% within 40–80 minutes, while generalizing across tasks, base policies, and tactile representations.

Limitation. Despite these results, OMNITACTUNE still inherits common limitations of real-world RL, including the need for manual resets and hardware wear under repeated contact-rich interactions, especially when using fragile vision-based tactile sensors. Future work should reduce reset and hardware costs through safer and more automated real-world training, while further improving motion priors and data augmentation, such as incorporating world models [43, 70–72] to generate more physically plausible visuo-tactile rollouts for pretraining and online policy improvement. Moreover, adapting it into other embodiments and tactile sensor is a straightforward extension.

References

[1] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, M. Z. Irshad, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, V. Guizilini, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2025. URL https://arxiv.org/abs/2310.08864.

[2] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

[3] S. Kareer, D. Patel, R. Punamiya, P. Mathur, S. Cheng, C. Wang, J. Hoffman, and D. Xu. Egomimic: Scaling imitation learning via egocentric video, 2024. URL https://arxiv.org/abs/2410.24221.

[4] R. Zheng, D. Niu, Y. Xie, J. Wang, M. Xu, Y. Jiang, F. Castañeda, F. Hu, Y. L. Tan, L. Fu, T. Darrell, F. Huang, Y. Zhu, D. Xu, and L. Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data, 2026. URL https://arxiv.org/abs/2602.16710.

[5] R. Punamiya, S. Kareer, Z. Liu, J. Citron, R.-Z. Qiu, X. Cai, A. Gavryushin, J. Chen, D. Liconti, L. Y. Zhu, P. Aphiwetsa, B. Li, A. Cheluva, P. Kuppili, Y. Liu, D. Patel, A. Gao, H.-Y. Chung, R. Co, R. Zbizika, J. Liu, X. Xu, H. Xiong, G. Chen, S. Oliani, C. Yang, X. Wang, J. Fort, R. Newcombe, J. Gao, J. Chong, G. Matsuda, A. Doriwala, M. Pollefeys, R. Katzschmann, X. Wang, S. Song, J. Hoffman, and D. Xu. Egoverse: An egocentric human dataset for robot learning from around the world, 2026. URL https://arxiv.org/abs/2604.07607.

[6] W. Yuan, S. Dong, and E. H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[7] K. Yu, Y. Han, Q. Wang, V. Saxena, D. Xu, and Y. Zhao. Mimictouch: Leveraging multimodal human tactile demonstrations for contact-rich manipulation, 2025. URL https://arxiv.org/abs/2310.16917.

[8] B. Huang, Y. Wang, X. Yang, Y. Luo, and Y. Li. 3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing, 2025. URL https://arxiv.org/abs/2410.24091.

[9] H. Li, Y. Zhang, J. Zhu, S. Wang, M. A. Lee, H. Xu, E. Adelson, L. Fei-Fei, R. Gao, and J. Wu. See, hear, and feel: Smart sensory fusion for robotic manipulation, 2022. URL https://arxiv.org/abs/2212.03858.

[10] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation, 2025. URL https://arxiv.org/abs/2503.02881.

[11] Y. Dou, F. Yang, Y. Liu, A. Loquercio, and A. Owens. Tactile-augmented radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26529–26539, 2024.

[12] X. Zhu, B. Huang, and Y. Li. Touch in the wild: Learning fine-grained manipulation with a portable visuo-tactile gripper. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=WabVVQKTUF.

[13] L. Fu, G. Datta, H. Huang, W. C.-H. Panitch, J. Drake, J. Ortiz, M. Mukadam, M. Lambeta, R. Calandra, and K. Goldberg. A touch, vision, and language dataset for multimodal alignment. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=tFEOOH9eH0.

[14] R. Gao, Y. Dou, H. Li, T. Agarwal, J. Bohg, Y. Li, L. Fei-Fei, and J. Wu. The objectfolder benchmark: Multisensory learning with neural and real objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17276–17286, 2023.

[15] F. Liu, C. Li, Y. Qin, J. Xu, P. Abbeel, and R. Chen. Vitamin: Learning contact-rich tasks through robot-free visuo-tactile manipulation interface, 2025. URL https://arxiv.org/abs/2504.06156.

[16] R. Hoque, P. Huang, D. J. Yoon, M. Sivapurapu, and J. Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video, 2026. URL https://arxiv.org/abs/2505.11709.

[17] B. Huang, J. Xu, I. Akinola, W. Yang, B. Sundaralingam, R. O'Flaherty, D. Fox, X. Wang, A. Mousavian, Y.-W. Chao, and Y. Li. VT-refine: Learning bimanual assembly with visuo-tactile feedback via simulation fine-tuning. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=bOVF8Rj33i.

[18] P. Dan, K. Kedia, A. Chao, E. W. Duan, M. A. Pace, W.-C. Ma, and S. Choudhury. X-sim: Cross-embodiment learning via real-to-sim-to-real, 2025. URL https://arxiv.org/abs/2505.07096.

[19] W. Wan, J. Fu, X. Yuan, Y. Zhu, and H. Su. Lodestar: Long-horizon dexterity via synthetic data augmentation from human demonstrations, 2025. URL https://arxiv.org/abs/2508.17547.

[20] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

[21] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning, 2025. URL https://arxiv.org/abs/2410.21845.

[22] W. Xiao, H. Lin, A. Peng, H. Xue, T. He, Y. Xie, F. Hu, J. Wu, Z. Luo, L. J. Fan, G. Shi, and Y. Zhu. Self-improving vision-language-action models with data generation via residual rl, 2025. URL https://arxiv.org/abs/2511.00091.

[23] Z. Zhou, A. Peng, Q. Li, S. Levine, and A. Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data, 2025. URL https://arxiv.org/abs/2412.07762.

[24] C. Xu, J. T. Springenberg, M. Equi, A. Amin, A. Esmail, S. Levine, and L. Ke. Rl token: Bootstrapping online rl with vision-language-action models, 2026. URL https://arxiv.org/abs/2604.23073.

[25] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[26] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[27] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song. Flow as the cross-domain manipulation interface, 2024. URL https://arxiv.org/abs/2407.15208.

[28] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

[29] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[30] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[31] L. Y. Zhu, P. Kuppili, R. Punamiya, P. Aphiwetsa, D. Patel, S. Kareer, S. Ha, and D. Xu. Emma: Scaling mobile manipulation via egocentric human data, 2025. URL https://arxiv.org/abs/2509.04443.

[32] Y. Liu, W. C. Shin, Y. Han, Z. Chen, H. Ravichandar, and D. Xu. Immimic: Cross-domain imitation from human videos via mapping and interpolation, 2025. URL https://arxiv.org/abs/2509.10952.

[33] K. Yu, S. Zhang, H. Soora, F. Huang, H. Huang, P. Tokekar, and R. Gao. Genflowrl: Shaping rewards with generative object-centric flow in visual reinforcement learning, 2025. URL https://arxiv.org/abs/2508.11049.

[34] Y. Han, Z. Chen, K. A. Williams, and H. Ravichandar. Learning prehensile dexterity by imitating and emulating state-only observations. *IEEE Robotics and Automation Letters*, 9(10): 8266–8273, 2024.

[35] Z. Wang, B. He, K. Yu, S. Lee, R. Gao, F. Huang, and Y. Aloimonos. Humanego: Zero-shot robot learning from minutes of human egocentric videos, 2026.

[36] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[37] A. Singh, K. Torshizi, K. Habib, K. Yu, R. Gao, and P. Tokekar. Afford2act: Affordance-guided automatic keypoint selection for generalizable and lightweight robotic manipulation, 2026. URL https://arxiv.org/abs/2510.01433.

[38] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild, 2022. URL https://arxiv.org/abs/2207.09450.

[39] R. Bhirangi, V. Pattabiraman, E. Erciyes, Y. Cao, T. Hellebrekers, and L. Pinto. Anyskin: Plug-and-play skin sensing for robotic touch, 2024. URL https://arxiv.org/abs/2409.08276.

[40] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, 2020. ISSN 2377-3774. doi:10.1109/lra.2020.2977257. URL http://dx.doi.org/10.1109/LRA.2020.2977257.

[41] S. Kim and A. Rodriguez. Active extrinsic contact sensing: Application to general peg-in-hole insertion, 2022. URL https://arxiv.org/abs/2110.03555.

[42] Y. Han, K. Yu, R. Batra, N. Boyd, C. Mehta, T. Zhao, Y. She, S. Hutchinson, and Y. Zhao. Learning generalizable vision-tactile robotic grasping strategy for deformable objects via transformer. *IEEE/ASME Transactions on Mechatronics*, 30(1):554–566, 2025. doi:10.1109/TMECH.2024.3400789.

[43] Z. Xu and Y. She. Letac-mpc: Learning model predictive control for tactile-reactive grasping, 2024. URL https://arxiv.org/abs/2403.04934.

[44] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang. Rotating without seeing: Towards in-hand dexterity through touch, 2023. URL https://arxiv.org/abs/2303.10880.

[45] I. Guzey, B. Evans, S. Chintala, and L. Pinto. Dexterity from touch: Self-supervised pre-training of tactile representations with robotic play, 2023. URL https://arxiv.org/abs/2303.12076.

[46] J. J. Liu, Y. Li, K. Shaw, T. Tao, R. Salakhutdinov, and D. Pathak. Factr: Force-attending curriculum training for contact-rich policy learning, 2025. URL https://arxiv.org/abs/2502.17432.

[47] C. Sferrazza, Y. Seo, H. Liu, Y. Lee, and P. Abbeel. The power of the senses: Generalizable manipulation from vision and touch through masked multimodal learning, 2023. URL https://arxiv.org/abs/2311.00924.

[48] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts, 2019. URL https://arxiv.org/abs/1810.02890.

[49] X. Xu, Y. Hou, Z. Liu, and S. Song. Compliant residual DAgger: Improving real-world contact-rich manipulation with human corrections. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=cjcm5LYVWm.

[50] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid. Vividex: Learning vision-based dexterous manipulation from human videos, 2025. URL https://arxiv.org/abs/2404.15709.

[51] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

[52] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

[53] M. Nakamoto, Y. Zhai, A. Singh, M. S. Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

[54] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

[55] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement – residual rl for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

[56] S. Haldar, J. Pari, A. Rai, and L. Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations, 2023. URL https://arxiv.org/abs/2303.01497.

[57] I. Guzey, Y. Dai, G. Savva, R. Bhirangi, and L. Pinto. Bridging the human to robot dexterity gap through object-oriented rewards, 2024. URL https://arxiv.org/abs/2410.23289.

[58] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation, 2024. URL https://arxiv.org/abs/2403.07870.

[59] Y. Kuang, S. Park, K. Fragkiadaki, and S. Tulsiani. Dex4d: Task-agnostic point track policy for sim-to-real dexterous manipulation, 2026. URL https://arxiv.org/abs/2602.15828.

[60] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

[61] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything, 2023. URL https://arxiv.org/abs/2304.02643.

[62] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos, 2024. URL https://arxiv.org/abs/2410.11831.

[63] Z. Zhao, S. Haldar, J. Cui, L. Pinto, and R. Bhirangi. Touch begins where vision ends: Generalizable policies for contact-rich manipulation, 2025. URL https://arxiv.org/abs/2506.13762.

[64] R. Feng, Y. Zhou, S. Mei, D. Zhou, P. Wang, S. Cui, B. Fang, G. Yao, and D. Hu. Anytouch 2: General optical tactile representation learning for dynamic tactile perception, 2026. URL https://arxiv.org/abs/2602.09617.

[65] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021. URL https://arxiv.org/abs/2107.09645.

[66] D. Luo, K. Yu, A.-H. Shahidzadeh, C. Fermüller, Y. Aloimonos, and R. Gao. Controltac: Force- and position-controlled tactile data augmentation with a single reference image, 2025. URL https://arxiv.org/abs/2505.20498.

[67] B. Huang and Y. Li. Flexitac: A low-cost, open-source, scalable tactile sensing solution for robotic systems, 2026. URL https://arxiv.org/abs/2604.28156.

[68] C. Higuera, A. Sharma, C. K. Bodduluri, T. Fan, P. Lancaster, M. Kalakrishnan, M. Kaess, B. Boots, M. Lambeta, T. Wu, and M. Mukadam. Sparsh: Self-supervised touch representations for vision-based tactile sensing, 2024. URL https://arxiv.org/abs/2410.24090.

[69] J. Zhao, Y. Ma, L. Wang, and E. H. Adelson. Transferable tactile transformers for representation learning across diverse sensors and tasks, 2024. URL https://arxiv.org/abs/2406.13640.

[70] Y. Guo, T. Lee, L. X. Shi, J. Chen, P. Liang, and C. Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model, 2026. URL https://arxiv.org/abs/2602.12063.

[71] S. Yang, Y. Du, K. Ghasemipour, J. Tompson, L. Kaelbling, D. Schuurmans, and P. Abbeel. Learning interactive real-world simulators, 2024. URL https://arxiv.org/abs/2310.06114.

[72] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

[73] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

[74] A. . Team, J. Aldaco, T. Armstrong, R. Baruch, J. Bingham, S. Chan, K. Draper, D. Dwibedi, C. Finn, P. Florence, S. Goodrich, W. Gramlich, T. Hage, A. Herzog, J. Hoech, T. Nguyen, I. Storz, B. Tabanpour, L. Takayama, J. Tompson, A. Wahid, T. Wahrburg, S. Xu, S. Yaroshenko, K. Zakka, and T. Z. Zhao. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation, 2024. URL https://arxiv.org/abs/2405.02292.

[75] N. R. Arachchige, Z. Chen, W. Jung, W. C. Shin, R. Bansal, P. Barroso, Y. H. He, Y. C. Lin, B. Joffe, S. Kousik, and D. Xu. Sail: Faster-than-demonstration execution of imitation learning policies, 2025. URL https://arxiv.org/abs/2506.11948.

Appendix

A Implementation Details

Data Collection System

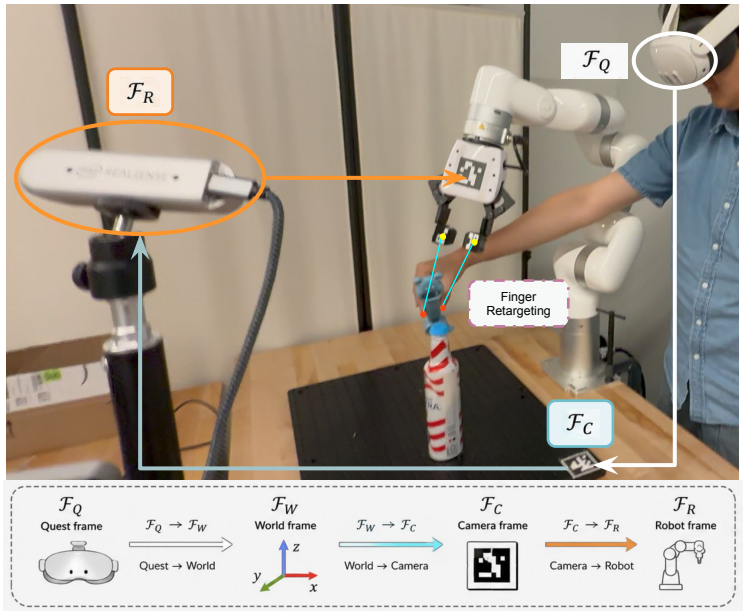

Figure 7: Visualization of our data collection system, for both hand and finger retargeting.

Human video collection and Quest pose retargeting. For each task, we collect human videos using a third-view RGB camera and a Meta Quest headset. The RGB camera records the in-scene videos for human hand operations, while the Quest headset provides the 6-DoF hand pose and the finger keypoints in the Quest frame. Following prior VR-based data collection systems [58], we calibrate the Quest frame, the world frame, and the robot base frame via an ArUco marker placed on the table. Let ${}^{q}\mathbf{T}_h(t)$ denote the tracked hand pose in the Quest frame at time t, ${}^{w}\mathbf{T}_q$ show the transform from the Quest frame to the world frame, and ${}^{r}\mathbf{T}_w$ denote the transform from the world frame to the robot base. The robot-aligned hand pose is obtained by

$$ {}^{r}\mathbf{T}_h(t) = {}^{r}\mathbf{T}_w \, {}^{w}\mathbf{T}_q \, {}^{q}\mathbf{T}_h(t). \tag{1} $$

We synchronize the third-view video and Quest pose stream, and linearly interpolate the lower-frequency stream from the quest. To reduce tracking noise, we apply a small smoothing filter to the tracked hand translation and use spherical interpolation for rotations.

Finger pose tracking and retargeting. For demonstrations that require precise grasp, we further use the Quest hand skeleton to estimate finger motion and retarget them to the binary grasping signal. Because human hand's skeleton and the robot gripper have different morphology and degrees of freedom, we do not directly copy human joint angles. Instead, we retarget the finger geometry into a compact binary gripper command. Specifically, we extract the thumb and index fingertip positions, their relative distance, and the palm normal direction from the human hand skeleton. the finger distance is computed from the normalized thumb-index distance:

$$ a_t^{\text{grip}} = \text{clip}\left(\alpha \frac{\|\mathbf{p}_t^{\text{thumb}} - \mathbf{p}_t^{\text{index}}\|_2 - d_{\min}}{d_{\max} - d_{\min}} + \beta,\ a_{\min}, a_{\max} \right), \tag{2} $$

where $\mathbf{p}_t^{\text{thumb}}$ and $\mathbf{p}_t^{\text{index}}$ are the tracked fingertip positions, $d_{\min}$ and $d_{\max}$ are calibration distances for closed and open hand poses, and α, β map the human aperture to the robot gripper range. We further detect grasp intent by thresholding the temporal change of the thumb-index distance and the absolute aperture value, which produces a binary grasp signal used for grasping and placing.

Teleoperation data collection. In addition to human videos, we collect robot demonstrations through Meta Quest headset via an OpenTeach-style interface [58]. The operator controls the robot end-effector with hand motion, while the system records synchronized RGB observations, robot proprioception, gripper state, and executed delta actions. We use the same calibrated third-view RGB camera and robot setup as in the human-video demonstrations. Each teleoperated episode is stored as $\{I_t, \mathbf{q}_t, \mathbf{a}_t\}_{t=1}^{T}$, where I_t is the third-view RGB observation, $\mathbf{q}_t$ is the robot proprioceptive state, and $\mathbf{a}_t$ is the executed delta Cartesian end-effector action with gripper command. Teleoperation data is mainly used to train robot-data-based base policies, including ACT, Diffusion Policy, and $\pi_{0.5}$. For contact-rich tasks like **Charger Insertion**, **Box Opening**, and **Cap Opening**, it's impractical to collect high-quality during the contact stage because human cannot get the contact event with their hand [7]. Therefore, in our main results, we use human flow policies for all four tasks and use teleoperation policies only for the **Peg-in-Hole**. More analysis has been shown in Appendix. C.1

A.1 Flow Generator

Our flow generator follows the object-centric design of Im2Flow2Act and GenFlowRL [27, 33]. Given the first RGB frame I_0, we first localize the manipulated object using DINOv2 features and prompt SAM to obtain an object mask M_0. We then sample $N = 32$ keypoints inside the object mask and track them through the demonstration video using CoTracker 3 [62]. This generates an object keypoint trajectory

$$\mathbf{K}_{1:T} = \{\mathbf{k}_t^i\}_{t=1,i=1}^{T,N}. \tag{3}$$

The flow generator G_ϕ predicts a task-specific object flow from the initial image, keypoints, and task condition:

$$\hat{\mathbf{F}} = G_\phi(I_0, \mathbf{K}_0, z), \tag{4}$$

where $\mathbf{K}_0$ is the initial set of object keypoints and z denotes the task condition. We initialize G_ϕ from the pretrained flow generator of Im2Flow2Act and fine-tune it on our collected demonstrations. For each task, we fine-tune the generator with 50 demonstrations using LoRA adapters [73] while keeping the main pretrained backbone frozen. This improves task-specific flow generation while avoiding overfitting to the small demonstration set.

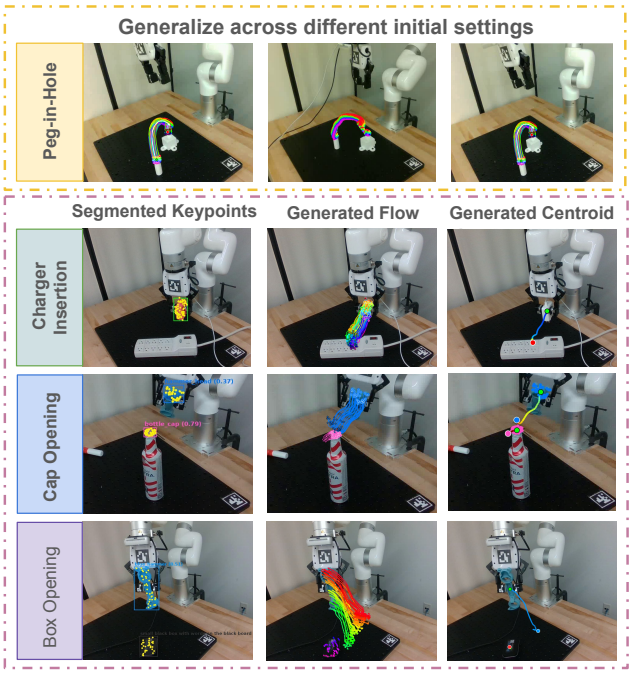

Figure 8: Generated flows for each tasks.

The generator is trained with a point-wise flow prediction loss and a temporal smoothness regularizer:

$$\mathcal{L}_{\text{flow}} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N}\left\|\hat{\mathbf{k}}_t^i - \mathbf{k}_t^i\right\|_2^2 + \lambda_{\text{smooth}}\sum_{t=2}^{T-1}\left\|\hat{\mathbf{k}}_{t+1} - 2\hat{\mathbf{k}}_t + \hat{\mathbf{k}}_{t-1}\right\|_2^2. \tag{5}$$

At test time, the generator predicts a complete object-centric flow from the initial scene. To formulate our flow policy and the object-centric reward, we sparsify the generated trajectory into L subgoals and use these subgoals as compact guidance for the flow-based base policy and dense object-centric reward during RL. This follows from the observation that object flow provides a cross-embodiment manipulation interface with less embodiment gap.

A.2 Base Visual Policies

Flow policy. The flow policy converts generated and observed object flows into compact object-centric policy inputs. At time t, we compute the current object centroid $\mathbf{c}_t$, the relative transfor-

mation between current object centroid and the initial object centroid$\mathbf{T}^{\mathrm{rel}}_{t_0\to t}$, the selected generated lookahead centroid $\hat{\mathbf{c}}_\ell$, the relative transformation between the generated centroid and the initial centroid$\hat{\mathbf{T}}^{\mathrm{rel}}_{t_0\to \ell}$, and the remaining transformation between the current centroid and the generated lookahead centroid $\mathbf{T}^{\mathrm{rel}}_{t\to \ell}$. The policy input is

$$\mathbf{o}_t^{\mathrm{flow}} = [\mathbf{c}_0^{3D},\, \mathbf{c}_t,\, \mathbf{T}^{\mathrm{rel}}_{t_0\to t},\, \hat{\mathbf{c}}_\ell,\, \hat{\mathbf{T}}^{\mathrm{rel}}_{t_0\to \ell},\, \mathbf{T}^{\mathrm{rel}}_{t\to \ell}], \tag{6}$$

where $\mathbf{c}_0^{3D}$ is the initial 3D object centroid used for reaching and grasping. The flow policy predicts an action chunk:

$$\mathbf{a}^{\mathrm{b}}_{t:t+K} = \pi_\theta^{\mathrm{flow}}(\mathbf{o}_t^{\mathrm{flow}}, \mathbf{q}_t), \tag{7}$$

where $\mathbf{q}_t$ is robot proprioception and K is the action horizon. Unlike diffusion-based visuomotor policies, we implement the flow policy as a lightweight action-chunking policy over low-dimensional object-centric states. In practice, all modules are small MLPs with LayerNorm. This design keeps inference fast and stable during online RL, while they can follow the generated motion priors learned from expert demonstrations. During execution, the current lookahead subgoal ℓ is kept fixed until the observed object transformation is close to the generated subgoal, and then the policy switches to the next sparse subgoal along the generated flow. We sparsify the generated trajectory into L subgoals, which enable the robot to reach the goal in a closed-loop manner. This closed-loop subgoal tracking is similar in spirit to point-track-conditioned policies such as Dex4D [59].

Diffusion Policy. We train Diffusion Policy (DP) [26] on teleoperated robot demonstrations. The policy takes third-view RGB observations and robot proprioception as input and predicts a sequence of future end-effector actions through conditional denoising diffusion. We use the same image encoder, action horizon, and observation horizon as the prior work across all DP-based comparisons. Because DP can produce smoother action chunks than one-step behavior cloning, it provides a strong visual base policy for contact-rich tasks. However, it still open-loop for short-horizon motions and lacks direct contact feedback, so it may fail during the contact-rich manipulation tasks and require the close-loop corrections.

ACT. We train Action Chunking Transformer (ACT) [74] on the same teleoperation dataset. ACT encodes RGB observations and proprioception with a transformer-based conditional VAE and predicts an action chunk instead of a single action. We use the standard ACT training objective, including the reconstruction loss for actions and the KL regularization term for the latent variable. At test time, we use temporal ensembling over overlapping action chunks to reduce action jitter. Same as DP, although it can predict smooth action chunks, ACT still lacks closed-loop control with contact reasoning, making it a useful stress test for our residual tactile adaptation pipeline.

$\pi_{0.5}$**.** For VLA experiments, we fine-tune $\pi_{0.5}$ [28] on our teleoperation demonstrations. The model takes RGB observations from a single camera and a language instruction describing the task, such as "insert the peg into the hole" or "insert the charger into the port", and predicts low-level robot actions. We use parameter-efficient fine-tuning via LoRA [73] for the in our **Peg-in-Hole** task and keep the visual-language backbone fixed. Notably, since all the other settings only using a single 3-rd view RGB camera, $\pi_{0.5}$ also uses only one camera.

A.3 Baselines

PLD*. We adapt PLD [22] to our setting by using the frozen visual base policy as the generalist policy and training a residual actor with real-world interaction. Different from its original setting, the residual policy receives the same visual/flow + tactile observations as our method and is trained with SAC. Different from the original PLD, since the expert demo doesn't have tactile input, we cannot initialize the critic with the offline data. During the warm-start stage, we only autonomous rollouts the base policy to restore the replay buffer with the random critic. It therefore tests whether residual RL alone is sufficient without our fine-tuning our the tactile encoder and initialize the critic.

PLD (Visual Only). This baseline follows the visual-only residual RL setting more closely. We initialize the critic using the available offline visual demonstrations and train a residual policy on top of the flow base policy. The policy does not use tactile observations, tactile rewards, or tactile encoder optimization. This baseline tests whether the improvement comes from residual RL on the visual policy alone or from adapting tactile feedback.

ViTAL. We implement ViTAL as a visuo-tactile real-world RL baseline that learns a visuo-tactile policy and critic from scratch without any critic initialization, replay buffer initialization, and encoder optimization. The policy receives flow observations, proprioception, and tactile observations, and is trained using the same online interaction budget. This baseline evaluates the importance of using warm-start stage in critic initialization and encoder optimization.

ACT + Tactile. For the ACT tactile baseline, we simply add tactile observations to the ACT policy through feature concatenation. The RGB image is encoded by the original visual encoder, and the tactile image is encoded by the AnyTouch2 [64] used in our method. The two features are concatenated with proprioception and passed to the ACT transformer and predict action chunks. This baseline is trained by imitation learning from the teleoperated demonstrations with tactile observations. Unlike our method, it requires paired visuo-tactile demonstrations and does not improve through online tactile practice.

RDP. For the diffusion-based tactile baseline, we implement Reactive Diffusion Policy (RDP) [10], a slow-fast visuo-tactile policy for contact-rich manipulation. The slow branch predicts a visual motion plan from RGB observations. Then, the fast branch encode the latent plan from the slow policy with the tactile feedback, reacts to the contact and refines the action during execution. We train RDP on the same amount of visuo-tactile teleoperation data used by the other visuo-tactile imitation baselines. This baseline tests whether a SOTA visuo-tactile diffusion policy can match the data-efficiency of our online tactile residual adaptation or not.

$\pi_{0.5}$ + Tactile. For the VLA tactile baseline, we follow recent tactile-augmented VLA designs such as LeFlexiTac [67]. We encode tactile observations as additional tactile tokens and append them to the visual-language-action input during supervised fine-tuning. The model is fine-tuned on visuo-tactile teleoperation data with the same data budget as the other imitation baselines. Compared with our residual adaptation, this baseline directly fine-tune the VLA policy through supervised fine-tuning via LoRA [73], and therefore requires paired teleoperated, visual-tactile demonstrations for each task.

A.4 Warm-start Implementation

During warm-start, we execute the frozen base policy in the real world and set the residual action to zero:

$$\mathbf{a}_t = \mathbf{a}_t^{\mathrm{b}}, \qquad \mathbf{a}_t^{\mathrm{r}} = \mathbf{0}. \tag{8}$$

The goal of this stage is not to improve the policy immediately, but to collect on-policy tactile experience under the deployment distribution of the base policy. As shown in [23], without initializing the replay buffer and bootstrapping the critic is easily to lead the policy forget. In our setting, since the expert doesn't contains tactile feedback, we need to initialize the buffer and critic through online data during interaction. During data collection, each transition is stored as

$$(\mathbf{q}_t,\, \mathbf{z}_t^f,\, \mathbf{o}_{t-1:t}^{\tau},\, \mathbf{a}_t^{\mathrm{b}},\, r_t,\, \mathbf{q}_{t+1},\, \mathbf{z}_{t+1}^f,\, \mathbf{o}_{t+1-1:t+1}^{\tau}), \tag{9}$$

where $\mathbf{z}_t^f$ is the flow/object-centric representation, $\mathbf{o}_{t-1:t}^{\tau}$ is concatenated two tactile observations, and r_t is the multi-sensory reward.

We initialize the tactile encoder from a pretrained tactile representation model, such as AnyTouch2 [64], Sparsh [68], or T3 [69]. For image-based tactile encoders, the backbone is frozen, and only the final projection layer and a lightweight adapter of the encoder are optimized. For

marker-based tactile observations, we train a two-layer MLP encoder from scratch. The tactile representation is computed from consecutive tactile observations:

$$\mathbf{z}_t^\tau = E_\psi(\mathbf{o}_{t-1:t}^\tau). \tag{10}$$

We only update the tactile encoder on contact transitions, where contact is detected by marker displacement or tactile depth thresholding. The flow-tactile critic follows the SAC formulation and is implemented as twin three-layer MLP critics. Each critic takes the robot state, flow representation, tactile representation, and executed action as input:

$$Q_{\eta_i}(\mathbf{q}_t, \mathbf{z}_t^f, \mathbf{z}_t^\tau, \mathbf{a}_t), \quad i \in \{1, 2\}. \tag{11}$$

During warm-start, since the residual action is set to zero, the executed action is given by the frozen base policy, i.e., $\mathbf{a}_t = \mathbf{a}_t^{\text{b}}$. The twin critics are optimized with the Bellman loss:

$$\mathcal{L}_Q = \sum_{i=1}^{2} \mathbb{E}_{\mathcal{B}} \left[\left(Q_{\eta_i}(\mathbf{q}_t, \mathbf{z}_t^f, \mathbf{z}_t^\tau, \mathbf{a}_t) - y_t \right)^2 \right], \tag{12}$$

where the target is computed as

$$y_t = r_t + \gamma \min_{i=1,2} \bar{Q}_{\bar{\eta}_i}(\mathbf{q}_{t+1}, \mathbf{z}_{t+1}^f, \mathbf{z}_{t+1}^\tau, \mathbf{a}_{t+1}). \tag{13}$$

Here, $\bar{Q}_{\bar{\eta}_i}$ denotes the target critic network, and $\mathbf{a}_{t+1}$ is also produced by the frozen base policy during warm-start. The tactile projection layer is trained with both critic supervision and a reconstruction regularizer:

$$\mathcal{L}_E = \mathcal{L}_Q + \lambda_{\text{rec}} \mathcal{L}_{\text{rec}}. \tag{14}$$

The reconstruction decoder is kept frozen, so only the tactile projection layer and lightweight adapter are updated. This warm-start stage bootstraps both the tactile representation and the tactile-aware critic before learning residual actions.

ControlTac trajectory-level augmentation. ControlTac [66] is a SOTA tactile data augmentation method. Because the warm-start buffer contains only a small number of real contact trajectories, we use it to increase tactile data diversity without additional robot interaction. ControlTac is a controllable tactile generation model that synthesizes realistic tactile images from a reference tactile observation, conditioned on desired contact-force changes and contact-pose variations. In our pipeline, it is used only to augment tactile observations using the contact force: the robot states, actions, flow features, rewards, and terminal labels are kept unchanged, while the tactile images are replaced by generated variants that simulate different but physically plausible contact conditions.

After each warm-start rollout, we first identify contact segments using the tactile marker displacement threshold. For each contact trajectory, we perform trajectory-level generation rather than frame-wise independent generation, so that the augmented tactile sequence remains temporally consistent across the rollout. Specifically, for each augmentation, we sample a force perturbation $\Delta f \in [-3, 10]$ and apply it consistently to all tactile samples in the same trajectory. We also apply domain randomization to each generated sample by sampling a pose/control offset from $\{-1, -0.5, 0, 0.5, 1\}$, which introduces small variations in contact location and deformation. For every real trajectory, we generate two augmented tactile trajectories. The augmented transitions are then added to the replay buffer together with the real transitions and are used for tactile encoder adaptation and flow-tactile critic bootstrapping. This augmentation exposes the critic and tactile encoder to richer contact-force and contact-pose variations while preserving the same task-level motion and reward structure.

For the implementation details of ControlTac, given a real tactile image, we first need to remove the visual marker pattern from the underlying tactile deformation because the training dataset of ControlTac is markless. Specifically, we detect the black marker locations and construct a binary marker mask, where marker regions are set to one and the remaining tactile surface is set to zero. We then remove the markers from the original tactile image through image inpainting, producing a marker-free tactile observation that preserves the contact-induced color and geometry changes.

This marker-free image is used as the reference input to ControlTac, which generates new tactile images under different force conditions by perturbing the normal force, e.g., $\Delta F = -3$N and $\Delta F = +3$N. Since the generated outputs do not contain the original marker pattern, we compose the generated marker-free image with the original marker mask by copying the marker pixels back to their corresponding locations. The progress is visualized in Fig. 9

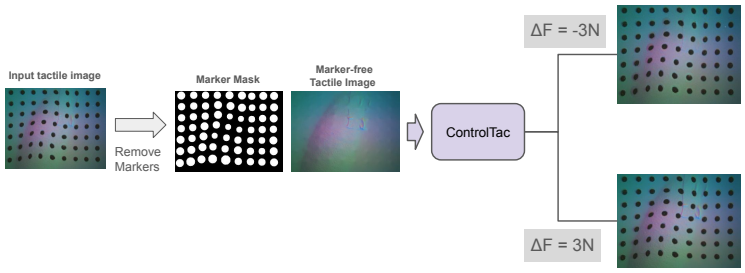

Figure 9: Visualization of ControlTac Generation Process.

A.5 Online Residual Policy Learning

After warm-start, we keep the base policy frozen and train both a residual tactile policy and the critic via SAC. The final action is

$$\mathbf{a}_t = \mathbf{a}_t^{\mathrm{b}} + \mathbf{s}_t \mathbf{a}_t^{\mathrm{r}}. \tag{15}$$

The residual policy is conditioned on proprioception, flow features, tactile features, the contact gate, and the base action chunk:

$$\mathbf{a}_t^{\mathrm{r}} = \pi_\theta^{\mathrm{r}}(\mathbf{q}_t,\, \mathbf{z}_t^f,\, \mathbf{g}_t \cdot \mathbf{z}_t^\tau,\, \mathbf{a}_t, \mathbf{a}_{t:t+K}^{\mathrm{b}}), \tag{16}$$

where

$$g_t = \mathbf{1}[m_t > \epsilon_{\mathrm{contact}}] \tag{17}$$

is a contact-aware tactile gate and m_t is the marker displacement magnitude. The gate prevents the policy from overusing tactile features before contact and encourages tactile-driven corrections only when contact is present. For the other variables, $\mathbf{q}_t$ is the robot proprioception, $\mathbf{z}_t^f$ is the flow representations, $\mathbf{a}_t$ is the current base action, and $\mathbf{a}_{t:t+K}^{\mathrm{b}}$ is the current action chunk, which can provides the temporal reference for residual policy to generate the correction.

For stable real-world learning, we constrain the residual action magnitude:

$$\mathbf{a}_t^{\mathrm{r}} = s_t \tilde{\mathbf{a}}_t^{\mathrm{r}}, \tag{18}$$

where s_t is a scheduled residual scale, defined as 0–0.15.

The final SAC target is:

$$y_t^{\mathrm{rl}} = r_t + \gamma \left[\min_{i=1,2} \bar{Q}_{\bar{\eta}_i}(\mathbf{q}_{t+1}, \mathbf{z}_{t+1}^f, \mathbf{z}_{t+1}^\tau, \mathbf{a}_{t+1}^{\mathrm{b}} + s_{t+1}\tilde{\mathbf{a}}_{t+1}^{\mathrm{r}}) - \alpha \log \pi_\theta^{\mathrm{r}}(\tilde{\mathbf{a}}_{t+1}^{\mathrm{r}} \mid \mathbf{s}_{t+1}) \right]. \tag{19}$$

In our experiments, s_t is gradually increased during online learning so that early residual updates remain conservative and later updates can make stronger contact corrections. This follows the same principle as residual RL methods that keep the learned correction small relative to the base policy to avoid unsafe exploration.

A.6 Reward Shaping Details

We use a normalized multi-sensory reward:

$$r_t = \frac{w_r r_t^{\mathrm{reach}} + w_g r_t^{\mathrm{grasp}} + w_f r_t^{\mathrm{flow}} - w_s r_t^{\mathrm{safety}}}{w_r + w_g + w_f}, \tag{20}$$

where r_t^{reach}, r_t^{grasp}, and r_t^{flow} are normalized to $[0, 1]$. We use $w_r = 1.0$, $w_g = 0.5$, $w_f = 2.0$, and $w_s = 1.0$ for all tasks. Because the positive reward is divided by $w_r + w_g + w_f$, the maximum reward is 1 when all positive components are fully satisfied and the safety penalty is not triggered.

Before grasping, the reaching reward encourages the end-effector to move toward the initial 3D object centroid:

$$r_t^{\text{reach}} = 1 - \text{clip}\left(\frac{\|\mathbf{p}_t^{ee} - \mathbf{c}_0^{3D}\|_2}{d_{\max}^{\text{reach}}}, 0, 1\right), \tag{21}$$

where $d_{\max}^{\text{reach}} = 0.15$ m.

After grasping, the flow reward tracks sparse subgoals from the generated object flow:

$$d_t^{\text{flow}} = \left\| \mathbf{T}_{t_0 \to t}^{\text{rel}} - \hat{\mathbf{T}}_{t_0 \to \ell}^{\text{rel}} \right\|_2, \tag{22}$$

$$r_t^{\text{flow}} = \frac{\ell_t}{L} + \frac{1}{L}\left(1 - \text{clip}\left(\frac{d_t^{\text{flow}}}{d_{\max}^{\text{flow}}}, 0, 1\right)\right), \tag{23}$$

where $L = 30$ is the number of sparse flow subgoals, $\ell_t \in \{0, \ldots, L-1\}$ is the number of reached subgoals before the current one, and $d_{\max}^{\text{flow}} = 0.05$. A flow subgoal is considered reached when $d_t^{\text{flow}} < \epsilon_{\text{flow}}$, where $\epsilon_{\text{flow}} = 0.03$.

The tactile grasp reward encourages stable contact:

$$r_t^{\text{grasp}} = \mathbf{1}\left[\frac{1}{|\Omega|}\sum_{u \in \Omega} D_t^{\tau}(u) > \epsilon_{\text{depth}}\right], \tag{24}$$

where D_t^{τ} is the tactile depth image, Ω is the valid tactile region, and $\epsilon_{\text{depth}} = 0.10$. For marker-based tactile sensing, we instead compute the average marker displacement magnitude:

$$m_t = \frac{1}{|\mathcal{M}|}\sum_{j \in \mathcal{M}} \left\| \Delta \mathbf{u}_t^j \right\|_2, \tag{25}$$

where $\mathcal{M}$ is the set of detected markers and $\Delta \mathbf{u}_t^j$ is the displacement of marker j from its reference position. Marker-based contact is detected when $m_t > \epsilon_{\text{contact}}$, where $\epsilon_{\text{contact}} = 1.5$ pixels.

The safety penalty detects excessive contact:

$$r_t^{\text{safety}} = \mathbf{1}[m_t > \epsilon_{\text{safety}}], \tag{26}$$

where $\epsilon_{\text{safety}} = 8.0$ pixels. If the safety penalty is triggered, we terminate the episode and reset the environment.

A.7 Training and Inference Details

We train all supervised models on a single NVIDIA RTX A6000 GPU. This includes flow-generator fine-tuning, the flow-policy training, and the supervised training or fine-tuning of ACT, DP, and $\pi_{0.5}$.

For real-world RL, we follow a similar asynchronous training setup as prior real-world RL systems such as HIL-SERL [21] and WSRL [23]: policy execution and network optimization run in parallel, so the robot can continue collecting interaction data while the learner updates the critic, residual policy, and tactile encoder from the replay buffer. During both warm-start and online RL, we update the trainable networks every 500 optimization steps.

In the warm-start stage, the residual action is set to zero, where only the critic and tactile encoder are optimized. After each warm-start episode, we run ControlTac [66] augmentation on the collected contact segments and add the augmented tactile transitions to the replay buffer. ControlTac augmentation is executed asynchronously together with flow-generator inference, so tactile augmentation does not block robot execution.

For deployment-time inference, the robot runs the frozen visual base policy and the residual tactile policy on a single NVIDIA RTX 5080 GPU.

A.8 Hyperparameters

Unless otherwise specified, we use the same hyperparameters across all tasks and base policies. All actions are represented in the robot end-effector frame and normalized to $[-1, 1]$ before being passed to the policy. The control frequency is 10 Hz, and each episode has a maximum length of 300 control steps.

Table 3: Implementation hyperparameters used in OMNITACTUNE.

Component	Hyperparameter	Value
Data collection and retargeting		
Camera	Third-view RGB frame rate	30 Hz
Quest	Quest hand tracking rate	60 Hz
Pose smoothing	Translation smoothing window	5 frames
Finger retargeting	Closed/open hand distances $(d_{\min}, d_{\max})$	$(5, 12)$ cm
Finger retargeting	Gripper range $[a_{\min}, a_{\max}]$	$[0, 1]$
Finger retargeting	Aperture scale/offset (α, β)	$(1.0, 0.0)$
Grasp detection	Aperture / velocity threshold	0.35 / -0.02 m/s
Flow generator and flow policy		
Object keypoints	Number of tracked keypoints N	32
Flow generator	Demonstrations per task	50
Flow generator	LoRA rank / alpha	$16/32$
Flow generator	Learning rate / batch size / epochs	1×10^{-4} / 8 / 500
Flow loss	Smoothness weight λ_{smooth}	0.1
Sparse flow	Number of sparse subgoals L	30
Flow policy	Action chunk horizon K	8
Warm-start and tactile representation		
Warm-start	Duration	12 min
Replay buffer	Maximum size	1×10^4 transitions
Tactile latent	Latent dimension d_τ	32
Encoder loss	Reconstruction weight λ_{rec}	0.1
Encoder update	Contact-only update	Yes
ControlTac trajectory-level augmentation		
ControlTac	Augmented trajectories per real trajectory	2
ControlTac	Force perturbation Δf	$\mathcal{U}[-3, 10]$
ControlTac	Pose/control offset candidates	$\{-1, -0.5, 0, 0.5, 1\}$
ControlTac	Generation granularity	trajectory-level
Online residual RL		
RL algorithm	Backbone	SAC
SAC	Discount / target update (γ, τ)	$(0.99, 0.005)$
SAC	Actor / critic learning rate	3×10^{-4}
SAC	Encoder learning rate	1×10^{-4}
SAC	Batch size / updates per step	256 / 1
Policy Residual scale	Schedule s_t	$0.05 \rightarrow 0.10 \rightarrow 0.15$
Residual scale	Schedule interval	every 500 online steps

Table 4: Reward, contact-gate, and safety hyperparameters. The positive reward is normalized to $[0, 1]$, while the safety term is applied as an additional penalty.

Component	Hyperparameter	Value
Reward weights and normalization		
Reward	Reaching weight w_r	1.0
Reward	Grasp/contact weight w_g	0.5
Reward	Flow weight w_f	2.0
Reward	Safety penalty weight w_s	1.0
Reward	Positive reward normalization	$w_r + w_g + w_f = 3.5$
Reward	Positive reward range	$[0, 1]$
Safety reward	Safety penalty range	$[-1, 0]$
Reaching and flow reward		
Reaching reward	Distance normalization $d_{\max}^{\text{reach}}$	0.15 m
Flow reward	Number of sparse subgoals L	30
Flow reward	Flow error normalization $d_{\max}^{\text{flow}}$	0.05
Flow reward	Flow subgoal threshold ϵ_{flow}	0.03
Subgoal switching	Policy subgoal threshold $\epsilon_{\text{subgoal}}$	0.03
Contact detection, tactile gate, and safety		
Tactile gate	Marker gate threshold ϵ_{gate}	1.5 px
Contact detection	Marker contact threshold $\epsilon_{\text{contact}}$	1.5 px
Contact detection	Tactile depth threshold ϵ_{depth}	0.10
Safety reset	Marker safety threshold ϵ_{safety}	8.0 px
Safety reset	Termination condition	$m_t > \epsilon_{\text{safety}}$

B Experimental Settings

Peg-in-Hole Task. In this task, the robot grasps a cylindrical peg and inserts it into a target receptacle placed at different tabletop locations. The task evaluates whether the policy can generalize the reaching and grasping motion across spatial variations while still performing precise contact-rich insertion. More details have been shown in Fig. 10, where the tolerance of the insertion task is 5mm. Although the visual base policy can usually bring the peg close to the hole, small pose errors during insertion often lead to jamming or failed contact. Tactile feedback is therefore critical for detecting contact, correcting local misalignment, and maintaining a stable grasp during the final insertion stage.

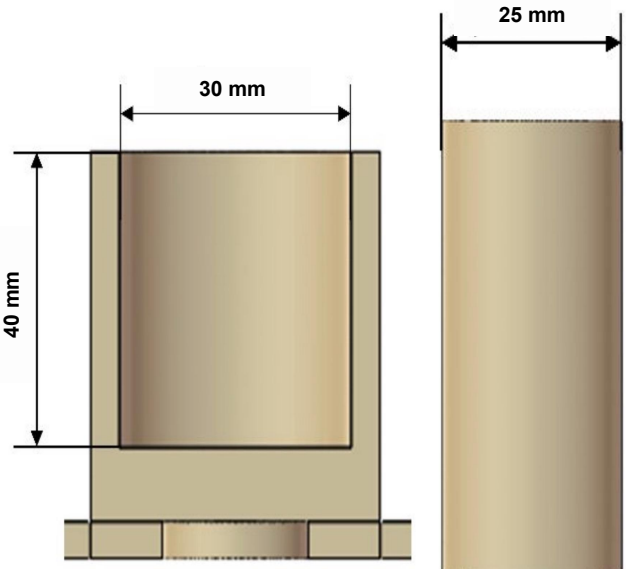

Figure 10: Setting of Peg-in-Hole task.

Charger Insertion Task. The robot has been grasped a charger and plan to insert it into a phone charging port. This task is substantially more precise than standard insertion because both the charger head and the socket tolerance are very small. As shown in Fig. 11, the metal prong width is only about 7 mm, while the corresponding outlet slot length is about 10 mm. For the smaller side, the metal prong thickness is about 1.5 mm and the socket width is about 2 mm, leaving only a narrow clearance for successful insertion. As a result, visual guidance alone often brings the charger near

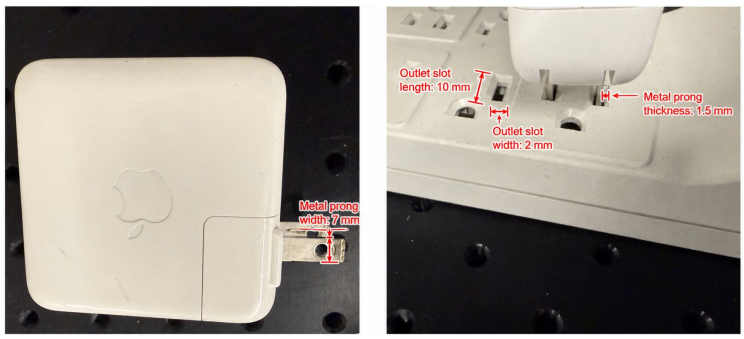

Figure 11: Setting of Charger Insertion task.

the port but cannot reliably resolve the final contact alignment. The robot must use tactile feedback to explore contact, detect lateral misalignment, and make small corrective motions before insertion. To improve robustness, we apply small domain randomization for the initial robot position during both training and inference by perturbing the initial pose within 5 cm along the x, y, and z directions and within 10° around the vertical axis. Since the charger is already grasped and the task is challenging, we restrict the action space of the residual policy to translational corrections in x, y, and z and rotational correction in yaw.

Cap Opening Task. In this task, the robot grasps a cap opener and uses it to open a bottle cap. Unlike pure insertion tasks, this setting requires maintaining a precise tool pose while the contact geometry changes dynamically during levering. The opener must first establish contact with the cap edge, then preserve this contact while applying a rotational motion to lift the cap. Failure modes include slipping off the cap, contacting the wrong region, or applying force with an incorrect tool orientation. This task therefore tests whether tactile residual adaptation can stabilize dynamic contact and correct the tool pose during forceful interaction. Similar to Charger Insertion,

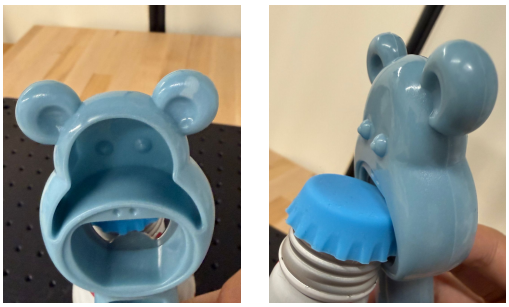

Figure 12: Setting of Cap Opening task.

we apply small domain randomization for the initial robot position during both training and inference by perturbing the initial pose within 5 cm along the x, y, and z directions and within 10° around the vertical axis. The setting is visualized in Fig. 12.

Box Opening Task. The robot uses a lever-type opener to open the GelSight box by catching and lifting a very thin edge. This task is particularly challenging because the available working edge is only about 1.3 mm thick, as shown in Fig. 13. The robot must precisely align the opener with this narrow edge, establish contact without slipping, and maintain the contact while executing a dynamic levering motion. Small visual or pose errors can easily cause the opener to miss the edge or slide away from it. Thus, the task strongly depends on tactile feedback for detecting edge contact, maintaining stable engagement, and correcting the manipulation trajectory during opening. Same as the other two tasks, we apply small domain randomization for the initial robot position during both training and inference by perturbing the initial pose within 5 cm along the x, y, and z directions and within 10° around the vertical axis.

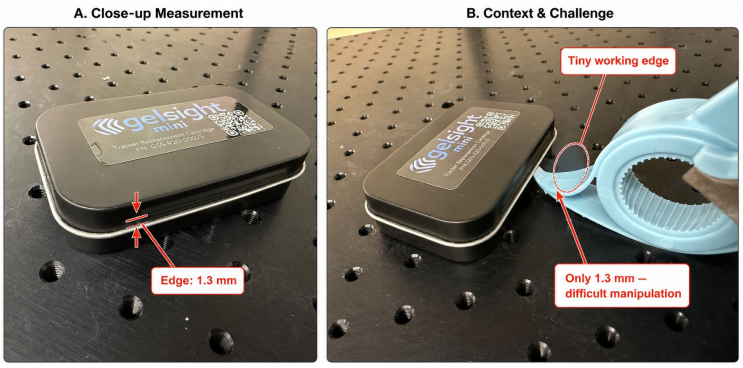

Figure 13: Setting of Box Opening task.

C Additional Experiments

C.1 Analysis of Collected Data and Base Policies

In this section, we want to analyze the trajectory quality of the base policies and the data. To evaluate their quality, we propose to use the smoothness of the tracked keypoints centroid on the object, since all sources share this representation. Following prior work [75], we use two complementary smoothness metrics: SPARC and LDLJ.

SPARC measures the arc length of the normalized Fourier magnitude spectrum of the speed profile, capturing high-frequency oscillations in the trajectory:

$$\text{SPARC} \triangleq -\int_0^{\omega_c} \left[\left(\frac{1}{\omega_c} \right)^2 + \left(\frac{d\hat{V}(\omega)}{d\omega} \right)^2 \right]^{1/2} d\omega, \tag{27}$$

where $\hat{V}(\omega) = V(\omega)/V(0)$ is the normalized magnitude spectrum of v_t, and ω_c is an adaptive cutoff frequency determined by a spectrum amplitude threshold. A larger SPARC value, i.e., closer to zero, indicates a smoother trajectory with fewer high-frequency components.

We further use LDLJ, which measures smoothness in the time domain by penalizing abrupt changes in acceleration:

$$\text{LDLJ} \triangleq -\log \left(\frac{(t_2 - t_1)^5}{v_{\text{peak}}^2} \int_{t_1}^{t_2} \left\| \frac{d^2 v(t)}{dt^2} \right\|_2^2 dt \right), \tag{28}$$

where t_1 and t_2 denote the start and end time of the trajectory, and $v_{\text{peak}} = \max_{t \in [t_1, t_2]} v(t)$ is the peak speed. LDLJ is dimensionless and penalizes jerky motions caused by rapid acceleration and deceleration changes. We report mean SPARC and LDLJ over successful rollouts, denoted as M-SPARC and M-LDLJ, where larger values indicate smoother motion.

We compare four trajectory sources that share the same object-centric keypoint representation: human demonstrations, teleoperation demonstrations, the policy trained from human demonstrations, and the policy trained from teleoperation demonstrations. In addition to quantitative metrics, we visualize successful trajectories from the same initial setting for all four groups, allowing direct comparison of the trajectory shape and contact-stage behavior.

Table 5 and Fig. 14 shows that human demonstrations and the human policy produce substantially smoother object trajectories than their teleoperation counterparts. Human demonstrations achieve an M-SPARC of -2.889 and M-LDLJ of -18.03, while teleoperation demonstrations drop to -7.320 and -19.70, indicating more oscillations and jerkier motion. This gap becomes larger after policy learning: the human policy remains smooth with M-SPARC -2.664 and M-LDLJ -17.29, whereas the teleoperation policy further degrades to M-SPARC -9.521 and M-LDLJ -20.89.

These results suggest that teleoperation data is less suitable for contact-rich manipulation. During teleoperation, the human operator cannot directly feel contact forces at the object, making precise

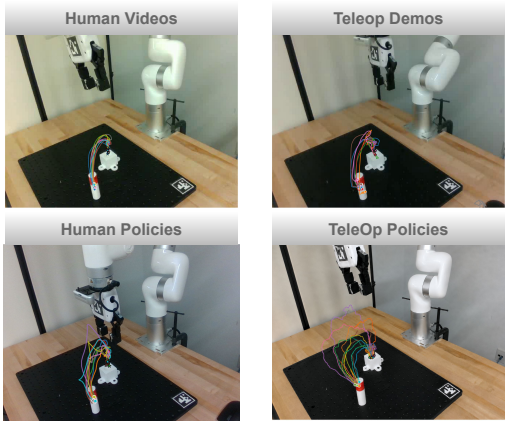

Figure 14: Visualization of centroid trajectories.

Table 5: Trajectory smoothness comparison using the centroid of tracked object keypoints.

Group	M-SPARC ↑	M-LDLJ ↑
Human Demos	−2.889	−18.03
Teleop Demos	−7.320	−19.70
Human Policy	−2.664	−17.29
TeleOp Policy	−9.521	−20.89

contact exploration and insertion difficult. As a result, teleoperation trajectories often contain noisy corrections and jerky motions, especially during the contact-rich phase. When a policy imitates such data, these artifacts can be amplified, producing unstable contact behavior and making tactile residual refinement more difficult. This also indicates that collecting teleoperation data for more challenging contact-rich tasks can become impractical, as the demonstrations themselves may not provide sufficiently smooth and reliable motion priors.

C.2 Ablation Studies

Reward Shaping We conduct ablation study for our multi-sensory reward shaping method in the Peg-in-Hole task. As shown in Fig. 15, removing any component of our reward design leads to slower learning and lower final performance, demonstrating the importance of multi-sensory reward shaping. The reaching reward provides guidance for grasping the object, while the dense flow reward further offers object-centric motion supervision during insertion. In addition, the tactile reward captures grasping signals and adding penalties for unexpected behaviors.

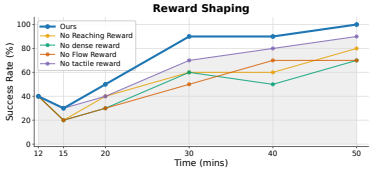

Figure 15: Ablation Study of the Multi-sensory Reward Shaping

Residual Policy Design We further ablate our residual policy design in Fig. 16. Our method uses trajectory-level keypoint guidance, which is temporally sparser than per-step keypoint conditions but still providing a structured motion prior. As shown in the figure, per-step keypoint will over-constrain the policy, which leads the base policy such as ACT failed because of its highly varied actions. In contrast, trajectory-level keypoints leaves enough reactivity for residual RL to refine local contact-rich behaviors. We also find that keypoint-conditioned policy is more effective than using visuo-tactile observations, since keypoints provide a compact object-centric guidance. Finally, contact-aware gating also improves performance by reducing unnecessary corrections in non-contact stage and enabling stronger tactile-driven corrections during contact.

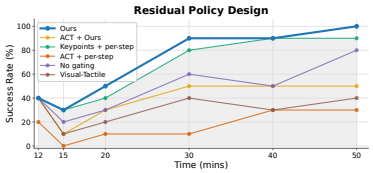

Figure 16: Ablation between our residual policy design and per-step keypoint reaching policy, visuo-tactile policy, or policy after removing contact-aware gating.

Warm-Start Strategies We then conduct ablation study for the design choices in our warm-start stage. As shown in the Fig. 17, removing warm-start optimization on the tactile encoder and critic leads to unstable early stage and weak overall performance, which indicates that optimizing the tactile representation and critic during warm-start is important for stable residual RL. We also observe that removing ControlTac [66] also results in unstable performance, highlighting that tactile augmentation in the warm-start stage can refine the tactile encoder more efficiently. In addition, removing the critic loss or the reconstruction loss both degrades final performance, showing that the two objectives play complementary roles in extracting task-level features and restoring tactile structures.

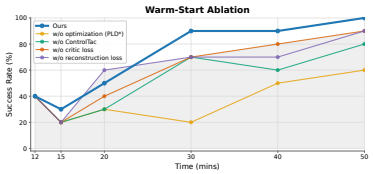

Figure 17: Ablation study of the warm-start stage in showing the importance of optimization of both tactile encoder and critic.

Action Space Design We further ablate our action space design in Fig. 18. As shown in the figure, removing the scheduler leads to slower learning and lower final performance, suggesting that stage-aware action scheduling is important for balancing exploration and stable refinement during contact-rich manipulation. We also observe that removing action scaling significantly hurts performance, indicating that unconstrained residual corrections are harder to optimize and can easily destabilize training. In addition, using a larger action scale of 0.5 with scheduler improves over removing action scaling, but still underperforms our full design, showing that $0, 15$ is an appropriate ratio. Overall, these results validate that both the scheduler and action scaling are important for making residual actions more stable and sample-efficient.

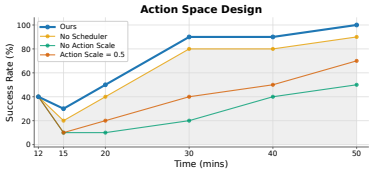

Figure 18: Ablation study of the action space in showing the importance of scheduler and the scale number.

D Failure Cases

In this section, we further analyze representative failure cases in the **Cap Opening** and **Box Opening** tasks, as shown in Fig. 19. First, even when the object reaches the cap, the contact force may be insufficient or imprecisely applied, leading to slipping rather than successful opening. Second, the policy may fail to align the object with the box edge, causing the contact point to miss the effective opening region. Third, the object pose can become tilted during manipulation, which changes the contact geometry and prevents the robot from applying force in the desired direction. These failures suggest that cap opening and box opening require not only reaching the correct location, but also maintaining a stable object pose and generating accurate contact-rich corrective motions.

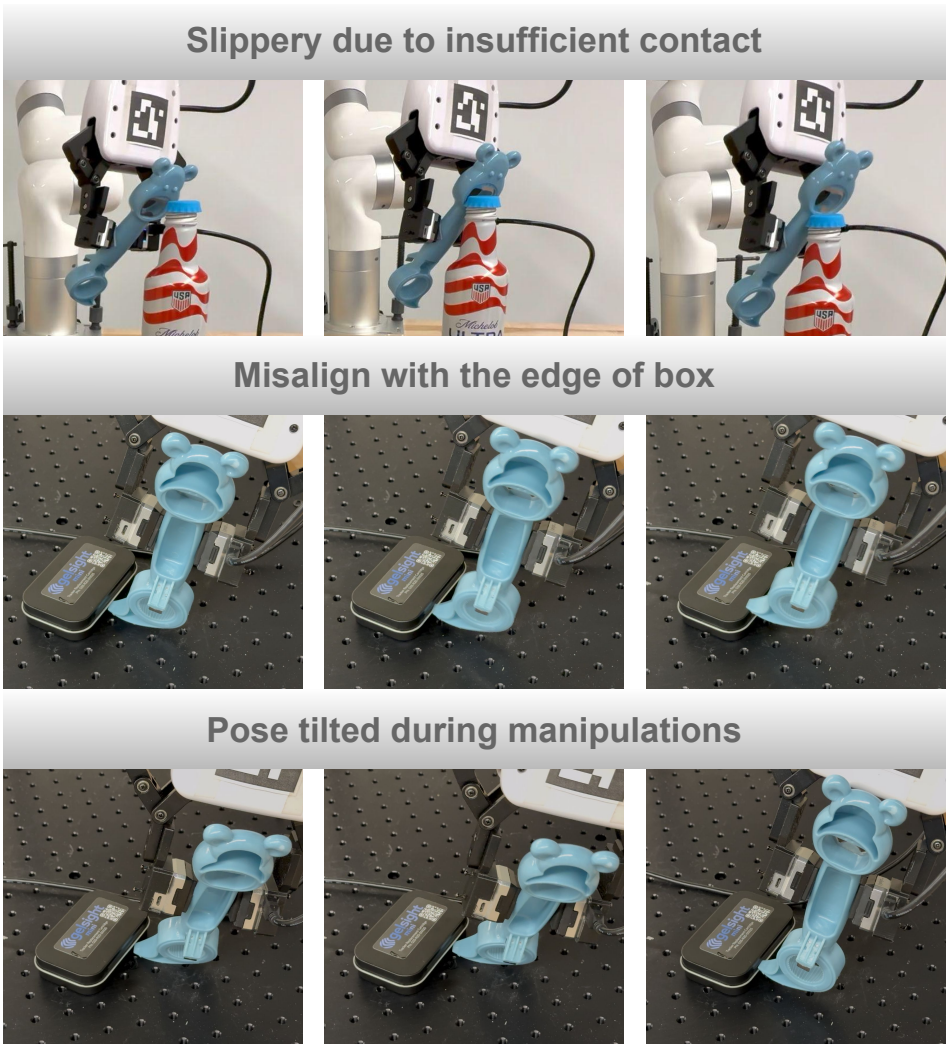

Figure 19: Visualization of the failure cases on **Cap Opening** and **Box Opening**.

E Additional plots

In this section, we visualize the training curve of the success rates across different visual base policies and tactile representations, which are additional results of Sec. 4.2 and Sec. 4.4. The plots are shown in Fig. 20 and Fig. 21.

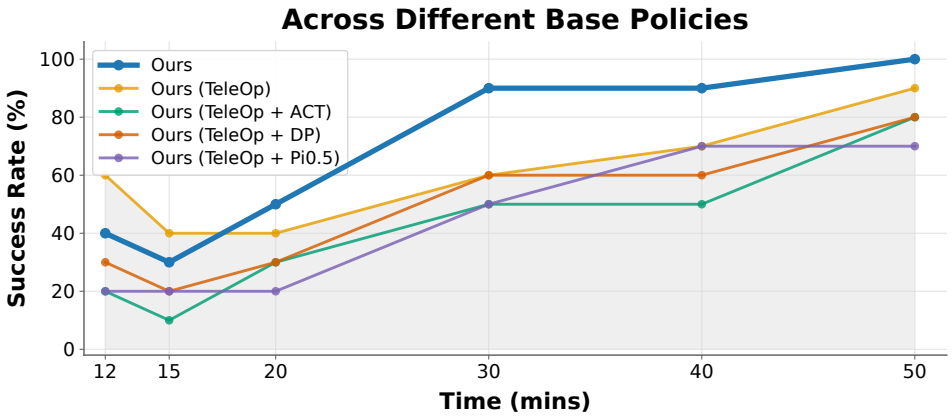

Figure 20: The training success rates for different policies, which is shown in Sec. 4.2.

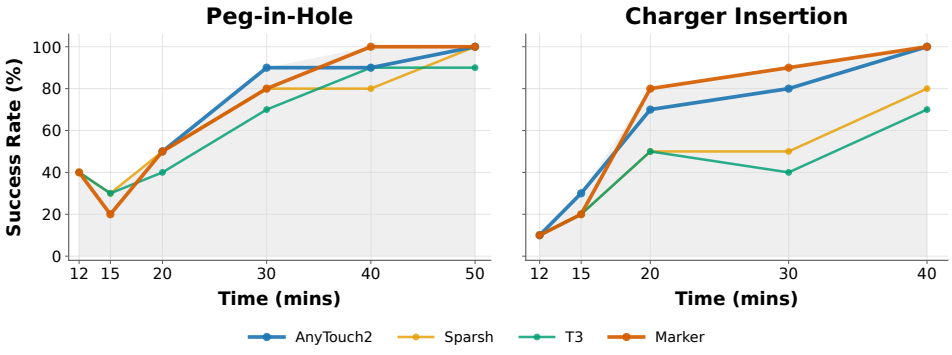

Figure 21: The training success rates for different policies, which is shown in Sec. 4.4.