UniDex-ViTac: Learning Unified Visuo-Tactile Dexterous Manipulation Policy from Human Video Data

Hyesung Lee, Si-Hwan Heo and Sungwook Yang

***Abstract*— Learning dexterous robotic manipulation directly from human videos is fundamentally challenged by the kinematic embodiment gap and the lack of contact information that is unobservable in videos. To address these limitations, we present UniDex-ViTac, a unified visuo-tactile imitation learning framework that distills physically feasible, contact-rich trajectories generated by residual RL specialists into a single multi-task generalist policy. Crucially, our generalist operates on an expressive visuo-tactile representation that explicitly fuses global 3D point clouds with local binary tactile feedback. By effectively reasoning over both spatial geometry and local contact events, UniDex-ViTac achieves a 68.3% success rate in simulation and demonstrates robust Sim2Real transfer on a physical 16-DoF hand, achieving a 66.4% average success rate across diverse seen and unseen objects. Project page: https://unidex-vitac.github.io/.**

I. Introduction

Imitation learning (IL) from human experts [1]–[6] has enabled multi-fingered robotic hands to acquire complex manipulation skills. Nevertheless, collecting expert robot demonstrations through teleoperation remains costly, labor-intensive, and dependent on specialized hardware.

To reduce this burden, recent work has increasingly explored in-the-wild human videos as a scalable source of supervision. Existing methods can be broadly grouped into two categories. The first learns implicit latent actions [7] or dynamics representations directly from video. Although attractive for large-scale pre-training, this approach typically requires vast amounts of data and still relies on expensive robot demonstrations to adapt the learned representations to the substantial domain gap between human and robot actions.

The second category uses explicit geometric representations, such as MANO [8] hand meshes and object poses, to construct Human–Object Interaction (HOI) trajectories. Compared with latent video representations, HOI data provides more direct geometric guidance for manipulation. However, converting such trajectories into robot control remains challenging. First, the embodiment gap between human anatomy and robotic hardware leads to severe kinematic mismatches. Second, human videos do not provide direct access to latent physical quantities such as contact and force. Yet these signals are essential for contact-rich dexterous manipulation and are difficult to infer reliably from vision alone.

To address the embodiment gap, prior work often combines explicit human guidance with goal-conditioned reinforcement learning (RL) [9]–[13]. However, learning a single monolithic policy across diverse tasks and a high-degree-of-freedom (DoF) action space often leads to unstable exploration and optimization. Recent methods, such as Vividex [14], alleviate this issue by generating feasible trajectories in simulation and then training point-cloud-based policies. While promising, policies that rely only on vision and proprioception remain fundamentally limited in contact-rich settings, where local tactile feedback becomes crucial once visual observations are ambiguous or heavily occluded.

In this work, we present **UniDex-ViTac**, a unified visuo-tactile policy learning framework for robust multi-task dexterous manipulation. Our method first trains object-specific residual RL specialists that adapt coarse HOI trajectories into physically feasible robot motions, thereby bridging the embodiment gap while preserving task-relevant interaction patterns. We then collect successful, contact-rich rollouts from these specialists and distill them into a single multi-task generalist policy. Importantly, although the fusion of global 3D point-cloud geometry and binary tactile signals [15] has shown promise in reinforcement learning for in-hand manipulation, its effectiveness in imitation learning for contact-rich dexterous manipulation remains underexplored. We therefore investigate this visuo-tactile representation in a distilled imitation learning setting and show that it enables the policy to jointly reason about scene geometry and local contact events. Together, these design choices improve robustness in simulation and support effective Sim2Real transfer across both seen and unseen objects.

II. METHOD

Fig. 1 provides an overview of our proposed framework. In the following sections, we describe each component in detail.

A. Multi-Task Human Demonstration Dataset Setup

In this paper, we utilize the DexYCB [16] dataset, which contains human demonstrations of grasping and hand-over tasks. In this dataset, hand pose and shape are represented

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (No. RS-2024-00464386, No. RS-2025-25396144), and by the Korea Institute of Science and Technology (KIST) Institutional Program.

H. Lee, S.-H. Heo, and S. Yang are with the Center for Humanoid Research, Korea Institute of Science and Technology, Seoul 02792, Republic of Korea. H. Lee is also with the Kim Jaechul Graduate School of AI, Korea Advanced Institute of Science and Technology, Seoul 02455, Republic of Korea. ({hs981002, hershey, swyang}@kist.re.kr).

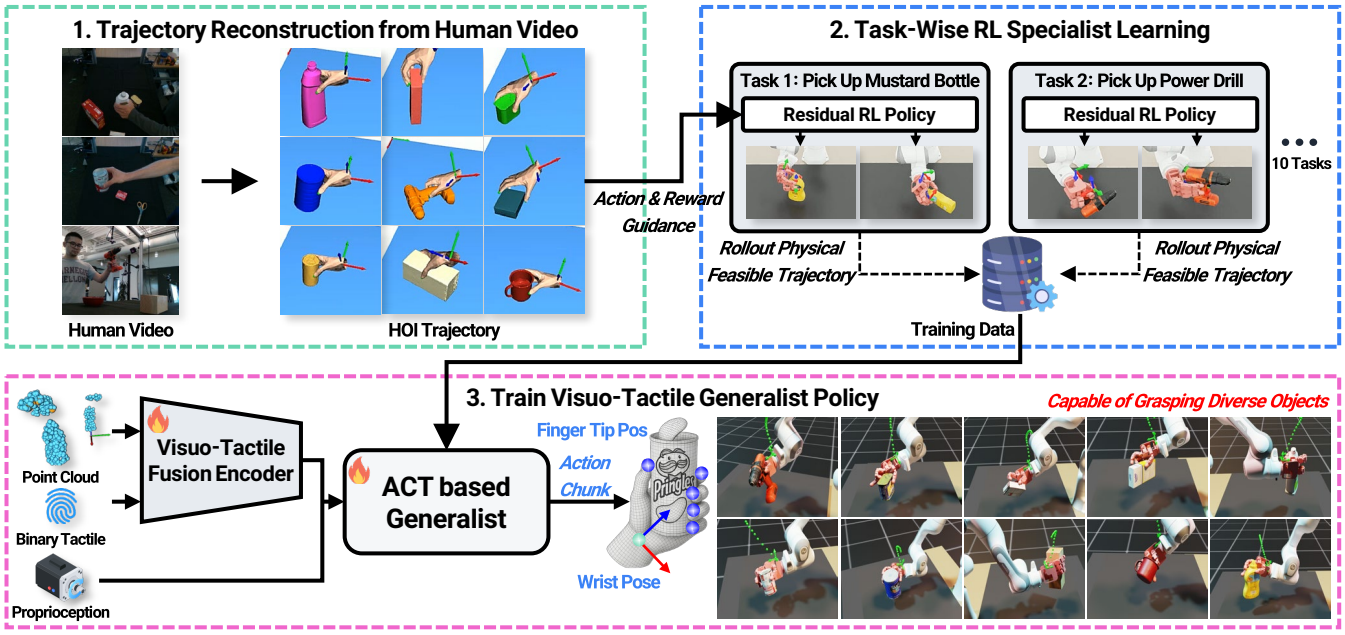

Fig. 1: The proposed pipeline for learning a unified generalist policy: (1) Reconstruct HOI trajectories from human video data. (2) Train a Residual RL specialist policy for each object. (3) Build a dataset by rolling out trajectories from RL specialists and train an ACT [3]-based visuo-tactile generalist policy capable of grasping diverse objects.

using the MANO model. From this MANO representation, we extract the wrist pose $\mathbf{w}_t$ and the wrist-relative fingertip positions $\mathbf{h}_t$ for each timestep t. These are then combined with the time-varying object pose $\mathbf{o}_t$ to construct HOI trajectories. Although DexYCB encompasses hand-over motions for 21 objects, we do not use the full dataset. Instead, we applied a filtering process to exclude trajectories with unstable initial interaction states. These filtered instances included objects placed on top of other objects, suspended unnaturally in mid-air, or leaning against other surfaces. Ultimately, a refined dataset of 50 video demonstrations, covering 10 distinct objects (5 videos per object), was selected and utilized for our experiments.

B. Residual RL Specialist Learning

To refine coarse, kinematically-derived HOI trajectories into physically feasible robot motions, we train a distinct reinforcement learning specialist policy for each object using its corresponding video demonstrations. Specifically, we employ the Proximal Policy Optimization (PPO) [17] algorithm with task-space residual action learning to adapt these reference trajectories to viable robotic actions. To accelerate convergence, we provide privileged information to both the actor and critic networks during training. Furthermore, to expand the spatial diversity of the limited HOI data, we apply domain randomization by translating and rotating the entire trajectory around the z-axis, centered on the object's initial world position. Detailed formulations of the reward function, observation space, and residual action design are provided in Appendix A.

Each specialist was trained for 1,000 policy update iterations across 8,192 parallel Isaac Lab simulation environments. The total training time for this process was approximately two days on a system equipped with a single NVIDIA RTX 5090 GPU.

C. Training the Unified Visuo-Tactile Generalist Policy

To distill specialist policies into a unified generalist, we curated a high-quality trajectory dataset by rolling out the pre-trained specialists. Throughout this paper, a trial is counted as a success if the robot grasps the object, lifts it by at least 20 cm, and holds this pose for at least 3 s. A trajectory is retained only if the underlying specialist succeeds across five independent rollouts under varied initial conditions, yielding a final dataset of $N = 10{,}000$ trajectories from 1,000 rollouts per specialist.

Each trajectory $\tau^{(i)}$ is a sequence of observation–action pairs,

$$\tau^{(i)} = \{(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)})\}_{t=0}^{T^{(i)}-1}, \tag{1}$$

where $\mathbf{s}_t$ denotes the observation at time step t and $\mathbf{a}_t$ denotes the corresponding action.

a) Observation: The observation $\mathbf{s}_t$ of our visuo-tactile generalist policy consists of three components: a point cloud $\mathbf{P}_t$, a proprioceptive state vector $\mathbf{p}_t$, and a binary contact state of fingertips $\mathbf{c}_t$:

$$\mathbf{s}_t = \big(\mathbf{P}_t,\, \mathbf{p}_t,\, \mathbf{c}_t\big). \tag{2}$$

The point cloud $\mathbf{P}_t \in \mathbb{R}^{512\times 6}$ consists of 512 points, where each point stores its 3D position (x, y, z) and a 3D one-hot feature vector. Among these 512 points, 508 are sampled from the camera depth map using Farthest Point Sampling (FPS), and the remaining 4 points correspond to the fingertip positions obtained from forward kinematics (FK). As

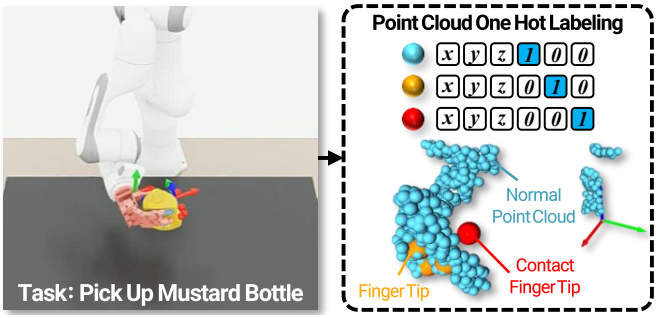

Fig. 2: Visualization of the point-cloud observation $\mathbf{P}_t$, composed of 508 FPS-sampled camera points and 4 FK fingertip points.

illustrated in Fig. 2, the one-hot feature encodes the semantic type of each point: $(1, 0, 0)$ for camera points, $(0, 1, 0)$ for non-contacting fingertips, and $(0, 0, 1)$ for contacting fingertips.

The proprioceptive state $\mathbf{p}_t$ concatenates the robot's kinematic features,

$$\mathbf{p}_t = \text{concat}\big(\mathbf{q}_{\text{hand},t},\, \mathbf{q}_{\text{arm},t},\, \mathbf{w}_t,\, \mathbf{f}_t\big) \in \mathbb{R}^{42}, \tag{3}$$

where $\mathbf{q}_{\text{hand},t} \in \mathbb{R}^{16}$ and $\mathbf{q}_{\text{arm},t} \in \mathbb{R}^{7}$ are the hand and arm joint positions, $\mathbf{w}_t \in \mathbb{R}^7$ is the wrist pose (3D position and quaternion), and $\mathbf{f}_t \in \mathbb{R}^{12}$ is the collection of fingertip positions expressed in the world frame.

The binary contact state $\mathbf{c}_t \in \{0, 1\}^4$ encodes whether each fingertip is in contact with the object or environment. Although the contact information is partially reflected in the point-cloud labels via the fingertip points, we additionally provide $\mathbf{c}_t$ as a separate, low-dimensional input token to the policy.

b) Action: The action $\mathbf{a}_t$ is defined as a target in task space, consisting of the desired end-effector wrist pose and wrist-relative fingertip targets:

$$\mathbf{a}_t = \big(\mathbf{w}_t^{\text{des}},\, \mathbf{f}_t^{\text{rel}}\big), \tag{4}$$

where $\mathbf{w}_t^{\text{des}}$ is the desired wrist pose (with rotation represented in the 6D continuous representation), and $\mathbf{f}_t^{\text{rel}}$ denotes the desired fingertip positions expressed in the local wrist frame. To execute these task-space actions, they are converted into joint-space commands; the arm tracks the wrist pose using a Damped Least Squares (DLS) inverse kinematics (IK) solver, while the hand joint configurations are computed using an analytical IK solver to reach the fingertip targets. The detailed neural network architecture of the Visuo-Tactile Generalist policy is provided in Appendix B.

III. EXPERIMENTS AND RESULTS

We evaluate our framework to address three core questions: **(RQ1) Sensory vs. Explicit State:** Does direct visuo-tactile observation yield more robust manipulation than explicit state representations (e.g., object ID and 6D pose)? **(RQ2) Modality & Fusion Ablation:** What is the impact of individual visuo-tactile modalities and their fusion architectures on performance? And **(RQ3) Sim2Real Transfer:** Can the policy, trained entirely in simulation, successfully transfer to a physical robot and generalize to unseen objects?

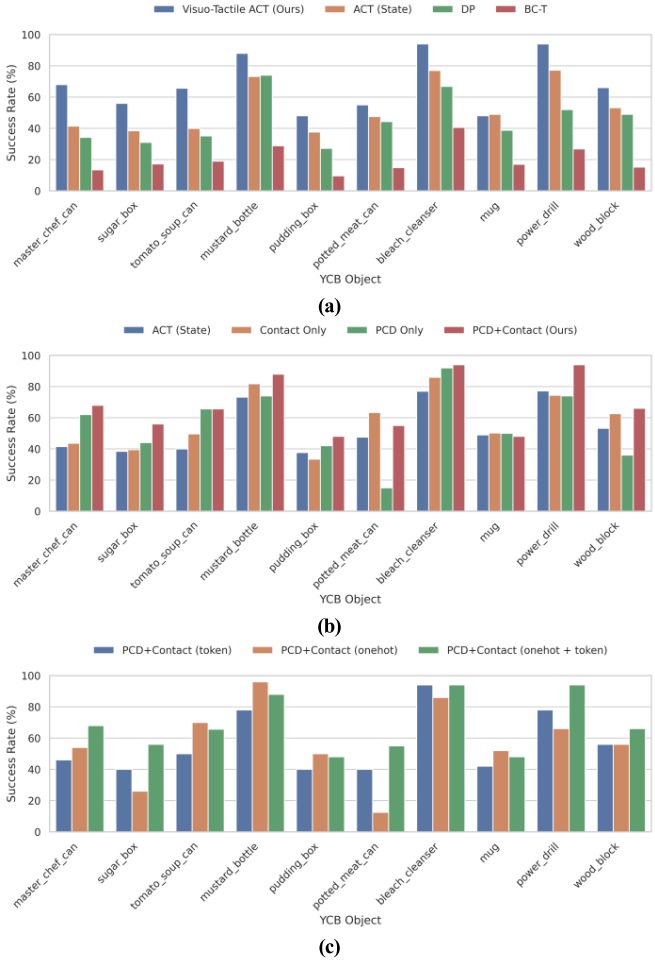

Fig. 3: Simulation results on 10 YCB objects. **(a)** Comparison with state-based baselines. **(b)** Ablation on input modalities. **(c)** Ablation on contact–point-cloud fusion.

A. Sensory vs. Explicit State Representation (RQ1)

To address **RQ1**, we compare our visuo-tactile policy against three state-based baselines: **ACT (State)**, **Diffusion Policy (DP)** [1], and **BC-Transformer (BC-T)** [2]. Unlike our method, these baselines rely on explicit state information, including the one-hot object ID and 6D object pose. All policies were trained on the same 10,000 specialist-generated trajectories, with our model trained for significantly fewer epochs (1,500 vs. 5,000 for baselines). All baselines were trained until their validation loss plateaued, and we report each method's best-performing checkpoint for a fair comparison.

Fig. 3(a) details the individual success rates across the 10 YCB objects. As demonstrated in these per-object results, our visuo-tactile policy consistently outperforms the baselines. Overall, it achieves an average success rate of **68.3%** across all objects, substantially surpassing ACT (53.4%), DP (45.2%), and BC-T (20.2%). The performance gap in the plot is most evident for small cylindrical objects (e.g., `master_chef_can`, `tomato_soup_can`), where our method shows a 17%–26% improvement.

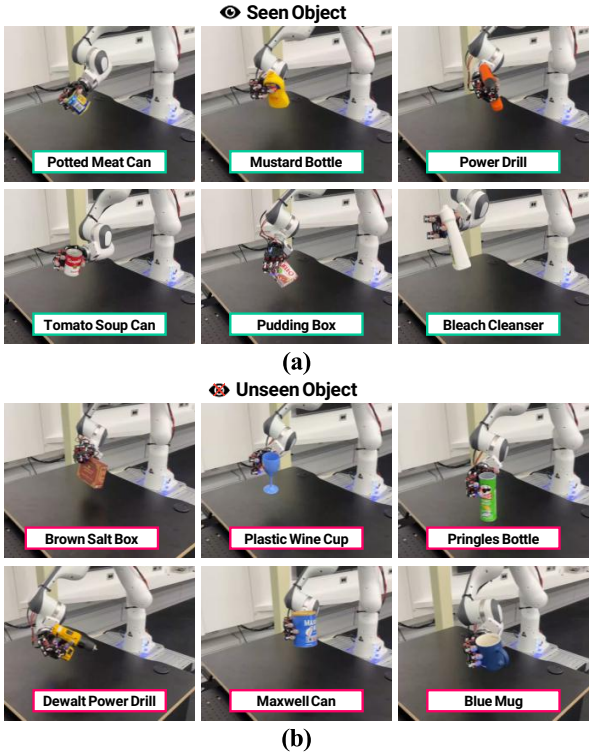

Fig. 4: Snapshots of real-world experiments evaluating our Visuo-Tactile Generalist Policy on (a) seen objects and (b) unseen objects with varying shapes and properties.

These per-object results suggest that explicit state representations are often underconstrained for learning dexterous manipulation. While state-based policies must infer complex geometric properties (e.g., local curvature, stable grasp regions) indirectly from proprioception and object ID, our policy directly leverages dense 3D point clouds and contact-aware labels. This richer representation allows the agent to learn more reliable manipulation strategies that generalize across diverse object geometries.

B. Ablation on Visuo-Tactile Inputs and Fusion (RQ2)

To address **RQ2**, we evaluate the individual contributions of point-cloud and binary contact modalities, alongside their architectural fusion strategies.

Policy Input Modalities: Fig. 3(b) details the per-object success rates across different input configurations. While adding only binary contact (*Contact Only*) to the state-based baseline improves the overall average success to 58.4%, relying solely on point clouds (*PCD Only*, 55.5% avg.) struggles with occluded small items (e.g., `potted_meat_can`) or large objects lacking clear volumetric cues. Our full visuo-tactile model effectively resolves these object-specific failures, achieving the highest overall average (68.3%) and proving the two modalities are highly complementary.

Contact–Point-Cloud Fusion: Having established the need for both inputs, Fig. 3(c) compares per-object performance across fusion architectures. Partial integrations (*token only* or *onehot only*) yield modest gains over the *PCD Only* baseline but still fail on challenging geometries. By contrast, our dual representation—*PCD+Contact (onehot + token)*—consistently excels across objects. Embedding contact locally in the point labels while simultaneously providing a global contact token equips the Transformer with the richest, most structured representation for robust manipulation.

TABLE I: Real-world experimental results comparing the success rates (out of 10 trials per object) of the vision-only policy and the visuo-tactile fusion policy.

Category	Object Name	PC Only	PC + Tac
Seen Objects	`bleach_cleanser`	10/10	9/10
	`mustard_bottle`	7/10	7/10
	`power_drill`	7/10	8/10
	`potted_meat_can`	4/10	7/10
	`tomato_soup_can`	3/10	5/10
	`pudding_box`	2/10	6/10
	Average	**55.0%**	**70.0%**
Unseen Objects	`maxwell_coffee_can`	9/10	9/10
	`brown_salt_box`	2/10	4/10
	`plastic_wine_cup`	7/10	6/10
	`blue_mug`	0/10	2/10
	`pringles_bottle`	9/10	10/10
	Average	**54.0%**	**62.0%**
Overall Average		**54.5%**	**66.4%**

C. Real-World Deployment (RQ3)

To investigate **RQ3**, we deployed our policy on a Franka Emika Panda robot with a 16-DoF dexterous hand to evaluate Sim2Real transfer across both *seen* and *unseen* objects. Detailed hardware specifications regarding the real-world binary tactile sensors are provided in Appendix C.

As summarized in Table I, our **Point Cloud + Binary Tactile** policy consistently outperforms the **Point Cloud Only** baseline, particularly on small objects (e.g., `potted_meat_can`, `pudding_box`) where hand-induced visual occlusion severely degrades the vision-only policy and tactile feedback provides reliable contact cues.

Nonetheless, we also observe a few cases (e.g., `bleach_cleanser`, `plastic_wine_cup`) where tactile feedback slightly decreases the success rate, which we attribute to the discrepancy in contact signals between simulation and the real world—an important direction for future work. Qualitatively, the robot still achieved stable grasps on both familiar instances (Fig. 4(a)) and unseen objects with diverse geometries and textures (Fig. 4(b)), demonstrating that our visuo-tactile fusion policy transfers effectively to the real world.

IV. Conclusion

We presented a novel pipeline for learning visuo-tactile dexterous manipulation policies from human video data. By fusing 3D point clouds with binary tactile one-hot encodings, we successfully trained a multi-task generalist policy validated in both simulation and reality.

However, critical limitations remain. First, a sim-to-real gap in contact sensing occasionally degrades performance on certain objects, where noisy tactile activations conflict

with an otherwise reliable vision-based grasp. Second, our binary contact representation discards essential continuous force magnitude and precise spatial localization data. Finally, the current pipeline incurs high computational costs, requiring up to two days for specialist training and trajectory collection.

Future work will explore advanced visuo-tactile architectures that integrate continuous force and precise localization, moving beyond binary bottlenecks to achieve more robust and efficient real-world manipulation.

REFERENCES

[1] 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*, p. 02783649241273668, 2023.

[2] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[3] 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.

[4] L. Heng, H. Geng, K. Zhang, P. Abbeel, and J. Malik, "Vitacformer: Learning cross-modal representation for visuo-tactile dexterous manipulation," *arXiv preprint arXiv:2506.15953*, 2025.

[5] 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.

[6] J. Cao, Q. Zhang, J. Sun, J. Wang, H. Cheng, Y. Li, J. Ma, K. Wu, Z. Xu, Y. Shao, *et al.*, "Mamba policy: Towards efficient 3d diffusion policy with hybrid selective state models," *arXiv preprint arXiv:2409.07163*, 2024.

[7] S. Ye, J. Jang, B. Jeon, S. Joo, J. Yang, B. Peng, A. Mandlekar, R. Tan, Y.-W. Chao, B. Y. Lin, *et al.*, "Latent action pretraining from videos," *arXiv preprint arXiv:2410.11758*, 2024.

[8] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *arXiv preprint arXiv:2201.02610*, 2022.

[9] Y. Chen, C. Wang, Y. Yang, and C. K. Liu, "Object-centric dexterous manipulation from human motion data," *arXiv preprint arXiv:2411.04005*, 2024.

[10] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik, *et al.*, "Dexteritygen: Foundation controller for unprecedented dexterity," *arXiv preprint arXiv:2502.04307*, 2025.

[11] S. Zhao, X. Zhu, Y. Chen, C. Li, X. Zhang, M. Ding, and M. Tomizuka, "Dexh2r: Task-oriented dexterous manipulation from human to robots," *arXiv preprint arXiv:2411.04428*, 2024.

[12] K. Li, P. Li, T. Liu, Y. Li, and S. Huang, "Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6991–7003.

[13] X. Liu, J. Adalibieke, Q. Han, Y. Qin, and L. Yi, "Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references," *arXiv preprint arXiv:2502.09614*, 2025.

[14] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid, "Vividex: Learning vision-based dexterous manipulation from human videos," *arXiv preprint arXiv:2404.15709*, 2024.

[15] Y. Yuan, H. Che, Y. Qin, B. Huang, Z.-H. Yin, K.-W. Lee, Y. Wu, S.-C. Lim, and X. Wang, "Robot synesthesia: In-hand manipulation with visuotactile sensing," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6558–6565.

[16] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield, *et al.*, "Dexycb: A benchmark for capturing hand grasping of objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9044–9053.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

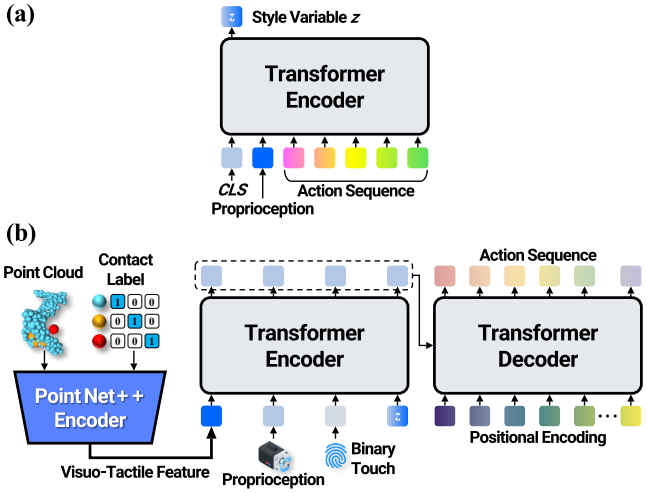

Fig. 5: Architecture of our ACT-based generalist policy: The observation consists of a 3D point cloud, a proprioceptive state vector, and a binary fingertip contact state. These components are encoded into tokens (including a CVAE latent style variable) and fed into a Transformer, which predicts a 30-step action chunk.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

APPENDIX

A. Residual RL Specialist Details

We train the RL Residual Specialist in simulation using PPO. To accelerate learning, both the actor and critic networks receive an identical observation vector $\mathbf{o}_t^{\text{spec}} = (\mathbf{s}_t^{\text{priv}}, \mathbf{p}_t, \mathbf{b}_t, \mathbf{f}_t, \hat{\boldsymbol{\tau}}_t, \mathbf{e}_t)$. This observation encapsulates the strictly privileged physical state variables $\mathbf{s}_t^{\text{priv}}$ (which include joint torques, joint velocities, object linear and angular velocities, and fingertip velocities), the combined robot proprioceptive state $\mathbf{p}_t$, binary contact states $\mathbf{b}_t$, contact forces $\mathbf{f}_t$, the current motion reference $\hat{\boldsymbol{\tau}}_t$, and the raw object intrinsic parameters $\mathbf{e}_t$ (which include object mass, friction coefficient, 6D pose, and scale). Both the actor and critic networks are implemented as multilayer perceptrons (MLPs) with hidden layer sizes of [1024, 1024, 512, 256], using ELU activations.

Rather than generating joint targets directly, the actor predicts a raw task-space residual action signal $\mathbf{u}_t$ at a policy frequency of 20 Hz. To ensure motion smoothness and stability, we apply an exponential moving average (EMA) filter to the raw policy output: $\tilde{\mathbf{u}}_t = \alpha \mathbf{u}_t + (1-\alpha)\tilde{\mathbf{u}}_{t-1}$, where α is the EMA smoothing coefficient. The actual task-space residual $\Delta\mathbf{a}_t$ is computed by integrating the filtered command over time. To prevent the policy from diverging excessively from the motion guidance, the integrated residual is strictly clamped:

$$\Delta\mathbf{a}_t = \text{clip}\left(\Delta\mathbf{a}_{t-1} + s \cdot \tilde{\mathbf{u}}_t \cdot \Delta t,\ \Delta\mathbf{a}_{\min},\ \Delta\mathbf{a}_{\max}\right), \quad (5)$$

where s is a scaling factor and $\Delta t = 0.05\,\text{s}$ is the inference time step. The maximum cumulative deviations are bounded

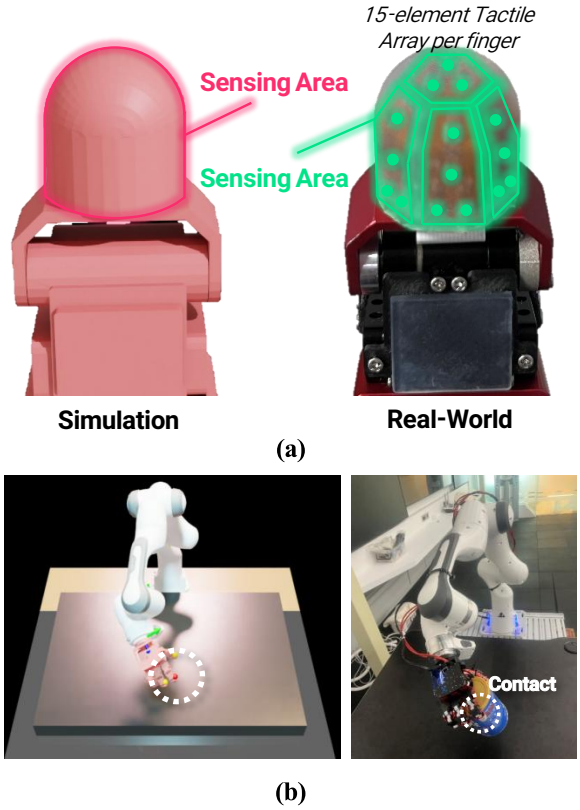

Fig. 6: Binary tactile sensing regions. (a) Binary tactile sensing regions in simulation and in the real-world setup. In simulation, the sensing region covers the pad side of the fingertip. In the real-world setup, the dotted area indicates the physical tactile sensor, while the outlined area denotes the overall sensable region. (b) Visualization of the real-world binary tactile sensing signals in the digital-twin simulation.

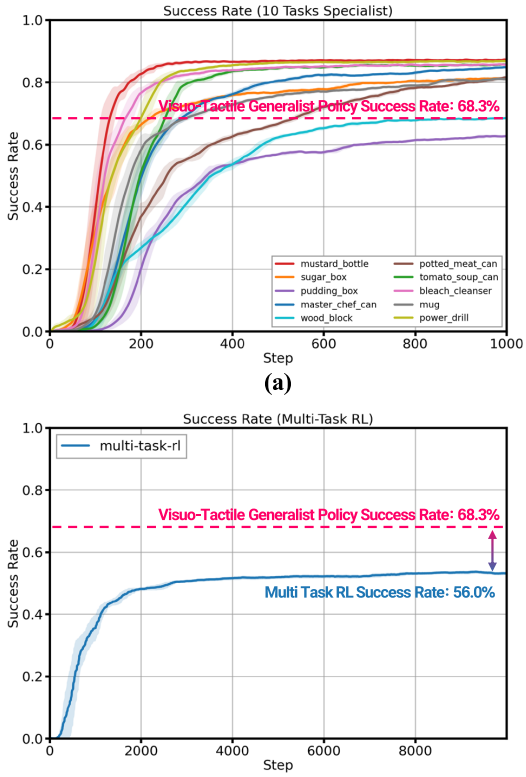

Fig. 7: Learning efficiency comparison between specialist and multi-task RL policies. (a) Ten individual specialists rapidly reach 70–85% success within 1,000 steps, while (b) a single multi-task policy converges slowly to only 56% after 10,000 steps.

to $\pm 0.1\,\mathrm{m}$ for wrist translation, $\pm 40^\circ$ for wrist orientation, and $\pm 0.05\,\mathrm{m}$ for fingertip translations. The refined task-space target is obtained by superimposing this clamped residual $\Delta \mathbf{a}_t$ onto the motion reference $\hat{\boldsymbol{\tau}}_t$.

The reward function is formulated to preserve the kinematic structure of the motion reference while ensuring physical feasibility, defined as a weighted sum:

$$r_t = \omega_{\text{obj}} r_{\text{obj}} + \omega_{\text{hand}} r_{\text{hand}} + \omega_{\text{wrist}} r_{\text{wrist}} + \omega_{\text{contact}} r_{\text{contact}} + \omega_{\text{penalty}} r_{\text{penalty}}. \tag{6}$$

The tracking terms (r_{obj}, r_{hand}, r_{wrist}) utilize exponential negative distance functions for Euclidean position and angular rotation to encourage the object, fingertips, and wrist to closely follow their respective references. The contact reward $r_{\text{contact}} = \|\mathbf{b}_t\|_1$ promotes stable grasping by maximizing the number of activated contact sensors. Finally, the penalty term $r_{\text{penalty}} = p_{\text{col}} + p_{\text{reg}}$ discourages undesired physical behaviors; p_{col} penalizes excessive impact forces on the fingertips and robot body to ensure hardware safety, while the regularization penalty $p_{\text{reg}} = -a_v \|\dot{\mathbf{q}}_t\|_2$ heavily penalizes excessive joint velocities to induce smooth and stable motions.

B. Generalist Policy Architecture

As our generalist policy, we employ an ACT-based model [3] (Action Chunking with Transformers), while replacing the original image-based vision backbone. Fundamentally, the ACT network adopts a Conditional Variational Autoencoder (CVAE) architecture, as illustrated in Fig. 5. Specifically, the overall architecture consists of a CVAE encoder (Fig. 5a) and a policy network (Fig. 5b). During the training phase, both the CVAE encoder and the policy network are trained jointly. However, the CVAE encoder is exclusively used for training, while only the policy network is utilized during deployment for inference.

Given the current observation $\mathbf{s}_t = (\mathbf{P}_t, \mathbf{p}_t, \mathbf{c}_t)$, the policy first converts each component into a token in the Transformer input space. The point cloud $\mathbf{P}_t$ is processed by a PointNet++ [18] based encoder to produce a point-cloud latent token. The proprioceptive state $\mathbf{p}_t$ is mapped to a second token via a linear projection. The contact state $\mathbf{c}_t$ is embedded through a small MLP to form a third token. These observation tokens are then fed into the Transformer encoder to condition the action generation process.

C. Real-World Tactile Sensor Hardware

The binary tactile sensing setup used in both simulation and the real-world environment is illustrated in Fig. 6. In

simulation, during the rollout of specialist trajectories, a binary tactile signal is set to 1 when the total contact force applied to the pad region of a fingertip exceeds 1.0 N, and 0 otherwise.

In the real-world setup, each finger of the dexterous hand is instrumented with barometric-based pressure sensors, with 15 sensors distributed across the hand in total. To calibrate the real-world threshold for each sensor, we applied a constant 1.0 N force to the sensor for 10 seconds and recorded the mean raw output value (within the sensor's full range of 0–8192). This per-sensor mean was then used as the binary-contact threshold. For each fingertip, the binary tactile signal is set to 1 whenever at least one of the sensors attached to that fingertip exceeds its corresponding threshold, and 0 otherwise.

D. Multi-Task RL vs. Single-Task Specialist

To justify our choice of training per-object specialists rather than a single unified RL policy, we investigate whether a single monolithic policy can fundamentally learn diverse dexterous manipulation tasks jointly. To this end, we train a multi-task RL baseline on all 10 object manipulations simultaneously across 8,192 parallel environments, using the same settings as our specialists.

Fig. 7 compares the learning curves of the two approaches. The gap is striking: the multi-task policy converges to only $\sim$56% success even after 10,000 update steps, whereas the individual specialists reach 70–85% within just 1,000 steps. This confirms that task decomposition with per-object specialists is a substantially more efficient learning strategy than a single monolithic multi-task policy.

Moreover, our visuo-tactile generalist policy, distilled from the trajectories rolled out by these individual specialists, achieves a 68.3% average success rate across all 10 tasks in simulation—substantially higher than the $\sim$56% ceiling reached by direct multi-task RL. This suggests that decomposing the problem into per-object specialists and subsequently distilling their trajectories into a unified policy is a more effective route to a multi-task generalist than training a single monolithic RL policy end-to-end.