GET-Zero: Graph Embodiment Transformer for Zero-shot Embodiment Generalization

Austin Patel[1] and Shuran Song[1]

***Abstract*— This paper introduces GET-Zero, a model architecture and training procedure for learning an embodiment-aware control policy that can immediately adapt to new hardware changes without retraining. To do so, we present Graph Embodiment Transformer (GET), a transformer model that leverages the embodiment graph connectivity as a learned structural bias in the attention mechanism. We use behavior cloning to distill demonstration data from embodiment-specific expert policies into an embodiment-aware GET model that conditions on the hardware configuration of the robot to make control decisions. We conduct a case study on a dexterous in-hand object rotation task using different configurations of a four-fingered robot hand with joints removed and with link length extensions. Using the GET model along with a self-modeling loss enables GET-Zero to zero-shot generalize to unseen variation in graph structure and link length, yielding a 20% improvement over baseline methods. All code and qualitative video results are on our project website.**

I. Introduction

Learning algorithms have significantly improved robots' ability to adapt to external environments, yet most robots today cannot tolerate minor changes in their internal hardware. From missing links caused by damage to added joints for design improvements, hardware modifications often require retraining existing policies with embodiment-specific data.

Recent methods develop embodiment-aware policies that condition on the hardware configuration of a robot design to control a class of embodiments [1], [2], [3], [4], [5], [6], which enables efficient data reuse across embodiments. However, these method often have incomplete embodiment representations which ignore the graph connectivity of the robot, and thus cannot generalize well to an embodiment with a varied graph structure. Improving the performance under graph variations would expand the applicability of embodiment-aware methods in dexterous manipulation, where the number of fingers may vary.

The goal of this work is to introduce an embodiment representation that explicitly leverages the graph connectivity of the robot to improve zero-shot control of robots with variations in graph structure. We present GET-Zero, which introduces the **Graph Embodiment Transformer (GET)** model architecture that enables **zero-shot adaptation** to new embodiments (Fig. 1). The key idea of GET is to leverage the robot graph connectivity as a structural bias in the transformer attention mechanism. By training this model using behavior data from a variety of robot embodiments, we can then zero-shot control previously unseen embodiments with varied geometry, number of joints, and graph structure. GET-Zero consist of three components:

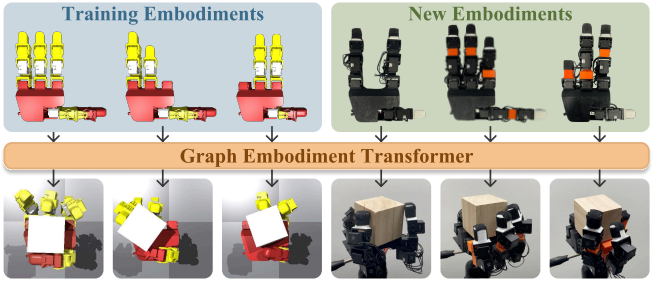

Fig. 1. **GET-Zero** is an embodiment-aware policy that is able to zero-shot generalize to unseen embodiment designs with varied geometry, number of joints, and graph structure.

1) **Graph Embodiment Transformer (GET).** GET is a modified transformer architecture [7] that encodes joints as separate tokens and uses the embodiment connectivity as a learned bias in the transformer attention mechanism. Our graph attention bias modulates communication based on how the joints are physically connected. This extends prior embodiment-aware transformer models which do not encode the embodiment connectivity [3] or have incomplete graph representations [4].
2) **Embodiment-aware Distillation.** To train GET-Zero we first collect demonstration data from embodiment-specific experts. By conditioning on embodiment information, we then distill knowledge from embodiment-specific experts into an embodiment-aware GET model using behavior cloning (BC) that controls both training and unseen embodiments with a *single* set of network weights. Prior methods [2], [3], [4] use RL to simultaneously learn an embodiment-aware model while learning to complete the task across all embodiments. In contrast, our method simplifies data generation by independently learning experts, then jointly distilling expert behavior.
3) **Self-modeling Loss.** During the distillation BC phase, we introduce a self-modeling loss to predict the position of each joint in 3D space (i.e., forward kinematics). We found that this simple and general supervision improves zero-shot performance. Self-supervision is common in NLP [8] and vision [9] domains, but is less explored in cross-embodiment learning [10], [11], [12].

To validate GET-Zero, we conduct a case study to learn an embodiment-aware, dexterous in-hand object rotation policy across different hardware configurations of a multi-fingered robot hand. In particular, we use the LEAP Hand [13], which is a low-cost, four-fingered hand with four joints

[1]Stanford University
Project Page: https://get-zero-paper.github.io
Corresponding Author: auspatel@stanford.edu

per finger consisting of 3D printed components and off-the-shelf motors. We create 44 hardware configurations of this hand by removing different combinations of finger joints and associated links. Next, we follow the GET-Zero training procedure to distill the expert behavior from these hands into a GET model. Our experiments in simulation and real world demonstrate how our graph encoding and self-modeling improve zero-shot capability.

II. Related Work

Cross-Embodiment pretraining with embodiment-specific finetuning. Recent approaches leverage large-scale pretraining, either on visual data, robot trajectories, language, or tasks as an initialization for policy learning or generalist agents [14], [15], [16], [17], [18], [19], [20]. Other approaches leverage cross-embodiment robot data to initialize a generalist agent, and then fine-tune the base model or an action head to adapt to new robot embodiments [21], [22]. These method often assume a unified action space, which is reasonable for navigation [23] or for robot arm policies where the end-effector pose is a shared representation across arms [22], [24], [21], [25]. However, unified actions spaces are often not sufficient for tasks where precise, low-level joint actions matter, such as dexterous manipulation [26], [27], [28], [29]. Our method uses the robot graph structure as an embodiment representation to perform joint-level control for tasks where a unified action space is not possible.

Embodiment-aware policy architectures. Most related to our approach is the line of work developing embodiment-aware policies. Prior methods [30], [31], [2] structure a graph neural network (GNN) to match the embodiment graph with joints as nodes and links as edges to control stick-like walking characters in simulation. [3] shows that a GNN matching the embodiment graph is outperformed with the fully connected attention mechanism in transformers [7] despite having no graph encoding. [4] extends this model with dynamics and kinematics encodings, demonstrating zero-shot generalization to these properties. However, these methods do not generalize well to unseen embodiment graphs due to no [3] or limited [4] graph representations. Our method GET-Zero extends the transformer architecture in [3], [4] to include an explicit graph representation that improves zero-shot generalization to unseen embodiment graphs.

III. Method: Graph Embodiment Transformer

We present Graph Embodiment Transformer (GET), an embodiment-aware transformer [7] that uses the embodiment graph as a structural bias in network architecture (Fig. 2). An embodiment graph consists of nodes representing local joint information, directed edges representing links connecting parent and child joints, and undirected edges between joints at the start of independent serial chains. GET represents each joint as separate transformer tokens containing both local sensory observations and local hardware properties (§III-A). The graph edges (i.e., links) are encoded through a learned attention bias in the self-attention layers (§III-B).

A. Embodiment Tokenization

For a robot with J joints, there are J input tokens to the transformer encoder containing local observations and J corresponding output tokens for per-joint actions. The encoder supports a variable number of tokens, enabling compatibility as the number of joints varies across embodiments.

Observations can either be local l to specific joints or global g across the entire embodiment. Additionally, observations are either fixed f or variable v if they change during execution. This yields four types: variable local o_{vl} (e.g., joint angle/velocity), variable global o_{vg} (e.g., a global time encoding, or environment state), fixed local o_{fl} (e.g., joint position in rest pose, or joint limit ranges) and fixed global o_{fg} (e.g., a task identifier). H past history steps, indicated $t \to t-H$, are included for variable observations. The local observations for joint j are defined as $o_{*l,j}$. The transformer token T_j for joint j at timestep t is constructed as $T_j^t = [o_{vg}^{t\to t-H}, o_{fg}, o_{vl,j}^{t\to t-H}, o_{fl,j}]$. The tokens pass through a linear embedding before entering the transformer encoder.

B. Graph Encoding

One challenge is that the transformer encoder has no direct graph encoding mechanism in the original implementation [7]. Input tokens are permutation invariant without a positional encoding mechanism, but are often linearly encoded with sinusoidal or learned positional encodings. Prior embodiment-aware methods address this in multiple ways. [3] use no positional encoding meaning no graph representation is present. [4] linearizes the graph using a depth-first search ordering then apply a learned linear positional encoding. However, this approach is sensitive to graph variations as the DFS ordering is not unique, which empirically caused a $\sim 75\%$ drop in performance with the opposite node order in [4]. [32] use the adjacency matrix as a binary attention mask to limit communication early in the transformer to adjacent nodes.

Our GET model uses a learned attention bias in the encoder self-attention mechanism to explicitly encode the graph similar to the Graphormer work by [33]. The original self-attention mechanism [7] computes $A \in \mathbb{R}^{J\times J}$ attention scores where A_{ij} represents the score computed between joints i and j. To encode the embodiment graph, we learn two separate biases to the attention scores (A):

Spatial Encoding. The spatial encoding computes the shortest path distance (SPD) ϕ^{SPD} between node i and node j as $\phi^{\text{SPD}}(i,j)$, treating all edges as undirected. An embedding table s is indexed with the SPD distance as $s_{\phi^{\text{SPD}}(i,j)}$ to get a learned scalar that is added to the attention score A_{ij}.

Parent-Child Encoding. Unlike Graphormer [33] that only encodes undirected graph in the attention mechanism, we introduce a new parent-child attention bias to encode directed features. For joints i and j, we compute the parent distance $\phi^P(i,j) = \phi^{\text{SPD}}(i,j)\mathbb{1}\{i \text{ is parent of } j\}$ which is the distance between i and j if j is the child of i at some distance along the forward kinematic chain, otherwise 0. We compute an analogous formula for the child distance as $\phi^C(i,j) = \phi^{\text{SPD}}(i,j)\mathbb{1}\{i \text{ is child of } j\}$. There are associated

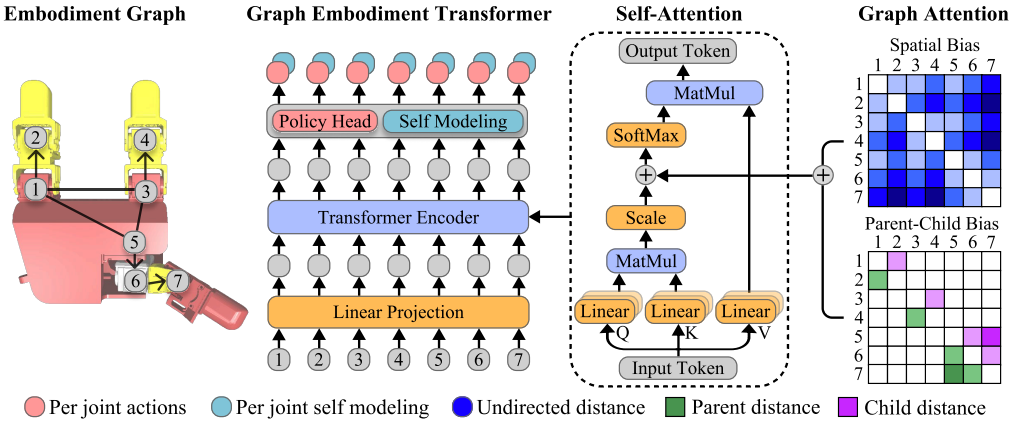

Fig. 2. **Graph Embodiment Transformer (GET).** GET is an embodiment-aware model based on a transformer encoder. Each joint forms separate tokens containing local sensory and embodiment information. The self-attention layers use an undirected (Spatial Bias) and directed (Parent-Child Bias) graph distance to bias the attention scores between joints according to the embodiment graph (grid color intensity indicates distance between nodes). A policy head predicts actions and a self modeling head predicts meta-properties about the embodiment, such as forward kinematics.

scalar embedding tables p and c, meaning directed edges have distinct biases from parent-to-child or child-to-parent.

Attention Bias Computation. For each head and encoder layer, we add attention bias embeddings s, p, and c to compute attention as: $\hat{A}_{ij} = A_{ij} + s_{\phi^{\text{SPD}}(i,j)} + p_{\phi^{\text{P}}(i,j)} + c_{\phi^{\text{C}}(i,j)}$. One key aspect of this approach is that this attention bias is invariant to the ordering of the input tokens unlike the depth-first linearization approach [4].

Output Heads. After passing J tokens through the transformer encoder, GET produces J corresponding features. We include a policy head to predict actions and a self-modeling head used to predict meta-features about the embodiment. More details on the heads are discussed in §IV-C.

IV. Case Study: In-hand Object Rotation

We conduct a case study on applying GET-Zero to in-hand object rotation with variations of LEAP Hand [13]. Applying GET-Zero requires three stages (Fig. 3): 1) generate embodiment variations (§IV-A), 2) train RL embodiment-specific experts (§IV-B), and 3) distill knowledge from experts into an embodiment-aware GET architecture (§IV-C).

A. Procedural Embodiment Generation

The LEAP Hand [13] consists of 3D printed components and off-the-shelf motors, making it highly configurable. We make variations to the hand graph by removing joints and to the links through link length extensions. Further details are in Section VII-A.

B. Train Embodiment-Specific Experts

We train embodiment-specific RL experts using PPO [34] in Isaac Gym [35] to complete the in-hand rotation task, adapted from the rewards and environments in [13]. These learned policies operate only on proprioceptive joint states and predict delta joint targets that guide a PD controller. We pick the best of five RL seeds per embodiment and filter embodiments that do not complete a full 2π rotation within 30s. This results in 44 training embodiments (19% of total design space) with associated expert policies.

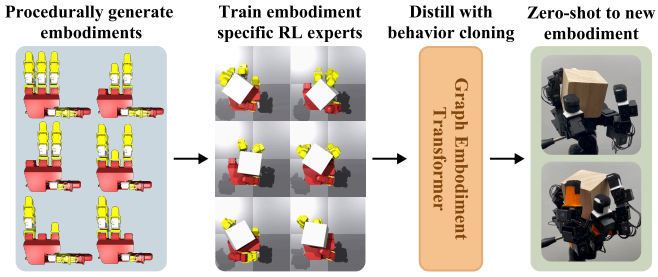

Fig. 3. **Training procedure.** We follow a teacher-student paradigm, where the teachers are *separate* embodiment-specific experts trained using RL. Then we distill knowledge from experts into a *single* embodiment-aware transformer (i.e., the student policy) using behavior cloning. GET-Zero takes as input embodiment definition and proprioception, and infers proper actions to perform an in-hand rotation task for an unseen embodiment.

C. Distill Experts into Embodiment-Aware Transformer

We roll out embodiment-specific experts to collect 7 hours worth of demonstrations per embodiment. This data is combined with embodiment information to form the dataset used to train the GET model with behavior cloning (BC).

GET Training Setup. We train the GET architecture (§III) using BC. Specifically, our token input has normalized joint angles and PD target joint states (o_{vl}), 3D joint position and joint rotation with respect to the parent joint from the URDF file (o_{fl}), and a sinusoidal phase encoding with a period of two seconds to encode cyclic progression of the rotation cycle (o_{vg}). The directed embodiment graph is an additional input used in the graph attention encoding (§III-B).

Output Heads. A policy head predicts per-joint single-step action. Each joint has a PD target state that is initialized to the starting joint state. The action head predicts delta actions that are added to this target state, which sets the target state for the low-level PD controller. We supervise these actions using L_2 loss with the demonstration data.

A self-modeling head predicts the current 3D pose of each joint in the robot local frame. This forward kinematics (FK) task is simple and generally applicable across embodiments, yet requires the network to use the spatial relationships from the graph and current joint angles along the FK chain. We

supervise with the ground truth FK pose with L_2 loss.

V. Experiments

We procedurally generate 236 graph variations of the LEAP hand [13] by removing various combinations of joints and associated links from the embodiment (§IV-A). From these, we obtain 44 embodiments with graph variations with associated expert RL policies that achieve a baseline level of performance (§IV-B). These 44 embodiments serve as training embodiments and only have graph variations (link length extensions are not seen during training). Our experiments consist of using demonstration data from these embodiments to train various baselines and ablations of the GET architecture to evaluate zero-shot performance to unseen embodiment graphs and link extensions.

Comparisons. MetaMorph [4] and Amorpheus [3] models are two prior embodiment-aware transformer methods. We re-implement the positional encoding aspects of these method in our setup to focus on comparing the impact of our graph representation and self-modeling rather than differences in task (locomotion v.s. manipulation), training procedure (RL v.s. BC) or observation format.

- **ET.** The Embodiment Transformer (ET) matches the architecture of GET except without the graph encoding (§III-B) or the self modeling. This ET baseline most closely matches the encoding method in Amorpheus [3]. Note that this model has no connectivity information about the joints.
- **ET+DFS.** This baseline uses positional encoding from MetaMorph [4] that linearizes the embodiment graph in depth-first search (DFS) ordering and applies a learned linear positional encoding.

We also ablate different components of our method such as self-modeling loss (SL), spatial encoding (SE), parent-child encoding (PE). Results are summarized in Tab. I.

Evaluation Metric. The task performance is measured by average rotational velocity along the yaw axis in degrees/second including standard deviation across five seeds. We compute results by rolling out the embodiment-aware policies in a physics simulator for a total of 42 minutes of execution time per embodiment per seed. Each category averages over 10 unseen embodiments.

We were able to train embodiment-specific RL experts for all 10 embodiments with graph variations with sufficient performance (2π rotation within 30 seconds), validating that they are reasonable targets for zero-shot generalization.

A. Key Results and Findings

Table I shows GET-Zero's performance on training embodiments (Training Graph) and on multiple types of zero-shot embodiments: unseen graph variations (New Graph), unseen link length variations (New Geo) and both (New Graph & Geo). Link length extensions were *never* seen during training.

GET-Zero improves zero-shot capabilities to graph and geometry variations. Compared to the best performing baseline for each category, GET-Zero achieves a 16% improvement with zero-shot to new graph, an 10% improvement with zero-shot to link length variations, and a 20% improvement with zero-shot to both graph and link length variations (Tab. I).

TABLE I
Simulation results (average rotational velocity deg/sec). RL experts are embodiment-specific policies. ET: Embodiment transformer, DFS: depth-first graph encoding, SL: self-modeling loss, SE: spatial graph encoding, PE: parent-child graph encoding

	Training Graph	New Graph	New Geo	New Graph&Geo
RL experts	16.50	—	—	—
ET [3]	14.82±0.15	8.68±0.34	12.92±0.36	8.12±0.46
ET+DFS [4]	15.03±0.37	6.45±0.28	14.37±0.34	6.27±0.22
ET+SL	14.68±0.33	8.19±0.74	12.71±0.72	7.60±1.12
ET+PE+SE	16.03±0.19	9.65±0.23	15.30±0.29	9.05±0.42
ET+PE+SL	16.10±0.22	9.56±0.47	15.59±0.32	8.81±0.42
ET+SE+SL	16.24±0.21	10.04±0.15	15.74±0.25	**9.75±0.20**
ET+PE+SE+SL	**16.32±0.24**	**10.07±0.58**	**15.80±0.29**	**9.75±0.54**

Self-modeling improves performance only with graph encoding present. We observe that adding self-modeling (SL) to the baseline (ET) decreases zero-shot performance by an average of 4.6% across the three categories, but yields an average increase of 5.1% when added to the model with graph encoding (ET+PE+SE) (Table I).

Spatial and parent-child encoding help policy learning. Relative to ET with self-modeling (ET+SL), adding the parent-child graph encoding (ET+PE+SL) improves performance by 19% and adding the spatial graph encoding (ET+SE+SL) improves performance by 25% on average across zero-shot tasks. Adding the parent-child bias when the spatial bias is already present provides no statistically significant improvement, indicating that our directed encoding is not critical for performance.

Depth-first graph linearization lowers performance under graph variations. The DFS linearization (ET+DFS) serves as a simple, yet incomplete graph representation as the DFS ordering is not unique. When only link length variations are present, linearization improves zero-shot performance over no positional encoding (ET) by 11% (Tab. I). However, with unseen graph variations, performance drops by 26%. This result intuitively makes sense as the linearized positional encoding overfit to training embodiment graphs, but fail to generalize to unseen graphs due to the aforementioned issue with DFS. This motivates the need for the more complete graph representation used in GET-Zero.

VI. Conclusion

We present GET-Zero, an embodiment-aware model architecture and training procedure that enables zero-shot control of new robot designs. Through a case study on an in-hand object rotation task, we demonstrate the ability of our model to control a wide range of hardware configurations of a multi-fingered hand under variations in embodiment graph and geometry. Our results in simulation and real show that the graph encoding and self-modeling features in GET-Zero improve cross-embodiment transfer. We hope that GET-Zero serves as a useful method for the robotics community to share knowledge across similar robot designs.

REFERENCES

[1] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," https://octo-models.github.io, 2023.

[2] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *ICML*, 2020.

[3] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson, "My body is a cage: the role of morphology in graph-based incompatible control," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=N3zUDGN5lO

[4] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, "Metamorph: Learning universal controllers with transformers," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=Opmqtk_GvYL

[5] M. Shafiee, G. Bellegarda, and A. Ijspeert, "Manyquadrupeds: Learning a single locomotion policy for diverse quadruped robots," *arXiv preprint arXiv:2310.10486*, 2023.

[6] N. Bohlinger, G. Czechmanowski, M. P. Krupka, P. Kicki, K. Walas, J. Peters, and D. Tateo, "One policy to run them all: Towards an end-to-end learning approach to multi-embodiment locomotion," in *Workshop on Embodiment-Aware Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=HVWusz2zv5

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[10] C. Ying, Z. Hao, X. Zhou, X. Xu, H. Su, X. Zhang, and J. Zhu, "Peac: Unsupervised pre-training for cross-embodiment reinforcement learning," *arXiv preprint arXiv:2405.14073*, 2024.

[11] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi, "Xirl: Cross-embodiment inverse reinforcement learning," *Conference on Robot Learning (CoRL)*, 2021.

[12] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song, "XSkill: Cross embodiment skill discovery," in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: https://openreview.net/forum?id=8L6pHd9aS6w

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

[14] 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 *arXiv preprint arXiv:2307.15818*, 2023.

[15] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik, "Robot learning with sensorimotor pre-training," in *Conference on Robot Learning*. PMLR, 2023, pp. 683–693.

[16] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas, "A generalist agent," *Transactions on Machine Learning Research*, 2022, featured Certification, Outstanding Certification. [Online]. Available: https://openreview.net/forum?id=1ikK0kHjvj

[17] I. Schubert, J. Zhang, J. Bruce, S. Bechtle, E. Parisotto, M. Riedmiller, J. T. Springenberg, A. Byravan, L. Hasenclever, and N. Heess, "A generalist dynamics model for control," *arXiv preprint arXiv:2305.10912*, 2023.

[18] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, S. Kirmani, B. Zitkovich, F. Xia, C. Finn, and K. Hausman, "Open-world object manipulation using pre-trained vision-language models," in *arXiv preprint*, 2023.

[19] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "Palm-e: An embodied multimodal language model," in *arXiv preprint arXiv:2303.03378*, 2023.

[20] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*. PMLR, 2018, pp. 651–673.

[21] O. X.-E. Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, H. Su, H.-S. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Kim, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Majd, K. Rana, K. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. hua Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui, "Open X-Embodiment: Robotic learning datasets and RT-X models," https://arxiv.org/abs/2310.08864, 2023.

[22] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, *et al.*, "Robocat: A self-improving foundation agent for robotic manipulation," *arXiv preprint arXiv:2306.11706*, 2023.

[23] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "Vint: A foundation model for visual navigation," *arXiv preprint arXiv:2306.14846*, 2023.

[24] J. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, D. Sadigh, and S. Levine, "Pushing the limits of cross-embodiment learning for manipulation and navigation," *arXiv preprint arXiv:2402.19432*, 2024.

[25] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space. an action space for reinforcement learning in contact rich tasks," in *Proceedings of the International Conference of Intelligent Robots and Systems (IROS)*, 2019.

[26] L. Han and J. Trinkle, "Dextrous manipulation by rolling and finger gaiting," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, 1998, pp. 730–735 vol.1.

[27] J.-P. Saut, A. Sahbani, S. El-Khoury, and V. Perdereau, "Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2907–2912.

[28] D. Rus, "In-hand dexterous manipulation of piecewise-smooth 3-d objects," *The International Journal of Robotics Research*,

vol. 18, no. 4, pp. 355–381, 1999. [Online]. Available: https://doi.org/10.1177/02783649922066268

[29] Y. Bai and C. K. Liu, "Dexterous manipulation using both palm and fingers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1560–1565.

[30] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=S1sqHMZCb

[31] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros, "Learning to control self- assembling morphologies: A study of generalization via modularity," in *NeurIPS*, 2019.

[32] C. Sferrazza, D.-M. Huang, F. Liu, J. Lee, and P. Abbeel, "Body transformer: Leveraging robot embodiment for policy learning," in *Workshop on Embodiment-Aware Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=IbXqRpANPD

[33] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?" in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. [Online]. Available: https://openreview.net/forum?id=OeWooOxFwDa

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

[35] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance GPU based physics simulation for robot learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=fgFBtYgJQX_

TABLE II

GET-ZERO SIM-TO-REAL EVALUATION. [COL 1] AN AR TAG TRACKS CUBE ROTATION. [COL 2] THE UNMODIFIED LEAP HAND (SEEN DURING TRAINING). [COL 3-7] ZERO-SHOT EMBODIMENTS WITH UNSEEN GRAPH VARIATIONS AND/OR LINK LENGTH EXTENSIONS (ORANGE). WE REPORT (ACROSS 3 MINUTE TRIAL): 1) AVERAGE CUBE ANGULAR VELOCITY IN DEGREES/SECOND 2) REAL AVERAGE VELOCITY AS A % OF SIM AVERAGE VELOCITY FOR THE SAME EMBODIMENT AND POLICY (SIM-TO-REAL GAP) AND 3) NUMBER OF TIMES THE CUBE FELL OFF THE HAND.

Variation	Training	New Graph	New Graph	New Geo	New Graph+Geo	New Graph+Geo
ET	11.5, 55%, 0	5.9, 50%, 0	9.2, 93%, 1	8.6, 57%, 2	9.6, 67%, 18	-0.6, -6%, 0
GET-Zero	20.5, 70%, 0	8.3, 65%, 0	21.8, 133%, 0	19.0, 83%, 1	9.0, 53%, 16	1.2, 9%, 0

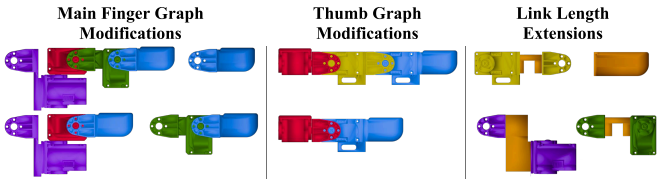

Fig. 4. **Finger Variations.** We procedurally generate variations of the fingers in the LEAP Hand [13] by removing joints and links as well as adding in 1.5cm link length extensions (orange).

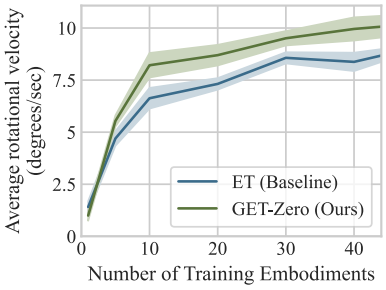

Fig. 5. **Impact of training embodiments on zero-shot graph generalization**. We observe that even with fewer training embodiments, GET-Zero achieves reasonable in-hand rotation performance (5 seeds).

ACKNOWLEDGMENT

We thank Kenneth Shaw, Prof. Pathak's lab at CMU, and the Prof. Liu's Movement lab at Stanford for sharing LEAP Hand hardware. We also thank Huy Ha, Xiaomeng Xu, Mengda Xu and Haochen Shi for their helpful feedback and fruitful discussions. This work was supported in part by Sloan Fellowship and NSF #2132519. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

VII. SUPPLEMENTARY MATERIALS

A. Embodiment Variations

The hand features three main fingers which are identical in structure and a thumb which has a separate structure. For the main fingers, we construct five variations of the fingers (4 combinations + no finger config) and two different variations of the thumb by removing various combinations of joints (Fig. 4). This yields $5^3 \times 2 = 250$ embodiment configurations from which we require 1) at least two fingers have at least one joint and 2) that there is at least one main finger with two joints, yielding 236 graph variation embodiments. We additionally introduce 1.5cm link length extensions shown in orange in Fig. 4 to the 236 designs to generate additional hand designs.

B. Zero-shot with Fewer Training Embodiments

To validate that zero-shot performance isn't solely due to a large number of training embodiments that might be similar to the zero-shot embodiments, we evaluate our method on smaller subsets of training embodiments as shown in Fig. 5. GET-Zero achieves reasonable rotation performance even with only 10 training embodiments, indicating that zero-shot performance works when demonstration data is only available for fewer embodiments. The 44 training embodiments for results in Tab. I is 19% of the 236 total graph variation embodiments generated, though not all 236 embodiments are equally capable of completing the rotation task.

C. Real-world Evaluation

We present results from a real-world evaluation of GET-Zero on unseen embodiments in Tab. II following the same evaluation methods as for simulation. We observe GET-Zero outperforms the ET baseline for both the original LEAP Hand [13] as well as unseen graph and link length variations. Empirically, the ET policy struggles to continuously rotate the cube, and we observe high finger precision is required for stable control. We observe a sim-to-real gap, but GET-Zero achieves zero-shot performance above simulation for the third embodiment. For another embodiment (second from right), a missing index finger causes the hand to drop the cube many times. For the right-most embodiment, a shortened index finger and thumb make it challenging for the hand to start the rotation cycle.