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]

I. Introduction

Robot manipulation has made rapid progress in recent years due to advances in imitation learning [42, 9], reinforcement learning [17, 29], and vision-language-action (VLA) models [20, 3, 4]. 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 [6, 5, 13, 10, 19]. 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 [35, 41, 44], 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.

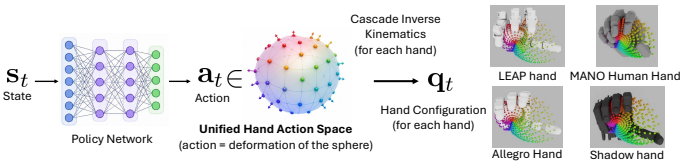

Fig. 1. In our unified hand action space, an action is represented as the deformation of a canonical sphere. A deformed sphere is mapped to hand configurations of various embodiments (LEAP [27], Allegro [36], MANO Human [25] and Shadow [31]).

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 [26], we train manipulation policies directly in the proposed action space for dexterous in-hand manipulation. Fig. 1 illustrates our framework. 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:

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

- 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.

II. 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 dexterous manipulation skills through large-scale simulation training and domain randomization [1, 7, 14, 22, 40]. More recent efforts have explored learning dexterous behaviors from demonstrations and large-scale video datasets [28, 8, 32, 39, 21, 30]. 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 [5], Octo [13], and OpenVLA [20] 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 [11], Universal Actions [43], XL-VLA [16], and One Hand to Rule Them All [34]. 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 [33, 18, 12, 38, 37, 15]. For example, D(R,O) Grasp [33] proposes a unified representation of robot-object interactions for cross-embodiment dexterous grasping, while RobotFingerPrint (RFP) [18] 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.

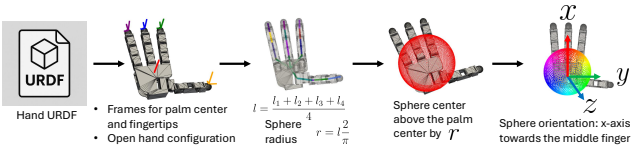

Fig. 2. Illustration of creating a sphere for a robotic hand given its URDF.

III. 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 parameterized across robotic hands.

A. 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).

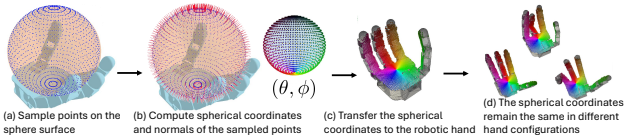

Fig. 3. Construction of the unified hand surface correspondence.

B. 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.

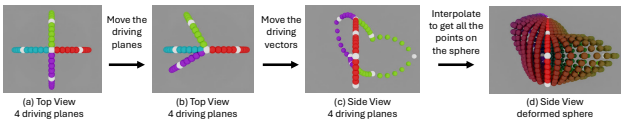

Fig. 4. Sphere deformation parameterization in the Unified Hand Action Space (UHAS). (a) Initial configuration of four driving planes. (b) Rotating the driving planes controls the lateral deformation $\Delta\theta$. (c) Radial displacement of the driving vectors controls Δr. (d) The final deformed sphere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

C. Sphere Deformation Action Space

Directly deforming the entire sphere surface yields a prohibitively high-dimensional action space. Let (θ, ϕ, r) denote spherical coordinates, where θ is the azimuthal angle, ϕ is the polar angle, and $r = 1$ on the reference sphere. We therefore parameterize deformations using only $(\Delta\theta, \Delta r)$, corresponding to lateral and radial displacements. This simplification is enabled by anchoring the sphere's north pole to the palm frame.

To obtain a compact action representation, we define a sparse set of *driving planes* passing through the sphere center at fixed azimuthal angles θ_{plane} (Fig. 4). Driving planes parameterize lateral deformations $\Delta\theta$. Within each plane, a set of *driving vectors* at fixed polar angles ϕ parameterize radial deformations Δr.

The full deformation field is reconstructed by interpolation. We first interpolate $\Delta\theta$ between neighboring driving planes, then interpolate Δr in the (θ, ϕ) domain using the driving vectors. Despite its compactness, this representation produces a rich set of sphere deformations suitable for dexterous manipulation. For example, five driving planes with two driving vectors per plane yield a 15-dimensional continuous action space $(5 + 2 \times 5)$.

D. Cascade Inverse Kinematics

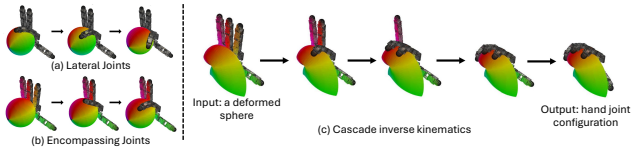

Fig. 5. We classify hand joints into (a) lateral joints and (b) encompassing joints and; (c) Illustration of the cascade inverse kinematics algorithm on a deformed sphere.

We map deformed sphere surfaces to embodiment-specific joint configurations using a novel *Cascade Inverse Kinematics* (CIK) algorithm. Through the surface correspondences in Section III-B, each hand surface point is associated with fixed spherical coordinates (θ, ϕ) on the canonical sphere. Given a deformed sphere, CIK computes the joint configuration $\mathbf{q}$ that places the corresponding hand surface points onto the target sphere geometry.

a) Joint Classification: As illustrated in Fig. 5(a,b), hand joints are classified by their dominant effect on sphere coordinates. Lateral joints primarily affect the azimuthal angle θ, producing side-to-side finger motion, while encompassing joints mainly affect the radial distance r and polar angle ϕ, allowing the finger to conform to the sphere surface. This decomposition enables efficient cascaded inverse kinematics.

b) Lateral Joint Mapping: For each lateral joint, we precompute a lookup table by sweeping its range of motion and recording the resulting fingertip azimuthal angle θ. During inference, the policy outputs a target deformation $\Delta\theta$, and the corresponding lateral joint adjustment $\Delta\mathbf{q}_{\text{lateral}}$ is recovered directly from the lookup table.

c) Encompassing Cascade: After applying lateral adjustments, encompassing joints are solved sequentially from the finger root to the fingertip. Each joint solves a one-dimensional inverse kinematics problem that places downstream surface points onto the deformed sphere. This cascade progressively conforms the finger geometry to the target deformation while maintaining kinematic feasibility. Additional details are provided in Appendix A.

d) Sphere Controller: Combining sphere deformation with CIK yields a *sphere controller* that maps UHAS actions to embodiment-specific joint configurations. During execution, the policy predicts sphere deformations that are converted into joint commands through CIK. The lightweight implementation runs at up to **150 Hz**, enabling real-time control across diverse robotic hands.

IV. In-hand Manipulation Experiments

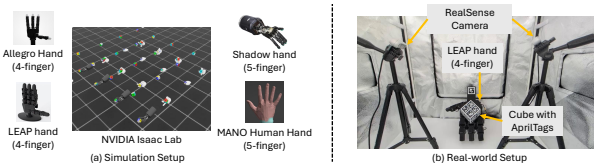

Fig. 6. (a) Simulation setup with 4 hands (b) Our real-world setup of the LEAP hand

TABLE I

IN-HAND CUBE REORIENTATION RESULTS. METRICS ARE REPORTED AS (SUCCESS RATE / AVERAGE CONSECUTIVE REORIENTATIONS).

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 II

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

A. Experimental Setup

a) Tasks: We evaluate in-hand cube reorientation in simulation and the real world. In simulation, we adapt the NVIDIA Isaac Lab Repose Cube task [23] to use our sphere controller on four hands: LEAP [27], Allegro [36], Shadow [31], and a simulated MANO human hand [25] (Fig. 6(a)). Each episode requires the hand to sequentially reach 10 target cube orientations. In the real world, we deploy on a LEAP Hand (Fig. 6(b)) and run each trial until the cube falls. A reorientation attempt is considered successful if the target orientation is reached within 30 seconds.

b) Evaluation Metrics: We report **Success Rate**, the percentage of successful reorientation attempts, and **Average Consecutive Reorientations**, the average number of successful reorientations before the first cube drop (maximum 10 in simulation). We compare against a joint-space policy that directly predicts joint positions from joint states.

c) Manipulation Policies: Policies are trained using PPO [26]. Actions are represented in the proposed Unified Hand Action Space, while observations consist of homogeneous hand representations expressed in the canonical sphere frame. Together, these provide a shared observation and action space across robotic hands with different kinematic structures and numbers of fingers, enabling a single policy to operate across all embodiments. Additional implementation details are provided in Appendix B.

B. Simulation Results

a) Main Results: Table I compares four training settings on in-hand cube reorientation: *Single-Hand*, where a separate policy is trained for each hand; *Joint Control*, a joint-space baseline; *Multi-Hand*, where a single policy is trained across all hands using UHAS; and *Zero-shot*, where the target hand is excluded from training. UHAS achieves consistently strong performance across all hands and generally outperforms the joint-space baseline. Notably, a single Multi-Hand policy achieves performance comparable to hand-specific policies, while Zero-shot transfer remains effective on unseen hands despite substantial differences in morphology and kinematics. These results demonstrate the effectiveness of UHAS for cross-embodiment dexterous manipulation.

b) Cross-Morphology Generalization: Table II evaluates transfer across 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 on all four hands. The transferred policies are deployed directly using the corresponding UHAS representation. The results show strong in-distribution performance and meaningful transfer across morphologies. Although a gap remains relative to in-distribution performance, transferred policies achieve substantial success on unseen hands, demonstrating that UHAS generalizes across different finger counts and kinematic structures. Ablation studies of our method are provided in Appendix E.

TABLE III

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**

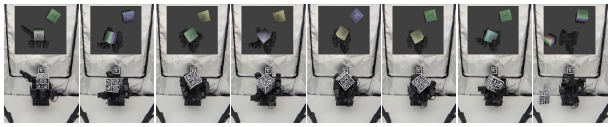

Fig. 7. An example of a real-world run for in-hand cube reorientation.

C. Real-World Results with Sim-to-Real Transfer

We deploy our method on a physical LEAP Hand [27] using an AprilTag-based cube pose estimator [2, 24] (Fig. 7). To reduce the sim-to-real gap, we perform system identification of the hardware setup. We evaluate each method over 10 trials, where the policy runs until the cube falls. Table III reports the number of consecutive successful reorientations before failure.

A substantial sim-to-real gap remains despite system identification and domain randomization. However, all UHAS variants outperform the joint-space baseline, including the zero-shot model. The LEAP-only model achieves the best performance, while the multi-hand model is slightly more conservative. Additional details and results are provided in Appendix C, Appendix D, and Appendix F.

V. 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] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

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

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arxiv:2410.24164*, 2024.

[4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

[7] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

[8] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[10] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le,

Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng 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.

[11] Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[12] Xin Fei, Zhixuan Xu, Huaicong Fang, Tianrui Zhang, and Lin Shao. T (r, o) grasp: Efficient graph diffusion of robot-object spatial transformation for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2510.12724*, 2025.

[13] Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Ria Doshi, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[14] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel 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.

[15] Xingyi He, Adhitya Polavaram, Yunhao Cao, Om Deshmukh, Tianrui Wang, Xiaowei Zhou, and Kuan Fang. Generate, transfer, adapt: Learning functional dexterous grasping from a single human demonstration. *arXiv preprint arXiv:2601.05243*, 2026.

[16] Guangqi Jiang, Yutong Liang, Jianglong Ye, Jia-Yang Huang, Changwei Jing, Rocky Duan, Pieter Abbeel, Xiaolong Wang, and Xueyan Zou. Cross-hand latent representation for vision-language-action models. *arXiv preprint arXiv:2603.10158*, 2026.

[17] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke,

and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[18] Ninad Khargonkar, Luis Felipe Casas, , Balakrishnan Prabhakaran, and Yu Xiang. Robotfingerprint: Unified gripper coordinate space for multi-gripper grasp synthesis. 2024.

[19] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[20] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

[21] Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan 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.

[22] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim 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.

[23] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel 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.

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

[25] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017.

[26] Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

[27] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023.

[28] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, pages 654–665. PMLR, 2023.

[29] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

[30] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *arXiv preprint arXiv:2505.07813*, 2025.

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

[32] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[33] Zhenyu Wei, Zhixuan Xu, Jingxiang Guo, Yiwen Hou, Chongkai Gao, Zhehao Cai, Jiayu Luo, and Lin Shao. D (r, o) grasp: A unified representation of robot and object interaction for cross-embodiment dexterous grasping. *arXiv preprint arXiv:2410.01702*, 2024.

[34] Zhenyu Wei, Yunchao Yao, and Mingyu Ding. One hand to rule them all: Canonical representations for unified dexterous manipulation. *arXiv preprint arXiv:2602.16712*, 2026.

[35] Ruoshi Wen, Guangzeng Chen, Zhongren Cui, Min Du, Yang Gou, Zhigang Han, Liqun Huang, Mingyu Lei, Yunfei Li, Zhuohang Li, Wenlei Liu, Yuxiao Liu, Xiao Ma, Hao Niu, Yutao Ouyang, Zeyu Ren, Haixin Shi, Wei Xu, Haoxiang Zhang, Jiajun Zhang, Xiao Zhang, Liwei Zheng, Weiheng Zhong, Yifei Zhou, Zhengming Zhu, and Hang Li. Gr-dexter technical report. *arXiv preprint arXiv:2512.24210*, 2025.

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

[37] Yuliang Wu, Yanhan Lin, WengKit Lao, Yuhao Lin, Yi-Lin Wei, Wei-Shi Zheng, and Ancong Wu. Dexgrasp-zero: A morphology-aligned policy for zero-shot cross-embodiment dexterous grasping. *arXiv preprint arXiv:2603.16806*, 2026.

[38] Zhiyuan Wu, Rolandos Alexandros Potamias, Xuyang Zhang, Zhongqun Zhang, Jiankang Deng, and Shan Luo. Cedex: Cross-embodiment dexterous grasp generation at scale from human-like contact representations. *arXiv preprint arXiv:2509.24661*, 2025.

[39] Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.

[40] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground, 2025. URL https://arxiv.org/abs/2502.08844.

[41] Zongzheng Zhang, Jingrui Pang, Zhuo Yang, Kun Li, Minwen Liao, Saining Zhang, Guoxuan Chi, Jinbang Guo, Huan ang Gao, Modi Shi, Dongyun Ge, Yao Mu, Jiayuan Gu, Rui Chen, Hao Dong, Huazhe Xu, Li Yi, Yixin Zhu, Hang Zhao, Pengwei Wang, Shanghang Zhang, Guocai Yao, Jianyu Chen, Hongyang Li, and Hao Zhao. Dexora: Open-source vla for high-dof bimanual dexterity. *arXiv preprint arXiv:2605.18722*, 2026.

[42] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[43] Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22508–22519, 2025.

[44] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

APPENDIX 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 III-B 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.

a) 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*.

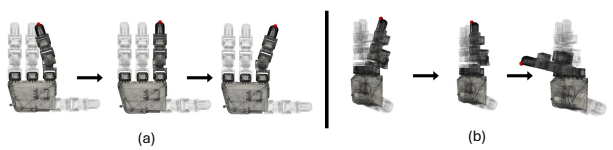

Fig. 8. Illustration of the joint classification procedure in the Cascade Inverse Kinematics (CIK) algorithm. (a) Classification of lateral joint based on their effect on the fingertip azimuthal angle θ. (b) Classification of encompassing joints based on their effect on the fingertip radial distance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

b) 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.

c) 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.

d) 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 IV.

APPENDIX 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-A.

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 III-B).

a) 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 III-B). 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.

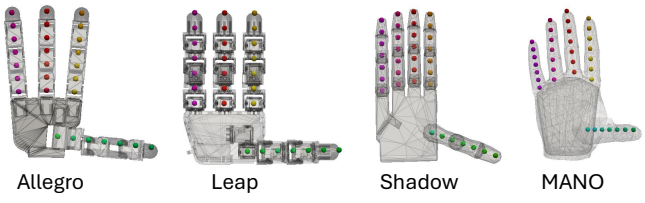

Fig. 9. Homogeneous observations across different robotic hands.

b) 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 $\phi = 60^\circ$ and $\phi = 120^\circ$, 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.

A. Training

a) Hyperparameters: All models were trained on a single NVIDIA A5000 GPU using the RSL-RL implementation of Proximal Policy Optimization (PPO) [26]. Training is performed in the custom NVIDIA Isaac Lab Cube Reposing environment [23] running on Isaac Sim 4.5.0 with the PhysX physics simulator. The complete set of PPO training hyperparameters is summarized in Table IV.

TABLE IV
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

b) 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 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

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

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 V.

TABLE V
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

TABLE VI
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) 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 VI 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.

APPENDIX C
CUBE POSE ESTIMATION

We track a 3D-printed cube based on an open source design [24]. Each of its six faces carries four distinct `tag36h11` AprilTags [2] (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.

APPENDIX 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.

A. 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 VI, 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.

B. 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.

TABLE VII
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 $\pm$ 2.6	9.3 $\pm$ 2.0	**9.6 $\pm$ 1.5**	9.1 $\pm$ 2.4
Training Time (h)	5.3	**4.5**	6.5	5.5

APPENDIX E
ABLATION STUDIES

We present ablation studies on different design choices of our method.

a) Ablation on Driving Vectors: 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 VII, 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.

b) 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 VIII, 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 VIII
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 $\pm$ 2.1	9.3 $\pm$ 2.0	9.4 $\pm$ 1.8	**9.5 $\pm$ 1.8**
Training Time (h)	4.9	**4.5**	4.8	4.7

c) 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 IX, 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 IX
ABLATION ON THE NUMBER OF DRIVING PLANES.

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

d) 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 X 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 X
ZERO-SHOT PERFORMANCE WITH AND WITHOUT DOMAIN RANDOMIZATION (PD GAINS, EFFORT, OBJECT SCALE, AND VECTOR ϕ). RESULTS REPORTED AS SUCCESS RATE / AVERAGE CONSECUTIVE REORIENTATIONS ± STD.

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

APPENDIX 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 [36].

A. 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 XI summarizes the results.

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.

B. Finetuning on Unseen Hands in Simulation

Table XII 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.

TABLE XI

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

TABLE XII

FAST ADAPTATION FROM MANO TO UNSEEN ROBOTIC HANDS. 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**

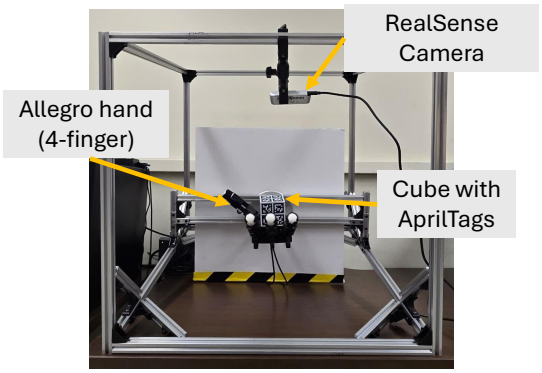

Fig. 10. Illustration of our real-world setup for the Allegro hand

C. Real-World Results on the Allegro Hand

We deploy our method on a physical Allegro Hand [36] 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 XIII reports the number of consecutive successful reorientations achieved before failure across trials, along with the mean.

TABLE XIII

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.