DexUMI: Using Human Hand as the Universal Manipulation Interface for Dexterous Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: We present DexUMI - a data collection and policy learning framework that uses the human hand as the natural interface to transfer dexterous manipulation skills to various robot hands. DexUMI includes hardware and software adaptations to minimize the embodiment gap between the human hand and various robot hands. The hardware adaptation bridges the kinematics gap using a wearable hand exoskeleton. It allows direct haptic feedback in manipulation data collection and adapts human motion to feasible robot hand motion. The software adaptation bridges the visual gap by replacing the human hand in video data with high-fidelity robot hand inpainting. We demonstrate DexUMI's capabilities through comprehensive real-world experiments on two different dexterous robot hand hardware platforms, achieving an average task success rate of 86%.

Keywords: Dexterous Manipulation, Learning from Human, Imitation Learning

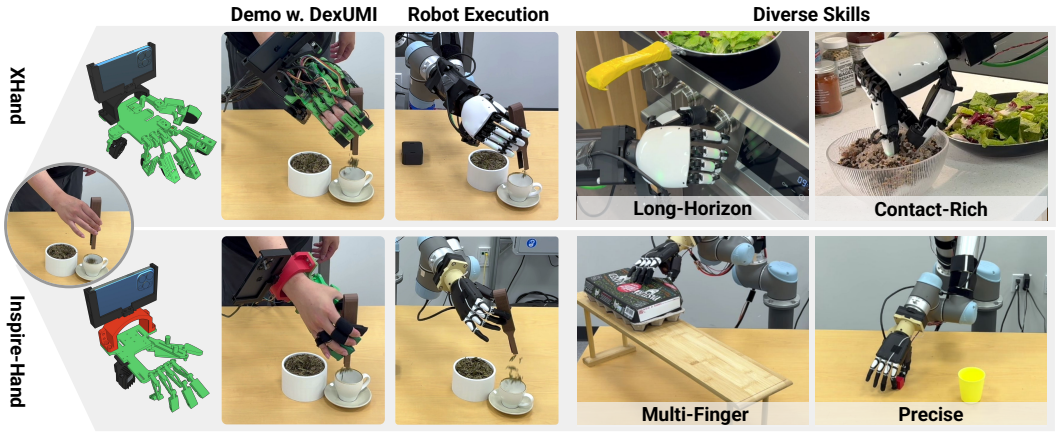

Figure 1: **DexUMI** transfer dexterous human manipulation skills to various robot hand by using wearable exoskeletons and a data processing framework. We demonstrate DexUMI's capability and effectiveness on both underactuated (e.g., Inspire) and fully-actuated (e.g., XHand) robot hand for a wide variety of manipulation tasks.

1 Introduction

Human hands are incredibly dexterous in a wide range of tasks. Dexterous robot hands are designed with the hope of replicating this capability. However, it remains a significant challenge to transfer skills from human hands to robotic counterparts due to their substantial *embodiment gap*. This gap manifests in various forms, such as differences in kinematic structures, contact surface shape, available tactile information, and visual appearance.

What further complicates this challenge is the diversity of dexterous hand hardware designs available today. Each robotic hand presents different engineering trade-offs in degrees of freedom, motor

∗ Indicates equal contribution

Submitted to the 9th Conference on Robot Learning (CoRL 2025). Do not distribute.

ranges, actuation mechanisms, and overall dimensions. The solution for reducing the embodiment gap must handle the vast hardware design space. Teleoperation has become a popular manipulation interface for dexterous hands. However, teleoperation can be difficult due to the spatial observation mismatch and the lack of direct haptic feedback. These problems do not exist when human hand can perform the manipulation task directly. In other words, human hand itself is a better manipulation interface. In this paper, we ask the following question:

How can we minimize the embodiment gap, so that we can use the human hand as the universal manipulation interface for diverse robot hands?

To answer this question, we propose **DexUMI**, a framework with hardware and software adaptation components that is designed to minimize the action and observation gaps.

The **hardware adaptation** takes the form of a wearable hand exoskeleton. A user can directly collect manipulation data while wearing it. The exoskeleton is designed for each target robot hand through *a hardware optimization framework* that refines exoskeleton parameters (e.g., link lengths) to closely match the robot finger trajectories while maintaining wearability for the human hand. The hardware adaption provides the following benefits:

- **Intuitive demonstration with direct haptic feedback:** Unlike teleoperation systems, the wearable exoskeleton has no spatial mismatch and allows users to directly contact objects during manipulation, making the demonstration intuitive and doable without a robot.
- **Records feasible motion for the robot hand:** The exoskeleton constrains human hand motions to match the kinematics of the target hand, ensuring the recorded motion is transferable.
- **Capturing precise joint action:** Unlike retargeting methods, our exoskeleton reads precise joint angles directly from encoders, eliminating inaccuracies due to visual fingertip tracking.
- **Matching tactile information for learning:** Most handheld grippers for data collection [1–3] do not record the tactile information. Our design includes additional tactile sensors on the fingertip to record the same tactile info as what the robot hand would record.

Our **software adaptation** takes the form of a data processing pipeline that bridges the visual observation gap between human demonstration and robot deployment. This processing pipeline first removes the human hand and exoskeleton from the demonstration video using video segmentation, then inpaints the video with the corresponding robot hand and environment backgrounds that match the target action. This adaptation ensures visual input consistency between training and robot deployment, despite visual differences between human and robotic hands.

With both hardware and software adaptation layers, DexUMI allows us to collect data on various tasks with minimal kinematic and visual gaps then transfer skills to robots. Comprehensive real-world experiments demonstrate DexUMI's capability on two different dexterous hand types: a 6-DoF Inspire hand [4] and a 12-DoF XHand [5]. Our approach achieves 3.2 times greater data collection efficiency compared to teleoperation and an average success rate of 86% across four tasks , including long-horizon and complex tasks requiring multi-finger contacts.

2 Related Work

Although extensive work has studied how to enable learning in simulated environments [6–20], we focus on reviewing real world data collection methods.

Teleoperation: Teleoperation is a popular interface for dexterous manipulation. Hand control is achieved with motion capture gloves [21–25], virtual-reality devices [26–28], or camera-based tracking [29–35]. Most approaches employ optimization-based retargeting to map human fingertips to robot hand. While being adaptable to different robot platforms, retargeting struggles with fundamental morphological differences between human and robot hands, especially the thumb flexibility [36]. Recent work by Zhou et al. [37] introduced a hand exoskeleton for direct joint mapping, but the mechanical structural differences limit the mapping accuracy. Additionally, teleoperation or kinesthetic teaching [38] require the robot hardware to be present, limiting the flexibility of data collection. In contrast, DexUMI collects manipulation data without physical robots.

Figure 2: **Exoskeleton Design.** The optimized exoskeleton design shares the same joint-to-fingertip position mapping as the target robot hand while maintaining the wearability. The exoskeletons utilizes the encoder to precisely capture the joint action and 150° DFoV camera to record the information-rich visual observation. An iPhone is rigidly mounted to track the wrist pose through the ARKit.

Human hand video: Learning manipulation skills from human hand video is an attractive direction. Prior works have explored learning affordance [39–42] or extracting human and object pose [43–47] from video. Though showing promising results, many of these works either require additional real-world robot data or need to learn the policy in simulation and depend on privileged information, such as object pose, to deploy the policy in the real world.

Wearable devices: Another line of work focuses on designing wearable devices for data collection, such as portable hand-held grippers [1–3, 48–57]. These approaches have demonstrated promising results in scaling real-robot manipulation skills. However, these systems primarily target simple parallel/pinch grippers and cannot be easily adapted to multi-fingered systems. Alternatively, Dexcap [58] uses motion capture gloves for in-contact data collection. However, it still relies on retargeting methods and human-correction data through teleportation. In contrast, our method eliminates these requirement, enabling direct policy deployment with data collected through DexUMI. Recently, Wei and Xu [59] and Fang et al. [60] proposed hand-over-hand systems for dexterous hands. These works require the actual robot hand to be available and lifted by the human hand.

3 Hardware Adaptation to Bridge the Embodiment Gap

This section introduces our hardware adaptation, which is a wearable exoskeleton design that adapts human motion to feasible robot actions. While the final exoskeleton design is robot-specific, the principles of the design framework can be shared. We introduce the design framework in two parts: mechanism design optimization (§3.1) and sensor integration (§3.2).

3.1 Exoskeleton Mechanism Design

Modern robot hands often closely mimic human hands anatomically, meaning that a hand exoskeleton would compete for space with the human hand wearing it. The biggest challenge is for the thumb, whose pronation–supination movement can sweep a large volume and cause significant collision between the human thumb and a naively designed exoskeleton. Our exoskeleton design has two goals to achieve:

1. *Shared joint-action mapping:* The exoskeleton and the target robot hand must share the same joint-to-fingertip position mapping, including their limits, so the action can transfer.
2. *Wearability:* The exoskeleton must allow sufficient natural movements of the user's hand.

While the first goal can be mathematically defined, the wearability goal is hard to write down concretely. Our solution is to parameterize the exoskeleton design and formulate the wearability requirements as constraints on the design parameters, then find a solution that accommodates wearability while preserving kinematic relationships by solving an optimization. To make the optimization feasible, we prioritize the exact kinematics of fingertip links, while allowing greater flexibility in the kinematics of links less likely to contact objects.

E.1 Design initialization: We initialize the design with parameterized robot hand models based on URDF files (See Fig. 3). When such detailed designs are unavailable (e.g., the Inspire-Hand's finger mechanisms), we substitute them with equivalent general linkage designs with the same DoFs (e.g., a four-bar linkage) and allow optimization to find parameters that best match the observed kinematic behavior. Please see Appendix for details.

E.2 Bi-level optimization objective: Our optimization objective maximizes the following similarity: $\max_{\mathbf{p}} \mathcal{S}(\mathcal{W}^{\text{tip}}_{\text{exo}}(\mathbf{p}), \mathcal{W}^{\text{tip}}_{\text{robot}})$, where $\mathcal{W}^{\text{tip}}_{\text{exo}}$ and $\mathcal{W}^{\text{tip}}_{\text{robot}}$ represent the fingertip workspaces (set of all possible fingertip pose in SE(3)) for the exoskeleton and robot hand, respectively. $\mathbf{p} = \{j_1, ..., j_n, l_1, ..., l_m\}$ is the exoskeleton design parameters including joint positions $j_i \in \mathbb{R}^3$ in the wrist coordinate (i.e., flange) and linkage lengths l_j. The function $\mathcal{S}(\cdot, \cdot)$ represents a similarity metric between the two workspaces, which quantifies how closely the exoskeleton's fingertip pose distribution matches that of the robot hand. In practice, the $\mathcal{S}(\cdot, \cdot)$ is implemented as minimization by sampling configurations from both workspaces. Given a set of K robot hand configurations $\theta_{\text{robot},k}$ and N exoskeleton configurations $\theta_{\text{exo},n}$:

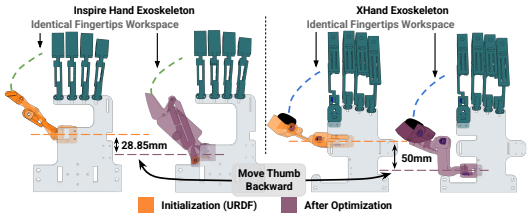

Figure 3: **Mechanism Optimization.** To avoid thumb collision between human hand and exoskeleton, the hardware optimization step allows us to move the exoskeleton thumb backward while still preserving the original fingertip and joint mapping in SE(3) space.

$$\mathcal{S}(\mathcal{W}^{\text{tip}}_{\text{exo}}(\mathbf{p}), \mathcal{W}^{\text{tip}}_{\text{robot}}) = -\Big(\sum_{k=1}^{K} \min_{\boldsymbol{\theta}_{\text{exo}}} \| \mathcal{F}^{\text{tip}}_{\text{exo}}(\mathbf{p}, \boldsymbol{\theta}_{\text{exo}}) - \mathcal{F}^{\text{tip}}_{\text{robot}}(\boldsymbol{\theta}_{\text{robot},k}) \|^2 + \sum_{n=1}^{N} \min_{\boldsymbol{\theta}_{\text{robot}}} \| \mathcal{F}^{\text{tip}}_{\text{exo}}(\mathbf{p}, \boldsymbol{\theta}_{\text{exo},n}) - \mathcal{F}^{\text{tip}}_{\text{robot}}(\boldsymbol{\theta}_{\text{robot}}) \|^2 \Big) \tag{1}$$

where $\mathcal{F}^{\text{tip}}_{\text{exo}}$ and $\mathcal{F}^{\text{tip}}_{\text{robot}}$ are the forward kinematics for the exoskeleton and robot hand respectively. Optimizing the first term encourages the exoskeleton to cover the robot hand's workspace by finding exoskeleton configurations closest to the sampled robot hand configurations. The second term requires $\mathcal{W}^{\text{tip}}_{\text{exo}}(\mathbf{p}) \subseteq \mathcal{W}^{\text{tip}}_{\text{robot}}$, ensuring the exoskeleton's fingertip workspace remains within the robot hand's capabilities, preventing generation of unreachable poses outside the robot hand's workspace.

E.3 Constraints: We apply bound constraints $j_i \in \mathcal{C}_i$ and $l_j^{\min} \leq l_j \leq l_j^{\max}$, which are empirically selected to ensure that the exoskeleton can be comfortably worn. For example, we want to move the thumb swing joint closer to the wrist along the x-axis under MANO [61] convention to avoid collision between the human thumb's pronation–supination movement and that of the exoskeleton.

3.2 Sensor Integration

Sensors on the exoskeleton need to satisfy the following design objectives:

1. *Capture sufficient information:* the sensors need to capture ALL the information necessary for policy learning, which includes: robot action such as joint angle (*S.1*) and wrist motion (*S.2*), as well as observations in both vision (*S.3*) and tactile (*S.4*).
2. *Minimize embodiment gap:* the sensory information should have minimal distribution shift between human demonstration and robot deployment.

S.1 Joint capture & mapping. To precisely capture joint actions, our exoskeleton integrates joint encoders at every *actuated* joint – using resistive position encoders for both the XHand and Inspire-hand. We choose the `Alps` encoder [62] for its size and precision. Due to the joint friction and motor backlash, the mapping between exoskeleton joint encoder θ^i_{exo} and robot hand motor $\mathcal{M}^i_{\text{robot}}$ values is often non-linear, therefore, we train a simple regression model for each joint to obtain this mapping. To calibrate the regression model, we collect a set of paired data by uniformly sampling K motor values on the physical robot for each finger and then find the corresponding exoskeleton joint value by overlaying the visual observation between the robot hand and exoskeleton. This process creates a paired dataset for us to train the regression model.

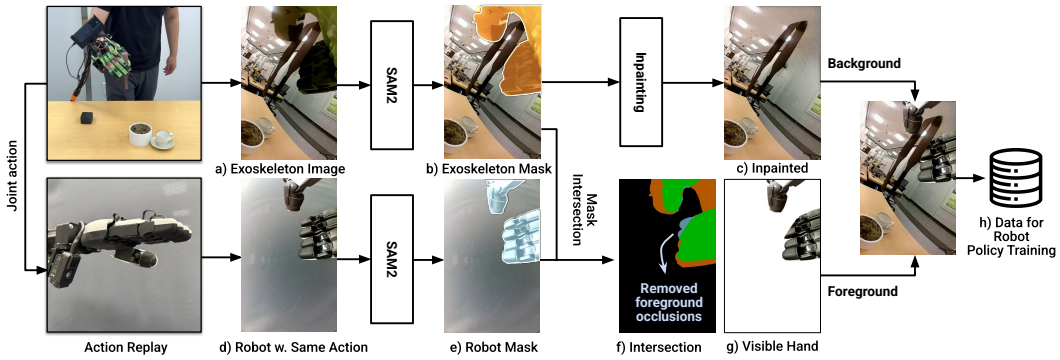

Figure 4: **Bridging the Visual Gap.** To convert the visual observation into policy training data, we first segment the exoskeleton using SAM2 (b) and inpaint the missing background (c). The corresponding joint action (a) is replayed on the dexterous hand to obtain the robot hand image (d). SAM2 is applied to obtain the robot mask (e). The intersection (f) of the exoskeleton mask (b) and robot mask (e) identifies the visible part of the hand during interaction. Finally, we replace pixels in the inpainted background (c) with the visible robot hand (g).

S.2 Wrist pose tracking. We use iPhone ARKit to capture the 6DoF wrist pose, as smartphones represent the most accessible devices capable of providing precise spatial tracking. This tracking device is only needed for data collection, not for robot deployment.

S.3 Visual observation. We mounted a 150° diagonal field of view (DFoV) wide-angle camera `OAK-1` [63] under the wrist for both the exoskeleton and the target robot dexterous hand. This positioning was chosen to effectively capture hand-object interactions. Critically, the camera poses in the wrist frame were identical for the exoskeleton and the robot hand, which maintains visual consistency between training and deployment.

S.4 Tactile sensing. The wearable exoskeleton allows users to directly contact objects and receive haptic feedback. However, this human haptic feedback cannot be directly transferred to the robotic dexterous hand. Therefore, we install tactile sensors on the exoskeleton to capture and translate these tactile interactions. To ensure consistent sensor readings, we install the same type of tactile sensors on the exoskeleton as those used on the target robot hand. For XHand, we use the electro-magnetic tactile sensor that comes with the hand. For the Inspire-Hand, we install the same resistive tactile sensor `Force Sensitive Resistor` [64] for both the exoskeleton and the robot hand.

4 Software Adaptation to Bridge the Visual Gap

Fig. 4 shows the visual gap between human demonstration (a) and robot deployment (h). To bridge this visual gap, we developed a data processing pipeline to adapt the demonstration image into what the robot will see as if the robot hand was collecting data. This adaptation uses off-the-shelf pretrained models to ensure generalizability. The adaptation takes four steps:

V.1 Segment human hand and exoskeleton. Firstly, we segment (Fig. 4b) the human hand and exoskeleton on observation videos using SAM2 [65]. Since SAM2 requires initial prompt points, we established a protocol where the human operator always begins with the same hand gesture, allowing us to reuse the same prompt points for all demonstrations.

V.2 Inpaint environment background. With segmentation, we remove the human hand and the exoskeleton pixels from the image data. Then we use ProPainter [66], a flow-based inpainting method, to fully refill (Fig. 4c) the missing areas [67–69].

V.3 Record corresponding robot hand video. Next, to render robot hand properly into the video, we replay the recorded joint action on the robot hand and record another video with only the robot hand (Fig. 4d). This step does not involve the robot arm. We then used SAM2 again to extract the robot hand pixels (Fig. 4e) and discard the background. Notice, it is possible to train an image generation model to output the robot hand image based on the actions, but it requires additional model training.

V.4 Compose robot demonstrations. The last step is to merge the inpainted-background-only video with robot-hand-only video. It is crucial to maintain proper occlusion relationships: the robot hand

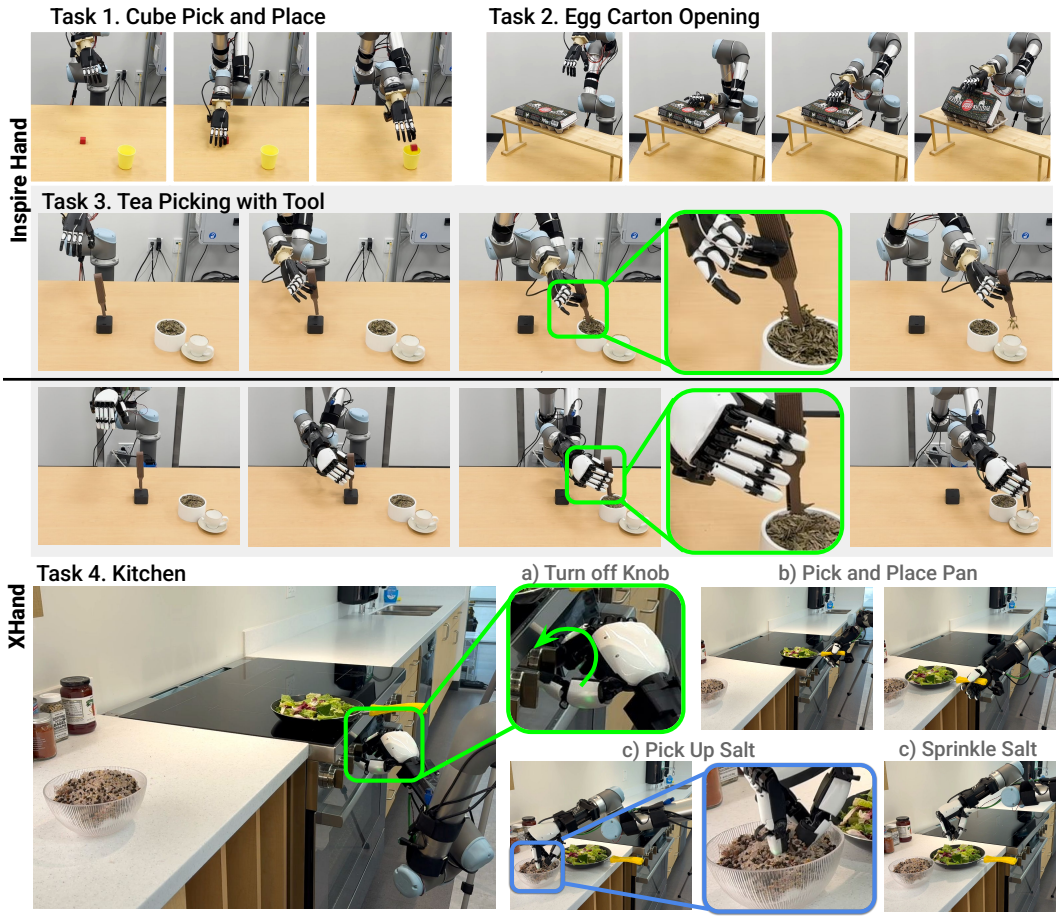

Figure 5: **Policy Rollout:** We evaluate DexUMI's capabilities across challenging real-world tasks. The **Cube** task tests basic picking precision. The **Egg Carton** task evaluates multi-finger coordination. The **Tea Picking** task assesses performance on contact-rich manipulation requiring millimeter-level fine-grained fingertip actions. Finally, the **Kitchen** task tests capabilities on long-horizon high-precision actions to manipulate a knob, move a pan using both the side of thumb and index finger (beyond just fingertips), and utilize tactile sensing for visually challenging salt picking tasks.

does not always appear on top. We developed an occlusion-aware compositing approach leveraging: (1) our consistent under-wrist camera setup, and (2) the kinematic and shape similarity between the exoskeleton and robot hand. We compute a visible mask (Fig. 4f) by intersecting the exoskeleton mask and robot hand mask. Rather than naively overwriting pixels, we selectively replace pixels in the inpainted observation with robot hand pixels only if those pixels are present in the visible mask. This preserved natural occlusion relationships between the hand and objects when viewed from our under-wrist camera perspective. This approach generated visually coherent robot manipulation demonstrations that maintained proper spatial relationships.

Imitation learning. Our imitation learning policy $p(\mathbf{a_t}|o_t, f_t)$ takes processed visual observation o_t and tactile sensing f_t as input. The output is a sequence of actions $\{a_t, \ldots, a_{t+L}\}$ of length L, starting from the current time t, denoted as $\mathbf{a_t}$. The robot action a_t includes a 6-DOF end-effector action and N-DOF hand action where N depends on the specific robot hand hardware.

5 Evaluation

Target robot hands: We evaluate DexUMI across two different robot hands:

- *Inspire Hand (IHand):* A twelve-DoF (six active DoFs) underactuated hand. The thumb has two active and two passive DoFs, while each remaining finger has one active and one passive DoF.
- *XHand:* A fully-actuated hand with twelve active DoFs. The thumb contains three DoFs, the index finger has three DoFs, and each of the remaining fingers has two DoFs.

Method			Inspire Hand				XHand				
Action	Tactile	Visual	Cube	Carton	Tea		Tea		Kichen		
					tool	leaf	tool	leaf	knob	pan	salt
Rel	Yes	Inpaint	**1.00**	0.85	**1.00**	0.85	**1.00**	**0.85**	**0.95**	**0.95**	**0.75**
Abs	Yes	Inpaint	0.10	0.35	0.80	0.00	**1.00**	0.25	0.50	0.45	0.00
Rel	No	Inpaint	0.95	**0.90**	**1.00**	**0.90**	0.95	0.80	**0.95**	**0.95**	0.15
Abs	No	Inpaint	0.90	0.85	0.90	0.60	**1.00**	0.75	0.60	0.60	0.0
Rel	No	Mask	0.60	0.10	0.90	0.50	/	/	/	/	/
Rel	No	Raw	0.20	0.05	0.85	0.05	/	/	/	/	/

Table 1: **Evaluation Results.** We report stage-wise accumualted success rate. The experiments compare different combinations of finger action representation (Absolute vs Relative), tactile feedback (Yes vs No), and visual rendering approaches (Inpaint vs Mask/Raw).

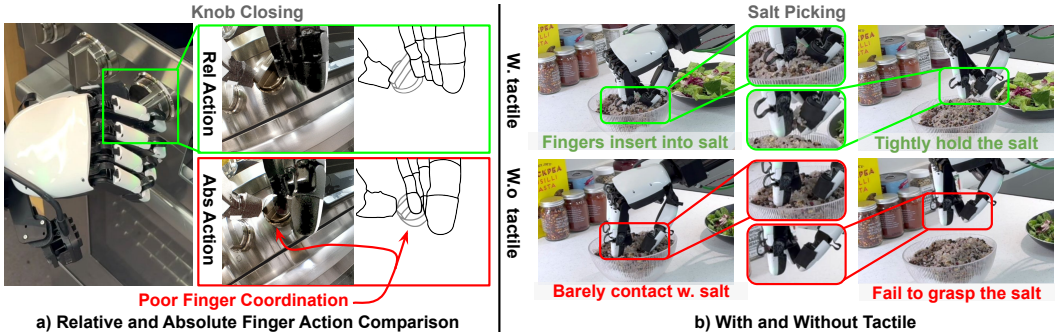

Figure 6: **Comparisons.** a) The policy outputs relative hand actions yield more precise action and demonstrate better multi-finger coordination. Note, we draw a sketch for the knob closing for better visualization. b) Even with noisy tactile sensor reading, the tactile significantly improve tasks which is visually challenging.

Tasks: We evaluate DexUMI across four different real-world tasks:

- **Cube** [IHand]: Pick up a 2.5cm wide cube from a table and place it into a cup. This evaluates the basic capabilities and precision of the DexUMI system.
- **Egg Carton** [IHand]: Open an egg carton with multiple fingers: the hand needs the index, middle, ring, and little fingers to apply downward pressure on the carton's top while simultaneously using the thumb to lift the front latch.
- **Tea** [IHand & XHand]: Grasp tweezers from the table and use them to transfer tea leaves from a teapot to a cup. The main challenge is to stably operate the deformable tweezers with multi-finger contacts.
- **Kitchen** [XHand]: The task involves four sequential steps: turn off the stove knob; transfer the pan from the stove top to the counter; pick up salt from a container; and lastly, sprinkle it over the food in the pan. The task tests DexUMI's capability over long-horizon tasks with precise actions, tactile sensing and skills beyond using fingertips.

Comparison: We evaluate the impact of policy action space choices, tactile sensing, and software adaptation on system performance.

- *Relative vs. Absolute finger action:* We compare the form of finger action trajectory: absolute position or relative trajectory proposed by [1]. We always use relative position for wrist action.
- *With vs. Without tactile sensing:* We trained policies with and without tactile sensor input.
- *With vs. Without software adaptation:* We examine two variants without software adaptation: (1) Mask, which replaces pixels occupied by the exoskeleton (during training) or robot hand (during inference) with a green color mask, and (2) Raw, which simply passes unmodified images containing the exoskeleton as policy input.

Evaluation protocol: For each evaluation episode, the test objects are randomly placed on the table at initialization. We conduct 20 evaluation episodes per task, maintaining consistent initial object configurations across our method and all baselines. For long horizon tasks, we report stage-wise accumulated success rate in Tab. 1.

5.1 Key Findings

DexUMI framework enables efficient dexterous policy learning: As shown in Tab. 1, the DexUMI system achieves high success rates across all four tasks on two robot hands. The system handles precise manipulation, long-horizon tasks, and coordinated multi-finger contact, while effectively generalizing across diverse manipulation scenarios.

Relative finger trajectories are more robust to noise and hardware imperfections: Tab. 1 shows relative finger trajectory consistently achieves better success across all tasks. Fig. 6 shows more insights: relative trajectory can make critical contact events more reliable. We hypothesize two reasons for this difference: 1. Relative action has a simpler distribution than absolute and is thus easier to learn; 2. Relative action learns a reactive behavior where the delta action keeps accumulating until a key event is reached (e.g. fingers close on contact). However, the absolute action learns a static mapping and would stall if the mapping has errors.

Only relative finger trajectories can benefit from the noisy tactile feedback: An interesting observation in Tab. 1 is how having tactile affects the results differently. The tactile sensor on the XHand can drift and become inconsistent after experiencing high pressure. Therefore, in most cases, having tactile makes the results worse. We observed that only with relative trajectory can the policy benefit from having such tactile sensing. For the Inspire hand, the tactile sensors we manually installed are even more noisy (See section §3.2 for details), then all methods become worse after adding tactile sensor as input. However, policies with relative trajectory still suffer less performance drop compared with the ones with absolute trajectory.

Tactile feedback improves performance on tasks with clean force profiles: We try to understand what kind of task would benefit from having tactile sensing. We focused on the XHand as its tactile sensors provide cleaner readings. We observed that tactile feedback significantly improved performance on picking up salt. This task highlights the effect of tactile because 1) The tactile sensors give a clear, large reading when the fingers touch the bowl of salt. 2) There is little useful visual information close to grasping as the camera view is mostly blocked by the bowl. In this case, we found that tactile feedback completely changes policy behavior. With tactile sensors, the fingers always insert into the salt first then close the fingers. Without tactile feedback, the fingers attempt to grasp the salt sometimes in the air. On the contrary, tactile info does not help in tweezer manipulation, which lacks strong correlation between hand motion and force feedback. Holding a tweezer only triggers minimal tactile sensor readings.

DexUMI framework enables efficient dexterous hand data collection: We compared data collection efficiency across three ways: DexUMI, bare human hand, and teleoperation on the tea-picking-with-tool task. The same human operator collected data using each approach within 15-minute sessions. We computed the collection throughput (CT) based on the number of successful demonstrations acquired. As illustrated in Fig. 7, while DexUMI remains slower than direct human hand manipulation, it achieves 3.2 times greater efficiency than traditional teleoperation methods, significantly reducing the time required for dexterous manipulation data collection.

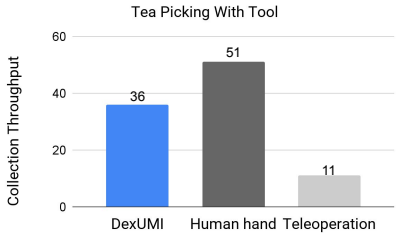

Figure 7: **Efficiency:** Collection throughput (CT) within 15-minute. Though DexUMI still slower than bare hand, it achieves significant higher efficiency than teleportation.

6 Conclusion

We present DexUMI, a scalable and efficient data collection and policy learning framework that uses the human hand as an interface to transfer human hand motion to precise robot hand actions while providing natural haptic feedback. Through extensive challenging real-world experiments, we demonstrate DexUMI's capability in learning dexterous manipulation policies for precise, contact-rich, and long-horizon tasks. Our work establishes a new approach to collecting real-world dexterous hand data efficiently and at scale beyond traditional teleoperation.

7 Limitation and Future Work

We would like to discuss DexUMI's limitations from three different aspects: hardware adaptation, software adaptation, and existing robot hand hardware.

Hardware Adaptation:

- *Per robot hand exoskeleton design:* Although DexUMI demonstrates generalizability across underactuated and fully-actuated hands, our optimization framework still requires hardware-specific tuning, especially for wearability. One future work direction is fully automated optimization formulation given robot hand model and some description of the human hand. Further, our hardware optimization framework can potentially leverage generative models [70] to increase efficiency and accuracy when design space grows.
- *Fingertips Matching:* Our current formulation focuses only on matching the fingertip workspace between the designed exoskeleton and target robot hand. It would be interesting for future work to also model remaining potential contact geometries such as the palm.
- *Wearability:* The hardware optimization pipeline makes the exoskeleton wearable and allows humans to operate it relatively easily for extended periods. However, wearability could be further improved by integrating soft materials, such as TPU for parts that contact the human hand. Additionally, constrained by both the design of the target hand and 3D printing material strength, users might still experience limitations in fully stretching certain fingers.
- *Reliability of Tactile Sensors:* Throughout our experiments, we found that reliable tactile sensors are key to maintaining consistent tactile observation between the exoskeleton and corresponding robot hand, thereby reducing the embodiment gap. In our implementation, the resistive tactile sensors added to the Inspire hand and its exoskeleton proved sensitive to their attachment way on fingers. Meanwhile, the electromagnetic tactile sensors on the XHand and its exoskeleton showed a tendency to drift after exposure to high pressure. Since the human hand generates more force than the robot hand, tactile sensor readings frequently drift when humans operate the exoskeleton. Future work can also incorporate other types of tactile sensors, such as vision-based tactile sensors [71–73] and capacitive F/T sensors [74].
- *Material Limitations:* Our experiments demonstrate that DexUMI is able to capture fine-grained fingertip actions such as closing tweezers. However, we sometimes found that encoders cannot precisely capture human motion due to 3D printing material strength limitations; occasionally, the human hand slightly distorts the exoskeleton linkage when manipulating objects. In such cases, encoders are unable to capture this distortion.

Software Adaptation:

- *Robot Hand Image:* Currently, we still require real-world robot hardware to obtain robot hand images. However, this requirement could be eliminated by implementing an image generation model that receives motor values as input and produces corresponding hand pose images as output.
- *Inpainting Quality:* Throughout our experiments, we found that the current software adaptation pipeline can already yield high-fidelity robot hand images. Nevertheless, we observed that illumination effects on the robot hand cannot be fully reproduced, and some areas in the image appear blurred due to limitations in the inpainting process.
- *Camera Location:* DexUMI currently requires the camera to be rigidly attached to the robot hand/exoskeleton and does not support a moving camera. However, it would be feasible to collect a dataset and train an image generation model that receives the relative pose between the camera and hand, along with hand pose information, to generate the corresponding hand pose image from any given camera position.

Existing Robot Hand Hardware:

- *Precision:* Throughout our experiments, we found that both the Inspire Hand and XHand lack sufficient precision due to backlash and friction. For example, the fingertip location of the Inspire Hand differs when moving from 1000 to 500 motor units compared to moving from 0 to 500

motor units. Although the desired motor value is the same in both cases, the final fingertip position varies. We observed this phenomenon in both robot hands. Consequently, when fitting regression models between encoder and hand motor values, we can typically ensure precision in only "one direction"—either when closing the hand or opening it. This inevitably causes minor discrepancies in the inpainting and action mapping processes. Further, we found that the XHand mapping between motor command and fingertip location slightly differs across time shifts or after each reboot.

- *Size Discrepancy:* The size difference between the robot hand and the human hand may cause wearability issues. For example, if the robot hand is twice as large as the human hand, it becomes difficult for both the human hand and the exoskeleton to reach the joint configurations required by the robot hand.
- *Co-design:* Many of these wearability issues arise from design constraints in existing commercial hardware. An interesting direction would be to explore a reverse design paradigm: first designing an exoskeleton that is comfortable and fully operable for humans, and then using that exoskeleton as the foundation for designing the robot hand.

References

[1] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. arXiv preprint arXiv:2402.10329, 2024.

[2] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation. arXiv preprint arXiv:2112.01511, 2021.

[3] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. arXiv preprint arXiv:2311.16098, 2023.

[4] Generic. inspire hand, . URL https://inspire-robots.store/collections/the-dexterous-hands/products/the-dexterous-hands-rh56dfx-series?variant=42735794422004.

[5] Generic. Xhand, . URL https://www.robotera.com/en/goods1/4.html.

[6] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 39(1):3–20, 2020.

[7] T. Lin, Z.-H. Yin, H. Qi, P. Abbeel, and J. Malik. Twisting lids off with two hands. arXiv preprint arXiv:2403.02338, 2024.

[8] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang. Lessons from learning to spin" pens". arXiv preprint arXiv:2407.18902, 2024.

[9] L. Sievers, J. Pitz, and B. Bäuml. Learning purely tactile in-hand manipulation with a torque-controlled hand. In 2022 International conference on robotics and automation (ICRA), pages 2745–2751. IEEE, 2022.

[10] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. arXiv preprint arXiv:2407.02274, 2024.

[11] M. Yang, C. Lu, A. Church, Y. Lin, C. Ford, H. Li, E. Psomopoulou, D. A. Barton, and N. F. Lepora. Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch. arXiv preprint arXiv:2405.07391, 2024.

[12] Y. Han, M. Xie, Y. Zhao, and H. Ravichandar. On the utility of koopman operator theory in learning dexterous manipulation skills. In Conference on Robot Learning, pages 106–126. PMLR, 2023.

[13] A. Agarwal, S. Uppal, K. Shaw, and D. Pathak. Dexterous functional grasping. arXiv preprint arXiv:2312.02975, 2023.

[14] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 5977–5984. IEEE, 2023.

[15] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang. Rotating without seeing: Towards in-hand dexterity through touch. arXiv preprint arXiv:2303.10880, 2023.

[16] G. Khandate, S. Shang, E. T. Chang, T. L. Saidi, Y. Liu, S. M. Dennis, J. Adams, and M. Ciocarlie. Sampling-based exploration for reinforcement learning of dexterous manipulation. arXiv preprint arXiv:2303.03486, 2023.

[17] B. Huang, Y. Chen, T. Wang, Y. Qin, Y. Yang, N. Atanasov, and X. Wang. Dynamic handover: Throw and catch with bimanual hands. arXiv preprint arXiv:2309.05655, 2023.

[18] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. In Conference on Robot Learning, pages 1722–1732. PMLR, 2023.

[19] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. arXiv preprint arXiv:2412.01791, 2024.

[20] T. G. W. Lum, A. H. Li, P. Culbertson, K. Srinivasan, A. D. Ames, M. Schwager, and J. Bohg. Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer. arXiv preprint arXiv:2410.23701, 2024.

[21] H. Zhang, S. Hu, Z. Yuan, and H. Xu. Doglove: Dexterous manipulation with a low-cost open-source haptic force feedback glove, 2025. URL https://arxiv.org/abs/2502.07730.

[22] K. Shaw, Y. Li, J. Yang, M. K. Srirama, R. Liu, H. Xiong, R. Mendonca, and D. Pathak. Bimanual dexterity for complex tasks. arXiv preprint arXiv:2411.13677, 2024.

[23] H. Liu, Z. Zhang, X. Xie, Y. Zhu, Y. Liu, Y. Wang, and S.-C. Zhu. High-fidelity grasping in virtual reality using a glove-based system. In 2019 International Conference on Robotics and Automation (ICRA), pages 5180–5186, 2019. doi:10.1109/ICRA.2019.8794230.

[24] M. Schwarz, C. Lenz, A. Rochow, M. Schreiber, and S. Behnke. Nimbro avatar: Interactive immersive telepresence with force-feedback telemanipulation. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5312–5319. IEEE, 2021.

[25] 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.

[26] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations. arXiv preprint arXiv:2412.14803, 2024.

[27] 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. arXiv preprint arXiv:2407.03162, 2024.

[28] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. In 8th Annual Conference on Robot Learning.

[29] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation. arXiv preprint arXiv:2403.07870, 2024.

[30] S. Yang, M. Liu, Y. Qin, R. Ding, J. Li, X. Cheng, R. Yang, S. Yi, and X. Wang. Ace: A cross-platform and visual-exoskeletons system for low-cost dexterous teleoperation. In 8th Annual Conference on Robot Learning.

[31] 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. In Robotics: Science and Systems, 2023.

[32] S. Han, B. Liu, R. Cabezas, C. D. Twigg, P. Zhang, J. Petkau, T.-H. Yu, C.-J. Tai, M. Akbay, Z. Wang, et al. Megatrack: monochrome egocentric articulated hand-tracking for virtual reality. ACM Transactions on Graphics (ToG), 39(4):87–1, 2020.

[33] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. In 2023 ieee international conference on robotics and automation (icra), pages 5954–5961. IEEE, 2023.

[34] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 9164–9170, 2020. doi:10.1109/ICRA40945.2020.9197124.

[35] Y. Qin, H. Su, and X. Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. IEEE Robotics and Automation Letters, 7(4): 10873–10881, 2022. doi:10.1109/LRA.2022.3196104.

[36] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 5962–5969, 2023. doi:10.1109/ICRA48891.2023.10160547.

[37] J. Zhou, B. Liang, J. Huang, I. Zhang, P. Abbeel, and M. Tomizuka. Global-local interface for on-demand teleoperation. arXiv preprint arXiv:2502.09960, 2025.

[38] Y. Hou, Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, S. Feng, B. Burchfiel, and S. Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control. arXiv preprint arXiv:2410.09309, 2024.

[39] A. Kannan, K. Shaw, S. Bahl, P. Mannam, and D. Pathak. Deft: Dexterous fine-tuning for real-world hand policies. arXiv preprint arXiv:2310.19797, 2023.

[40] P. Mandikal and K. Grauman. Learning dexterous grasping with object-centric visual affordances. In 2021 IEEE international conference on robotics and automation (ICRA), pages 6169–6176. IEEE, 2021.

[41] Y.-H. Wu, J. Wang, and X. Wang. Learning generalizable dexterous manipulation from human grasp affordance. In Conference on Robot Learning, pages 618–629. PMLR, 2023.

[42] A. Gavryushin, X. Wang, R. J. Malate, C. Yang, X. Jia, S. Goel, D. Liconti, R. Zurbrügg, R. K. Katzschmann, and M. Pollefeys. Maple: Encoding dexterous robotic manipulation priors learned from egocentric videos. arXiv preprint arXiv:2504.06084, 2025.

[43] S. Park, S. Lee, M. Choi, J. Lee, J. Kim, J. Kim, and H. Joo. Learning to transfer human hand skills for robot manipulations. arXiv preprint arXiv:2501.04169, 2025.

[44] 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.

[45] P. Mandikal and K. Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In Conference on Robot Learning, pages 651–661. PMLR, 2022.

[46] J. Ye, J. Wang, B. Huang, Y. Qin, and X. Wang. Learning continuous grasping function with a dexterous hand from human demonstrations. IEEE Robotics and Automation Letters, 8(5): 2882–2889, 2023.

[47] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In European Conference on Computer Vision, pages 570–587. Springer, 2022.

[48] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. Robotics and Automation Letters, 2020.

[49] H. Ha, Y. Gao, Z. Fu, J. Tan, and S. Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. arXiv preprint arXiv:2407.10353, 2024.

[50] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. In Conference on Robot learning, pages 1992–2005. PMLR, 2021.

[51] K. Doshi, Y. Huang, and S. Coros. On hand-held grippers and the morphological gap in human manipulation demonstration. arXiv preprint arXiv:2311.01832, 2023.

[52] P. Praveena, G. Subramani, B. Mutlu, and M. Gleicher. Characterizing input methods for human-to-robot demonstrations. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 344–353. IEEE, 2019.

[53] F. Sanches, G. Gao, N. Elangovan, R. V. Godoy, J. Chapman, K. Wang, P. Jarvis, and M. Liarokapis. Scalable. intuitive human to robot skill transfer with wearable human machine interfaces: On complex, dexterous tasks. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6318–6325. IEEE, 2023.

[54] M. Seo, H. A. Park, S. Yuan, Y. Zhu, and L. Sentis. Legato: Cross-embodiment imitation using a grasping tool. IEEE Robotics and Automation Letters, 10(3):2854–2861, Mar. 2025. ISSN 2377-3774. doi:10.1109/lra.2025.3535182. URL http://dx.doi.org/10.1109/LRA.2025.3535182.

[55] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation. arXiv preprint arXiv:2410.18647, 2024.

[56] F. Liu, C. Li, Y. Qin, A. Shaw, J. Xu, P. Abbeel, and R. Chen. Vitamin: Learning contact-rich tasks through robot-free visuo-tactile manipulation interface. arXiv preprint arXiv:2504.06156, 2025.

[57] Z. Liu, C. Chi, E. Cousineau, N. Kuppuswamy, B. Burchfiel, and S. Song. Maniwav: Learning robot manipulation from in-the-wild audio-visual data. In 8th Annual Conference on Robot Learning, 2024.

[58] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation, 2024. URL https://arxiv.org/abs/2403.07788.

[59] D. Wei and H. Xu. A wearable robotic hand for hand-over-hand imitation learning. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 18113–18119. IEEE, 2024.

[60] H.-S. Fang, B. Romero, A. Hu, L. Wang, E. Adelson, and P. Agrawal. Dexo: Hand exoskeleton system for teaching robot dexterous manipulation in-the-wild. 2023. URL https://fang-haoshu.github.io/files/DEXO.pdf.

[61] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: modeling and capturing hands and bodies together. ACM Transactions on Graphics, 36(6):1–17, Nov. 2017. ISSN 1557-7368. doi:10.1145/3130800.3130883. URL http://dx.doi.org/10.1145/3130800.3130883.

[62] Alps Alpine. Alps alpine rdc506018a rotary position sensor, 2025. URL https://www.digikey.com/en/products/detail/alps-alpine/RDC506018A/19529120. Accessed: March 23, 2025.

[63] Generic. Oak-1 w ov9792, 2025. URL https://shop.luxonis.com/products/oak-1-w?variant=44051403604191. Accessed: March 23, 2025.

[64] Generic. Zd10-100 force sensitive resistor (fsr) pressure sensor, 2025. URL https://www.amazon.com/Pressure-ZD10-100-Resistance-Type-Resistor-Sensitive/dp/B07MHTWR1C/. Accessed: March 23, 2025.

[65] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714, 2024.

[66] S. Zhou, C. Li, K. C. Chan, and C. C. Loy. Propainter: Improving propagation and transformer for video inpainting. In Proceedings of the IEEE/CVF international conference on computer vision, pages 10477–10486, 2023.

[67] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. arXiv preprint arXiv:2207.09450, 2022.

[68] P. Li, T. Liu, Y. Li, M. Han, H. Geng, S. Wang, Y. Zhu, S.-C. Zhu, and S. Huang. Ag2manip: Learning novel manipulation skills with agent-agnostic visual and action representations. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 573–580. IEEE, 2024.

[69] L. Y. Chen, C. Xu, K. Dharmarajan, M. Z. Irshad, R. Cheng, K. Keutzer, M. Tomizuka, Q. Vuong, and K. Goldberg. Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. arXiv preprint arXiv:2409.03403, 2024.

[70] X. Xu, H. Ha, and S. Song. Dynamics-guided diffusion model for robot manipulator design. arXiv preprint arXiv:2402.15038, 2024.

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

[72] B. Romero, H.-S. Fang, P. Agrawal, and E. Adelson. Eyesight hand: Design of a fully-actuated dexterous robot hand with integrated vision-based tactile sensors and compliant actuation. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1853–1860. IEEE, 2024.

[73] Y. Liu, X. Xu, W. Chen, H. Yuan, H. Wang, J. Xu, R. Chen, and L. Yi. Enhancing generalizable 6d pose tracking of an in-hand object with tactile sensing. IEEE Robotics and Automation Letters, 9(2):1106–1113, 2023.

[74] H. Choi, J. E. Low, T. M. Huh, G. A. Uribe, S. Hong, K. A. Hoffman, J. Di, T. G. Chen, A. A. Stanley, and M. R. Cutkosky. Coinft: A coin-sized, capacitive 6-axis force torque sensor for robotic applications. arXiv preprint arXiv:2503.19225, 2025.

[75] P. Contributors. Placo: A python library for planning and control, 2025. URL https://placo.readthedocs.io/. Accessed: May 8, 2025.

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

[77] X. Xu, Y. Yang, K. Mo, B. Pan, L. Yi, and L. Guibas. Jacobinerf: Nerf shaping with mutual information gradients. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16498–16507, 2023.

[78] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Robotics: Science and Systems, 2023.

[79] X. Xu, D. Bauer, and S. Song. Robopanoptes: The all-seeing robot with whole-body dexterity. arXiv preprint arXiv:2501.05420, 2025.

Appendix

A Additional Experiment Results

We show the processed visual observation by the software adaptation layer in policy training data in Fig. 8. Our software adaptation bridges the visual gap by replacing the human hand and exoskeleton in visual observations recorded by the wrist camera with high-fidelity robot hand inpainting. Though the overall inpainting quality is good, we found there are still some deficiencies in the output caused by:

- **Imperfect Segmentation from SAM2:** In most cases, SAM2 can segment the human hand and exoskeleton effectively. However, we notice SAM2 sometimes misses some small areas on the exoskeleton.
- **Quality of inpainting method:** We use flow-based inpainting to replace the human and exoskeleton pixels with background pixels. Though the overall quality is high, some areas remain blurry. We add Gaussian blur augmentation to the images during policy training to make the policy less sensitive to this blurriness.
- **Robot hand hardware limitations:** Throughout our experiments, we found that both the Inspire Hand and XHand lack sufficient precision due to backlash and friction. For example, the fingertip location of the Inspire Hand differs when moving from 1000 to 500 motor units compared to moving from 0 to 500 motor units. Consequently, when fitting regression models between encoder and hand motor values, we can typically ensure precision in only "one direction"—either when closing the hand or opening it. This inevitably causes minor discrepancies in the inpainting and action mapping processes.
- **Inconsistent illumination:** Similar to prior work [69], we found that illumination on the robot hand might be inconsistent with what the robot experiences during deployment. Therefore, we add image augmentation including color jitter and random grayscale during policy training to make the learned policy less sensitive to lighting conditions.
- **3D-printed exoskeleton deformation:** The human hand is powerful and can sometimes cause the 3D-printed exoskeleton to deform during operation. In such cases, the encoder value fails to reflect this deformation. Consequently, the robot finger location might not align with the exoskeleton's actual finger position.

B Evaluation Details

B.1 Initial State Selection

For each task, we manually select a set of initial states for the environment. Objects are placed as diversely as possible within the environment. This set of initial states is shared across all methods. We achieve consistency by placing an additional side camera to record images of all selected initial states. When starting a new evaluation episode, we visualize an image overlay between the recorded pre-selected initial state and the current initial state. We carefully adjust the current setup until it matches the pre-selected initial state with near pixel-perfect alignment.

Note that due to differences in wrist camera placement relative to the robot flange between the XHand and Inspire Hand, some initial states viable for the Inspire Hand cannot be completed by the XHand. For example, if the tea cup is positioned more than 45° to the left of the tea pot (image space), the XHand's wrist camera cannot capture the tea cup after grasping the tea due to its camera positioning (the XHand thumb has a larger range of motion, requiring us to rotate the wrist camera more toward the thumb direction to obtain clearer visual observations). Consequently, the XHand and Inspire Hand do not strictly share the same set of initial states for the Tea Picking Using Tool task. Nevertheless, we ensure their initial states remain within similar distributions and maintain as much diversity as possible.

For the kitchen task, the large workspace presents challenges for a fixed-base single UR5 to cover diverse initial states, particularly regarding the seasoning bowl location, as the stove and knob posi-

Figure 8: **Inpainting Results.** The visual observations in the original collected dataset contain exoskeletons and human hands. The software adaptation layer replaces these pixels with corresponding robot hand images while preserving the natural occlusion relationships during hand-object interactions.

tions are fixed. Despite these constraints, we maximize the diversity of bowl placement within the kinematically feasible workspace.

B.2 Success Criteria

Cube Picking: The robot must pick up the red cube and place it into the yellow cup. If the cup falls over after the cube is already placed in it, we still count the episode as successful.

Egg Carton: We define task success as when the lid is lifted up with its box at an angle greater than 30° and the egg box remains stable on the shelf.

Tea Picking Using Tool: This task consists of two sub-tasks. We define tool picking success as the robot's ability to steadily hold the tweezers and move them to the tea pot. We define leaf picking

success as the robot's ability to use tweezers to 1) grasp at least one tea leaf from the pot and 2) transfer at least half of the grasped tea leaf into the cup. Subsequent sub-tasks automatically count as failures if the previous sub-task fails, even if the robot can successfully complete the later sub-tasks.

Kitchen Manipulation: This task consists of three sub-tasks. We define knob closing success as the robot hand rotating the knob by at least 60° from its initial position. We define pan moving success as the robot moving the pan from the stove to the counter without dropping it during transfer. We define the salt task success as the robot 1) grasping some seasoning from the bowl and 2) sprinkling it inside the pan. Subsequent sub-tasks automatically count as failures if the previous sub-task fails, even if the robot can successfully complete the later sub-tasks.

B.3 Policy Execution

The learned policy predicts 16 steps of future actions, but the robot only executes the first 8 steps and discards the rest. The policy executes at 10 Hz, while the UR5 executes commands at 125 Hz. The Inspire Hand executes at 10 Hz, and the XHand executes at 60 Hz. The 10 Hz policy commands are linearly interpolated to match the desired hardware execution frequency.

The action output by the policy contains two components: relative UR5 end-effector action and hand action. The relative end-effector action from the learned policy is converted to absolute by adding the relative action to the current UR5 absolute position in the UR5 base frame. For hand actions, if the action type is absolute, the desired motor value is sent directly to the robot hand for execution. If the hand action type is relative, we first read the current hand motor position, add the relative hand action to it, and then send the result for execution.

For the XHand, we found that creating a virtual current hand motor position improves performance compared to reading the current position directly from hardware. Unlike the Inspire Hand motor, which is self-locking, the XHand finger position slightly drifts after encountering external forces (such as the restoring force of tweezers). The 10 Hz policy isn't reactive enough to adjust for this real-time drifting. Consider the following scenario: the robot hand attempts to close the tweezers to grasp tea leaves. The current motor value obtained by calling the hardware API might already be outdated due to the restoring force of the tweezers (causing fingers to spread wider) when robot execution begins. To address this issue, we initialize a virtual current hand motor position by reading the actual motor position at the beginning of the evaluation. Once the evaluation begins, we update this virtual hand motor position by adding the executed relative hand actions. With this virtual hand motor position approach, finger actions become less impacted by physical drifting, resulting in more precise and reliable grasping operations.

C Exoskeleton Design Details

C.1 Inspire Hand

Underactuated hands like the Inspire Hand typically incorporate closed-loop kinematics, such as four-bar linkages, which cannot be directly represented in URDF. As a result, we cannot initialize the exoskeleton design for the Inspire Hand directly from its URDF model. Instead, our approach is to capture the finger kinematic behavior—specifically, the fingertip poses—and use equivalent general linkage designs with the same degrees of freedom (DoFs) as an initial template for the finger mechanisms. This allows the optimization process to identify parameters that best match the observed kinematics.

To achieve this, we employed a motion capture system (see Fig. 9) to record the fingertip poses in SE(3) space. We 3D-printed marker mounting components for each finger and flange and installed them on the Inspire Hand. For the index, middle, ring, and pinky fingers, each of which has a single DoF, we uniformly sampled 16 motor command values from the lower limit (0) to the upper limit (1000), sent the commands to the fingers, and recorded the corresponding fingertip poses.

For the thumb, which has 2 DoFs—swing and bend—we first fixed the swing value and then uniformly sampled the bend motor values. For example, as shown in Fig. 9d, we set the swing motor to 400 and recorded the fingertip poses by varying the bend motor command. We repeated this procedure for swing values of 0, 200, 400, 600, 800, and 1000.

After obtaining the fingertip poses in the flange coordinate system, we applied the same bi-level optimization formulation defined in Equation 1 in main paper to determine design parameters for each finger. For all five fingers, we employed four-bar linkages as the linkage designs. For each sampled design parameter, We simulate the fingertip poses using PlaCo [75]. For thumb, we minimized the overall loss across all swing motor values, since the thumb's structural configuration should remain consistent regardless of the swing motor value.

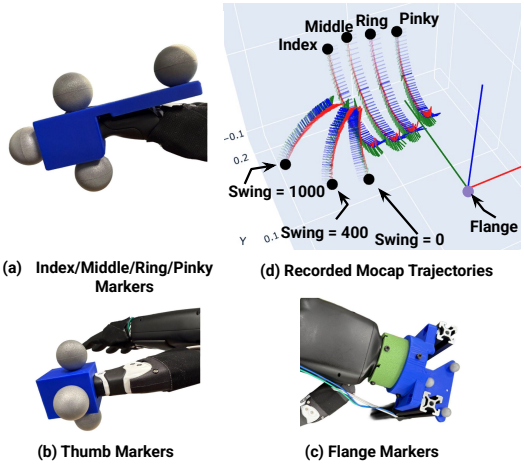

Figure 9: **Inspire Mocap:** We use motion capture system to record fingertips trajectories in the flange coordinate. We attached marker on fingers and flange to capture the fingertip pose in flange coordinate.

From the optimized design parameters to the physical implementation, we apply three additional steps to ensure that the exoskeleton mask consistently covers the real Inspire Hand. First, we extend the length of the last link of each finger in the exoskeleton design by 3 mm beyond the optimized value. This guarantees that the exoskeleton mask always fully covers the last link of the actual Inspire Hand. Second, we increase the width of the thumb's four-bar linkage to eliminate any hollow regions in the camera's field of view, thereby maintaining the visual integrity of a continuous exoskeleton mask. Third, we conservatively tighten the joint limits by 5° at each joint to ensure the mask continues to cover the real Inspire Hand even when structural deformation occurs due to the limited strength of the 3D-printed PLA-CF material.

C.2 XHand

Since the URDF file of the XHand is well-organized, with each joint origin defined at the location of its corresponding rotary joint, we can directly extract link lengths from the URDF structure. In cases where the exact values are not specified, we can perform reverse modeling using the STL meshes from URDF file to recover geometric features near each joint and manually measure link lengths in CAD software.

Joint limits are also specified in the URDF file and are implemented in the exoskeleton design by physically constraining the link motion to prevent rotation beyond the specified range. Similar to the Inspire Hand exoskeleton design, we adopt a conservative strategy when applying these limits setting slightly tighter bounds on each joints. For example, if the actual joint rotation range is -110° to 20°, the corresponding exoskeleton limit is set to -105° to 15°. This precaution accounts for possible deformation of the 3D-printed exoskeleton links under human-applied torque, which can introduce unintended joint deflection. Without this buffer, the exoskeleton might deform beyond the physical limits of the XHand, leading to an embodiment gap.

When converting the link lengths to the actual exoskeleton design, two primary constraints must be considered. The first is *wearability*. To ensure that the human operator can comfortably wear the exoskeleton, the structure must be hollowed out as much as possible, allowing the finger to pass through unobstructed. The second constraint is *material strength*. Through empirical testing, we determined that the optimal minimum structural width for 3D-printed PLA-CF material is 4 mm. Therefore, any part expected to experience significant stress is reinforced to be at least 4 mm thick in the final design.

D Sensor Details

D.1 Joint Encoder

Our exoskeleton uses Alps RDC506018A rotary sensors as encoders at every joints. These are resistive sensors whose resistance varies approximately linearly with absolute angular position.

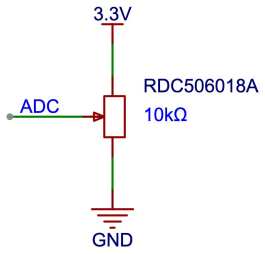

Figure 10: **Joint Encoder Circuit:** The rotary sensor acts as a variable resistor with three output pins. As it rotates with the joint, the voltage on the ADC line changes approximately linearly.

As shown in Fig. 10, when the joint rotates, the voltage on the ADC line changes proportionally. This analog voltage signal is then sampled by an Analog-to-Digital Converter (ADC) on a microcontroller unit (MCU). Then the joint angle α_{joint} can be estimated as:

$$\alpha_{\text{joint}} = \frac{V_{\text{ADC}}}{3.3\,\text{V}} \times 360^\circ$$

However, this simple voltage divider circuit has a significant failure mode: if the power supply (3.3 V in our case) is unstable due to temperature drift in semiconductor components or ripple from DC-DC converters and LDOs, the joint angle reading will drift accordingly. To mitigate this issue, we simultaneously measure the supply voltage through another ADC channel. Instead of dividing by a fixed 3.3 V, we normalize the sensor voltage using the measured supply voltage when computing the joint angle:

$$\alpha_{\text{joint}} = \frac{V_{\text{ADC}}}{V_{\text{supply}}} \times 360^\circ$$

This voltage normalization runs in real time on the MCU. After computing the joint angles, the MCU packs all joint values into a single data packet with a fixed 2-byte header and a checksum tail. The header simplifies decoding by allowing the receiver to locate a known keyword in variable-length data streams, while the checksum ensures packet integrity. The final data packet is transmitted to the host computer via a Universal Asynchronous Receiver-Transmitter (UART) interface.

D.2 Tactile

For commercial dexterous hands without built-in tactile sensors (e.g., the Inspire Hand in our evaluation), we use a simple and low-cost Force-Sensitive Resistor (FSR) as the tactile sensor. When no force is applied, the FSR exhibits a resistance of several megaohms, while under significant force, the resistance drops to the kiloohms range. As shown in Fig. 11, the FSR is incorporated into a simple voltage divider circuit to produce an analog voltage signal. The divider resistor R_1 is selected to be comparable to the minimum resistance of the FSR. Since the FSR resistance is approximately inversely proportional to the applied force, we can express the force using a constant scale factor k as:

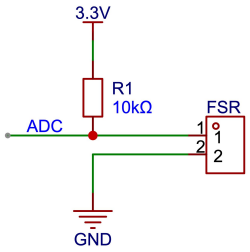

Figure 11: **Voltage Divider Circuit:** This simple voltage divider circuit converts the resistance change of the FSR sensor into an analog voltage on the ADC line.

$$F = k\left(\frac{V_{\text{supply}}}{V_{\text{ADC}}} - 1\right)$$

In our experimental setup, the same FSR sensor is mounted on both the dexterous hand and the exoskeleton. For simplicity, we directly use the V_{ADC} reading as a proxy for tactile input.

For hands equipped with onboard tactile sensors (e.g., the XHand), we install the same type of sensor as used in the hand. In our setup, this sensor is a magnet-based tactile array capable of measuring three-dimensional forces across 120 points on its surface. The force data is output via an SPI communication interface using a proprietary protocol. By configuring this interface on our embedded system, the force array can be successfully transmitted to the host machine.

E Data Collection and Policy Training details

E.1 Data Collection

We collected 310 trajectories for Cube Picking task policy training, 175 trajectories for Egg Carton Opening task policy training, and 400 trajectories for Tea Picking Using Tools policy training (for both Inspire Hand and XHand). For the kitchen task, we collected 370 trajectories covering all four sub-tasks, plus an additional 100 trajectories focused solely on knob closing.

For the Inspire Hand, all data types—including wrist position from ARKit, policy visual observations from the wrist-mounted camera, joint angles from encoders, and tactile feedback—were recorded at 45 FPS. For the XHand, we recorded at 30 FPS, as the tactile sensor readings became unstable at higher recording frequencies. For each data type, we recorded the receive timestamp t_{receive} when the data arrived at the recording buffer.

We wear green gloves when collecting data with exoskeleton as we use green PLA-CF to 3D-printed the exoskeleton. We found consistent color helps SAM2 to yield better segmentation results.

E.2 Training Data Latency Management

There is an inherent latency between the time when sensors capture data and when that data actually arrives in the recording buffer. To ensure our imitation learning policy receives properly aligned observations (visual observations, tactile sensor readings) and actions (joint encoder readings), we calculate the actual data capture time using $t_{\text{capture}} = t_{\text{receive}} - l_{\text{sensor}}$, where l_{sensor} refers to the latency from capture to receive for a particular sensor. We measure the iPhone and OAK camera latency by reading a rolling QR code displayed on a computer monitor showing the current computer system time, as proposed in UMI [1]. The camera and iPhone latency is calculated as $l_{\text{camera}} = t_{\text{receive}} - t_{\text{display}} - l_{\text{display}}$, where l_{display} represents the monitor refresh rate.

The encoder latency is adjusted by examining the overlay image between the recorded exoskeleton image and the corresponding robot hand image from action replay. If the encoder latency is set too high, the robot hand fingers will execute future actions and lead in the overlay image. If the encoder latency is set too low, the robot hand fingers will lag behind the exoskeleton fingers in the overlay image. We tune the encoder latency until the exoskeleton fingers and robot fingers are perfectly aligned. Once all data timestamps are adjusted, we linearly interpolate the joint angles and tactile readings to obtain data points properly aligned with the camera timestamps. Finally, We downsample the data by a factor of 3 to reduce the policy training time.

E.3 Policy Training

We process the visual observations with pretrained DINO-V2 [76, 77]. Before passing the visual observations into DINO-V2, we augment it with random crop, color jitter, random grayscale and Gaussian Blur. We concatenate the CLS token from DINO-V2 with tactile sensor readings as input to the diffusion policy [78, 79]. The policy predicts 16 steps of robot actions, which contain both 6-DoF robot end-effector relative actions and hand actions (6-DoF for Inspire Hand and 12-DoF for XHand). We train the models for 400 epochs across all tasks for both types of hands. The pretrain DINO-V2 is not frozen and updated during the policy training.