Cross-Embodiment Robot Manipulation via a Unified Hand Action Space

Luis Felipe Casas[1] **Robert Teal**[1] **Keval Shah**[1] **Abhijit Tadepalli**[2]
Wanxin Jin[2] **Yu Xiang**[1]

[1]Intelligent Robotics and Vision Lab, University of Texas at Dallas
[2]Intelligent Robotics and Interactive Systems Lab, Arizona State University

{Luis.CasasMurillo, Robert.Teal, Keval.Shah, Yu.Xiang}@utdallas.edu
{vtadepa1, wjin}@asu.edu

Abstract: Robot manipulation policies are typically tied to specific robotic hand embodiments, limiting the transfer of learned behaviors across platforms with different kinematic structures. In this work, we propose the Unified Hand Action Space (UHAS), a sphere-based unified action representation for cross-embodiment dexterous manipulation. UHAS represents robotic hand actions as geometric deformations of a canonical sphere and uses a Cascade Inverse Kinematics (CIK) algorithm to map the shared representation to embodiment-specific joint configurations. Using reinforcement learning, we train dexterous manipulation policies directly in the proposed action space for in-hand cube reorientation tasks. We evaluate our method in both simulation and real-world experiments across multiple robotic hands, including the Allegro Hand, LEAP Hand, Shadow Hand, and MANO Human Hand. Experimental results demonstrate effective dexterous manipulation, zero-shot transfer to unseen hands, rapid finetuning across embodiments, and successful real-world deployment. Our experiments show that the proposed UHAS representation enables stable dexterous control and cross-embodiment policy transfer across robotic hands.[1]

Keywords: Cross-Embodiment Manipulation, Unified Action Space, Reinforcement Learning, In-Hand Manipulation

1 Introduction

Robot manipulation has made rapid progress in recent years due to advances in imitation learning [1, 2], reinforcement learning [3, 4], and vision-language-action (VLA) models [5, 6, 7]. While recent policies have demonstrated impressive capabilities in grasping, pick-and-place, dexterous manipulation, and long-horizon task execution, most large-scale robot learning systems and datasets remain dominated by relatively simple end-effectors such as parallel-jaw grippers [8, 9, 10, 11, 12]. In contrast, dexterous robotic hands remain comparatively underexplored due to their highly diverse kinematic structures, action parameterizations, and control interfaces. As a result, existing dexterous manipulation systems often rely on embodiment-specific action spaces, datasets, and retraining procedures [13, 14, 15], making cross-embodiment transfer across robotic hands challenging.

To address this challenge, we investigate the problem of robot learning in a unified hand action space for robot manipulation. *The central idea of our approach is to represent hand actions through the deformation of a canonical sphere representation.* Instead of defining actions directly in embodiment-specific joint spaces, we model manipulation actions as geometric deformations on a shared spherical surface. This representation provides a continuous and embodiment-agnostic interface that captures the semantic behavior of robotic hands while abstracting away low-level hardware differences.

[1]Data, code, and videos for the project are available at https://irvlutd.github.io/UHAS/.

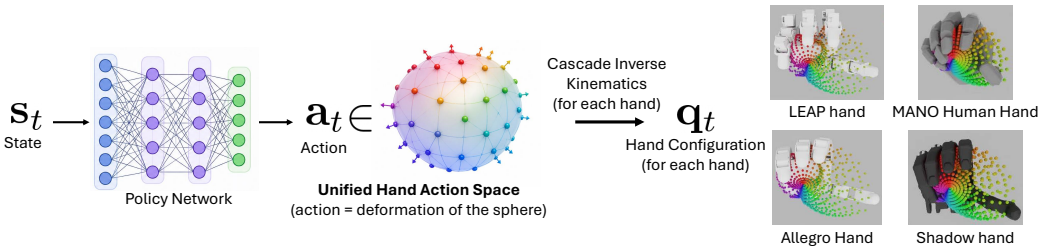


Figure 1: In our unified hand action space, an action is represented as the deformation of a canonical sphere. A deformed sphere is mapped to hand configurations of various embodiments (LEAP [16], Allegro [17], MANO Human [18] and Shadow [19]).

Our sphere-based action representation is motivated by the observation that many robotic hands interact with objects through spatial contact patterns around an object-centered workspace despite their diverse mechanical designs. By representing hand actions as deformations of a canonical sphere, we obtain a shared geometric action space that generalizes across different hand morphologies. Embodiment-specific hand motions are recovered through the proposed inverse kinematics algorithm that maps sphere deformations to executable joint configurations. This unified representation enables cross-embodiment policy transfer, data sharing across heterogeneous robotic platforms, and scalable policy learning for future robot foundation models.

In this work, we build upon the proposed sphere-based unified hand action space to develop a framework for cross-embodiment dexterous manipulation. We introduce a geometric action representation based on sphere deformations together with a cascade inverse kinematics algorithm that maps the unified representation to embodiment-specific hand joint configurations. Using reinforcement learning [20], we train manipulation policies directly in the proposed action space for dexterous in-hand manipulation. We evaluate our method on in-hand cube reorientation tasks across multiple robotic hands with different kinematic structures. Experimental results demonstrate that the proposed representation enables stable dexterous control, effective multi-hand policy learning, and meaningful zero-shot transfer to previously unseen robotic hands. The main contributions of this work are:

- We propose the Unified Hand Action Space (UHAS), a sphere-based geometric action representation that models robotic hand actions as deformations of a canonical sphere.
- We develop a Cascade Inverse Kinematics (CIK) algorithm that maps the unified sphere representation to heterogeneous robotic hands with different kinematic structures.
- We demonstrate dexterous in-hand manipulation, multi-hand policy learning, and cross-embodiment transfer across multiple robotic hands in simulation and the real world, including zero-shot transfer to unseen hands.

2 Related Work

Dexterous Manipulation. Dexterous in-hand manipulation remains a fundamental challenge in robotics due to the high-dimensional action spaces and complex contact dynamics involved in controlling multi-finger robotic hands. Prior works have demonstrated that reinforcement learning can learn sophisticated dexterous manipulation skills through large-scale simulation training and domain randomization [21, 22, 23, 24, 25]. More recent efforts have explored learning dexterous behaviors from demonstrations and large-scale video datasets [26, 27, 28, 29, 30, 31]. In this work, we focus on dexterous in-hand cube reorientation and investigate how a unified geometric action representation can improve manipulation transfer across robotic hands with different kinematic structures.

Unified and Cross-Embodiment Action Representations. Recent robot learning systems increasingly aim to support multiple robotic embodiments through shared representations and generalist policies. RT-2 [9], Octo [10], and OpenVLA [5] demonstrate the potential of large-scale robot foundation models, but most existing approaches still rely on embodiment-specific action spaces. Several recent works have explored unified representations for cross-embodiment learning, including CrossFormer [32], Universal Actions [33], XL-VLA [34], and One Hand to Rule Them All [35]. These methods use different action heads, latent actions or canonical joint-space representations.

Our work introduces the Universal Hand Action Space (UHAS), where manipulation actions are represented as geometric deformations of a canonical sphere shared across robotic hands.

Geometric Representations and Cross-Hand Transfer. Geometric representations have recently emerged as an effective abstraction for cross-embodiment robotic grasping and manipulation [36, 37, 38, 39, 40, 41]. For example, D(R,O) Grasp [36] proposes a unified representation of robot-object interactions for cross-embodiment dexterous grasping, while RobotFingerPrint (RFP) [37] establishes dense correspondences between gripper surface points and a canonical sphere for grasp synthesis. Inspired by these geometric representations, we extend the sphere correspondence idea beyond grasp representation and propose the Universal Hand Action Space (UHAS), where sphere deformations themselves define the manipulation action space.

3 Unified Hand Action Space

The Unified Hand Action Space (UHAS) is motivated by the central question of whether robotic hand control can be expressed through geometric deformations of a canonical sphere surface. A sphere provides a natural unified representation because it defines a continuous, closed, and topology-agnostic surface that can be consistently parameterized across robotic hands with different kinematic structures and numbers of fingers. Moreover, many grasping and dexterous manipulation behaviors can be interpreted as coordinated radial and angular interactions around an object-centered workspace, making the sphere a convenient geometric abstraction for hand control.

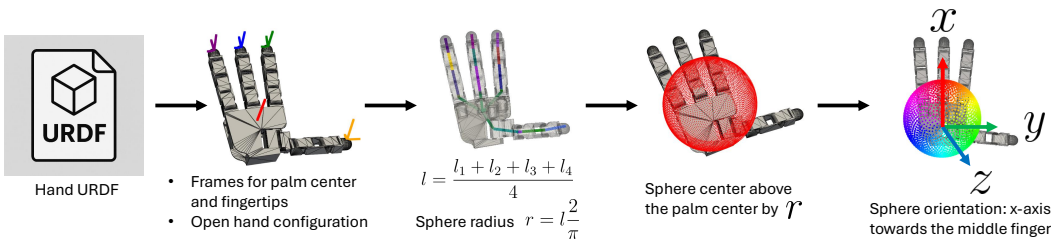


Figure 2: Illustration of the process of creating a sphere for a robotic hand given its URDF.

3.1 Automatic Sphere Creation for a Robotic Hand

Given a robotic hand URDF model, we automatically construct the canonical sphere through kinematic analysis and geometric normalization, as illustrated in Fig. 2. We first identify the palm and fingertip coordinate frames in an open-hand configuration, where the fingers are fully extended and the fingertip normals approximately align with the palm normal. We then compute the palm center by averaging the finger root positions and measure the average distance l from the palm center to the fingertips. The sphere radius is defined as $r = \frac{2l}{\pi}$, placing the sphere within the natural grasping workspace of the hand and approximately covering a 90° arc from the palm center to the fingertips.

Next, the sphere center is positioned along the outward palm normal at a distance r from the palm center, placing the sphere within the natural grasping workspace of the hand where the fingers tend to converge during grasping and dexterous manipulation. To obtain a canonical orientation shared across embodiments, the sphere coordinate frame is defined such that its positive z-axis aligns with the outward palm normal and its positive x-axis aligns with the middle finger direction. The y-axis is then uniquely determined by the resulting xz-plane according to the right-hand rule.

To remove embodiment-specific scale differences, all distances in the sphere coordinate frame are normalized by the hand-specific radius r. This normalization maps the canonical sphere to a *unit sphere*, producing a scale-invariant representation shared across robotic hands with different kinematic structures and numbers of fingers. The resulting normalized sphere coordinate system defines the Unified Hand Action Space (UHAS), enabling a unified geometric action representation for cross-embodiment policy learning and transfer. Although the UHAS operates on a normalized unit sphere, the corresponding hand-specific sphere parameters are preserved and later used by the Cascade Inverse Kinematics (CIK) algorithm to recover embodiment-specific hand configurations.

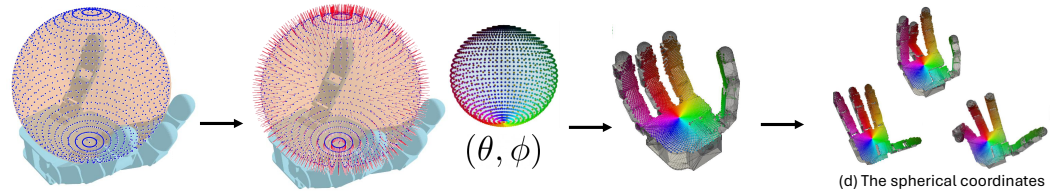


Figure 3: Construction of the unified hand surface correspondence.

3.2 Unified Hand Surface Correspondence

The key idea behind using the sphere for hand control is to establish dense geometric correspondences between the canonical sphere surface and the robotic hand. Once these correspondences are defined, deformations of the sphere can be directly transferred to the hand surface, enabling the sphere to serve as a unified action representation across embodiments.

As illustrated in Fig. 3, we first uniformly sample points on the sphere surface, as shown in Fig. 3(a). For each sampled point, we compute its spherical coordinates (θ, ϕ) together with its outward surface normal, as illustrated in Fig. 3(b), where θ denotes the azimuthal angle and ϕ denotes the polar angle. These spherical coordinates provide a canonical parameterization of the sphere surface independent of any specific robotic hand embodiment. We transfer the spherical coordinates to the robotic hand by projecting sampled sphere points onto nearby points on the interior hand surface, producing dense correspondences between the sphere and the palm and finger surfaces, as illustrated in Fig. 3(c). Although the 3D locations of the hand surface points vary under different hand configurations, their associated spherical coordinates remain unchanged, as shown in Fig. 3(d). This configuration-invariant parameterization enables different robotic hands to share a common spherical coordinate domain, allowing hand actions to be expressed as geometric deformations of the canonical sphere surface rather than embodiment-specific joint motions.

3.3 Sphere Deformation Action Space

Directly modeling point-wise deformations of the entire sphere surface is computationally impractical due to the high dimensionality of the resulting action space. Let (θ, ϕ, r) denote the spherical coordinates of a point on the normalized reference sphere, where θ is the azimuthal angle, ϕ is the polar angle, and $r = 1$ for the undeformed sphere. To obtain a compact action representation, we model sphere deformations using only $(\Delta\theta, \Delta r)$, corresponding to lateral angular displacements and radial expansions or contractions. This simplification is motivated by the construction of the UHAS, where the north pole of the sphere is rigidly anchored to the palm frame.

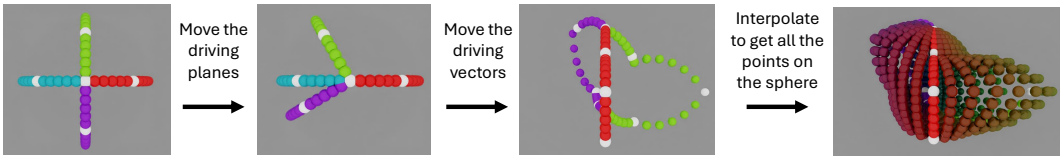


Figure 4: Sphere deformation parameterization in the Unified Hand Action Space (UHAS). (a) Initial configuration of four driving planes. (b) Rotating the driving planes controls the lateral deformation $\Delta\theta$. (c) Radial displacement of the driving vectors controls Δr. (d) The final deformed sphere reconstructed through interpolation.

We parameterize the sphere deformation using a sparse set of *control primitives*. Specifically, we introduce a finite number of *driving planes*, each defined as a plane passing through the sphere center at a fixed azimuthal angle θ_{plane} (Fig. 4). These driving planes control the lateral deformation component $\Delta\theta$. Within each driving plane, we further define a discrete set of control points at fixed

polar angles ϕ. The radial displacements of these control points, referred to as *driving vectors*, parameterize the radial deformation component Δr.

The complete deformation field over the sphere surface is reconstructed through interpolation. First, the $\Delta\theta$ deformation at every surface point is obtained by interpolating the values specified by neighboring driving planes. Next, the Δr deformation is computed through a two-dimensional interpolation of the driving-vector displacements in the (θ, ϕ) parameter space. Despite the compact parameterization, this representation can generate a rich set of sphere morphologies suitable for dexterous manipulation. Fig. 4 illustrates an example of a sphere deformation. *If we use five driving planes with two driving vectors per plane, it results in a 15-dimensional continuous action representation, i.e.,* $5 + 2 \times 5$. In practice, we define the driving planes such that they align with the hand fingers.

3.4 Cascade Inverse Kinematics

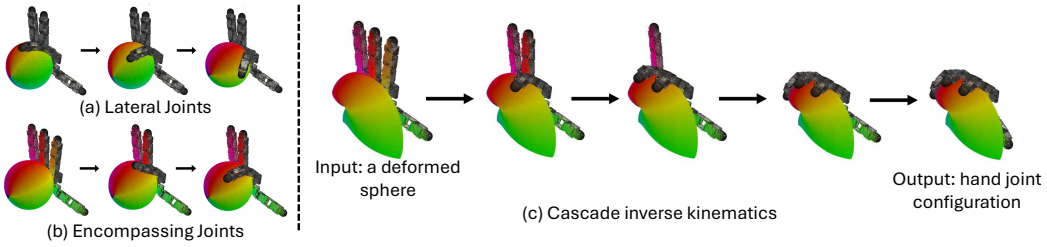


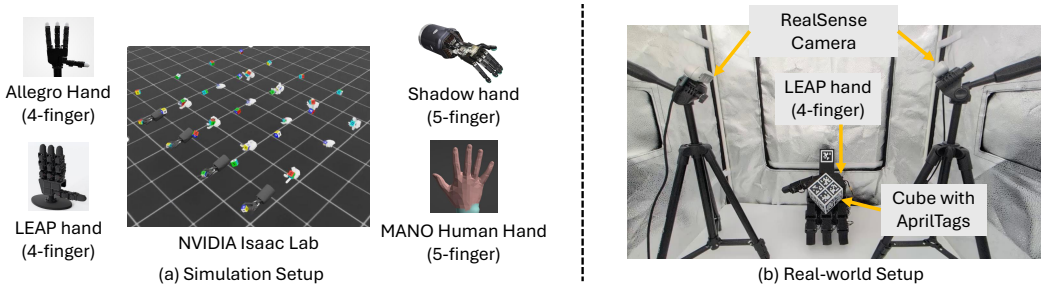
Figure 5: We classify hand joints into (a) lateral joints and (b) encompassing joints and; (c) Illustration of the cascade inverse kinematics algorithm on a deformed sphere.

We map deformed sphere surfaces to embodiment-specific hand joint configurations using a novel *Cascade Inverse Kinematics* (CIK) algorithm. Using the surface correspondences defined in Section 3.2, each hand surface point is associated with fixed spherical coordinates (θ, ϕ) on the canonical sphere. Given a deformed sphere, the corresponding target surface positions are obtained directly from the sphere geometry and used by the CIK algorithm to compute the hand joint configuration $\mathbf{q}$.

Joint Classification. Each hand joint is classified according to its dominant effect on the corresponding hand surface points in the sphere coordinate frame. As illustrated in Fig. 5(a), lateral joints primarily control the azimuthal angle θ of the fingertip and finger surface points, producing side-to-side finger motion. In contrast, encompassing joints in Fig. 5(b) mainly affect the radial distance r and polar angle ϕ, allowing the fingers to conform to the sphere surface. This decomposition enables the inverse kinematics problem to be solved efficiently in a cascaded manner.

Lateral Joint Mapping. To efficiently control lateral deformations, we precompute a lookup table by sweeping each lateral joint across its full range of motion while re-solving the encompassing joints on the undeformed reference sphere. For every sampled lateral-joint configuration, we record the resulting fingertip azimuthal angle θ. The lookup table provides a mapping between lateral joint configurations and θ displacements on the sphere surface. During inference, the target $\Delta\theta$ deformation is obtained directly from the displacement of the fingertip's corresponding sphere point computed in Section 3.2. The lateral joint adjustments $\Delta\mathbf{q}_{\text{lateral}}$ are then recovered through direct lookup in the precomputed table.

Encompassing Cascade. After the lateral adjustment stage, the encompassing joints are solved sequentially along the kinematic chain from the finger root to the fingertip. For each encompassing joint, we solve a one-dimensional inverse kinematics subproblem that places the downstream hand surface points onto their corresponding locations on the deformed sphere surface. This cascading procedure progressively conforms the finger geometry to the target sphere deformation while maintaining kinematic feasibility. The resulting joint configuration $\mathbf{q}$ is the final output of the CIK algorithm. The complete procedure is illustrated in Fig. 5(c). More details of the CIK algorithm are present in Appendix A.



(a) Simulation Setup (b) Real-world Setup

Figure 6: (a) Simulation setup with 4 hands (b) Our real-world setup of the LEAP hand

Sphere Controller. By combining the sphere deformation parameterization with the proposed Cascade Inverse Kinematics (CIK) algorithm, we obtain a *sphere controller* that maps sphere deformations in the UHAS to embodiment-specific hand joint configurations. During policy execution, the learned policy predicts sphere deformation parameters, which are converted into executable hand motions through CIK. Thanks to its lightweight sequential structure, CIK runs at up to **150 Hz**, enabling real-time control across diverse robotic hands.

4 In-hand Manipulation Experiments

4.1 Experimental Setup

Tasks. We evaluate our method on the task of in-hand cube reorientation in both simulation and the real world. i) For simulation, We use the Repose Cube simulation in NVIDIA Isaac Lab [42], and adapt it to use our sphere controller for four hands: Leap Hand [16] and Allegro Hand [17] (4 fingers each), and Shadow Hand [19] and a simulated MANO human hand model [18] (5 fingers each) as shown in Fig. 6(a). In each episode in simulation, the robotic hand is required to sequentially reach 10 target cube orientations. If the cube falls, the environment is reset and the remaining reorientation attempts continue until all 10 targets have been evaluated. ii) In the real world, we conduct experiments with a LEAP hand as shown in Fig. 6(b). For each trial in the real world, we run the policy until the cube falls. In both simulation and real-world experiments, each target orientation must be reached within 30 seconds. Otherwise, the attempt is considered a failure.

Evaluation Metrics. We report two primary evaluation metrics. The **Average Consecutive Reorientations** measures the average number of consecutive successful reorientations achieved before the first cube drop, with a maximum value of 10 in our simulation environment. The **Success Rate** measures the percentage of successful individual reorientation attempts, where a trial is considered successful if the cube reaches the target orientation without falling. All simulation results are evaluated using 1000 parallel environments. As a baseline, we compare against a policy trained to directly predict hand joint positions while using joint positions and velocities as the input state.

Manipulation Policies. We train in-hand manipulation policies using the RSL-RL implementation of PPO [20] with a lightweight actor-critic network. The policy outputs sphere deformations, where each finger is controlled by one driving plane with two driving vectors. To support both 4-finger and 5-finger hands within a unified action space, we add an extra driving plane at the ring-finger position for 4-finger embodiments. We adopt the same reward formulation as the original Isaac Lab [42] environment and use identical reward settings for all models and baselines. For observations, we use homogeneous observations expressed in the canonical sphere coordinate frame and normalized by the corresponding hand-specific sphere radius, enabling a shared observation space across robotic hands with different kinematic structures and numbers of fingers. Together with the unified action representation, *this allows a single policy to operate across all robotic hands used in our experiments.* Additional training details are provided in Appendix B.

Table 1: In-hand cube reorientation results. Metrics are reported as (Success Rate / Average Consecutive Reorientations). **Single-Hand**: trained only on the target hand. **Joint Control**: baseline joint-space controller trained on the target hand. **Multi-Hand**: trained on all four hands. **Zero-shot**: trained on all hands except the target hand.

Test Hand	Single-Hand	Joint Control	Multi-Hand	Zero-shot
Allegro	**99.1 / 9.6 ± 1.7**	98.5 / 9.2 ± 2.2	99.2 / 9.5 ± 1.9	95.3 / 7.7 ± 3.4
LEAP	**99.7 / 9.8 ± 1.1**	98.6 / 9.3 ± 1.2	99.1 / 9.5 ± 1.9	95.5 / 7.7 ± 3.5
Shadow	**99.3 / 9.6 ± 1.6**	98.0 / 9.1 ± 1.9	98.7 / 9.2 ± 2.3	85.7 / 4.4 ± 3.7
MANO	**99.8 / 9.9 ± 1.0**	99.6 / 9.8 ± 1.4	99.5 / 9.8 ± 1.2	98.1 / 8.9 ± 2.6

Table 2: Cross-morphology generalization across robotic hands with different numbers of fingers. Metrics are reported as (Success Rate / Average Consecutive Reorientations).

Test Hand	Train: Shadow + MANO (5-finger)	Train: Allegro + LEAP (4-finger)
Allegro (4F)	66.2 / 1.9 ± 1.9	**99.7 / 9.8 ± 1.2**
LEAP (4F)	80.8 / 3.7 ± 3.6	**99.8 / 9.9 ± 0.98**
Shadow (5F)	**98.6 / 9.3 ± 2.1**	83.2 / 4.0 ± 4.0
MANO (5F)	**99.7 / 9.8 ± 1.3**	95.0 / 7.6 ± 3.5

Table 3: Fast adaptation from MANO to unseen robotic hands. The policy is first trained on MANO and finetuned on the target hand for only 500 iterations. Metrics are Success Rate / Average Consecutive Reorientations.

Target Hand	Zero-shot	+500 Iter
Allegro	95.3 / 7.7 ± 3.4	**96.3 / 8.1 ± 3.3**
LEAP	95.5 / 7.7 ± 3.5	**96.2 / 8.0 ± 3.3**
Shadow	85.7 / 4.4 ± 3.7	**95.8 / 7.8 ± 3.5**

4.2 Simulation Results

Main Results. Table 1 evaluates the proposed sphere controller across four robotic hands on the in-hand cube reorientation task. We compare four training settings: *Single-Hand*, where a separate policy is trained for each hand individually; *Joint Control*, a baseline controller operating directly in joint space; *Multi-Hand*, where a single policy is jointly trained across all hands using the proposed UHAS representation; and *Zero-shot*, where the target hand is excluded during training and evaluated without finetuning.

The proposed UHAS representation achieves consistently strong performance across all hands and generally outperforms the joint-space baseline in both task success and long-horizon stability. Importantly, the Multi-Hand results show that a single shared policy can achieve performance comparable to embodiment-specific policies, while the Zero-shot results demonstrate meaningful transfer to previously unseen robotic hands despite substantial differences in kinematic structure and finger morphology. These results highlight the effectiveness of the proposed UHAS representation for cross-embodiment dexterous manipulation.

Cross-Morphology Generalization. Table 2 evaluates transfer across robotic hands with different numbers of fingers. We train one policy on the two 5-finger hands (MANO and Shadow) and another on the two 4-finger hands (Allegro and LEAP), then evaluate both policies on all four hands. During deployment, models are applied directly without additional processing, where 5-finger models use one driving plane with its vectors per finger, while 4-finger models duplicate the ring-finger plane and the corresponding observations. The results show strong in-distribution performance and meaningful transfer across hand morphologies. Although a performance gap remains compared to in-distribution performance, the transferred policies achieve substantial success on unseen hands, demonstrating that the proposed UHAS representation generalizes across different finger counts and kinematic structures.

Finetuning on Unseen Hands. Table 3 evaluates adaptation from the MANO hand to unseen robotic hands. Starting from a policy trained solely on MANO, we finetune the policy on each target hand for only 500 iterations, compared to approximately 4,500 iterations required for training from scratch. Despite this small finetuning budget, the success rate improves substantially across all target hands, recovering strong manipulation performance. These results demonstrate that the proposed UHAS representation enables both meaningful zero-shot transfer and rapid adaptation to new robotic hands.

Ablation Studies. We ablate the number of driving vectors per finger in the UHAS model by training policies with 1, 2, 3, and 4 vectors while keeping all other components fixed. All four hands are used for training on one NVIDIA A5000 GPU. The training time is measured as the number of iterations required to reach 90% of the maximum average consecutive reorientations (10). As shown in Table 4, 2-vectors achieves comparable performance to 3-vectors while requiring the shortest training time. In contrast, 1-vector provides insufficient control flexibility whose training time is longer than 2-vectors , while 4-vectors increase action-space complexity with limited performance gains. We therefore use the 2-vector configuration in our final model. Additional ablation studies are provided in Appendix E.

Table 4: Ablation on the number of driving vectors per driving plane.

# Driving Vectors	1	2	3	4
Success Rate	98.0	98.7	**99.5**	98.1
# Reorientations	8.8 ± 2.6	9.3 ± 2.0	$\mathbf{9.6 \pm 1.5}$	9.1 ± 2.4
Training Time (h)	5.3	**4.5**	6.5	5.5

4.3 Real-World Results with Sim-to-Real Transfer

Table 5: Real-world in-hand cube reorientation on the LEAP Hand over 10 independent trials. Entries report the number of consecutive successful reorientations before failure.

Method	1	2	3	4	5	6	7	8	9	10	MEAN
Baseline (Joint Control)	0	0	1	2	0	0	0	1	2	0	0.6
UHAS (Zero-Shot)	2	4	2	0	1	0	0	0	0	0	0.9
UHAS (Trained on Multi-Hand)	3	0	1	0	0	5	0	0	0	2	1.1
UHAS (Trained on LEAP Hand)	0	2	1	0	1	6	2	1	2	5	**2.0**

Figure 7: An example of a real-world run of our policy for in-hand cube reorientation.

We deploy our method on a physical LEAP Hand [16] along with a cube pose estimator based on AprilTags [43, 44] (Fig. 7). Details of our cube pose estimation algorithm are presented in Appendix C. To reduce the sim-to-real gap, we performed system identification on the entire hardware setup with details provided in Appendix D. We evaluate in-hand cube reorientation over 10 independent trials per method. In each trial, the policy runs continuously until the cube falls off the hand. Table 5 reports the number of consecutive successful reorientations achieved before failure in each trial, along with the mean across trials.

We observe a significant performance gap between simulation and the real world despite our efforts on system identification and domain randomization. However, all variants of our method outperform the joint-control baseline. Notably, even the zero-shot UHAS model achieved higher average consecutive reorientations than the baseline. We further observe that the model trained exclusively on the LEAP Hand achieves the best real-world performance (mean of 2.0). Interestingly, the model trained on multiple hands underperforms the single-hand LEAP model. We attribute this to the fact that single-hand training allows the policy to exploit the specific workspace and kinematics of the LEAP Hand, whereas multi-hand training forces the network to learn only movements that are feasible and safe across all hands. Given the relatively simple actor-critic architecture, this results in a more conservative policy when trained on multiple embodiments. Please see the supplementary video for the real-world deployment, and additional results in Appendix F.

5 Conclusion

In this work, we introduced the Unified Hand Action Space (UHAS), a sphere-based geometric action representation for dexterous manipulation across diverse robotic hands. By establishing dense correspondences between robotic hand surfaces and a canonical sphere representation, the proposed approach enables policies to operate in a shared action space independent of embodiment-specific joint parameterizations. We further proposed the Cascade Inverse Kinematics (CIK) algorithm to map sphere deformations to executable hand motions. Using reinforcement learning, we demonstrated dexterous in-hand cube reorientation across multiple robotic hands in both simulation and the real world, and showed meaningful cross-embodiment transfer through multi-hand, zero-shot, and finetuning experiments.

Limitations. Dexterous manipulation performance remains highly sensitive to low-level PD controller parameters, requiring extensive domain randomization for robust transfer across robotic hands. In addition, the in-hand cube reorientation task is sensitive to reward design and reinforcement learning hyperparameters. Finally, although UHAS enables transfer across diverse embodiments, performance degrades when transferring between substantially different hand morphologies, such as 4-finger and 5-finger hands.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under Grant Nos. 2346528 and 2520553, the NVIDIA Academic Grant Program Award, and a gift funding from XPeng.

References

[1] 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.

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

[3] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[4] 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.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

[6] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arxiv:2410.24164*, 2024.

[7] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling,

H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

[10] D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, R. Doshi, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

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

W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[12] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[13] R. Wen, G. Chen, Z. Cui, M. Du, Y. Gou, Z. Han, L. Huang, M. Lei, Y. Li, Z. Li, W. Liu, Y. Liu, X. Ma, H. Niu, Y. Ouyang, Z. Ren, H. Shi, W. Xu, H. Zhang, J. Zhang, X. Zhang, L. Zheng, W. Zhong, Y. Zhou, Z. Zhu, and H. Li. Gr-dexter technical report. *arXiv preprint arXiv:2512.24210*, 2025.

[14] Z. Zhang, J. Pang, Z. Yang, K. Li, M. Liao, S. Zhang, G. Chi, J. Guo, H. ang Gao, M. Shi, D. Ge, Y. Mu, J. Gu, R. Chen, H. Dong, H. Xu, L. Yi, Y. Zhu, H. Zhao, P. Wang, S. Zhang, G. Yao, J. Chen, H. Li, and H. Zhao. Dexora: Open-source vla for high-dof bimanual dexterity. *arXiv preprint arXiv:2605.18722*, 2026.

[15] R. Zheng, D. Niu, Y. Xie, J. Wang, M. Xu, Y. Jiang, F. Castañeda, F. Hu, Y. L. Tan, L. Fu, T. Darrell, F. Huang, Y. Zhu, D. Xu, and L. Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

[16] K. Shaw, A. Agarwal, and D. Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023.

[17] Wonik Robotics Co., Ltd. Allegro hand. https://www.allegrohand.com/. Accessed: 2025-08-08.

[18] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), Nov. 2017.

[19] The Shadow Robot Company. Shadow dexterous hand. https://www.shadowrobot.com/products/dexterous-hand/. Accessed: 2025-08-08.

[20] C. Schwarke, M. Mittal, N. Rudin, D. Hoeller, and M. Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

[21] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[22] Y. Chen, T. Wu, S. Wang, X. Feng, J. Jiang, Z. Lu, S. McAleer, H. Dong, S.-C. Zhu, and Y. Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

[23] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. V. Wyk, A. Zhurkevich, B. Sundaralingam, Y. Narang, J.-F. Lafleche, D. Fox, and G. State. 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.

[24] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, J. Fan, et al. Eureka: Human-level reward design via coding large language models. In *International conference on learning Representations*, volume 2024, pages 26516–26560, 2024.

[25] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel. Mujoco playground, 2025. URL https://arxiv.org/abs/2502.08844.

[26] K. Shaw, S. Bahl, and D. Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, pages 654–665. PMLR, 2023.

[27] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[28] 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. *arXiv preprint arXiv:2403.07788*, 2024.

[29] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.

[30] K. Li, P. Li, T. Liu, Y. Li, and S. Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6991–7003, 2025.

[31] T. Tao, M. K. Srirama, J. J. Liu, K. Shaw, and D. Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *arXiv preprint arXiv:2505.07813*, 2025.

[32] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[33] J. Zheng, J. Li, D. Liu, Y. Zheng, Z. Wang, Z. Ou, Y. Liu, J. Liu, Y.-Q. Zhang, and X. Zhan. Universal actions for enhanced embodied foundation models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22508–22519, 2025.

[34] G. Jiang, Y. Liang, J. Ye, J.-Y. Huang, C. Jing, R. Duan, P. Abbeel, X. Wang, and X. Zou. Cross-hand latent representation for vision-language-action models. *arXiv preprint arXiv:2603.10158*, 2026.

[35] Z. Wei, Y. Yao, and M. Ding. One hand to rule them all: Canonical representations for unified dexterous manipulation. *arXiv preprint arXiv:2602.16712*, 2026.

[36] Z. Wei, Z. Xu, J. Guo, Y. Hou, C. Gao, Z. Cai, J. Luo, and L. Shao. D (r, o) grasp: A unified representation of robot and object interaction for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2410.01702*, 2024.

[37] N. Khargonkar, L. F. Casas, , B. Prabhakaran, and Y. Xiang. Robotfingerprint: Unified gripper coordinate space for multi-gripper grasp synthesis. 2024.

[38] X. Fei, Z. Xu, H. Fang, T. Zhang, and L. Shao. T (r, o) grasp: Efficient graph diffusion of robot-object spatial transformation for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2510.12724*, 2025.

[39] Z. Wu, R. A. Potamias, X. Zhang, Z. Zhang, J. Deng, and S. Luo. Cedex: Cross-embodiment dexterous grasp generation at scale from human-like contact representations. *arXiv preprint arXiv:2509.24661*, 2025.

[40] Y. Wu, Y. Lin, W. Lao, Y. Lin, Y.-L. Wei, W.-S. Zheng, and A. Wu. Dexgrasp-zero: A morphology-aligned policy for zero-shot cross-embodiment dexterous grasping. *arXiv preprint arXiv:2603.16806*, 2026.

[41] X. He, A. Polavaram, Y. Cao, O. Deshmukh, T. Wang, X. Zhou, and K. Fang. Generate, transfer, adapt: Learning functional dexterous grasping from a single human demonstration. *arXiv preprint arXiv:2601.05243*, 2026.

[42] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Muñoz, X. Yao, R. Zurbrügg, N. Rudin, L. Wawrzyniak, M. Rakhsha, A. Denzler, E. Heiden, A. Borovicka, O. Ahmed, I. Akinola, A. Anwar, M. T. Carlson, J. Y. Feng, A. Garg, R. Gasoto, L. Gulich, Y. Guo, M. Gussert, A. Hansen, M. Kulkarni, C. Li, W. Liu, V. Makoviychuk, G. Malczyk, H. Mazhar, M. Moghani, A. Murali, M. Noseworthy, A. Poddubny, N. Ratliff, W. Rehberg, C. Schwarke, R. Singh, J. L. Smith, B. Tang, R. Thaker, M. Trepte, K. V. Wyk, F. Yu, A. Millane, V. Ramasamy, R. Steiner, S. Subramanian, C. Volk, C. Chen, N. Jawale, A. V. Kuruttukulam, M. A. Lin, A. Mandlekar, K. Patzwaldt, J. Welsh, H. Zhao, F. Anes, J.-F. Lafleche, N. Moënne-Loccoz, S. Park, R. Stepinski, D. V. Gelder, C. Amevor, J. Carius, J. Chang, A. H. Chen, P. de Heras Ciechomski, G. Daviet, M. Mohajerani, J. von Muralt, V. Reutskyy, M. Sauter, S. Schirm, E. L. Shi, P. Terdiman, K. Vilella, T. Widmer, G. Yeoman, T. Chen, S. Grizan, C. Li, L. Li, C. Smith, R. Wiltz, K. Alexis, Y. Chang, D. Chu, L. J. Fan, F. Farshidian, A. Handa, S. Huang, M. Hutter, Y. Narang, S. Pouya, S. Sheng, Y. Zhu, M. Macklin, A. Moravanszky, P. Reist, Y. Guo, D. Hoeller, and G. State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[43] AprilRobotics. Apriltag. https://github.com/AprilRobotics/apriltag.

[44] Y. Park and P. Agrawal. Aprilcube: 3d-printable fiducial targets for reliable 6-dof pose estimation, 2026. URL https://github.com/younghyopark/aprilcube.

A Cascade Inverse Kinematics

The Cascade Inverse Kinematics (CIK) algorithm maps a deformed canonical sphere to embodiment-specific joint configurations $\mathbf{q}$ for arbitrary robotic hands. Its objective is to position the hand surface points as close as possible to their corresponding locations on the input deformed sphere while respecting kinematic constraints. These correspondences are established during automatic sphere creation and surface projection (Section 3.2 and Fig. 3).

Traditional numerical inverse kinematics solvers are computationally expensive and unsuitable for real-time dexterous control. In contrast, CIK exploits the geometric structure of the Unified Hand Action Space together with a cascaded decomposition of joint responsibilities. In our real-world setup, the complete CIK pipeline runs at approximately **150 Hz**. In practice, CIK was never the runtime bottleneck. The dominant sources of latency were serial communication with the LEAP hand and AprilTag-based object pose estimation. We provide more details of the CIK algorithm below.

Joint Classification. We classify every joint of a robotic hand into one of two categories according to its dominant geometric effect on the fingertip position in the sphere coordinate frame. Classification is performed once per hand from its URDF in a base open-hand configuration. For each joint, we sweep it across its full range of motion while keeping all other joints fixed, and we record the resulting changes in the fingertip's spherical coordinates (θ, ϕ, r) via forward kinematics. In cases where a joint's rotation axis points toward the fingertip, we partially flex the remaining joints of the finger and repeat the sweep to determine its dominant effect. Joints are then categorized as follows:

- **Lateral joints** are those that predominantly affect the azimuthal angle θ of the fingertip, producing side-to-side finger motion.
- **Encompassing joints** are those that most strongly influence the radial distance r and polar angle ϕ of the fingertip, allowing the finger to conform to the sphere surface.

This joint classification procedure is illustrated in Fig. 8, where a joint is automatically classified according to the effect of the joint change on the position of the fingertip. Because the fingers are kinematically independent in the UHAS formulation, lateral and encompassing joints are solved sequentially but *independently for each finger*.

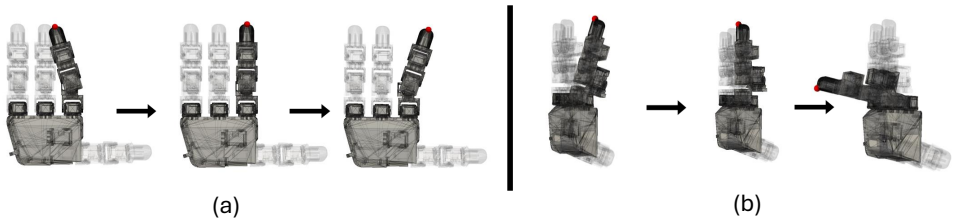


Figure 8: Illustration of the joint classification procedure in the Cascade Inverse Kinematics (CIK) algorithm. (a) Classification of lateral joint based on their effect on the fingertip azimuthal angle θ. (b) Classification of encompassing joints based on their effect on the fingertip radial distance r and polar angle ϕ.

Lateral Joint Mapping via Lookup Table. To enable fast lateral finger control, we precompute a lookup table for every lateral joint with the following steps:

1. Sweep the lateral joint across its full range of motion in uniform steps.
2. For each sampled value, fix the lateral joint and solve the encompassing joints on the *undeformed* reference sphere.

3. Record the resulting azimuthal angle $\theta_{\text{fingertip}}$ of the corresponding fingertip (surface point) on the canonical sphere frame.
4. Store the mapping: lateral joint value $q_{\text{lateral}} \mapsto \theta_{\text{fingertip}}$.

During inference, the policy outputs a lateral deformation $\Delta\theta$ for the driving plane aligned with the finger. We compute the fingertip target angle

$$\theta_{\text{fingertip}} = \theta_{\text{initial}} + \Delta\theta,$$

where θ_{initial} corresponds to the neutral (middle-of-range) pose of the finger. The precomputed lookup table directly yields the target lateral joint value q_{lateral}. The table is built offline once per hand and provides constant-time lateral solving at runtime.

Encompassing Joint Cascade. After the lateral joints have been set, we solve the encompassing joints sequentially in proximal-to-distal order along each finger's kinematic chain. For each encompassing joint i, we first use forward kinematics with the already computed parent joint values to transform the target sphere surface points associated with joint i and all its descendant links into the local coordinate frame of joint i. We then directly compute the joint angle that places both the current joint's surface correspondences and those of its children onto the deformed sphere surface, as illustrated in Fig. 5. Because each joint is solved exactly once in a single forward pass with no iterative numerical optimization, the cascade remains extremely lightweight. This cascaded procedure progressively conforms the finger geometry to the target sphere deformation while preserving kinematic feasibility. The final joint configuration $\mathbf{q}$ is the output of the complete CIK algorithm.

Sphere Controller. By combining the compact sphere-deformation action space with CIK we obtain the complete *Sphere Controller*. At every policy timestep the actor predicts driving-plane rotations ($\Delta\theta$) and driving-vector displacements (Δr). These parameters define a deformed sphere that is passed to CIK, which returns embodiment-specific joint positions $\mathbf{q}$ for the low-level controller. The lightweight sequential structure of CIK enables real-time execution across diverse robotic hands while remaining fully decoupled from any particular morphology. This design is central to the strong zero-shot transfer and rapid finetuning performance reported in Section 4.

B Policy Details

We provide implementation details of the reinforcement learning policies trained using the Unified Hand Action Space (UHAS). We describe the homogeneous observation space and the sphere-based action parameterization that enable effective cross-embodiment learning as well as training hyperparameters, domain randomization, and reward details in Appendix B.1.

A key requirement for training a single policy across robotic hands with different kinematic structures is the normalization of both *observation and action spaces*. Standard observation formulations for in-hand manipulation include goal rotation, object pose, object linear and angular velocities, the quaternion difference to the target orientation, joint positions and velocities, and previous actions. However, raw joint values cannot be used directly because their semantic meaning and numerical ranges vary significantly across embodiments, hindering cross-embodiment transfer. We therefore represent hand configuration through points defined via surface correspondences on the finger chains (Section 3.2).

Sphere Observation Space. We construct homogeneous observations from points sampled along each finger rather than from raw joint values. For every finger, we first discretize the kinematic chain into 7 equally spaced points from root to fingertip. The parent joint of each sampled point is determined by its closest surface correspondence on the canonical sphere (Section 3.2). Positions of these points under the current joint configuration are obtained via forward kinematics. Linear and angular velocities of the points are computed using the corresponding Jacobians evaluated at the current joint configuration and joint velocities.

For all policies presented in the main paper we retain only two points per finger: the middle point and the fingertip. This choice provides a compact yet informative representation of finger configuration while remaining consistent across hands with different numbers of fingers. For 4-finger hands we duplicate the ring-finger observations to preserve dimensional consistency with 5-finger embodiments. All point positions and velocities are expressed in the canonical sphere coordinate frame (Fig. 2) and normalized by the hand-specific sphere radius r. An illustration of the homogeneous observation points across different hands is shown in Fig. 9. A detailed ablation on the number of observation points is provided in Appendix E.

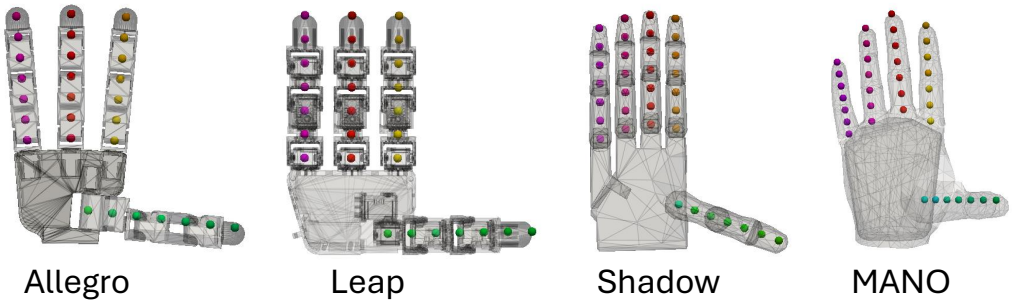


Figure 9: Homogeneous observations across different robotic hands.

Sphere Action Space. Actions are defined in the sphere deformation space introduced in Section 3.3. Each action consists of lateral deformations $\Delta\theta$ applied to a set of driving planes and radial deformations Δr applied to driving vectors within those planes. We attach one driving plane per fingertip and align its initial azimuthal angle with the corresponding fingertip θ on the canonical sphere. To enable a unified policy across both 4-finger and 5-finger hands while preserving independent actuation of the fifth finger, we introduce an additional driving plane at the ring-finger azimuthal position for all 4-finger embodiments. When computing deformations in the ring finger, the $\Delta\theta$ and Δr actions of the duplicated plane are averaged before interpolation.

Lateral actions are bounded to the extremal values stored in the precomputed lookup tables of each hand. We employ two driving vectors per plane, positioned at equally spaced polar angles $\phi = 60°$ and $\phi = 120°$, with radial displacements defined over the interval $[-2, 2]$. Prior to invoking the Cascade Inverse Kinematics (CIK) algorithm, we discard all sphere points whose radial coordinate r is negative. If an encompassing joint has no reachable points remaining after this filtering step, the joint is commanded to its fully closed configuration. This design choice was made after observing that permitting negative radial deformations substantially improves performance on the highly dynamic cube reorientation task, as it allows the policy to close fingers rapidly during aggressive reposing maneuvers. An ablation study on the number of driving vectors per plane is provided in Appendix E.

B.1 Training

Hyperparameters. All models were trained on a single NVIDIA A5000 GPU using the RSL-RL implementation of Proximal Policy Optimization (PPO) [20]. Training is performed in the custom NVIDIA Isaac Lab Cube Reposing environment [42] running on Isaac Sim 4.5.0 with the PhysX physics simulator. The complete set of PPO training hyperparameters is summarized in Table 6.

Reward Function. We adopt the reward formulation from the original NVIDIA Isaac Lab Reposing Cube environment. To discourage embodiment-specific exploitation—such as policies that rely predominantly on lateral joint motion while underutilizing encompassing joints—we augment the original reward with two additional regularization terms. These terms penalize deviations of the lateral and encompassing joint positions from a reference joint configuration. Let d denote the

Table 6: PPO training hyperparameters.

Hyperparameter	**Value**
Runner Configuration	
Num. steps per environment	16
Empirical normalization	True
Policy Network (Actor & Critic)	
Hidden layer dimensions	$[512, 512, 256, 128]$
Activation function	ELU
Initial action noise std.	1.0
PPO Algorithm	
Learning rate	5.0×10^{-4}
Learning rate schedule	adaptive
Clip parameter	0.2
Entropy coefficient	0.005
Num. learning epochs	5
Num. mini-batches	4
Value loss coefficient	1.0
Use clipped value loss	True
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Desired KL divergence	0.016
Max. gradient norm	1.0

object-to-goal distance and r_{rot} the orientation alignment reward. Let p_{lat} and p_{rad} denote the lateral and encompassing joint position penalties, respectively. The scalar reward at each timestep is then computed as

$$r = w_d\, d + w_r\, r_{\text{rot}} + w_{\text{lat}}\, p_{\text{lat}} + w_{\text{rad}}\, p_{\text{rad}} + b_{\text{success}} + p_{\text{fall}}, \tag{1}$$

where b_{success} is a large positive bonus awarded upon reaching the goal within tolerance of 0.1 radians, and p_{fall} is a negative penalty applied when the object falls. The reward scales used during training are reported in Table 7.

Table 7: Reward scales used during training.

Reward Term	**Scale**
Object-to-goal distance (w_d)	-10.0
Orientation alignment (w_r)	1.0
Reach goal bonus (b_{success})	250
Fall penalty (p_{fall})	-100
Lateral joint position penalty (w_{lat})	-0.016
Encompassing joint position penalty (w_{rad})	-0.004

Domain Randomization. We utilize domain randomization during policy training. These randomization strategies proved beneficial for improving cross-embodiment generalization. Randomizing the hand base inclination discouraged policies from relying on specific cube-palm interaction dynamics, such as assuming the cube would slide or settle into particular regions of the palm. Object scale randomization helped mitigate finger entrapment issues arising from differences in finger geometry across embodiments. Additionally, randomizing the driving vector azimuthal angle θ, joint effort limits, and PD gains altered the in-distribution dynamics of each hand which improved generalization, particularly across certain hands. An ablation study examining the contribution of these randomization components is provided in Appendix E. Table 8 shows the details of the domain randomization used for training. Note that the domain randomization strategy used for real-world deployment differed from the one described in this section. Details of real-world deployment are provided in Appendix D.

Table 8: Domain randomization ranges applied during training.

Parameter	Randomization Range
Object Properties	
Object scale	$[0.9, 1.1]\times$ nominal
Object mass	$[0.8, 1.2]\times$ nominal
Object static friction	$[0.2, 0.3]$
Object dynamic friction	$[0.15, 0.25]$
Robot Properties	
Robot mass	$[0.9, 1.1]\times$ nominal
Robot static / dynamic friction	$[0.75, 1.0]$
Joint friction	$[0.9, 1.1]\times$ nominal
Joint armature	$[1.00, 1.05]\times$ nominal
Joint effort limits	$[0.9, 1.1]\times$ nominal
Joint stiffness	$[0.75, 1.25]\times$ nominal
Joint damping	$[0.75, 1.25]\times$ nominal
Hand Pose & Action Space	
Hand base inclination	$15^\circ \pm 5^\circ$
Driving vector azimuthal angle (ϕ)	$\pm 15^\circ$

C Cube Pose Estimation

We track a 3D-printed cube based on an open source design [44]. Each of its six faces carries four distinct `tag36h11` AprilTags [43] (24 tags total). One or two Intel RealSense cameras each stream a rectified infrared image at 848×480 pixels and 60 Hz. A second camera viewpoint increases the number of visible tags on the cube that are not occluded by the hand. We also placed the hand in a well-lit location allowing for shorter exposure time to reduce motion blur during fast reorientation movements.

Each visible tag is localized independently via perspective-n-point pose estimation using its four corners and the camera intrinsics. Detected tags are only published if the Hamming decoded value is zero and the decision margin is at least 50. A separate standalone tag is placed at a known location on the base mount and serves as a common reference frame.

In the dual-camera configuration, messages timestamped within 100 ms of each other are paired. The transformation from the secondary to the primary camera frame is estimated from tags visible in both camera views. In the case of duplicate tags which both cameras see, the tag with the higher detection margin will be kept. Given the final set of visible tags, the cube pose is calculated by chaining the camera-to-tag pose with the calibrated cube-to-tag transform. Estimates whose translation deviates by more than 10 mm from the median or whose orientation deviates by more than 0.075 rad from a consensus quaternion are dropped. The remaining estimates are fused by their mean position and hemisphere-aligned quaternion averaging. The cube pose is published only when at least three separate tags are visible.

D System Identification

To enable reliable sim-to-real transfer, we performed system identification on the complete real-world hardware setup, including the LEAP Hand, its actuators, and the communication interface. Due to the limited serial communication bandwidth of the LEAP Hand, we operate the policy at a control frequency of 20 Hz during training and real-world deployment.

We model the servo control law of the LEAP Hand as $\text{current}(t) = K_p \Delta\theta(t) - K_d \Delta\dot{\theta}(t)$, where $\Delta\theta(t)$ denotes the joint position error. Directly mapping this current-based formulation to the implicit PD torque actuators available in simulation did not reproduce the motion observed on the physical robot. To resolve this mismatch, we performed system identification on the LEAP Hand by commanding random target positions while varying the commanded K_p and K_d gains. From the

recorded joint position, velocity, and current trajectories, we solved for the effective gains that best matched the recorded data. The identified values were subsequently used when training policies intended for real-world deployment.

The system identification revealed that the LEAP Hand motors behave as a nearly undamped system, with damping having a negligible effect on the observed dynamics. Furthermore, the effective proportional gain K_p differed from the manufacturer specifications: approximately 0.0786 Nm/rad per 100 motor units. The effective derivative gain K_d was found to be approximately 0.0014 Nm/(rad/s) per 100 motor units. We attribute these discrepancies primarily to the current-based position control mode employed by the motors.

By monitoring the motor state during operation, we observed that the LEAP Hand reliably enforces its internal velocity limits. Accordingly, we introduced velocity limits in simulation when training models intended for real-world deployment. These limits were themselves randomized during training to improve robustness.

D.1 Training for Real-World Deployment

For models deployed on the physical robot, we increased the range of domain randomization compared to simulation-only training. In addition to the randomization parameters described in Table 8, we randomized the velocity limits applied in simulation. We also adopted an asymmetric actor-critic architecture: the actor receives only positional information of the object and hand joints, while the critic has access to the full state, including linear and angular velocities of the object and hand. This design encourages the actor to learn policies that rely primarily on position feedback, which is more reliable under real-world sensing and communication noise.

D.2 Deployment on the Physical LEAP Hand

During real-world operation, we observed that serial communication with the LEAP Hand was unreliable, with frequent missed reads and writes. To mitigate this, we implemented additional software-level communication management to improve reliability. Despite these measures, certain motor state readings consistently failed through the manufacturer API.

We further limited the range of several joints on the physical hand. This was necessary because the joints exhibited overshoot and because the structural components of the LEAP Hand are relatively deformable under load. We also imposed a lower velocity limit than the hardware maximum to reduce overshooting observed during fast motions.

Finally, we observed that the fingers of the LEAP Hand gradually became loose after repeated use. To improve mechanical stability, we designed and 3D-printed a reinforced base that secures the root joints of the index, middle, and ring fingers using additional screws. We also observed significant deformation at the thumb root joint and therefore designed a custom attachment for it. Both the reinforced finger base and the thumb attachment will be released publicly alongside the paper.

E Additional Ablation Studies

We present additional ablation studies on two key design choices in our method: the number of homogeneous observation points per finger and the number of driving planes in the Unified Hand Action Space. In addition, we present an ablation study on domain randomization.

Ablation on Observation Points. We ablate the number of homogeneous observation points per finger. Fig. 9 illustrates the full set of candidate points distributed along each finger. From these points, we select different subsets as observations: one point at the fingertip, two points consisting of the finger midpoint and fingertip, or three and four equally spaced points along the finger. All models were trained amd tested on the four hands using one NVIDIA A5000 GPU while keeping all other components fixed. As reported in Table 9, increasing the number of observation points

provides only marginal improvements in success rate and average consecutive reorientations. These gains become negligible when domain randomization is applied, while training time and inference cost increase. We therefore use two observation points per finger (midpoint and fingertip) in our final models, as this configuration achieves a favorable trade-off between performance and computational efficiency. Additional details regarding the construction of the homogeneous observations are provided in Appendix B.

Table 9: Ablation on the number of homogeneous observation points per finger.

# Observation Points	**1**	**2**	**3**	**4**
Success Rate	98.8	98.7	99.0	**99.1**
# Reorientations	9.3 ± 2.1	9.3 ± 2.0	9.4 ± 1.8	$\mathbf{9.5 \pm 1.8}$
Training Time (h)	4.9	**4.5**	4.8	4.7

Ablation on Driving Planes. Additionally, we ablate the number of driving planes in the Unified Hand Action Space (UHAS). For the non-zero-shot models, we train and evaluate each hand using either the standard 5-plane UHAS or a 4-plane variant. The 4-plane version is constructed identically to our merging procedure for 4-finger hands: we compute the average deformation of the ring and pinky planes and then interpolate the canonical sphere points as usual. For the zero-shot models, we train on the other three hands and deploy using the 4-plane UHAS (again merging the ring and pinky planes). Consequently, the 4-plane zero-shot policies move the ring and pinky fingers in a coupled manner.

As shown in Table 10, using 4 or 5 driving planes yields very similar performance. Interestingly, the Shadow hand even achieves slightly better results with 4 planes than with 5. We attribute this to the fact that the pinky finger is rarely used to solve the cube reposing task; therefore, removing its dedicated driving plane has a negligible impact on task performance.

We attempted to train models using only 3 driving planes. However, these models were unable to reliably solve the reposing task, and training failed to converge to meaningful policies. This indicates that a minimum level of action-space expressiveness is required for stable learning on this task.

It is important to note that the small difference observed between 4 and 5 planes may be specific to this reposing task, the particular reward formulation and the reference cube position used. These factors had a significant influence on hand behavior during training. Therefore, the negligible impact of removing the fifth plane should not be assumed to hold for other manipulation tasks or reward designs.

Table 10: Ablation on the number of driving planes.

# Driving Planes	**4**	**5**
Shadow	$\mathbf{99.5 \,/\, 9.7 \pm 1.5}$	$99.3 \,/\, 9.6 \pm 1.6$
Shadow (Zero-shot)	$\mathbf{86.9 \,/\, 4.8 \pm 3.8}$	$85.7 \,/\, 4.4 \pm 3.7$
MANO	$99.6 \,/\, 9.8 \pm 1.3$	$\mathbf{99.8 \,/\, 9.9 \pm 1.0}$
MANO (Zero-shot)	$98.0 \,/\, 8.7 \pm 2.9$	$\mathbf{98.1 \,/\, 8.9 \pm 2.6}$

Ablation on Domain Randomization. We evaluate the impact of domain randomization on zero-shot cross-embodiment generalization. For each hand, we train models on the other three hands and evaluate zero-shot transfer performance. We compare models trained with and without randomization of PD gains, joint effort limits, object scale, and driving vector polar angle ϕ. Table 11 reports the results across all four hands. Randomization during training consistently improves zero-shot transfer, particularly for hands with larger morphological differences from the training set.

Table 11: Zero-shot performance with and without domain randomization (PD gains, effort, object scale, and vector ϕ). Results reported as Success Rate / Average Consecutive Reorientations $\pm$ std.

Test Hand	Without Randomization	With Randomization
Allegro	**96.7 / 8.2 $\pm$ 3.1**	95.3 / 7.7 $\pm$ 3.4
LEAP	95.3 / 7.6 $\pm$ 3.4	**95.5 / 7.7 $\pm$ 3.5**
Shadow	80.1 / 3.6 $\pm$ 3.6	**85.7 / 4.4 $\pm$ 3.7**
MANO	97.1 / 8.4 $\pm$ 2.9	**98.1 / 8.9 $\pm$ 2.6**

F Additional Experimental Results

We present additional experimental results that complement the main paper. We first analyze one-to-many zero-shot generalization by training policies on a single source hand and evaluating them on all other target hands. We then report real-world results on the Allegro Hand [17].

F.1 One-to-Many Generalization in Simulation

We train a policy on a single source hand and evaluate its zero-shot performance on all other target hands. This setting reveals how well a policy learned on one embodiment can transfer without any finetuning or exposure to other hands during training. Table 12 summarizes the results.

Table 12: One-to-Many zero-shot generalization. Rows indicate the source hand used for training; columns indicate the target hand used for evaluation.

Source \ Target	Allegro	LEAP	Shadow	MANO
Allegro	**99.1 / 9.6 $\pm$ 1.7**	55.4 / 1.1 $\pm$ 1.5	8.7 / 0.1 $\pm$ 0.3	17.5 / 0.0 $\pm$ 0.13
LEAP	95.3 / 7.8 $\pm$ 3.3	**99.7 / 9.8 $\pm$ 1.1**	65.8 / 1.8 $\pm$ 1.9	87.0 / 4.7 $\pm$ 3.7
Shadow	35.6 / 0.4 $\pm$ 0.7	59.4 / 1.6 $\pm$ 2.2	**99.3 / 9.6 $\pm$ 1.6**	97.6 / 8.7 $\pm$ 2.7
MANO	33.0 / 0.4 $\pm$ 0.68	31.0 / 0.4 $\pm$ 0.67	36.2 / 0.5 $\pm$ 0.9	**99.8 / 9.9 $\pm$ 1.0**

The results highlight two important phenomena. First, policies trained on a single hand frequently develop *exploitative behaviors* that are highly specific to that embodiment. For example, the Allegro-trained policy relies heavily on the hand's distinctive lateral joints to perform cube rotations. Because Shadow and MANO lack equivalent lateral actuation, this policy fails almost completely on those hands. Interestingly, it retains partial transfer to the LEAP Hand, likely because the LEAP Hand's lateral joints can produce somewhat similar motion when the fingers are flexed.

Second, hands with larger joint ranges of motion and workspace tend to generalize better to other embodiments. The LEAP Hand, which possesses the largest range of motion among the four, achieves the strongest overall zero-shot performance when used as the source. Similarly, the Shadow Hand generalizes reasonably well to MANO, but the reverse direction (MANO $\rightarrow$ Shadow) performs significantly worse, despite both hands having five fingers. This asymmetry suggests that greater kinematic flexibility during training enables the policy to learn more transferable behaviors, whereas policies trained on more constrained hands tend to overfit to their specific joint limits and motion patterns.

These findings raise an interesting question for future work: whether explicitly limiting the range of motion or action space during training on high-degree-of-freedom hands could improve robustness, or whether the current asymmetry is inherent to cross-embodiment transfer.

F.2 Real-World Results on the Allegro Hand

We deploy our method on a physical Allegro Hand [17] along with a cube pose estimator based on AprilTags. The real-world setup is shown in Fig. 10. We evaluate in-hand cube reorientation over 10 independent trials per method. In each trial, the policy runs continuously until the cube falls off the hand. Table 13 reports the number of consecutive successful reorientations achieved before failure across trials, along with the mean.

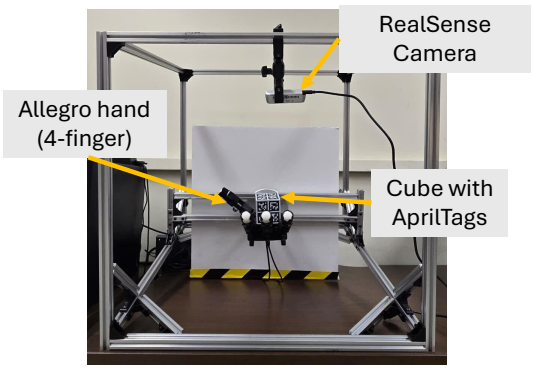


Figure 10: Illustration of our real-world setup for the Allegro hand

Table 13: Real-world in-hand cube reorientation on the Allegro Hand over 10 independent trials. Entries report the number of consecutive successful reorientations before failure.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**MEAN**
UHAS (Zero-Shot)	0	1	1	1	0	1	0	0	2	2	0.8
UHAS (Trained on Multi-Hand)	3	0	8	1	2	1	1	2	1	2	**2.1**
UHAS (Trained on Allegro Hand)	4	0	2	2	2	3	4	1	0	3	**2.1**

The Allegro-trained model achieves the highest real-world performance with a mean of 2.1 consecutive reorientations, tied with the multi-hand trained model. The zero-shot policy, however, performs substantially worse (mean 0.8), revealing a notable sim-to-real gap for cross-embodiment transfer, consistent with our LEAP Hand results. We also observe high variance across trials, with some runs reaching 8 reorientations while others fail immediately.