

ProxiDex: Learning Dynamics-Guided Proximity Policy for Dexterous Manipulation

Yushan Bai^{1,2,*} Boyu Zheng^{1,2,*} Zhiyang Mao⁴ Hongzheng Sun^{1,2}
Yuchuang Tong^{1,2,†} En Li^{1,2,3} Zhengtao Zhang^{1,2,3,†}

¹CAS Engineering Laboratory for Intelligent Industrial Vision, Institute of Automation, Chinese Academy of Sciences

²School of Artificial Intelligence, University of Chinese Academy of Sciences

³Beijing Zhongke Huiling Robot Technology Co., LTD.

⁴School of Intelligent Science and Technology, Xinjiang University

*Equal contribution. †Corresponding authors.

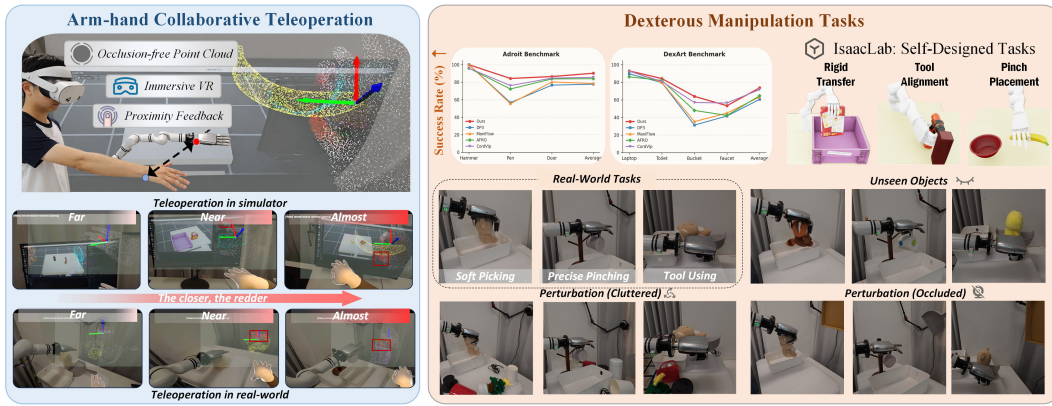


Figure 1: We propose ProxiDex, a framework that integrates teleoperation with proximity-policy learning. The left panel shows immersive teleoperation with proximity feedback, while the right panel demonstrates robust performance in both simulation and real-world dexterous manipulation.

Abstract: Multi-finger dexterous manipulation relies on stable hand-object interactions, yet these interactions are partially observable in practice. Visual observations are often occluded by the hand, tactile sensors introduce hardware-specific modalities and calibration burdens, and existing policies rarely model how these cues evolve under actions, making them brittle under contact uncertainty. To address these, we present ProxiDex, a dynamics-guided proximity policy framework that treats hand-object proximity as an interaction state for dexterous manipulation. ProxiDex reconstructs interaction point clouds and converts geometric distances into proximity cues, forming a hardware-agnostic contact representation that provides immersive feedback during VR teleoperation. Built on this representation, ProxiDex learns action-conditioned proximity dynamics with a coupled forward-inverse design: future observation latents are predicted from actions, while proximity variations are decoded from latent changes. Leveraging these dynamics, ProxiDex adaptively reweights proximity tokens across manipulation phases and uses dynamics-consistency supervision to guide policy inference, stabilizing action generation under unreliable visual feedback. Simulation and real-world experiments demonstrate improved success rates and robustness over representative baselines across standard, unseen objects, and perturbation scenarios. Additional visualizations are available at <https://proxidex.github.io/>.

Keywords: Dexterous Manipulation, Interaction Dynamics, Imitation Learning

1 Introduction

Dexterous manipulation with multi-fingered hands is a fundamental capability for human-like robotic systems [1]. Compared with parallel jaw grippers, dexterous hands require coordinated control of arm-level motion and fingertip-level interactions to accomplish complex tasks such as grasping, insertion, and tool use [2, 3]. A key requirement for learning such skills is the ability to represent local hand-object geometric interaction, which governs how contact is formed, transferred, and maintained during manipulation [4]. However, acquiring such interaction cues from demonstrations remains challenging. Vision-based teleoperation systems [5, 6] suffer from frequent occlusions that obscure critical contact regions, while tactile sensing provides limited scalability due to heterogeneous hardware designs and costly calibration across platforms [7, 8]. As a result, reliable local interaction information is often missing or inconsistent in collected data, limiting the ability of visuomotor policies to infer the underlying and evolving physical interaction structure.

This limitation exposes a broader gap in existing imitation learning approaches for dexterous manipulation. Most methods [9, 10, 11] directly learn end-to-end mappings from visual observations to actions, relying on implicit feature representations without explicitly modeling hand-object geometric proximity or its temporal evolution. Even when contact-related signals are incorporated, they are typically treated as auxiliary inputs without modeling how actions induce changes in interaction states or how interaction structure constrains future control [12, 13]. This leads to brittle behavior under occlusion, tracking noise, or contact uncertainty, where policies often generate physically inconsistent actions and fail to generalize beyond clean demonstrations.

To address these challenges, we propose **ProxiDex**, a dynamics-guided proximity policy learning framework that builds on hand-object geometric proximity as a coherent interaction representation and models its action-conditioned evolution for robust policy learning. As shown in Fig. 1, within a VR teleoperation setting [14], we reconstruct interaction point clouds and visualize them as immersive proximity feedback for operators. We further convert point-cloud distances into hardware-agnostic proximity feedback, enabling scalable inference of local interaction cues without tactile instrumentation. Based on this representation, ProxiDex learns forward-inverse latent dynamics that couple action-driven observation transitions with proximity variations, capturing the evolution of local hand-object interactions during manipulation. Leveraging these interaction dynamics, ProxiDex adaptively reweights proximity tokens across manipulation phases, balancing global observation guidance during motion with local proximity constraints during interaction. Dynamics consistency further guides online inference, stabilizing action generation under occlusion and tracking failures.

The main contributions of ProxiDex are summarized as follows: 1) We propose a data-construction pipeline and a VR-based teleoperation with proximity feedback, which provides scalable local interaction cues without tactile instrumentation. 2) We develop a forward-inverse latent dynamics model to capture action-conditioned evolution between observation transitions and proximity variations. 3) We introduce a trajectory-adaptive proximity policy with dynamics-consistency guided inference, improving robustness under occlusion and tracking failures.

2 Related Work

Demonstration data acquisition for dexterous manipulation. High-quality demonstration data are fundamental to learning dexterous manipulation. Existing data-collection systems often trade off sensing accuracy against deployment cost [9, 15]. Methods based on data gloves or tactile sensing devices [16, 17, 18] can provide fine-grained contact feedback, but they typically rely on expensive hardware and require calibration, maintenance, and cross-platform alignment. Vision-based teleoperation methods [6, 19, 20, 21] reduce deployment barriers, yet they struggle to capture local proximity relations reliably under complex hand-object occlusions. UMI-style devices [22, 23] and video-parsing methods [24, 25] offer promising scalability, but they are often constrained by specific end-effector designs or the quality of 3D reconstruction. In contrast, we adopt a VR-based

arm-hand co-teleoperation pipeline and explicitly recover interaction structure through geometric proximity, enabling scalable and interaction-aware data collection.

Imitation learning for dexterous manipulation. Imitation learning has advanced dexterous manipulation by modeling action distributions with Transformers or diffusion models [9, 10]. Yet commonly used 2D observations struggle to capture the fine spatial relationships required for multi-finger contact. Point-cloud-based policies improve geometric awareness [11, 26, 27], but these methods still fail to maintain a consistent representation of interaction states under occlusion. Reconstruction and pose-based methods recover more complete geometry [1], but mainly emphasize static geometric alignment rather than the temporal evolution of contact states. These limitations indicate that existing methods lack an explicit modeling of interaction as a structured and evolving state, motivating our explicit proximity-based interaction representation.

Dynamics-aware representation learning. Using dynamics models to enhance the physical awareness of policies has become an important direction in dexterous manipulation. Latent dynamics pretraining methods [28, 29] improve the dynamics sensitivity of observation encoders by predicting future states, but their latent representations often lack explicit physical meaning. Multimodal world models [30, 31, 32] assist action generation through future latent prediction, yet their complex model structures and inference pipelines can limit deployment efficiency. Geometry-evolution prediction method [33] can capture changes in point-cloud positions and velocities, but they remain limited in representing fine-grained contact logic in dexterous manipulation. Particle-based world models [34, 35] can reconstruct hand-object interaction processes, but they rarely model interaction explicitly as a bidirectional evolution of geometric proximity under actions. Motivated by these observations, we model interaction as a bidirectional evolution of geometric proximity under action influence, enabling explicit reasoning over contact dynamics.

3 Problem Formulation

Given an expert demonstration dataset $\mathcal{D} = \{\tau_i\}$, each trajectory $\tau = \{(o_t, a_t^E)\}_{t=1}^T$ consists of observations o_t and expert actions a_t^E . Imitation learning aims to train a parameterized policy π_θ to match expert behavior. For 3D point-cloud observations, an encoder f_ϕ is typically used to map o_t into a latent representation $z_t = f_\phi(o_t)$, from which the policy generates an action distribution $a_t \sim \pi_\theta(\cdot|z_t)$.

To improve the policy’s awareness of future physical evolution, prior methods often introduce a latent dynamics model h_ψ as auxiliary supervision. Given the current latent representation z_t and a future k -step sequence of expert actions $a_{t:t+k-1}^E$, the model predicts the future latent state and is optimized through a feature reconstruction loss:

$$\hat{z}_{t+k} = h_\psi(z_t, a_{t:t+k-1}^E), \quad \mathcal{L}_{dyn} = \mathbb{E}_{\tau \sim \mathcal{D}} [\|\hat{z}_{t+k} - f_\phi(o_{t+k})\|_2^2] \quad (1)$$

However, directly modeling the observation latent space is insufficient for contact-rich dexterous manipulation. First, frequent hand-object occlusions cause the raw observation o_t to miss fingertip contact regions, making interaction cues unobservable. Second, when only $\mathcal{L}_{dyn}$ is used to enforce future feature consistency, the model tends to prioritize salient motion patterns such as object-pose changes while overlooking the fine-grained evolution of hand-object distance and contact state. Our central motivation is therefore to construct a more physically meaningful interaction representation. We first reconstruct an occlusion-free hand-object interaction point cloud o_t^p to mitigate geometric information loss. We then introduce latent interaction dynamics, consisting of a forward dynamics model and an inverse differential module, to help the policy anticipate proximity and contact-state evolution and generate reliable actions under perturbation.

4 Method

We propose **ProxiDex** to address visual occlusion and missing local interaction cues in dexterous manipulation, as illustrated in Fig. 2. The method proceeds in three stages: (1) interaction-aware data

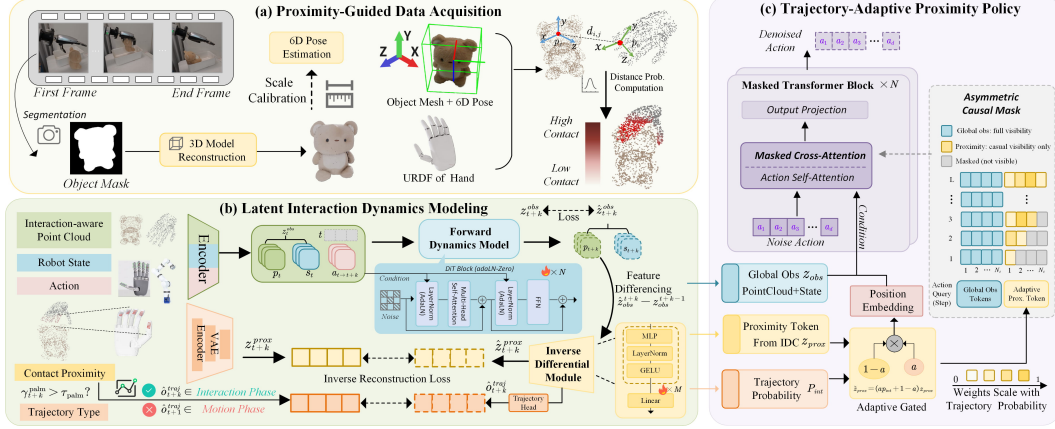


Figure 2: Overview of ProxiDex. It combines proximity-aware data acquisition, latent dynamics modeling, and trajectory-adaptive policy learning for robust dexterous manipulation.

acquisition, which reconstructs hand–object interaction point clouds from manipulation images and robot states (Sec. 4.1); (2) latent interaction dynamics modeling, which extracts region-level proximity cues and learns action-conditioned proximity evolution (Sec. 4.2); and (3) trajectory-adaptive policy inference, which incorporates these cues into a diffusion policy to reweight proximity tokens across manipulation phases, while dynamics-consistency guidance supports closed-loop inference for robust action generation (Secs. 4.3 and 4.4; Fig. 3).

4.1 Interaction-Aware Data Acquisition

To collect high quality demonstrations that reflect hand–object contact relations, we build an interaction-aware data collection pipeline that unifies visual observations, robot proprioception, and proximity information in the robot base frame, as illustrated in Appendix Fig. 10. Demonstrations are collected using a VR arm–hand co–teleoperation system based on Hand Tracking Streamer [14], which synchronously records robot states, target–object poses, and expert actions. Details of teleoperation control, action retargeting, and proximity feedback in VR are provided in Appendix A.1.

To represent hand–object interaction details, we first use Grounded-SAM [36] and SAM3D [37] to reconstruct the target object from a single initial image, followed by metric scale calibration to obtain the object model $\mathcal{M}_o$, and then track its 6D pose during demonstrations using FoundationPose++ [38]. The full procedure is described in Appendix A.2. Inspired by interaction-centric grasp representations [39], we sample surface points from both the reconstructed object mesh and the hand-link at time t . As shown in Fig. 2(a), the sampled points are then transformed into the robot base frame to obtain P_t^o and P_t^h . The interaction point cloud observation at time t is then defined as $o_t^p = P_t^o \cup P_t^h$. Compared with raw scene point clouds, the interaction point cloud preserves hand–object geometric relations more reliably and reduces the influence of background noise.

From o_t^p , we derive hand–object proximity observations by computing local distances between sampled hand surface points and object surface points. The resulting point-wise proximity strengths are then aggregated within each physical hand region S_k , yielding the region-level proximity vector $o_t^{\text{prox}} = [\gamma_t^1, \dots, \gamma_t^K]$. The detailed computation is provided in Appendix A.3. In addition, we use the palm-region response γ_t^{palm} to generate a trajectory-phase label $o_t^{\text{traj}} = \mathbb{I}(\gamma_t^{\text{palm}} > \tau_{\text{palm}})$, where $o_t^{\text{traj}} = 0$ denotes the approach phase and $o_t^{\text{traj}} = 1$ denotes the contact–interaction phase. This label provides a coarse motion-phase prior for subsequent models. Each demonstration frame is finally represented as $(o_t^p, o_t^{\text{prox}}, o_t^{\text{traj}}, s_t, a_t)$, where s_t is the robot proprioceptive state and a_t is the expert action. This representation jointly contains the interaction point cloud, region-level proximity, and trajectory-phase label, providing a physically interpretable interaction observation for latent dynamics modeling and policy learning.

4.2 Latent Interaction Dynamics Modeling

After obtaining the interaction-aware demonstration representation, we model latent interaction dynamics with self-supervised learning, encouraging the observation encoder to capture action-driven changes in hand-object proximity. As illustrated in Fig. 2(b), we first pretrain a VAE on the proximity observation o_t^{prox} , compressing region-level proximity responses into a proximity latent variable z_t^{prox} and reconstructing $\hat{z}_t^{\text{prox}}$ to obtain a stable contact representation. This proximity latent variable then serves as a physical anchor for aligning visual and contact cues.

Next, the observation encoder E_θ encodes the interaction point cloud o_t^p and robot state s_t into an observation latent variable z_t^{obs} . A forward dynamics model $\text{FDM}(\cdot)$ based on the DiT architecture [40] takes z_t^{obs} and the action sequence $a_{t:t+h-1}$ as input and predicts future observation latents. Meanwhile, the inverse differential module $\text{IDM}(\cdot)$ infers the proximity latent at the corresponding time step from predicted latent variable differences and aligns it with the VAE-derived z_t^{prox} . Details are provided in Appendix B.1:

$$\mathcal{L}_{\text{for}} = \frac{1}{H} \sum_{h=1}^H \|\text{FDM}(z_t^{\text{obs}}, a_{t:t+h-1}) - z_{t+h}^{\text{obs}}\|_2^2, \quad \mathcal{L}_{\text{inv}} = \frac{1}{H} \sum_{h=1}^H \|\text{IDM}(z_{t+h}^{\text{obs}} - z_{t+h-1}^{\text{obs}}) - z_{t+h}^{\text{prox}}\|_2^2 \quad (2)$$

$\text{FDM}(\cdot)$ constraint encourages z_t^{obs} to encode visual interaction and proprioceptive cues relevant to future state evolution, while $\text{IDM}(\cdot)$ alignment makes latent transitions sensitive to proximity variations. To distinguish trajectory phases, we introduce a trajectory head that predicts the interaction-phase probability p_{int}^t from z_t^{obs} , supervised by the trajectory label o_t^{traj} defined in Sec. 4.1 with a cross-entropy loss $\mathcal{L}_{\text{traj}}$. This stage is optimized with the following multi-task objective:

$$\mathcal{L}_{\text{stage1}} = \lambda_v \mathcal{L}_{\text{vae}} + \lambda_f \mathcal{L}_{\text{for}} + \lambda_c \mathcal{L}_{\text{inv}} + \lambda_t \mathcal{L}_{\text{traj}}. \quad (3)$$

After training, the observation encoder E_θ maps hand-object geometric observations into latent representations that capture the coupling between action-driven state transitions and proximity changes. These representations provide the state basis for downstream policy generation.

4.3 Trajectory-Adaptive Proximity Policy

During policy learning, we freeze the observation encoder trained in the previous stage and use a Transformer-based denoising policy to generate actions. The core design is a trajectory-adaptive asymmetric causal encoder. Specifically, the global observation feature z_t^{obs} is fully visible to all action steps and provides high-level guidance about the manipulation scene. In contrast, the proximity-token sequence z_t^{prox} is constrained by a causal memory mask M_{causal} , so that the action at step s can access only the current and historical proximity states (Fig. 2(c)). This branch is therefore responsible for local action refinement over time. Architectural details are given in Appendix B.2. During Stage-2 training, z_t^{prox} is obtained from the frozen VAE, whereas at inference it is replaced by the IDM estimate $\hat{z}_t^{\text{prox}}$, which is trained to match the same VAE latent space through $\mathcal{L}_{\text{inv}}$.

We further use the interaction-phase probability p_t^{int} from the trajectory head in Sec. 4.2 to adaptively regulate the proximity branch: the local interaction branch is suppressed during the motion phase and strengthened during the interaction phase. Given an expert action sequence A_0 , diffusion step k , noisy action A_k , and noise ϵ , the training objective of the second stage is:

$$\mathcal{L}_{\text{stage2}} = \mathbb{E}_{A_0, \epsilon, k} \left[\|\epsilon - \epsilon_\theta(A_k, k \mid z_t^{\text{obs}}, g(p_t^{\text{int}}) z_t^{\text{prox}})\|_2^2 \right] + \lambda_{\text{gate}} \mathcal{L}_{\text{gate}} \quad (4)$$

Here, $g(p_t^{\text{int}})$ is a gate conditioned on the interaction-phase probability, and $\mathcal{L}_{\text{gate}}$ regularizes the gate distribution. This design allows the policy to rely on global visual guidance during the motion phase and adaptively strengthen proximity feedback during the interaction phase, producing more stable contact-aware actions throughout dexterous manipulation.

4.4 Dynamics-Consistency Guided Policy Inference

During online inference, we combine the policy with the forward dynamics model $\text{FDM}(\cdot)$ to form a closed-loop controller with physical consistency checking, as shown in Fig. 3. At each time step, the

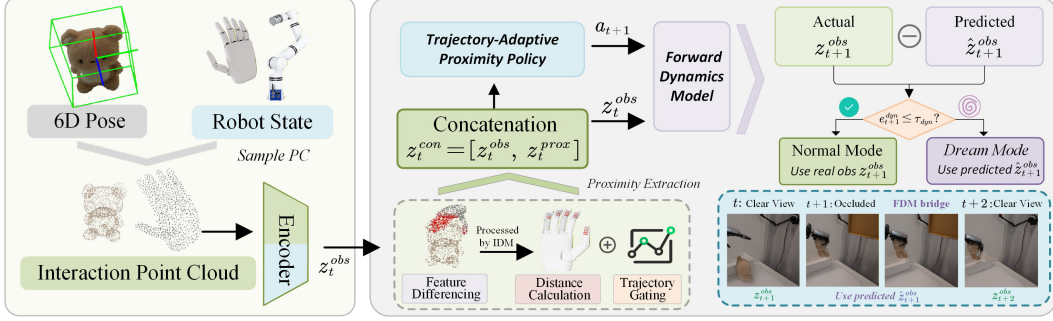


Figure 3: Overview of the Policy Inference Pipeline with Dynamics Consistency Guidance.

Table 1: Simulation results on dexterous manipulation benchmarks. We evaluate all methods on Adroit, DexArt, and three self-designed IsaacLab tasks.

Algorithm\Task	Obs.	Pretrain	Adroit 3 Tasks	DexArt 4 Tasks	Self-Designed 3 Tasks	Avg.
DP3 [11]	PC	×	77.8±2.4	60.6±0.7	80.8±2.3	71.8±1.7
ManiFlow [26]	PC	×	78.6±2.3	63.3±2.7	84.4±1.9	74.2±2.3
AFRO [28]	PC	✓	84.0±2.8	64.7±2.5	86.0±1.5	76.9±2.3
CordViP [1]	Int.PC	×	85.6±2.3	71.8±1.8	88.8±1.6	81.0±1.9
ProxiDex (Ours)	Int.PC	✓	90.3±1.4	73.7±2.2	91.1±1.7	83.9±1.8

observation encoder E_θ extracts the latent variable z_t^{obs} from the real-time interaction point cloud o_t^p and robot state s_t , and estimates the current proximity latent feature $\hat{z}_t^{prox}$ through the IDM($\cdot$). The trajectory head then outputs the interaction probability p_t^{int} , which adaptively modulates the proximity branch. The policy generates an action sequence conditioned on z_t^{obs} , $\hat{z}_t^{prox}$, and p_t^{int} , and executes it in a receding-horizon manner.

To handle occlusion or short-term perception failure, the system uses FDM($\cdot$) to predict the next latent variable and computes the dynamics deviation between the prediction and the real observation:

$$e_{t+1}^{dyn} = \|z_{t+1}^{obs} - \text{FDM}(z_t^{obs}, a_t)\|_2 \quad (5)$$

When $e_{t+1}^{dyn} \leq \tau_{dyn}$, ProxiDex regards real-time visual feedback as reliable and updates the policy state with the perceived latent observation. Otherwise, it switches to dynamics-dream mode, where action generation is maintained using latent variables autoregressively predicted by FDM($\cdot$). The system returns to regular closed-loop inference once the deviation e_{t+1}^{dyn} falls below the threshold τ_{dyn} . In deployment, FoundationPose++ runs at around 10 Hz, the proximity policy at about 28 Hz, and FDM($\cdot$) at about 20 Hz, which is sufficient for receding-horizon inference; implementation details are given in Appendix B.3.

5 Experiments

We evaluate ProxiDex in both simulation and real-world settings, focusing on two aspects: (i) whether interaction point clouds and proximity cues can improve dexterous manipulation performance; (ii) whether the learned interaction dynamics can maintain stable control under disturbances.

5.1 Simulation Experiments

Setup. We evaluate ProxiDex on 10 multi-finger tasks from Adroit [41], DexArt [42], and our customized IsaacLab environments [43]. Demonstrations for Adroit and DexArt are generated by reinforcement-learning experts, while customized tasks are collected through VR teleoperation. For each task, we collect 30 high-quality demonstrations covering representative hand-object contact patterns. Since object poses are directly available in simulation, we can accurately generate interaction point clouds o_t^p and region-level proximity features o_t^{prox} , as visualized in Appendix Fig. 21. Further details are provided in Appendices C.1 and C.2.

Baselines. We compare ProxiDex with four representative 3D point cloud policies. DP3 [11] and ManiFlow [26] use camera scene point clouds (PC) for end-to-end action prediction. AFRO [28] uses the same input with dynamics pretraining to evaluate the effect of latent dynamics. CordViP [1] adopts reconstructed interaction point clouds (Int.PC) and an arm-coordinated denoising policy.

Results. Table 1 summarizes the benchmark-level performance averaged over the evaluated tasks. ProxiDex achieves the best performance on all three task groups, reaching an overall success rate of 83.9% and outperforming DP3, ManiFlow, AFRO, and CordViP by 12.1, 9.7, 7.0, and 2.9 percentage points, respectively. On Adroit, DexArt, and our customized IsaacLab tasks, ProxiDex obtains 90.3%, 73.7%, and 91.1% success rates, exceeding CordViP by 4.7, 1.9, and 2.3 percentage points. Since CordViP also uses interaction point clouds, these gains mainly come from modeling fine-grained hand-object proximity and its temporal evolution. This is especially beneficial for contact-sensitive tasks such as Pen and Driller, where the policy must continuously adjust fingertip placement and wrist pose after contact under occlusion and object motion.

Ablation on Task Success Rate. To further analyze the contribution of each core component to task success, Fig. 4 presents task-wise ablations on six representative Adroit and DexArt tasks. ProxiDex performs best across all tasks, showing the joint benefit of interaction-centric representation and dynamics-aware policy learning. Removing proximity cues or replacing interaction point clouds with camera-observed scene point clouds causes clear drops, especially on contact-sensitive tasks such as Bucket and Pen, highlighting the importance of explicit hand-object interaction geometry under occlusion. Removing FDM, IDM, or closed-loop dynamics checking also reduces task success, indicating that latent dynamics modeling and online verification help improve execution stability during contact-rich manipulation.

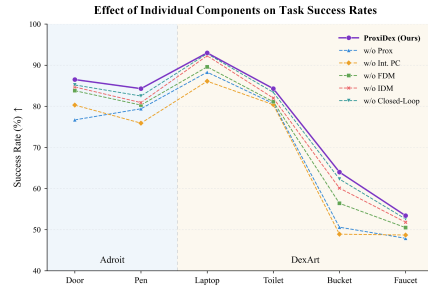


Figure 4: Ablation on Success Rate.

Ablation on Occlusion Robustness. To evaluate component contributions to occlusion robustness, we introduce a cuboid occluder into the depth-camera view for 2, 4, and 6 s, and measure recovery success on the Adroit Pen task. As shown in Fig. 5, full ProxiDex achieves 80% and 72% recovery under 2-s and 4-s occlusions. Removing FDM causes the largest drop, as future-latent prediction and dynamics takeover are no longer available. Without IDM, hand-object proximity is harder to recover from latent transitions, reducing success to 71% and 59%. Removing proximity cues or trajectory gating also weakens local interaction modeling and phase-adaptive modulation. Under 6-s occlusion, accumulated prediction errors reduce the full model to 35% recovery.

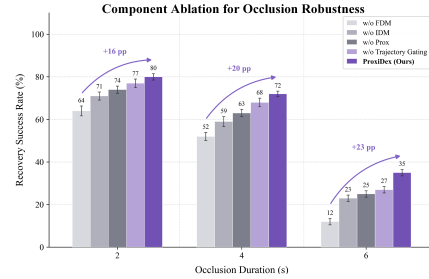


Figure 5: Ablation on Occlusion.

5.2 Real-World Experiments

Setup. The real-robot platform consists of a Realman RM75B 7-DoF robotic arm, a RealSense L515 RGB-D camera, and two dexterous hands. Specifically, a CasBot-PIL hand with 6 active DoFs is used for the standard tabletop manipulation tasks, while a 20-DoF Wuji Hand V1 is used for the more contact-rich manipulation tasks. As shown in Fig. 6 and Fig. 22, we design three tabletop tasks. For each task, we collect 50 demonstrations and evaluate the policy under in-distribution objects, unseen objects, and disturbance conditions. More details are provided in Appendix E.

Results on Tabletop Tasks. Table 2 shows that ProxiDex achieves the highest real-robot success rate, with an overall average of 72.6%. It outperforms CordViP, AFRO, ManiFlow, and DP3 by 9.7, 19.1, 22.9, and 32.6 percentage points, respectively. ProxiDex also obtains the best or tied-best

Table 2: Real-robot task success counts under in-distribution, unseen-object, perturbation, and contact-rich settings. The results show that ProxiDex achieves accurate and robust performance.

Tasks	In Distribution			Unseen Objects			Perturbation			Contact-Rich		Avg.
Algorithm	Pick	Pinch	Sweep	Pick	Pinch	Sweep	Pick	Pinch	Sweep	Twist	Flip	
DP3 [11]	14/20	10/20	12/20	18/40	9/40	16/40	14/30	13/30	14/30	2/20	2/20	40.0%
ManiFlow [26]	16/20	11/20	15/20	24/40	12/40	19/40	17/30	18/30	17/30	2/20	3/20	49.7%
AFRO [28]	17/20	11/20	16/20	25/40	15/40	24/40	17/30	17/30	16/30	3/20	5/20	53.5%
CordViP [1]	18/20	13/20	17/20	29/40	21/40	27/40	19/30	19/30	18/30	6/20	8/20	62.9%
ProxiDex-1	18/20	14/20	17/20	30/40	22/40	28/40	20/30	20/30	20/30	7/20	6/20	65.2%
ProxiDex-2	18/20	14/20	18/20	31/40	23/40	28/40	21/30	22/30	21/30	8/20	7/20	68.1%
ProxiDex	19/20	15/20	18/20	32/40	24/40	29/40	22/30	23/30	22/30	10/20	11/20	72.6%

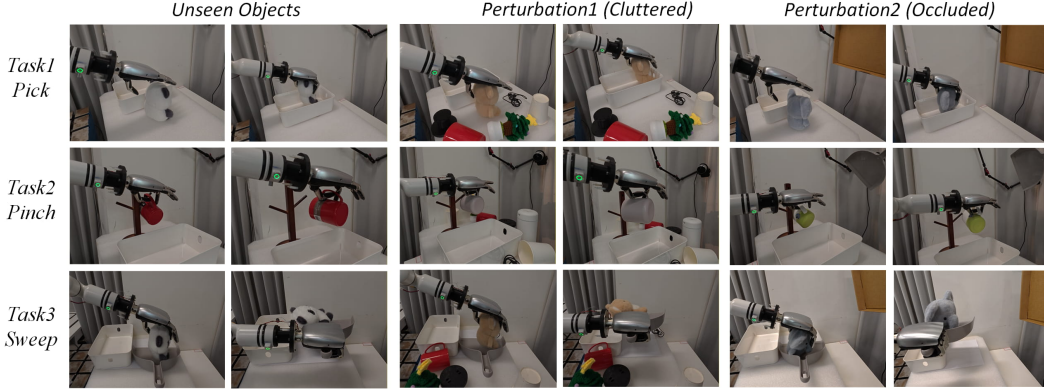


Figure 6: We introduce unseen objects, cluttered scenes, and visual occlusions into three tabletop manipulation tasks. The results show that ProxiDex can maintain stable hand-object interaction and complete the corresponding manipulation tasks in complex real-world scenarios.

result in nine task settings, demonstrating that the proposed interaction-centric representation and proximity-aware policy consistently improve real-world dexterous manipulation performance.

Generalization and robustness. Beyond the overall performance, Table 2 further shows that ProxiDex remains effective under distribution shifts and external disturbances. For unseen objects, it succeeds in 85/120 trials, compared with 77/120 for CordViP. Under perturbations, ProxiDex achieves 67/90 successful trials, while CordViP obtains 56/90. These results indicate that explicit hand-object proximity modeling improves robustness to object variations and external disturbances. In contrast, visible-scene point-cloud methods such as DP3 and ManiFlow degrade more clearly in Sweep, where hand occlusion and object pose changes make the observed scene geometry unstable.

Backbone Ablation. To further examine whether the gains only come from the policy backbone, the additional rows in Table 2 compare different policy backbones within ProxiDex. ProxiDex-1 with a U-Net backbone and ProxiDex-2 with a standard Transformer achieve 65.2% and 68.1% average success rates. The full ProxiDex further improves them by 7.4 and 4.5 percentage points, showing that the proposed backbone can emphasize hand-object proximity at critical contact moments. As shown in Fig. 7, the interaction-phase probability is

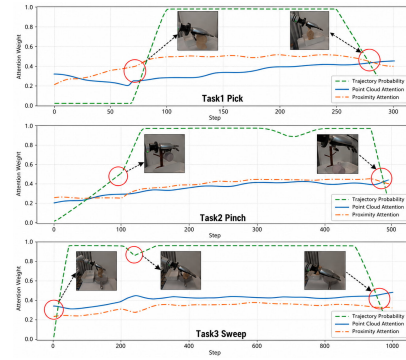


Figure 7: Attention visualization. Averaged Transformer attention weights of point-cloud and proximity tokens across three tabletop tasks.

used to adaptively adjust the attention weights between proximity and point-cloud tokens, allowing the policy to emphasize hand-object proximity at critical interaction moments.

Contact-Rich Task Analysis. We further evaluate two contact-rich tasks involving sustained finger-object interaction and object reorientation. As shown in Fig. 8, ProxiDex maintains stable interaction representations and completes complex in-hand manipulation even under short-term visual occlusion. The interaction point cloud provides local hand-object geometry, while dynamics prediction preserves interaction-state continuity when visual feedback becomes temporarily unreliable. As shown in Table 2, ProxiDex achieves the best performance on both contact-rich tasks, indicating that their combination improves stability during sustained contact and object reorientation.

Comparison with Real Tactile Sensing. To compare proximity with real tactile feedback, each fingertip of the PIL hand is equipped with a 2×4 tactile array. As shown in Fig. 9, proximity achieves competitive task performance compared with tactile sensing. Although proximity cannot replace force feedback, it captures changes in hand-object distance and spatial distribution before and around contact, providing useful geometric cues for pre-grasp adjustment and subsequent contact establishment.

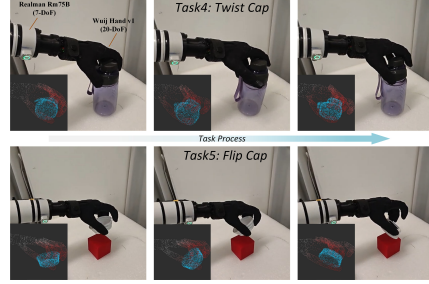


Figure 8: ProxiDex performance and interaction point-cloud visualization on contact-rich tasks.

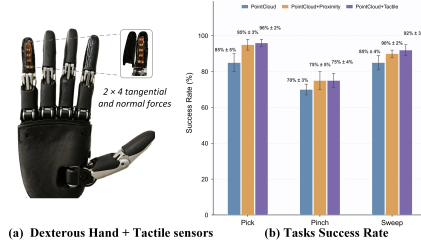


Figure 9: (a) PIL hand with tactile arrays. (b) Proximity-tactile comparison.

6 Conclusion

We presented ProxiDex, a proximity-policy learning framework for dexterous manipulation. ProxiDex constructs region-level proximity representations from reconstructed hand-object interaction point clouds and further learns action-conditioned latent interaction dynamics to guide trajectory-adaptive policy learning and closed-loop inference. By jointly modeling local hand-object geometry and its dynamic evolution, ProxiDex supports stable policy execution under short-term unreliable visual feedback, sustained contact, and complex object interactions. Simulation and real-world experiments show consistent performance across standard, unseen-object, disturbance, visual-occlusion, and contact-rich settings. Overall, these results demonstrate that ProxiDex provides a robust and physically meaningful interaction modeling framework for dexterous manipulation.

7 Limitations

Although ProxiDex shows strong stability, it still has several limitations. First, it depends on accurate object reconstruction and real-time pose tracking, which may be unreliable for small components, articulated objects, or high-precision manipulation. Second, dynamics takeover is mainly effective for short-term perception failures, while prolonged occlusion can accumulate autoregressive prediction errors and reduce control reliability. Finally, the current proximity representation remains geometry-driven and cannot measure real contact forces or precise fingertip force distributions. Future work will combine improved pose estimation, real tactile sensing, and large-scale human manipulation videos to enhance long-horizon recovery and fine-grained contact modeling.

Acknowledgments

This work was supported by the Hebei Provincial Science and Technology Plan Project under Grant No. 25241802D, the National Natural Science Foundation of China under Grant No. 62303457, and the New Generation Artificial Intelligence-National Science and Technology Major Project under Grant No. 2025ZD0122900.

References

- [1] Y. Fu, Q. Feng, N. Chen, Z. Zhou, M. Liu, M. Wu, T. Chen, S. Rong, J. Liu, H. Dong, et al. Cordvip: Correspondence-based visuomotor policy for dexterous manipulation in real-world. *arXiv preprint arXiv:2502.08449*, 2025.
- [2] Y. Bai, F. Chen, H. Sun, Y. Tong, E. Li, and Z. Zhang. Far-dex: Few-shot data augmentation and adaptive residual policy refinement for dexterous manipulation. *arXiv preprint arXiv:2603.10451*, 2026.
- [3] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023.
- [4] Z. Xu, Y. Wang, B. Abbatematteo, J. Preechayasomboon, S. Chan, N. Colonnese, and A. H. Memar. Contact-grounded policy: Dexterous visuotactile policy with generative contact grounding. *arXiv preprint arXiv:2603.05687*, 2026.
- [5] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpivot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.
- [6] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation. In *Conference on Robot Learning*, pages 2372–2395. PMLR, 2025.
- [7] B. Huang, Y. Wang, X. Yang, Y. Luo, and Y. Li. 3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing. In *Conference on Robot Learning*, pages 2557–2578. PMLR, 2025.
- [8] J. Ren, J. Zou, and G. Gu. Mc-tac: Modular camera-based tactile sensor for robot gripper. In *International Conference on Intelligent Robotics and Applications*, pages 169–179. Springer, 2023.
- [9] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- [10] 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.
- [11] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.
- [12] J. Li, T. Wu, J. Zhang, Z. Chen, H. Jin, M. Wu, Y. Shen, Y. Yang, and H. Dong. Adaptive visuo-tactile fusion with predictive force attention for dexterous manipulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3232–3239, 2025.
- [13] T. Wu, J. Li, J. Zhang, M. Wu, and H. Dong. Canonical representation and force-based pre-training of 3d tactile for dexterous visuo-tactile policy learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6786–6792, 2025.

- [14] Z. K. Weng. Hand tracking streamer: Meta quest vr app for motion capture and teleoperation, 2026. URL <https://github.com/wengmister/hand-tracking-streamer>.
- [15] H. Fang, C. Wang, Y. Wang, J. Chen, S. Xia, J. Lv, Z. He, X. Yi, Y. Guo, X. Zhan, et al. Airexo-2: Scaling up generalizable robotic imitation learning with low-cost exoskeletons. In *Conference on Robot Learning*, pages 198–220. PMLR, 2025.
- [16] H. Zhang, S. Hu, Z. Yuan, and H. Xu. Doglove: Dexterous manipulation with a low-cost open-source haptic force feedback glove. *arXiv preprint arXiv:2502.07730*, 2025.
- [17] R. Wen, J. Zhang, G. Chen, Z. Cui, M. Du, Y. Gou, Z. Han, J. Hu, L. Huang, H. Niu, et al. Dexterous teleoperation of 20-dof bytedexter hand via human motion retargeting. *arXiv preprint arXiv:2507.03227*, 2025.
- [18] K. Li, S. M. Wagh, N. Sharma, S. Bhadani, W. Chen, C. Liu, and P. Kormushev. Haptic-act: Bridging human intuition with compliant robotic manipulation via immersive vr. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1084–1091. IEEE, 2025.
- [19] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.
- [20] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. In *Conference on Robot Learning*, pages 2729–2749. PMLR, 2025.
- [21] R. Ding, Y. Qin, J. Zhu, C. Jia, S. Yang, R. Yang, X. Qi, and X. Wang. Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12248–12255. IEEE, 2025.
- [22] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.
- [23] T. Tao, M. K. Srirama, J. J. Liu, K. Shaw, and D. Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *arXiv preprint arXiv:2505.07813*, 2025.
- [24] H. Chen, T. Dong, T. Wu, L. Wang, Y. Jangir, Y. Niu, Y. Ye, H. Bharadhwaj, Z. Erickson, and J. Ichnowski. Dexterous manipulation policies from rgb human videos via 3d hand-object trajectory reconstruction. *arXiv preprint arXiv:2602.09013*, 2026.
- [25] J. Mu, S. Yang, Y. Bao, H. Bae, T. Wei, L. Xu, B. Li, H. Xu, and J. Pang. Deximit: Learning bimanual dexterous manipulation from monocular human videos. *arXiv preprint arXiv:2602.10105*, 2026.
- [26] G. Yan, J. Zhu, Y. Deng, S. Yang, R.-Z. Qiu, X. Cheng, M. Memmel, R. Krishna, A. Goyal, X. Wang, et al. Manifold: A general robot manipulation policy via consistency flow training. In *Conference on Robot Learning*, pages 2268–2293. PMLR, 2025.
- [27] Y. Fang, X. Zhang, H. Cheng, X. Zang, R. Song, and J. Zhao. Flow policy: Generalizable visuomotor policy learning via flow matching. *IEEE/ASME Transactions on Mechatronics*, 2025.
- [28] Q. Liang, B. Cai, M. Lai, S. Zhuang, T. Lin, Y. Qin, Y. Ye, J. Liang, and R. Xu. Bootstrap dynamic-aware 3d visual representation for scalable robot learning. *arXiv preprint arXiv:2512.00074*, 2025.

- [29] J. Lyu, K. Liu, X. Zhang, H. Liao, Y. Feng, W. Zhu, T. Shen, J. Chen, J. Zhang, Y. Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.
- [30] Y. Zheng, S. Gu, W. Li, Y. Zheng, Y. Zang, S. Tian, X. Li, C. Hao, C. Gao, S. Liu, et al. Omnivta: Visuo-tactile world modeling for contact-rich robotic manipulation. *arXiv preprint arXiv:2603.19201*, 2026.
- [31] C. Higuera, S. Arnaud, B. Boots, M. Mukadam, F. R. Hogan, and F. Meier. Visuo-tactile world models. *arXiv preprint arXiv:2602.06001*, 2026.
- [32] H. Yuan, W. Yi, Z. Zhang, W. Chen, Y. Mo, J. Yin, X. Li, X. Zeng, C. Wen, C. Lu, et al. Vtam: Video-tactile-action models for complex physical interaction beyond vlas. *arXiv preprint arXiv:2603.23481*, 2026.
- [33] Y. Zheng, J. Lyu, Y. Zhang, J. Chen, M. Yan, Y. Deng, X. Shi, X. Zhao, Y. Wang, Z. Zhang, et al. Emerging extrinsic dexterity in cluttered scenes via dynamics-aware policy learning. *arXiv preprint arXiv:2603.09882*, 2026.
- [34] Z. Hong, Y. Liu, H. Hou, B. Ai, J. Wang, T. Mu, Y. Qin, J. Gu, and H. Su. Learning particle-based world model from human for robot dexterous manipulation. In *3rd RSS Workshop on Dexterous Manipulation: Learning and Control with Diverse Data*, 2025.
- [35] Z. He, B. Ai, T. Mu, Y. Liu, W. Wan, J. Fu, Y. Du, H. I. Christensen, and H. Su. Scaling cross-embodiment world models for dexterous manipulation. *arXiv preprint arXiv:2511.01177*, 2025.
- [36] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.
- [37] X. Chen, F.-J. Chu, P. Gleize, K. J. Liang, A. Sax, H. Tang, W. Wang, M. Guo, T. Hardin, X. Li, et al. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.
- [38] W. Yan and J. Chu. FoundationPose++, Mar. 2025. URL <https://github.com/teal024/FoundationPose-plus-plus>.
- [39] Z. Wei, Z. Xu, J. Guo, Y. Hou, C. Gao, Z. Cai, J. Luo, and L. Shao. $\mathcal{D}(\mathcal{R}, \mathcal{O})$ Grasp: A Unified Representation of Robot and Object Interaction for Cross-Embodiment Dexterous Grasping. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4982–4988. IEEE, 2025.
- [40] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.
- [41] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.
- [42] C. Bao, H. Xu, Y. Qin, and X. Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.
- [43] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Munoz, X. Yao, R. Zurbrugg, N. Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Supplementary Material for ProxiDex

This supplementary material provides additional details on data acquisition, latent dynamics modeling, policy learning, and experimental protocols. It is organized as follows:

- Appendix A Proximity-guided data acquisition and geometric representation.
- Appendix B Training and closed-loop inference details of ProxiDex.
- Appendix C Simulation setup and qualitative analysis.
- Appendix D Additional experiments and ablation studies.
- Appendix E Real-world experiment setup and evaluation protocol.

A Proximity-Guided Data Acquisition and Geometric Representation

A.1 VR Teleoperation and Action Retargeting

Arm teleoperation. We use incremental inverse kinematics (IK) to retarget VR wrist motion to the robot end-effector. The system [14] first records the initial VR wrist position $\mathbf{x}_{\text{init}}$ and the initial robot end-effector position $\mathbf{x}_{\text{home}}$. At runtime, it computes the displacement of the current VR wrist position $\mathbf{x}_{\text{vr}}$ relative to the calibrated origin, and maps this displacement into the robot base frame using a scale vector $\mathbf{S}$ and a coordinate retargeting matrix $\mathbf{R}_{\text{map}}$:

$$\mathbf{x}_{\text{target}} = \mathbf{x}_{\text{home}} + \mathbf{R}_{\text{map}} (\mathbf{S} \circ (\mathbf{x}_{\text{vr}} - \mathbf{x}_{\text{init}})), \quad (6)$$

where $\mathbf{S}$ is an anisotropic scaling vector, $\mathbf{R}_{\text{map}}$ aligns the VR coordinate frame with the robot base frame, and $\circ$ denotes element-wise multiplication. The target pose is solved by numerical inverse kinematics under joint-limit and redundancy constraints, and the resulting joint commands are sent to the Realman RM75B arm through the low-level controller.

For orientation control, the system uses quaternion increments to synchronize the VR wrist orientation and the robot end-effector orientation. Let q_{vr} be the current VR wrist orientation, q_{init} the initial calibrated orientation, and q_{home} the initial robot end-effector orientation. The target orientation is

$$\Delta q = q_{\text{vr}} \otimes q_{\text{init}}^{-1}, \quad q_{\text{target}} = q_{\text{home}} \otimes \Delta q. \quad (7)$$

The Cartesian target is passed to a numerical IK solver in PyBullet, which computes joint commands under 7-DoF redundancy and joint-limit constraints. The resulting commands are synchronized to the RealMan RM75B arm through CAN-FD, enabling low-latency spatial following.

Dexterous-hand teleoperation. The VR detector provides 21 hand keypoints $\mathbf{u}_i$ [14]. We normalize them by removing the wrist translation and applying a scale factor, i.e., $\tilde{\mathbf{u}}_i = \eta(\mathbf{u}_i - \mathbf{u}_0)$, followed by a handedness correction when needed. A local wrist frame is estimated from the wrist and finger-root keypoints, and the normalized keypoints are reprojected into a canonical hand coordinate system. This step reduces differences among operators and provides a consistent geometric input for retargeting.

The retargeting problem converts human-hand topology into robot joint commands. We construct target vector flows $\hat{\mathbf{v}}_k$ that describe the relative geometry of the human fingers and solve for the robot joint configuration q_t^* by minimizing

$$q_t^* = \arg \min_q \sum_k w_k \rho_\delta (\|r_k(q) - \hat{\mathbf{v}}_k\|_2^2) + \lambda \|q - q_{t-1}\|_2^2. \quad (8)$$

Here, ρ_δ is the Huber loss for robustness to keypoint noise, and the second term regularizes temporal smoothness by penalizing abrupt joint changes. We further introduce a DexPilot-style grasp prior projection [5] and update the projection state with threshold hysteresis, improving fingertip closure while reducing hand jitter.

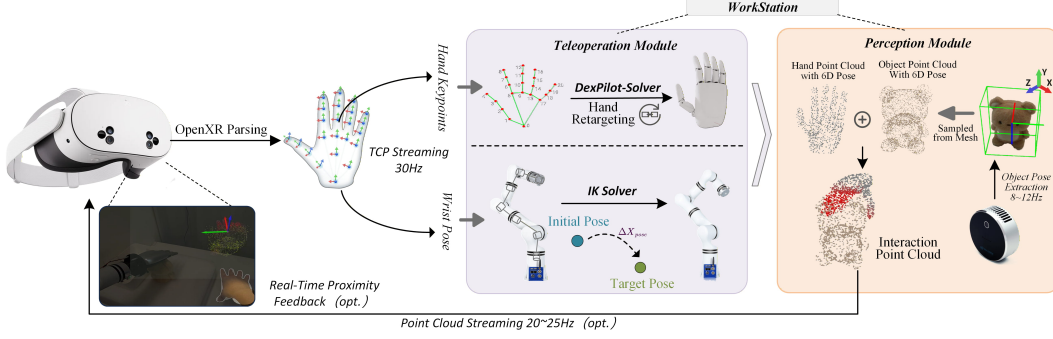


Figure 10: VR teleoperation pipeline in ProxiDex, where OpenXR hand keypoints and wrist poses are streamed to a workstation for hand–arm control, and the reconstructed hand–object interaction point cloud is sent back to the headset as optional proximity feedback for immersive teleoperation.

The teleoperation loop is sufficiently responsive for real-time operation. TCP hand-joint commands are transmitted at approximately 30 Hz, object-pose detection runs at around 10 Hz, and the interaction point-cloud stream is updated at about 23 Hz. These rates allow the operator to receive timely visual and proximity feedback during manipulation, as summarized in Fig. 10. In particular, when direct visual observation is degraded by hand-object occlusions, the proximity feedback provides an augmented-reality style cue that helps the operator perceive the hand-object distance and adjust the manipulation process more intuitively. With the teleoperation stream defined, the next step is to convert the reconstructed object geometry into a metric-scale representation for distance-based proximity computation.

A.2 Object Reconstruction and Metric-Scale Calibration

We further detail how ProxiDex reconstructs occlusion-free hand-object interaction point clouds from demonstration data. We first use Grounded-SAM[36] to segment the target object from the initial RGB observation, and then reconstruct the target mesh $\mathcal{M}_o$ using SAM3D[37]. Monocular reconstruction can complete invisible regions and provide a full geometric prior, but its output usually lacks reliable metric scale. Therefore, before using the mesh for interaction point-cloud synthesis and proximity computation, we perform scale calibration through the following procedure.

For each object, the system captures one aligned RealSense RGB-D observation and estimates a scale factor κ from the depth map. Given an RGB image, a metric depth map, and camera intrinsics, Grounded-SAM first produces the target object mask. Valid depth pixels inside the mask are then back-projected into the camera frame:

$$p(u, v) = \left(\frac{(u - c_x)d}{f_x}, \frac{(v - c_y)d}{f_y}, d \right), \quad (9)$$

where (u, v) denotes pixel coordinates, d is the depth value, and (c_x, c_y) and (f_x, f_y) are the principal point and focal lengths. After removing statistical outliers from the back-projected object point cloud, we compute its oriented bounding-box size e_{rgbd} .

We also load the vertices of the OBJ mesh reconstructed by SAM3D and compute its oriented bounding-box size e_{obj} . To avoid axis-order effects in scale estimation, both size vectors are sorted before comparison:

$$\mathbf{r} = \frac{\text{sort}(e_{\text{rgbd}})}{\text{sort}(e_{\text{obj}})}. \quad (10)$$

The final uniform scale coefficient is $\kappa = \text{median}(\mathbf{r})$, which is applied to the mesh vertices. This calibration preserves the shape-completion advantage of SAM3D while grounding subsequent distance calculations in metric geometry. Using the median of the sorted axis ratios also reduces sensitivity to a single noisy bounding-box dimension, which can appear when the RGB-D mask contains small boundary artifacts. The calibrated mesh therefore provides a stable object surface for the region-level proximity computation described next.

Table 3: Input data specifications across simulation and real-world environments.

Environment	Input Setting	Point-Cloud Input	Auxiliary Input	Action Dim.	Horizon / Notes
<i>Simulation Environments</i>					
Adroit	Scene Point Cloud	[512, 3] point_cloud	[24] low_dim state	28	n_{obs}
Adroit	Interaction Point Cloud	[3 × 512, 3] point_cloud	[24] low_dim state	28	$N = 3$, including hand and object
DexArt	Scene Point Cloud	[1024, 3] point_cloud	[32] low_dim state	22	n_{obs}
DexArt	Interaction Point Cloud	[2 × 1024, 3] point_cloud	[32] low_dim state	22	$N = 2$, including hand and object
IsaacLab	Scene Point Cloud	[1024, 3] point_cloud	[18] low_dim state	18	n_{obs}
IsaacLab	Interaction Point Cloud	[N × 1024, 3] point_cloud	[18] low_dim state	18	$N = 2$ for Box and $N = 3$ for other tasks
<i>Real-World Environment</i>					
Real World	Scene Point Cloud	[512, 3] point_cloud	[13] low_dim state	13	n_{obs}
Real World	Interaction Point Cloud	[N × 512, 3] point_cloud	[13] low_dim state	13	$N = 3$ for Sweep and $N = 2$ for other tasks

A.3 Geometric Proximity Representation

Given the metrically calibrated object mesh, the system samples points from the surfaces of object mesh and hand URDF links, and transforms them into the robot base frame using the object 6D pose tracked by FoundationPose++ [38] and robot forward kinematics. This yields the object point cloud P_t^o and the hand point cloud P_t^h . The interaction point cloud at frame t is defined as $o_t^p = P_t^o \cup P_t^h$. Compared with a raw scene point cloud, this representation explicitly preserves the relative geometry between the hand and object while reducing the influence of background points and self-occlusion.

Proximity is computed from geometric distances between the hand and object point clouds, without assuming any specific tactile sensor. For each hand sample $p_i^h \in P_t^h$, we first compute its nearest-neighbor distance to the object point cloud:

$$d_i^{\text{nn}} = \min_{p_j^o \in P_t^o} \|p_i^h - p_j^o\|_2 \quad (11)$$

In practice, small geometric interpenetrations may occur due to pose-tracking noise, calibration errors, or simplified hand/object meshes. Since the proximity representation is intended to describe near-field interaction rather than penetration depth, we treat such cases as saturated contact responses. Specifically, using the signed distance field $\phi_o(\cdot)$ of the object mesh, where negative values indicate points inside the object, the effective distance is clipped as

$$d_i = \begin{cases} d_i^{\text{nn}}, & \phi_o(p_i^h) \geq 0, \\ 0, & \phi_o(p_i^h) < 0. \end{cases} \quad (12)$$

This prevents occasional interpenetration artifacts from producing unstable proximity values while preserving consistent contact cues for policy learning. A Gaussian kernel then maps the effective distance to a normalized proximity response:

$$\eta_i = \exp\left(-\frac{1}{2} \left(\frac{d_i}{\sigma}\right)^2\right), \quad \eta_i \in [0, 1], \quad (13)$$

where σ controls the decay sensitivity. For VR visualization, η_i can be mapped to the hand point color $c_i = c_{\text{base}} + \eta_i(c_{\text{hot}} - c_{\text{base}})$, providing proximity guidance related to object distance. For policy training, we record the full o_t^p and proximity observation to maintain input consistency.

To obtain structured learning inputs, we group hand samples according to physical hand regions S_k , such as the palm, finger roots, and fingertips. Let $I_k = \{i \mid p_i^h \in S_k\}$ be the set of sample indices belonging to region S_k . The region-level proximity is

$$\gamma_t^k = \frac{1}{|I_k|} \sum_{i \in I_k} \eta_i. \quad (14)$$

The proximity observation at frame t is therefore

$$o_t^{\text{prox}} = [\gamma_t^1, \dots, \gamma_t^K]. \quad (15)$$

We further derive the trajectory-stage label from the palm response:

$$o_t^{\text{traj}} = \mathbb{I}(\gamma_t^{\text{palm}} > \tau_{\text{palm}}), \quad (16)$$

where $o_t^{\text{traj}} = 0$ denotes the approach stage and $o_t^{\text{traj}} = 1$ denotes the contact-interaction stage. Each demonstration frame is finally stored as $(o_t^p, o_t^{\text{prox}}, o_t^{\text{traj}}, s_t, a_t)$, consistent with the notation in the main paper. The concrete input dimensions of scene point clouds, interaction point clouds, auxiliary states, and actions across simulation and real-world environments are summarized in Table 3. In this representation, o_t^p keeps the complete hand-object geometry, o_t^{prox} compresses near-field distances into physically meaningful hand-region responses, and o_t^{traj} provides a coarse phase indicator. This separation is used in the following training stage, where the latent observation is explicitly aligned with proximity dynamics rather than only with visible geometry.

B Training and Inference Details of ProxiDex

ProxiDex is trained in two stages. The first stage learns a proximity-aware dynamics representation from interaction point clouds, robot states, and region-level proximity cues. The second stage freezes this representation and trains a Transformer-based diffusion policy, in which the predicted trajectory-stage probability adaptively modulates the proximity branch. This design allows the policy to emphasize global geometry during approach and exploit local proximity cues during dexterous manipulation. All training and inference are conducted on an Intel i9-14700K CPU and an NVIDIA RTX 4090D GPU.

B.1 Latent Interaction Dynamics Architecture

Given the interaction point cloud o_t^p and robot state s_t , the observation encoder E_θ extracts geometric features and fuses them with the state embedding to obtain the observation latent z_t^{obs} . This latent is shared by the forward dynamics model (FDM), inverse differential module (IDM), and trajectory head in Stage 1, and is subsequently used as the global condition for policy generation in Stage 2.

Before learning the interaction dynamics, we independently pretrain a proximity variational autoencoder (VAE) for 100 epochs to compress the region-level proximity observation o_t^{prox} into a compact latent representation. The encoder parameterizes a Gaussian posterior, while the decoder reconstructs the original proximity observation. In our implementation, the proximity input and latent dimensions are 124 and 16, respectively. The VAE is optimized using a reconstruction loss and a KL-divergence regularizer:

$$\mathcal{L}_{\text{vae}} = \lambda_{\text{rec}} \|\hat{o}_t^{\text{prox}} - o_t^{\text{prox}}\|_2^2 + \lambda_{\text{KL}} D_{\text{KL}}(q(z_t^{\text{prox}} | o_t^{\text{prox}}) \| \mathcal{N}(0, I)), \quad (17)$$

where $\lambda_{\text{rec}} = 1.0$ and $\lambda_{\text{KL}} = 0.01$. After pretraining, the VAE is frozen, and its posterior mean $\mu_{\text{prox}}(o_t^{\text{prox}})$ is used as a deterministic proximity target for interaction dynamics learning.

The forward dynamics model $\text{FDM}(\cdot)$ is implemented as a DiT-style conditional denoising model that predicts future observation features after action execution. Specifically, the future observation latent is perturbed and then denoised conditioned on the current latent and action sequence:

$$\begin{aligned} \hat{z}_{t+h}^{\text{obs},(\ell)} &= \sqrt{\bar{\alpha}_\ell} z_{t+h}^{\text{obs}} + \sqrt{1 - \bar{\alpha}_\ell} \epsilon, \\ \hat{z}_{t+h}^{\text{obs}} &= f_\theta \left(\hat{z}_{t+h}^{\text{obs},(\ell)}, \ell \mid z_t^{\text{obs}}, a_{t:t+h-1} \right), \end{aligned} \quad (18)$$

where ℓ denotes the diffusion step and ϵ is Gaussian noise. This forward-prediction objective encourages z_t^{obs} to encode not only object geometry and hand configuration, but also short-horizon interaction dynamics associated with approach, grasping, and contact transfer.

To connect the predicted state transition with local geometric proximity, ProxiDex introduces the inverse differential module $\text{IDM}(\cdot)$, implemented as an MLP with layer normalization and nonlinear activations. For one-step prediction, IDM maps the change in observation latent to the next-step proximity latent, which is aligned with the deterministic target provided by the frozen VAE:

$$\begin{aligned} \hat{z}_{t+1}^{\text{prox}} &= \text{IDM}(\hat{z}_{t+1}^{\text{obs}} - z_t^{\text{obs}}), \\ \mathcal{L}_{\text{inv}} &= \|\hat{z}_{t+1}^{\text{prox}} - \mu_{\text{prox}}(o_{t+1}^{\text{prox}})\|_2^2. \end{aligned} \quad (19)$$

Table 4: Key training and model hyperparameters for the Interaction Dynamics Model.

Parameter	Symbol	Value
<i>Policy and Diffusion Settings</i>		
Policy Horizon	T	16
Observation Horizon	T_o	4
Action Horizon	T_a	8
Diffusion Training Timesteps	K	100
<i>Dynamics and Proximity Modeling</i>		
Point-Cloud Feature Dimension	d_p	128
Future Prediction Steps	H_{dyn}	2
Proximity Latent Dimension	d_{prox}	16
Action Embedding Dimension	d_a	16
FDM Hidden Dimension	d_{fdm}	256
Proximity VAE Pretraining Epochs	–	100
Action Dropout / Negative Sampling Probability	–	0.1 / 0.1
Stage-1 Loss Weights	–	$\lambda_v = \lambda_f = \lambda_c = \lambda_t = 1.0$
Stage-2 Loss Weights	–	$\lambda_{\text{diff}} = 1.0, \lambda_{\text{gate}} = 0.001$
Reconstruction / KL Weights	–	$\lambda_{\text{rec}} = 1.0, \lambda_{\text{KL}} = 0.01$
<i>Optimization Settings</i>		
Optimizer	–	AdamW
Training Epochs	–	301
Batch Size	–	64
Learning Rate	–	5×10^{-5}
Weight Decay	–	1×10^{-6}
Learning Rate Schedule	–	Cosine

This alignment encourages the observation-latent transition to preserve action-induced changes in hand–object proximity and enables IDM to infer implicit proximity information from visual and robot-state dynamics.

In parallel, the trajectory-stage head predicts the interaction probability p_t^{int} from z_t^{obs} and is supervised by o_t^{traj} , as defined in Appendix A.3. Together, forward prediction, proximity-latent alignment, and trajectory-stage classification produce a physically meaningful interaction representation for downstream control.

As summarized in Table 4, the point-cloud feature and proximity latent dimensions are 128 and 16, respectively. The dynamics branch uses a hidden dimension of 256 and a short prediction horizon $H_{\text{dyn}} = 2$, allowing it to capture immediate action-induced interaction changes while limiting long-horizon prediction errors. Stage-1 training spans 301 epochs in total. We first pretrain the proximity VAE for 100 epochs. The VAE is then frozen, and the observation encoder, FDM, IDM, and trajectory head are trained for the remaining 201 epochs. The forward-dynamics, proximity-alignment, and trajectory-stage losses are assigned equal weights.

B.2 Trajectory-Adaptive Proximity Policy Learning

The Stage-2 policy backbone is a Transformer-based diffusion model. The policy is trained using VAE-derived proximity latents, whereas IDM-estimated proximity latents are used during online deployment. It takes a noised action sequence as input and denoises it under two types of conditions: the observation latent z_t^{obs} , which provides global information such as target geometry, object pose, and hand configuration; and the proximity latent z_t^{prox} , which provides local distance cues between hand regions and the object surface. To avoid overusing local feedback before contact, we gate the proximity branch according to the interaction probability p_t^{int} predicted by the trajectory-stage head in Stage 1:

$$z_t^{\text{prox}} = g(p_t^{\text{int}})z_t^{\text{prox}}, \quad g(p) = 1 - \alpha + \alpha p, \quad \alpha \in [0, 1]. \quad (20)$$

Table 5: Key training and model hyperparameters for the Proximity Policy.

Parameter	Symbol	Value
<i>Policy and Diffusion Settings</i>		
Policy Horizon	T	16
Observation Horizon	T_o	4
Action Horizon	T_a	8
Diffusion Training Timesteps	K	100
Inference Denoising Steps	–	10
<i>Proximity Policy Modeling</i>		
Point-Cloud Feature Dimension	d_p	128
Proximity Latent Dimension	d_{prox}	16
Interaction Logit Dimension	C_{int}	1
Gate Warm-up Steps	N_{warm}	2,000
Gate Temperature Range	τ	$2.0 \rightarrow 0.7$
Temperature Annealing Steps	N_τ	50,000
Gate Ramp-up Steps	N_α	20,000
Stage-2 Loss Weights	–	$\lambda_{\text{diff}} = 1.0, \lambda_{\text{gate}} = 10^{-3}$
<i>Transformer Backbone</i>		
Transformer Layers	L	8
Attention Heads	H	8
Embedding Dimension	d_{emb}	256
<i>Optimization Settings</i>		
Optimizer	–	AdamW
Training Epochs	–	301
Batch Size	–	128
Learning Rate	–	1×10^{-4}
Weight Decay	–	1×10^{-6}

During the first 2,000 training steps, we set $p_t^{\text{int}} = 0.5$ to stabilize policy optimization. After this warm-up period, the interaction probability is computed as $p_t^{\text{int}} = \text{sigmoid}(c_t/\tau)$, where c_t is the predicted interaction logit. The temperature τ is annealed from 2.0 to 0.7 over 50,000 steps, while α is linearly increased from 0 to 1 over 20,000 steps after warm-up. This schedule gradually introduces proximity modulation without abruptly changing the policy condition.

When p_t^{int} is low, the policy relies more on z_t^{obs} for global approach. As the interaction probability increases, the proximity branch is strengthened, enabling the policy to refine fingertip closure, contact maintenance, and release timing. To prevent the gate probability from prematurely saturating, we introduce a negative binary-entropy regularizer:

$$\mathcal{L}_{\text{gate}} = \mathbb{E} [p_t^{\text{int}} \log(p_t^{\text{int}} + \epsilon) + (1 - p_t^{\text{int}}) \log(1 - p_t^{\text{int}} + \epsilon)], \quad (21)$$

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{diff}} + \lambda_{\text{gate}} \mathcal{L}_{\text{gate}}.$$

where $\mathcal{L}_{\text{diff}}$ denotes the diffusion denoising loss and ϵ is a small constant for numerical stability.

For temporal policy generation, the proximity condition is projected into the policy embedding space and shared across the predicted action sequence, while the observation latent remains available as stable global context. This conditioning scheme preserves global planning ability while allowing the interaction probability to adaptively regulate the contribution of local proximity information.

The detailed hyperparameters for proximity-adaptive policy learning are summarized in Table 5. Overall, Stage 2 follows the same temporal window configuration as Stage 1 while using fewer denoising steps during inference for efficient online deployment. Since the Stage-1 representation is frozen, optimization mainly focuses on the diffusion policy and its trajectory-adaptive proximity conditioning.

Algorithm 1 Dynamics-Consistency Guided Policy Inference

Input: Multimodal observations $\{(P_t, S_t)\}_{t=0}^{T+1}$, encoder E , forward dynamics model FDM, diffusion policy π_θ , dynamics threshold τ_{dyn} , and scaling factor α

Output: Executed action sequence $\{a_t\}_{t=1}^T$

```

1:  $z_0^{\text{obs}} \leftarrow E(P_0, S_0)$ 
2:  $z_0^{\text{in}} \leftarrow z_0^{\text{obs}}$ 
3: Mode  $\leftarrow$  Normal
4: for  $t = 1$  to  $T$  do
  /* Observation encoding */
5:  $z_t^{\text{obs}} \leftarrow E(P_t, S_t)$ 
6: if Mode = Normal then
7:    $z_t^{\text{in}} \leftarrow z_t^{\text{obs}}$ 
8: else
9:    $z_t^{\text{in}} \leftarrow \hat{z}_t^{\text{obs}}$ 
10: end if
11:  $\Delta z_t \leftarrow z_t^{\text{in}} - z_{t-1}^{\text{in}}$ 
  /* Current proximity inference and adaptive gating */
12:  $\hat{z}_t^{\text{prox}} \leftarrow \text{IDM}(\Delta z_t)$ 
13:  $p_t^{\text{int}} \leftarrow h(z_t^{\text{in}})$ 
14:  $\bar{z}_t^{\text{prox}} \leftarrow (1 - \alpha + \alpha p_t^{\text{int}}) \hat{z}_t^{\text{prox}}$ 
  /* Action generation and latent dynamics prediction */
15:  $a_t \leftarrow \pi_\theta(z_t^{\text{in}}, \bar{z}_t^{\text{prox}})$ 
16: Execute  $a_t$  via the robot controller
17:  $\hat{z}_{t+1}^{\text{obs}} \leftarrow \text{FDM}(z_t^{\text{in}}, a_t)$ 
  /* Dynamics monitoring and mode switching */
18: Receive the next observation  $(P_{t+1}, S_{t+1})$ 
19:  $z_{t+1}^{\text{obs}} \leftarrow E(P_{t+1}, S_{t+1})$ 
20:  $e_{t+1}^{\text{dyn}} \leftarrow \|z_{t+1}^{\text{obs}} - \hat{z}_{t+1}^{\text{obs}}\|_2$ 
21: if  $e_{t+1}^{\text{dyn}} \leq \tau_{\text{dyn}}$  then
22:   Mode  $\leftarrow$  Normal
23:    $z_{t+1}^{\text{in}} \leftarrow z_{t+1}^{\text{obs}}$ 
24: else
25:   Mode  $\leftarrow$  Dream
26:    $z_{t+1}^{\text{in}} \leftarrow \hat{z}_{t+1}^{\text{obs}}$ 
27: end if
28: end for
```

B.3 Dynamics-Consistency Guided Policy Inference

During online inference, ProxiDex places policy generation and FDM prediction in the same loop. The system encodes the real-time interaction point cloud o_t^p and robot state s_t into z_t^{obs} , estimates $\hat{z}_t^{\text{prox}}$ through IDM, and obtains p_t^{int} from the trajectory head. The diffusion policy then generates an action sequence and executes actions in a receding-horizon manner.

After action execution, FDM predicts the next latent state and compares it with the encoded real observation:

$$e_{t+1}^{\text{dyn}} = \|z_{t+1}^{\text{obs}} - \text{FDM}(z_t^{\text{obs}}, a_t)\|_2. \quad (22)$$

If $e_{t+1}^{\text{dyn}} \leq \tau_{\text{dyn}}$, the system continues closed-loop control with real-time perception. If the deviation exceeds the threshold, the system treats the current visual feedback as potentially affected by occlusion, pose drift, or short-term tracking failure, and temporarily switches to dynamics takeover, using autoregressive FDM latents to maintain action generation. Once the deviation returns below the threshold, the system switches back to real observations and re-aligns the state.

This mechanism makes FDM both a training-time dynamics supervisor and an inference-time physical consistency checker. In deployment, FoundationPose++ provides object-pose updates at around 10 Hz, the Proximity Policy runs at about 28 Hz, and the learned world model runs at about 20 Hz. These runtime rates are sufficient for real-time closed-loop inference, enabling dynamics takeover to maintain action generation from predicted latents when visual pose tracking becomes temporarily unreliable. Algorithm 1 summarizes the procedure.



Figure 11: Visualization of the simulation tasks. We evaluate ProxiDex on ten dexterous manipulation tasks, including four DexArt tasks, three Adroit tasks, and three custom tasks built in IsaacLab.

C Simulation Experiments

C.1 Simulation Setup and Task Description

We evaluate ProxiDex on Adroit, DexArt, and three self-designed IsaacLab tasks, as illustrated in Fig. 11. Adroit covers classic dexterous manipulation tasks requiring fine finger coordination, such as Hammer, Door, and Pen, while DexArt provides articulated-object interactions closer to daily manipulation, including Laptop, Faucet, Toilet, and Bucket. Since the simulator directly provides object poses and robot joint states, we can reliably synthesize occlusion-free interaction point clouds o_t^p and compute region-level proximity observations o_t^{prox} . The interaction point clouds are visualized in Fig. 21, showing that they preserve active hand-object regions throughout manipulation.

The self-designed tasks are defined as follows:

- **Box.** The robot grasps a box-like object with multiple fingers and places it into a basket. This task evaluates grasp stability and collision-free transport around the basket rim.
- **Driller.** The robot side-grasps a drill and moves the drill tip above a target hole. This task requires stable tool orientation and accurate alignment after long-range motion.
- **Banana.** The robot pinches a banana-like object using the thumb and index finger and places it into a bowl. This task is sensitive to fingertip closure timing, contact location, and holding stability.

Together, these tasks cover rigid-object grasping, articulated-object manipulation, tool alignment, and fine fingertip pinching. The implementation details below keep the data split and evaluation protocol consistent across all methods, so that the comparisons isolate the effect of interaction point clouds, proximity features, and dynamics modeling.

C.2 Simulation Implementation Details

For Adroit and DexArt, demonstrations are generated by reinforcement-learning experts, while the self-designed IsaacLab demonstrations are collected through VR teleoperation. All methods use the same demonstration split, action horizon, and evaluation protocol. We report the mean success rate and standard deviation across repeated evaluations. Fig. 11 visualizes the ten simulation tasks before the quantitative comparison, and Table 6 provides the corresponding task-level results.

The detailed task-level results in Table 6 further support the overall advantage of ProxiDex. ProxiDex achieves the best ten-task average of 83.9%, outperforming DP3, ManiFlow, AFRO, and CordViP by 12.1, 9.7, 7.0, and 2.9 percentage points, respectively. The improvement is consistent across Adroit, DexArt, and the self-designed tasks, indicating that proximity-aware dynamics benefit both standard dexterous benchmarks and customized tabletop manipulation scenarios.

Table 6: Simulation results on dexterous manipulation benchmarks. We evaluate all methods on Adroit, DexArt, and three self-designed IsaacLab tasks.

Algorithm\Tasks	Adroit			DexArt				Self-Designed			Avg.
	Hammer	Door	Pen	Laptop	Faucet	Toilet	Bucket	Box	Driller	Banana	
DP3	100.0±0.0	76.7±4.7	56.7±2.6	89.7±0.9	41.7±0.5	79.7±0.9	31.3±0.5	90.3±3.2	74.3±2.6	77.8±1.2	71.8±1.7
ManiFlow	100.0±0.0	80.3±1.2	55.5±5.8	93.0±1.6	45.0±3.6	79.9±3.3	35.3±2.1	94.7±2.1	78.6±1.4	79.8±2.3	74.2±2.3
AFRO	96.0±2.8	83.8±3.4	72.3±2.1	86.3±2.5	42.3±1.8	82.2±3.4	48.0±2.3	97.2±1.3	79.4±1.5	81.4±1.8	76.9±2.3
CordViP	96.0±2.1	84.7±1.7	76.2±3.2	92.6±2.1	56.5±1.3	81.0±2.2	57.0±1.6	96.3±2.0	85.3±1.1	84.7±1.6	81.0±1.9
ProxiDex (Ours)	100.0±0.0	86.5±1.9	84.3±2.4	93.0±1.2	53.4±2.1	84.3±2.3	64.0±3.2	96.7±1.6	89.2±2.0	87.3±1.5	83.9±1.8



Figure 12: Visualization of hand-object interactions on the self-designed tasks, compared with typical failure cases from existing baselines.

A closer look at the task results shows that the gains are more evident in contact-rich tasks requiring sustained fingertip adjustment, such as Pen, Bucket, Driller, and Banana. These tasks involve continuous hand-object proximity regulation after initial contact, where the proposed proximity representation and adaptive policy modulation provide more informative feedback than geometry-only baselines. Although CordViP performs slightly better on Faucet, ProxiDex achieves the best aggregate performance by improving tasks with more complex contact-state transitions.

Fig. 12 provides qualitative evidence complementary to the quantitative results in Table 6, comparing representative executions on the self-designed tasks with typical failure cases of the baselines. Specifically, in the Box task, baseline methods often rely on a small distal contact region and can lose grasp stability during transport, whereas ProxiDex adjusts finger closure after approaching the target. In Driller, visible-point-cloud methods struggle to maintain the relative relation between the drill and the hole under occlusion; ProxiDex improves local adjustment near the hole through proximity-difference constraints. In Banana, ProxiDex forms more suitable fingertip contact regions, showing the value of region-level proximity for fine pinching.

D Additional Experiments and Analysis

D.1 Teleoperation System Comparison

To evaluate the benefit of immersive proximity feedback, we invited 20 participants to experience the three teleoperation systems shown in Fig. 13(c). After 10 minutes of training, participants collected 30 demonstrations under each setting. As shown in Fig. 13(a)(b), most participants found proximity feedback helpful, and 75% preferred ProxiDex. ProxiDex also reduces the average task completion time to 26.6 ± 5 s, outperforming the other two systems. This advantage mainly comes from more direct interaction feedback: compared with Vanilla VR, which is susceptible to hand-object occlusion, ProxiDex visualizes hand-object distances and spatial relations through color-coded proximity point clouds, enabling timely adjustment of grasp position and finger closure and thereby improving teleoperation experience and demonstration-collection efficiency.

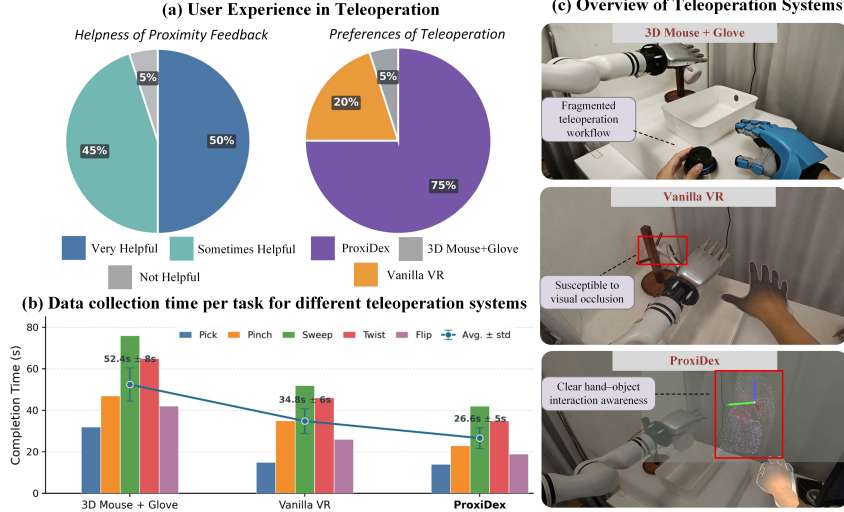


Figure 13: User study and teleoperation-system comparison. (a) User experience and system preference. (b) Task completion time. (c) Overview of the three teleoperation systems.

Table 7: Ablation study of key components on representative dexterous manipulation tasks.

Components	Door	Pen	Laptop	Faucet	Toilet	Bucket	Avg.
w/o Prox	76.7 $\pm$ 2.3	79.4 $\pm$ 1.6	88.3 $\pm$ 1.5	47.9 $\pm$ 1.5	80.7 $\pm$ 1.6	50.6 $\pm$ 1.3	70.6 $\pm$ 1.6
w/o Int. PC	80.3 $\pm$ 1.2	75.9 $\pm$ 2.4	86.1 $\pm$ 1.4	48.7 $\pm$ 1.4	80.4 $\pm$ 1.5	48.9 $\pm$ 2.0	70.1 $\pm$ 1.7
w/o FDM	83.8 $\pm$ 2.2	80.3 $\pm$ 1.7	89.6 $\pm$ 1.5	50.5 $\pm$ 1.7	81.1 $\pm$ 1.7	56.4 $\pm$ 1.8	73.6 $\pm$ 1.8
w/o IDM	84.7 $\pm$ 1.7	80.9 $\pm$ 2.1	92.3 $\pm$ 1.7	51.8 $\pm$ 1.3	82.0 $\pm$ 1.8	60.1 $\pm$ 1.5	75.3 $\pm$ 1.7
w/o Closed-Loop	85.2 $\pm$ 1.2	82.5 $\pm$ 1.2	92.8 $\pm$ 1.3	52.6 $\pm$ 1.1	83.4 $\pm$ 2.0	62.3 $\pm$ 1.6	76.5 $\pm$ 1.4
ProxiDex (Ours)	86.5$\pm$1.9	84.3$\pm$2.4	93.0$\pm$1.2	53.4$\pm$2.1	84.3$\pm$2.3	64.0$\pm$3.2	77.6$\pm$2.2

D.2 Ablation Study

Ablation on core components. Table 7 reports the task-level numerical results corresponding to the ablation summary in Fig. 4. The largest average drops occur when geometric interaction inputs are removed: removing interaction point clouds decreases the success rate from 77.6% to 70.1%, while removing proximity cues decreases it to 70.6%, with drops of 7.5 and 7.0 points, respectively. The degradation is especially clear on contact-sensitive tasks: Bucket drops by 15.1 and 13.4 points under these two removals, while Pen drops by 8.4 points without interaction point clouds. These results indicate that complete hand-object geometry and region-level proximity provide complementary cues for contact-rich manipulation.

From the dynamics perspective, removing FDM reduces the average success rate to 73.6%, corresponding to a 4.0-point drop, while removing IDM reduces it to 75.3%, a 2.3-point drop. FDM contributes more strongly to tasks that require predicting near-future contact consequences, whereas IDM improves sensitivity to local proximity changes caused by actions. Removing closed-loop feedback yields 76.5%, a smaller but consistent 1.1-point drop. This pattern suggests that most of the performance gain comes from representation and proximity-aware policy learning, while inference-time dynamics checking mainly stabilizes execution when perception is temporarily unreliable.

The component-level ablation motivates a separate analysis of the Stage-2 policy architecture: after the interaction representation is fixed, the remaining question is whether the policy backbone can use proximity information at the correct manipulation phase.

Ablation on the policy backbone. Complementing the real-robot backbone ablation in Table 2, Fig. 14 further evaluates the Stage-2 policy backbone in simulation while keeping Stage-1 pre-training unchanged. Replacing the proposed causal Transformer with a U-Net reduces the average success rate by about 3.1 percentage points, since temporal convolutions are less flexible in fusing

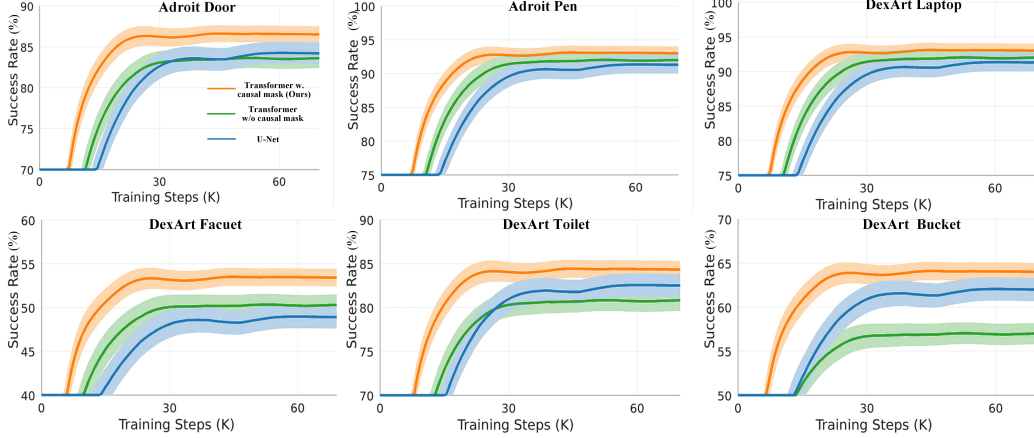


Figure 14: Policy-backbone ablation study on six Adroit and DexArt tasks. The proposed causal Transformer with adaptive weight converges faster and achieves higher final success rates.

Table 8: Ablation study on the latent dynamics prediction horizon H_{dyn} .

H_{dyn}	Door	Pen	Laptop	Faucet	Toilet	Bucket	Avg.
1	86.8$\pm$1.7	84.0 $\pm$ 2.2	92.6 $\pm$ 1.5	53.3 $\pm$ 1.9	80.7 $\pm$ 1.6	64.1 $\pm$ 1.8	76.9 $\pm$ 1.8
2	86.5 $\pm$ 1.9	84.3$\pm$2.4	93.0$\pm$1.2	53.4 $\pm$ 2.1	84.3$\pm$2.3	64.0 $\pm$ 3.2	77.6$\pm$2.2
4	85.9 $\pm$ 2.0	83.6 $\pm$ 1.9	93.0$\pm$1.5	53.7$\pm$1.4	81.1 $\pm$ 1.7	64.3$\pm$1.9	76.9 $\pm$ 1.7
8	84.7 $\pm$ 1.1	82.8 $\pm$ 1.8	92.1 $\pm$ 1.4	53.4 $\pm$ 1.5	82.0 $\pm$ 1.8	63.9 $\pm$ 1.7	76.5 $\pm$ 1.6
10	84.8 $\pm$ 1.2	82.6 $\pm$ 1.3	91.8 $\pm$ 1.1	52.8 $\pm$ 1.3	83.4 $\pm$ 2.0	63.2 $\pm$ 2.0	76.4 $\pm$ 1.5

multimodal tokens across manipulation phases. Replacing it with a standard Transformer reduces performance by about 2.8 percentage points, showing that causal proximity masking and trajectory-adaptive weighting are important for stable contact-aware action generation. These drops are smaller than the 7.0–7.5 point drops caused by removing proximity cues or interaction point clouds in Table 7, suggesting that the representation provides the primary contact information, while the backbone determines how effectively it is used over time.

Ablation on the latent dynamics prediction horizon. To examine how the dynamics prediction horizon affects latent dynamics modeling, we vary H_{dyn} and report the results in Table 8. $H_{\text{dyn}} = 2$ achieves the best average success rate of 77.6%, slightly outperforming the other settings. Although the differences are modest, shorter horizons provide limited supervision for contact evolution, while overly long horizons may accumulate prediction errors and shift the dynamics objective toward coarse trajectory fitting. We therefore use $H_{\text{dyn}} = 2$ by default, as it balances local contact sensitivity and prediction stability.

D.3 Data Scalability Evaluation

The number of demonstrations directly affects how well an imitation-learning policy covers contact patterns. To evaluate learning efficiency under few-shot and medium-scale data regimes, we vary the number of expert trajectories on six Adroit and DexArt tasks and compare DP3, ManiFlow, AFRO, CordViP, and ProxiDex. The demonstration sizes are $N \in \{5, 10, 20, 30, 50, 100\}$, and all other training settings remain unchanged.

As shown in Fig. 15, overall success rates improve as the number of demonstrations increases, but different methods exhibit different growth rates and saturation levels. DP3 and ManiFlow are unstable with limited data, suggesting that policies based only on visible scene point clouds require more samples to cover contact establishment and object-pose variation. AFRO improves sample efficiency through dynamics pretraining, but because it lacks explicit proximity supervision, its gains remain limited on strongly contact-driven tasks such as Bucket and Faucet. CordViP improves low-

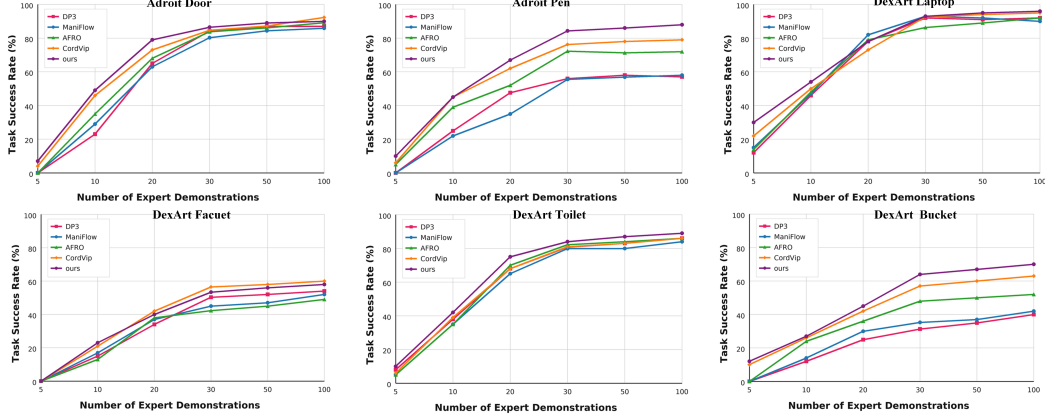


Figure 15: Scalability analysis with varying numbers of expert demonstrations on Adroit and DexArt tasks. ProxiDex consistently outperforms the baselines from 5 to 100 demonstrations, demonstrating scalable and data-efficient learning from collected demonstrations.

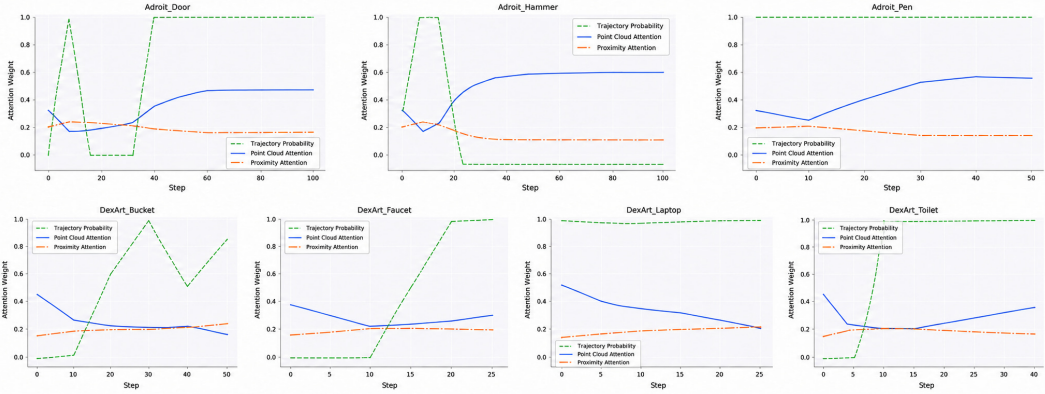


Figure 16: Attention weights visualization in simulation. We visualize the trajectory probability and averaged Transformer attention weights of point-cloud and proximity tokens across representative Adroit and DexArt tasks.

data performance through interaction point clouds, showing that recovering occlusion-free hand-object geometry is critical for few-shot learning.

ProxiDex achieves the highest average success rate across data scales. With 5, 10, 20, 30, 50, and 100 demonstrations, its six-task average success rates are approximately 11.5%, 40.0%, 64.0%, 77.5%, 80.0%, and 81.8%, respectively. The largest marginal gains occur before 30 demonstrations: the average increases by about 28.5 points from 5 to 10 demonstrations, 24.0 points from 10 to 20, and 13.5 points from 20 to 30. After 30 demonstrations, the curve begins to saturate, gaining only 4.3 points when moving from 30 to 100 demonstrations. Thus, 30 demonstrations already recover about 94.7% of the 100-demonstration performance, indicating that proximity representation provides a strong structured inductive bias for learning finger closure, contact maintenance, and release timing from limited data.

D.4 Attention Visualization in Simulation

We further visualize the attention weights in simulation environments, as shown in Fig. 16. The trajectory probability usually rises rapidly before contact or during the early contact-establishment stage, indicating that the policy can identify the transition from approach to manipulation. Point-cloud attention mainly reflects external geometric observations and contributes more to estimating the target pose, hand-object configuration, and reachability, so it is more active during early motion and pose adjustment. In contrast, proximity attention remains smoother and acts as a near-field

interaction modulation signal, constraining hand–object distance, contact stability, and manipulation robustness after contact becomes likely.

This visualization is consistent with the quantitative ablations above: the point-cloud branch provides global scene context, while the proximity branch becomes most useful when the task requires local contact refinement. Along with the change of trajectory probability, the policy adaptively balances attention between point-cloud geometry and proximity cues, leading to more stable dexterous control during contact establishment and subsequent manipulation.

D.5 Dream-Mode Robustness Analysis

To further characterize the stability range and error accumulation of Dream Mode under visual interruption, we measure end-effector pose deviations from the no-occlusion trajectory under controlled camera occlusions. As shown in Fig. 17, a 2-s occlusion causes only a transient deviation of about 1 cm, after which the system gradually realigns once visual tracking recovers. In contrast, under a 5-s occlusion, prediction error accumulates continuously: the deviation grows slowly at first, but increases rapidly after about 3.5 s to around 3 cm and further reaches about 5 cm, eventually causing arm drift and stalling. This indicates a reliable open-loop prediction horizon of approximately 3.5 s, which is sufficient for brief perception interruptions but not prolonged visual loss. This trend is also consistent with Fig. 5, where recovery performance decreases as occlusion duration increases. During inference, FDM evaluates latent-dynamics consistency at 20 Hz, and τ_{dyn} is set to the observation-feature discrepancy that first corresponds to a 1-cm end-effector pose deviation, enabling switching between normal visual feedback and dynamics takeover.

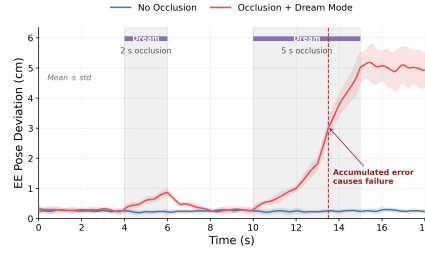


Figure 17: Dream-Mode robustness under controlled visual occlusion.

D.6 Pose-Tracking Analysis under Motion Speed

The preceding Dream-Mode robustness analysis (Sec. D.5) shows that short-term hand–object occlusion can be partially handled through latent prediction. Beyond occlusion, object motion speed is another important factor affecting the quality of interaction-point-cloud estimation. To evaluate this effect, we replay training trajectories at different end-effector speeds and measure the object pose-tracking error (ADD) and point-cloud error (Chamfer distance), both in millimeters. As shown in Fig. 18, both errors remain low up to 0.8 m/s, where the pose and point-cloud errors are approximately 8 mm and 6 mm, respectively. Beyond 0.8 m/s, both errors increase rapidly and approach 50 mm at 1.2 m/s, indicating degraded rotation estimation and point-cloud alignment under fast motion. Since proximity estimation relies on accurate hand–object relative geometry, such tracking errors directly reduce the reliability of proximity cues. Therefore, the current system is better suited to moderate-speed contact-rich manipulation, while highly dynamic motions remain limited by real-time pose-estimation accuracy.

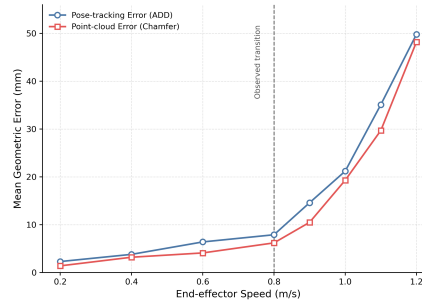


Figure 18: Pose and point-cloud errors at different speeds.



Figure 19: Real-world platform and evaluation objects. We use a RealMan RM75B arm with a 6-active-DoF dexterous hand and a fixed Intel RealSense L515 camera, and evaluate on seen and unseen objects with diverse shapes and sizes.

E Real-World Experiments

E.1 Experimental Setup and Task Description

Real-robot experiments evaluate ProxiDex under real perception noise, cross-object variation, and manipulation disturbances. The platform consists of a Realman RM75B 7-DoF arm, an Intel RealSense L515 RGB-D camera, and two dexterous hands, as summarized in Fig. 19. A 6-DoF CasBot-PIL hand is used for standard tabletop tasks, while a 20-DoF Wuji Hand V1 is used for more contact-rich manipulation. The policy must control not only the global arm pose but also finger closure timing and local contact state. The external RGB-D camera provides object pose tracking, and the reconstructed object point cloud is combined with robot forward kinematics to generate the hand-object interaction point cloud.

The three real-world tabletop tasks are **Pick**, **Pinch**, and **Sweep**. **Pick** requires the robot to grasp a tabletop object with multiple fingers and place it into a basket, testing grasp stability, transport, and release timing. **Pinch** requires the robot to form a small-area thumb-index contact and transfer the target object to the desired location, emphasizing fingertip alignment and stable closure. **Sweep** requires the robot to push a target object into a dustpan, grasp the dustpan handle, and pour the object into a container, involving pushing, tool grasping, and secondary contact transfer. In addition, Wuji Hand V1 is used for two contact-rich tasks involving sustained finger-object interaction and object reorientation. Together, these tasks cover whole-hand grasping, precise fingertip contact, and sustained in-hand interaction.

Demonstrations are collected using the VR teleoperation system described in Appendix A.1, with the operator controlling the arm and hand through Quest 3S. For each task, we collect 50 high-quality demonstrations while varying object initial positions, orientations, and some object types to improve adaptation to different geometric configurations. During point-cloud visualization, object pose estimates may exhibit mild jitter. This perturbation does not break action learning; instead, it increases observation variation in the training data and makes both the policy and the dynamics checker more robust to mild perception errors in deployment.

Fig. 22 visualizes real-world executions and their reconstructed interaction point clouds. The proximity responses remain concentrated near active hand-object contact regions, such as the grasping fingers in **Pick**, the thumb-index contact in **Pinch**, and the tool-object contact transfer in **Sweep**. This qualitative behavior links the real-world setup back to the geometric proximity representation in Appendix A.3 and motivates the evaluation protocol below.

E.2 Evaluation Protocol

We use three test settings to evaluate generalization and robustness. **In Distribution** uses the same objects and similar scene configurations as data collection, measuring task completion under standard conditions. **Unseen Objects** uses objects that do not appear in training and differ in shape, size, or surface appearance, testing generalization to object geometry changes. **Perturbation** intro-

duces complex tabletop backgrounds, clutter, short-term visual occlusion, or mild pose perturbations during execution, evaluating robustness under unstable perception.

Success criteria consider not only the final object location, but also contact stability and pose consistency during execution. For Pick, the robot must stably grasp the target and place it into the basket. For Pinch, it must form a stable thumb-index pinch and transfer the target to the specified area. For Sweep, it must sweep the target into the dustpan and complete pouring into the target container by grasping the dustpan. We evaluate ProxiDex under three test settings: standard execution, cross-object generalization, and disturbance robustness.

E.3 Failure Case Analysis

The results above collectively demonstrate the accuracy and robustness of ProxiDex, while Fig. 20 illustrates the remaining limitations of ProxiDex in fine-grained manipulation. Although ProxiDex achieves a practical balance between low-cost data collection and real-world deployment, its current proximity representation is still mainly geometry-driven and cannot fully capture precise fingertip force distribution. As shown in the zoomed failure case, the middle finger and thumb may occasionally fail to align symmetrically with the object during closure, resulting in unbalanced contact forces. In addition, during the cup-pinching task, the cup may exhibit slight shaking during detachment and transfer. This motion, together with partial hand-object occlusion, can reduce the accuracy and temporal stability of object pose estimation, thereby perturbing the interaction point cloud and affecting the stability of proximity-based feedback.

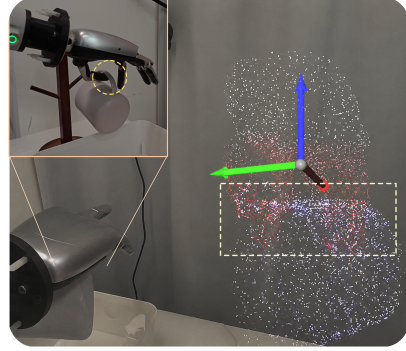


Figure 20: Failure cases under fine interaction. Finger alignment may occasionally be inaccurate, and object pose estimation may show mild jitter during contact-rich manipulation.

Future work can incorporate data gloves or lightweight tactile sensors to obtain higher-precision fingertip contact information, and use contrastive learning to align it with the geometric proximity representation proposed in this work. This would preserve the low-cost and cross-platform advantages of ProxiDex while improving force awareness and finger coordination in fine manipulation.

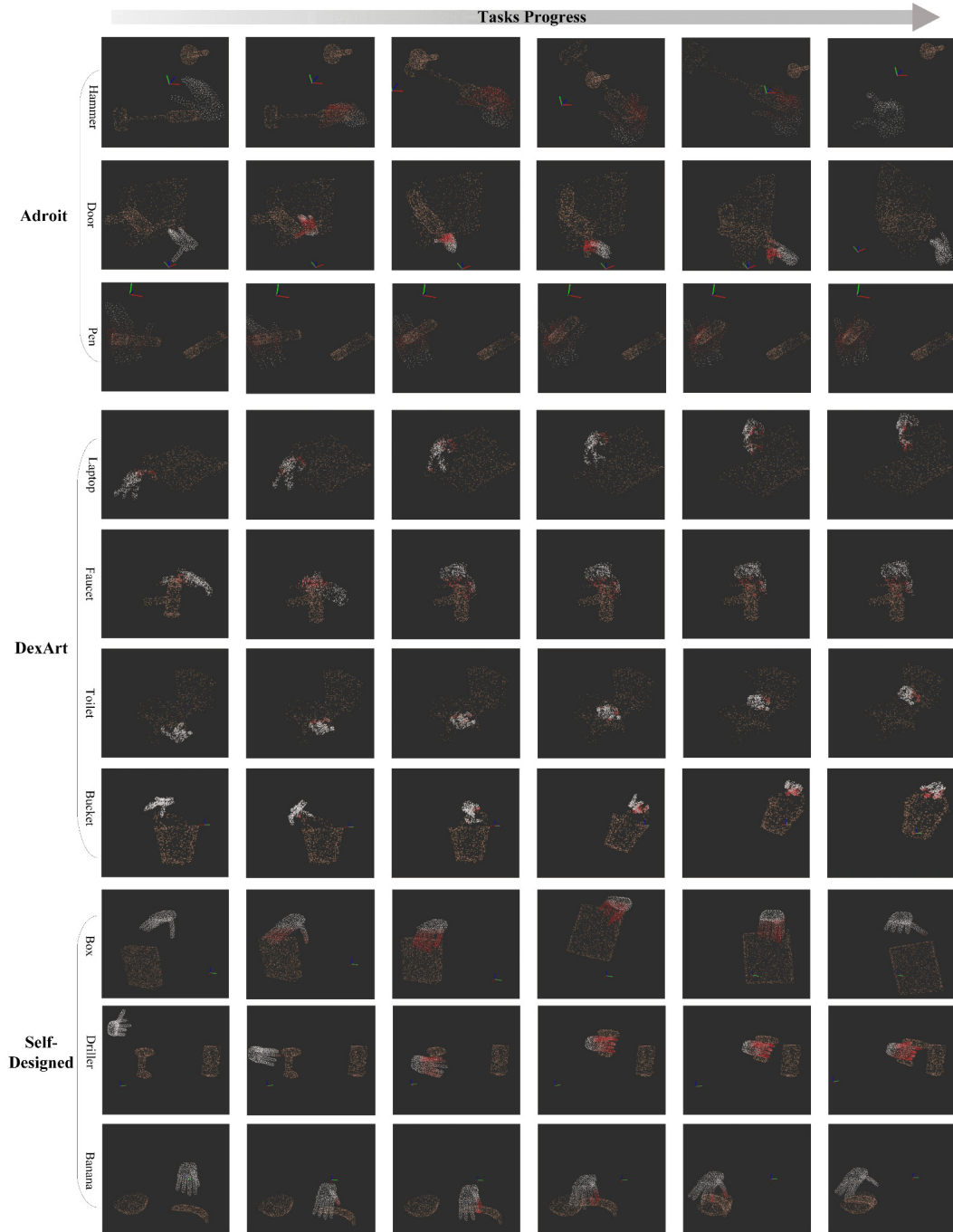


Figure 21: Visualization of interaction point clouds in simulation. Across Adroit, DexArt, and self-designed tasks, the processed hand-object point clouds preserve fine-grained interaction geometry, with red regions indicating stronger proximity responses along the task progress.

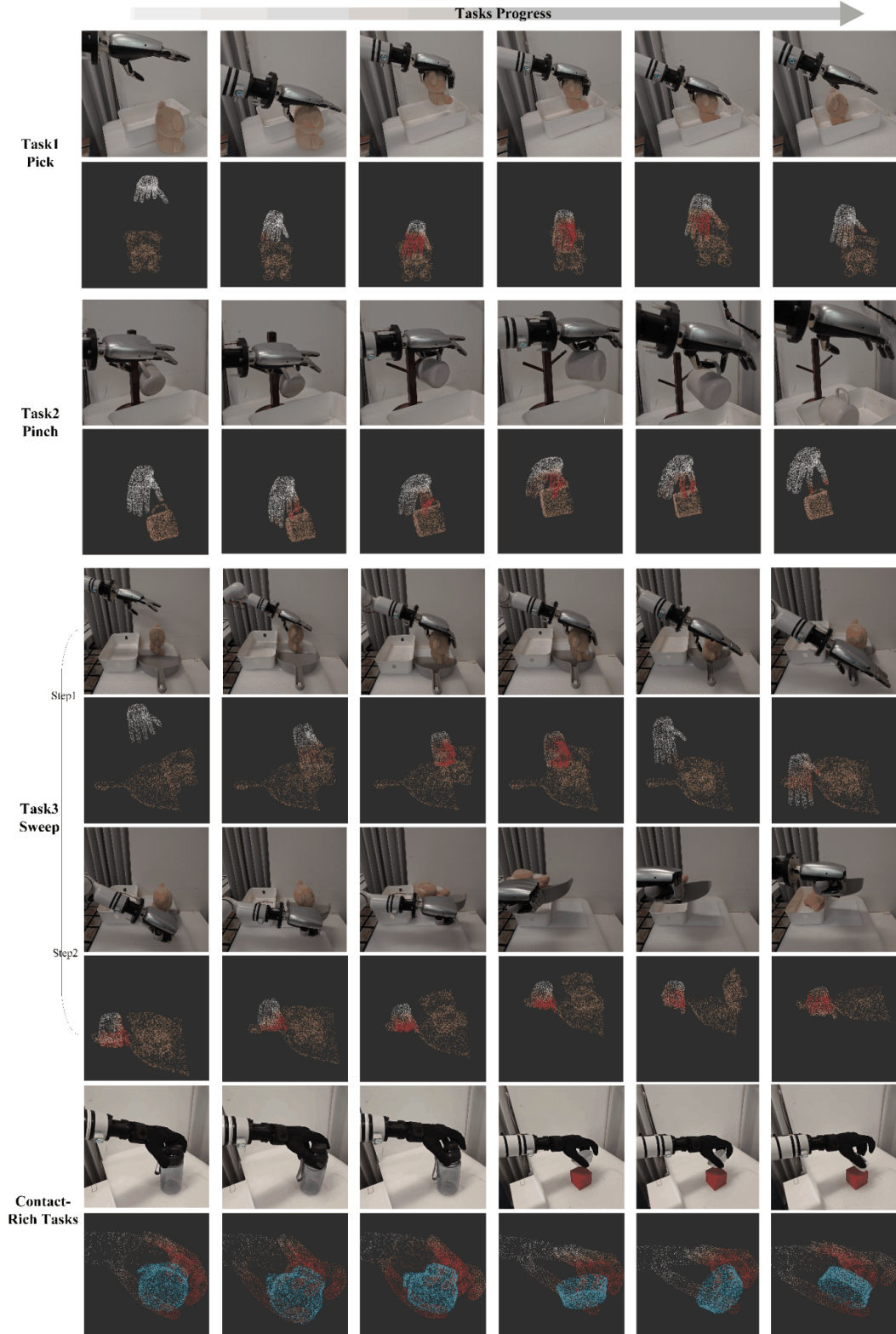


Figure 22: Visualization of real-world executions and the corresponding interaction point clouds. The reconstructed point-cloud sequences closely match the camera observations, while the color-coded proximity responses highlight local hand-object interaction regions during manipulation.