$\mathcal{D}(\mathcal{R}, \mathcal{O})$ Grasp: A Unified Representation of Robot and Object Interaction for Cross-Embodiment Dexterous Grasping

Zhenyu Wei[1,2*], **Zhixuan Xu**[1*], **Jingxiang Guo**[1], **Yiwen Hou**[1], **Chongkai Gao**[1], **Zhehao Cai**[1], **Jiayu Luo**[1], **Lin Shao**[1]

[1] National University of Singapore, [2] Shanghai Jiao Tong University

Zhenyu_Wei@sjtu.edu.cn, linshao@nus.edu.sg

Abstract: Dexterous grasping is a fundamental yet challenging skill in robotic manipulation, requiring precise interaction between robotic hands and objects. In this paper, we present $\mathcal{D}(\mathcal{R}, \mathcal{O})$ Grasp, a novel framework that models the interaction between the robotic hand in its grasping pose and the object, enabling broad generalization across various robot hands and object geometries. Our model takes the robot hand's description and object point cloud as inputs and efficiently predicts kinematically valid and stable grasps, demonstrating strong adaptability to diverse robot embodiments and object geometries. Extensive experiments conducted in both simulated and real-world environments validate the effectiveness of our approach, with significant improvements in success rate, grasp diversity, and inference speed across multiple robotic hands. Our method achieves an average success rate of **87.53%** in simulation in less than one second, tested across three different dexterous robotic hands. In real-world experiments using the LeapHand, the method also demonstrates an average success rate of **89%**. $\mathcal{D}(\mathcal{R}, \mathcal{O})$ Grasp provides a robust solution for dexterous grasping in complex and varied environments. The code, appendix, and videos are available on our project website at https://nus-lins-lab.github.io/drograspweb/.

Keywords: Dexterous Grasping, Robotic Manipulation

1 Introduction

Dexterous grasping is essential for robotic manipulation, but quickly obtaining diverse, high-quality grasps remains difficult due to robotic hands' complexity and high degrees of freedom. We can broadly categorize data-driven grasp generation methods into two types: those that utilize robot-centric representations, such as wrist poses and joint values [1, 2, 3], and those that rely on object-centric representations, such as contact points [4, 5, 6] or contact maps [7, 8, 9, 10].

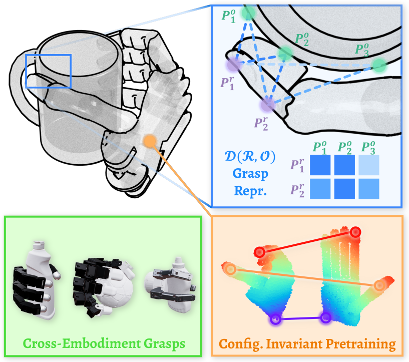

P_1^o
P_2^o
P_3^o
P_1^r
P_2^r
$\mathcal{D}(\mathcal{R}, \mathcal{O})$ Grasp Repr.
Cross-Embodiment Grasps
Config. Invariant Pretraining

Figure 1: We propose our model that utilizes configuration-invariant pretraining, predicts $\mathcal{D}(\mathcal{R}, \mathcal{O})$ representation, and obtains grasps for cross-embodiment from point cloud input.

The use of robot-centric representations like joint values in methods such as UniDexGrasp++ [3] offer fast inference by mapping observations to control commands but struggle with low sample efficiency and poor generalization across different robot embodiments. Object-centric methods, such as UniGrasp[4] and GenDexGrasp [7], effectively capture the geometry and contacts of objects, allowing for generalization across different

	Grasp Representation	Method Type	Cross Embodiment	Inference Speed	Sample Efficiency	Partial Object Point Cloud	Full-hand Contact (not only fingertips)	Optional Grasp Preference Interface
DFC [12]	Joint Values	Robot-centric	✓	✗✗	-	✗	✓	✗
UniDexGrasp++ [3]	Joint Values	Robot-centric	✗	✓✓	✗	✓	✓	✗
UniGrasp [4]	Contact Point	Object-centric	✓	✗	✓	✗	✗	✗
GeoMatch [5]	Contact Point	Object-centric	✓	✗	✓	✗	✓	✗
GenDexGrasp [7]	Contact Map	Object-centric	✓	✗	✓	✗	✓	✗
ManiFM [8]	Contact Map	Object-centric	✓	✗	✓	✗	✗	Contact Region
DRO-Grasp (Ours)	$\mathcal{D}(\mathcal{R},\mathcal{O})$	Interaction-centric	✓	✓	✓	✓	✓	Palm Orientation

Table 1: Dexterous grasp method comparison.

shapes and robots. However, these methods often require an additional time-consuming optimization step, making them less efficient [11].

To overcome the limitations of both paradigms, we propose $\mathcal{D}(\mathcal{R},\mathcal{O})$, a unified representation that captures the relationship between the robotic hand's grasp shape and the object. $\mathcal{D}(\mathcal{R},\mathcal{O})$ encapsulates both the articulated structure of the robot hand and the object's geometry, enabling direct inference of kinematically valid and stable grasps that generalize across various shapes and robot embodiments.

In conclusion, our primary contributions are as follows:

1. We introduce a novel representation, $\mathcal{D}(\mathcal{R},\mathcal{O})$ for dexterous grasping tasks. This interaction-centric formulation facilitates robust generalization across diverse robotic hands and objects.
2. We propose a configuration-invariant pretraining approach with contrastive learning to align features across different hand configurations, enabling effective grasp generation and cross-embodiment generalization.
3. We perform extensive experiments in both simulation and real-world settings, validating the efficacy of our proposed model in grasping novel objects with multiple robotic hands.

2 Method

Method Overview. First, we design an encoder network to learn representations from the point clouds of both the robot and the object. The robot encoder network is pretrained using our proposed configuration-invariant pretraining method (Sec. 2.1), which facilitates the learning of efficient robot embedding. Next, a CVAE model is used to predict the $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation, a point-to-point distance matrix between the robotic hand at its grasp pose and the object, to implicitly present the grasp pose (Sec. 2.2). From the $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation, we derive the 6D pose for each link, which serves as the optimization target for determining the joint values. This optimization process is notably straightforward and efficient (Sec. 2.3). Fig. 2 provides an overview of our proposed method, and further details are provided in Appendix A.

2.1 Configuration-Invariant Pretraining with Contrastive Learning

Learning dexterous grasping requires understanding the spatial relationships between the robot hand and the object. The goal is to match the hand in specific configurations with the object, but this is challenging due to the significant variations in local geometric features between open and grasp configurations. To address this, we introduce configuration-invariant pretraining, enabling the encoder network to capture self-articulation alignment across various configurations. This approach enhances the matching process between the robotic hand and the object.

We apply point-level contrastive learning on the robot hand's point clouds in both open and closed configurations to train the encoder network described in Sec. 2.2, ensuring feature consistency at corresponding points across different joint configurations.

2.2 $\mathcal{D}(\mathcal{R},\mathcal{O})$ Prediction

Given the robot point cloud under initial pose $\mathbf{P}^{\mathcal{R}} \in \mathbb{R}^{N_{\mathcal{R}}\times 3}$, and the object point cloud $\mathbf{P}^{\mathcal{O}} \in \mathbb{R}^{N_{\mathcal{O}}\times 3}$, our model aims to predict the point-to-point distance matrix $\mathcal{D}(\mathcal{R},\mathcal{O}) \in \mathbb{R}^{N_{\mathcal{R}}\times N_{\mathcal{O}}}$.

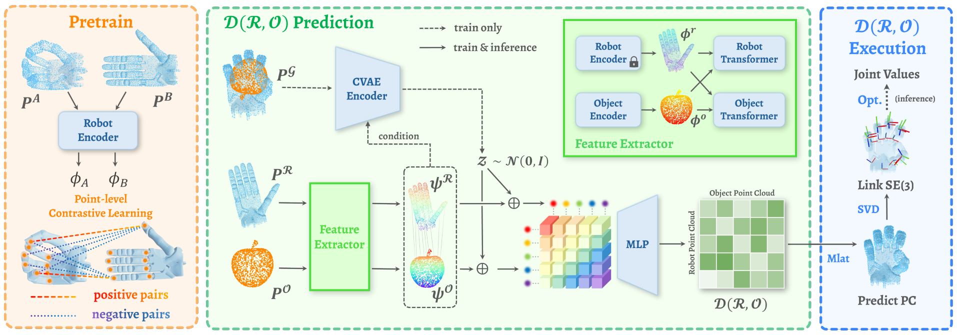


Figure 2: Overview of $\mathcal{D}(\mathcal{R},\mathcal{O})$ framework: We first pretrain the robot encoder with the proposed configuration-invariant pretraining method. Then, we predict the $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation between the robot and object point cloud. Finally, we extract joint values from the $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation.

Point Cloud Feature Extraction We start by extracting point cloud embeddings using a robot encoder and an object encoder, both sharing the same architecture. The robot encoder is initialized with pretrained parameters and remains frozen during training. These encoders extract point-wise features from the robot and object point clouds. To establish correspondences, we employ two multi-head transformers, integrating the relationships between the robot and object feature sets.

CVAE-based $\mathcal{D}(\mathcal{R},\mathcal{O})$ Prediction To achieve cross-embodiment grasp diversity, we employ a CVAE [13] network to capture variations across different combinations of hand, object, and grasp configurations. The CVAE encoder processes the robot and object point clouds in the grasp pose, along with the learned features, to output the latent variable z. This latent variable is then concatenated with the extracted point cloud features. We adopt the same kernel function $\mathcal{K}$ as Eisner et al. [14], which is non-negative and symmetric, to predict pairwise distances between the robot and object point clouds in the grasp pose. By computing distances across all pairs, we construct the complete $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation.

2.3 Grasp Configuration Generation from $\mathcal{D}(\mathcal{R},\mathcal{O})$

Given the predicted $\mathcal{D}(\mathcal{R},\mathcal{O})$, we first apply multilateration [15] to compute the robot's grasp point cloud based on the object point cloud. Directly solving inverse kinematics to obtain joint values from a point cloud is a non-trivial task. Next, we calculate the 6D pose of each link using rigid body registration techniques. Finally, we solve a simplified optimization problem using CVXPY [16] to determine the corresponding joint values.

3 Experiments

3.1 Evaluation Metric

Success Rate: We evaluate the success of grasping by determining whether the force closure condition is satisfied. To implement this evaluation criterion, we used the Isaac Gym simulator [17]. A simple grasp controller is applied to execute the predicted grasps in the simulation. We consider the grasp successful if the object's resultant displacement stays below 2 cm after applying the six directional forces.

Diversity: Grasp diversity is quantified by calculating the standard deviation of the joint values (including 6 floating wrist DoF) across all successful grasps.

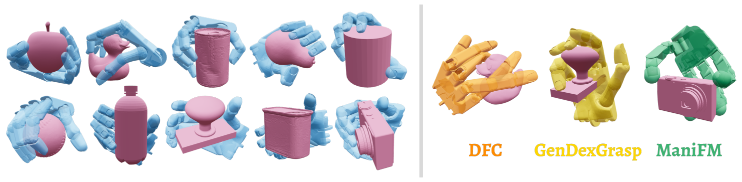


Figure 3: Visualization of all methods.

Method	**Success Rate (%) ↑**				**Diversity (rad.) ↑**			**Efficiency (sec.) ↓**		
	Barrett	Allegro	ShadowHand	Avg.	Barrett	Allegro	ShadowHand	Barrett	Allegro	ShadowHand
DFC [12]	86.30	76.21	58.80	73.77	**0.532**	**0.454**	0.435	>1800	>1800	>1800
GenDexGrasp [7]	67.00	51.00	54.20	57.40	0.488	0.389	0.318	14.67	25.10	19.34
ManiFM [8]	-	42.60	-	42.60	-	0.288	-	-	9.07	-
DRO-Grasp (w/o pretrain)	87.20	82.70	46.70	72.20	**0.532**	0.448	0.429	**0.49**	**0.47**	**0.98**
DRO-Grasp (Ours)	**87.30**	**92.30**	**83.00**	**87.53**	0.513	0.397	**0.441**	**0.49**	**0.47**	**0.98**

Table 2: Overall comparison with baselines.

Method	**Success Rate (%) ↑**			**Diversity (rad) ↑**		
	Barrett	Allegro	ShadowHand	Barrett	Allegro	ShadowHand
Single	84.80	88.70	75.80	0.505	**0.435**	0.425
Multi	**87.30**	**92.30**	**83.00**	**0.513**	0.397	**0.441**
Partial	84.70	87.60	81.80	0.511	0.401	0.412

Table 3: Comparison under different conditions. "Single" trains on one hand, "Multi" trains on all hands, and "Partial" trains and tests on partial point clouds.

Efficiency: The computational time required to achieve a grasp is measured, encompassing both network inference and the subsequent optimization steps.

3.2 Simulation Performance

We present a detailed comparison of $\mathcal{D}(\mathcal{R}, \mathcal{O})$ against DFC [12], GenDexGrasp [7], and ManiFM [8], as shown in Tab. 2. This evaluation, conducted on 10 previously unseen test objects using the Barrett, Allegro, and ShadowHand robotic hands, provides a comprehensive overview of distinct approaches to cross-embodiment grasping.

Our results demonstrate that $\mathcal{D}(\mathcal{R}, \mathcal{O})$ significantly outperforms all baselines in success rate, highlighting the effectiveness of our method. Fig. 3 illustrates grasps generated by our method alongside typical failure cases from the baselines. For successful grasps, the average displacement remains under 2 mm, with an average rotation below 1°, highlighting the firmness of our generated grasps.

The first two rows of Tab. 3 show a slight improvement in success rates when training across multiple robotic hands rather than a single one, confirming the cross-embodiment generalizability of our method. Additionally, our method significantly improves grasp generation speed, generating a grasp within 1 second. This fast computation is crucial for dexterous manipulation tasks.

For further experiments in simulation, please refer to Appendix C.1, C.2, C.3, C.4.

3.3 Real-Robot Experiments

We conducted real-world experiments with a uFactory xArm6 robot, equipped with the LEAP Hand [18] and the overhead Realsense D435 camera, as illustrated in Fig. 4. Our method achieved an average success rate of **89%** across 10 novel objects, showcasing its effectiveness in dexterous grasping and its generalizability to previously unseen objects. More details are provided in Appendix C.5.

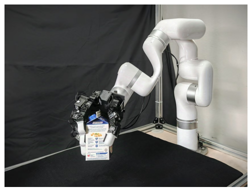


Figure 4: Real-world experiment setting.

4 Conclusion

This work presents a new method for improving dexterous grasping by introducing the $\mathcal{D}(\mathcal{R}, \mathcal{O})$ representation, which captures the essential interaction between robotic hands and objects. Unlike existing methods that rely heavily on either object or robot-specific representations, our approach bridges the gap by using a unified framework that generalizes well across different robots and object geometries. Additionally, our pretraining approach enhances the model's capacity to adapt to different hand configurations, making it suitable for a wide range of robotic systems. Experimental results confirm that our method delivers notable improvements in success rates, diversity, and computational efficiency.

References

[1] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023.

[2] W. Xu, W. Guo, X. Shi, X. Sheng, and X. Zhu. Fast force-closure grasp synthesis with learning-based sampling. *IEEE Robotics and Automation Letters*, 8(7):4275–4282, 2023.

[3] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023.

[4] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg. Unigrasp: Learning a unified model to grasp with multifingered robotic hands. *IEEE Robotics and Automation Letters*, 5(2):2286–2293, 2020.

[5] M. Attarian, M. A. Asif, J. Liu, R. Hari, A. Garg, I. Gilitschenski, and J. Tompson. Geometry matching for multi-embodiment grasping. In *Conference on Robot Learning*, pages 1242–1256. PMLR, 2023.

[6] S. Li, Z. Li, K. Han, X. Li, Y. Xiong, and Z. Xie. An end-to-end spatial grasp prediction model for humanoid multi-fingered hand using deep network. In *2021 6th International Conference on Control, Robotics and Cybernetics (CRC)*, pages 130–136. IEEE, 2021.

[7] P. Li, T. Liu, Y. Li, Y. Geng, Y. Zhu, Y. Yang, and S. Huang. Gendexgrasp: Generalizable dexterous grasping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8068–8074. IEEE, 2023.

[8] Z. Xu, C. Gao, Z. Liu, G. Yang, C. Tie, H. Zheng, H. Zhou, W. Peng, D. Wang, T. Chen, Z. Yu, and L. Shao. Manifoundation model for general-purpose robotic manipulation of contact synthesis with arbitrary objects and robots, 2024.

[9] D. Morrison, P. Corke, and J. Leitner. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv preprint arXiv:1804.05172*, 2018.

[10] J. Varley, J. Weisz, J. Weiss, and P. Allen. Generating multi-fingered robotic grasps via deep learning. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4415–4420. IEEE, 2015.

[11] A. Wu, M. Guo, and C. K. Liu. Learning diverse and physically feasible dexterous grasps with generative model and bilevel optimization. *arXiv preprint arXiv:2207.00195*, 2022.

[12] T. Liu, Z. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu. Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator. *IEEE Robotics and Automation Letters*, 7(1):470–477, 2021.

[13] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[14] B. Eisner, Y. Yang, T. Davchev, M. Vecerik, J. Scholz, and D. Held. Deep se (3)-equivariant geometric reasoning for precise placement tasks. *arXiv preprint arXiv:2404.13478*, 2024.

[15] A. Norrdine. An algebraic solution to the multilateration problem. In *Proceedings of the 15th international conference on indoor positioning and indoor navigation, Sydney, Australia*, volume 1315, 2012.

[16] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[17] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox. Gpu-accelerated robotic simulation for distributed reinforcement learning. In *Conference on Robot Learning*, pages 270–282. PMLR, 2018.

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

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

[20] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[21] Y. Zhou. An efficient least-squares trilateration algorithm for mobile robot localization. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3474–3479. IEEE, 2009.

[22] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[23] S. Brahmbhatt, C. Ham, C. C. Kemp, and J. Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8709–8719, 2019.

[24] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. *arXiv preprint arXiv:2210.02697*, 2022.

[25] AR Code. Ar code, 2022. URL `https://ar-code.com/`. Accessed: 2024-09-28.

[26] OpenCV Team. Opencv: Open source computer vision library, 2023. URL `https://opencv.org/`. Version 4.8.0, `https://docs.opencv.org/4.x/d9/d6c/tutorial_table_of_content_charuco.html`.

[27] L. Chen, Y. Qin, X. Zhou, and H. Su. Easyhec: Accurate and automatic hand-eye calibration via differentiable rendering and space exploration. *IEEE Robotics and Automation Letters*, 2023.

[28] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.

[29] H. S. Lab. Mplib: Motion planning library, 2023. URL `https://github.com/haosulab/MPlib`. Accessed: 2024-09-28.

Appendix

A Method Details

A.1 Configuration-Invariant Pretraining

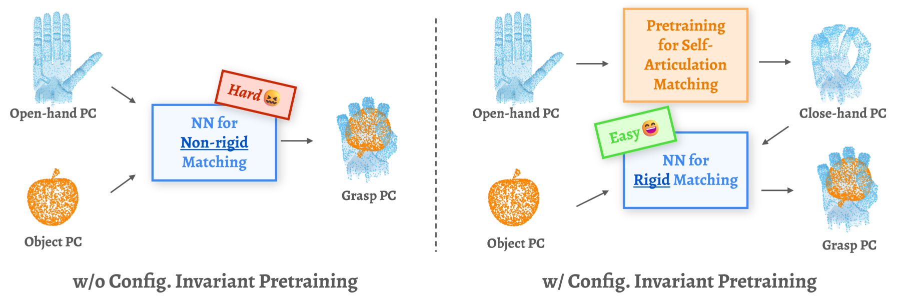

Open-hand PC
Object PC
Hard
NN for Non-rigid Matching
Grasp PC
w/o Config. Invariant Pretraining
Open-hand PC
Pretraining for Self-Articulation Matching
Close-hand PC
Easy
NN for Rigid Matching
Object PC
Grasp PC
w/ Config. Invariant Pretraining

Figure 5: Motivation for configuration-invariant pretraining.

The pertaining process breaks the problem into two simpler components: (1) self-articulation matching, which implicitly determines the joint values for the grasp configuration, and (2) wrist pose estimation. As shown in Fig. 5, leveraging configuration-invariant pretraining, we train the neural network to understand the self-articulation alignment across different configurations, thereby facilitating the matching process between the robot hand and the object.

Specifically, for each robot hand, we begin by uniformly sampling points on the surface of each link at the canonical pose, storing the resulting point clouds denoted as $\{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}$, where N_ℓ is the number of links. We define a point cloud forward kinematics model, FK $\left(q, \{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}\right)$ to map joint configurations to point clouds at new poses. For example, given a close-hand $q_\mathcal{A}$ and an open-hand configuration $q_\mathcal{B}$, where the wrist pose is the same or nearly identical, we obtain two point clouds $\mathbf{P}^\mathcal{A}, \mathbf{P}^\mathcal{B} \in \mathbb{R}^{N_\mathcal{R} \times 3}$, representing these two joint configurations. Here, $N_\mathcal{R}$ is the number of points in the robot point cloud, set to 512 in practice.

These point clouds are passed through the encoder network (as described in Sec. 2.2) to produce point-wise features $\phi^\mathcal{A}, \phi^\mathcal{B} \in \mathbb{R}^{N_\mathcal{R} \times D}$, where $D = 512$ is the feature dimension. The model applies point-level contrastive learning, aligning embeddings of positive pairs—points with the same index in both clouds—while separating negative pairs, weighted by the Euclidean distance in $\mathbf{P}^\mathcal{B}$. This process ensures that the features corresponding to the same positions on the robot hand remain consistent across different joint configurations. We define the resulting contrastive loss as:

$$\mathcal{L}_p = -\frac{1}{N_\ell} \sum_i \log \left[\frac{\exp\left(\left\langle \phi_i^\mathcal{A}, \phi_i^\mathcal{B} \right\rangle / \tau\right)}{\sum_j \omega_{ij} \exp\left(\left\langle \phi_i^\mathcal{A}, \phi_j^\mathcal{B} \right\rangle / \tau\right)} \right], \tag{1}$$

$$\omega_{ij} = \begin{cases} \frac{\tanh\left(\lambda \left\| p_i^\mathcal{B} - p_j^\mathcal{B} \right\|_2\right)}{\max\left(\tanh\left(\lambda \left\| p_i^\mathcal{B} - p_j^\mathcal{B} \right\|_2\right)\right)}, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the cosine similarity between two vectors, $p_i^\mathcal{B}$ represents the i-th point position in $\mathbf{P}^\mathcal{B}$. For the hyperparameters, we set $\tau = 0.1$ and $\lambda = 10$ in practice. Note that the learned features are finger configuration-invariant but dependent on the wrist pose.

A.2 $\mathcal{D}(\mathcal{R}, \mathcal{O})$ Prediction

Denote an open-hand configuration as q_init, of which the wrist pose can be either user-specified or randomly generated. Let the robot point cloud under q_init be $\mathbf{P}^\mathcal{R} = \text{FK}\left(q_\text{init}, \{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}\right) \in \mathbb{R}^{N_\mathcal{R} \times 3}$, and the object point cloud be $\mathbf{P}^\mathcal{O} \in \mathbb{R}^{N_\mathcal{O} \times 3}$, where $N_\mathcal{O}$ represents the number of points in the object point cloud, also set to 512 in practice. The objective of our neural network is to predict the point-to-point distance matrix $\mathcal{D}(\mathcal{R}, \mathcal{O}) \in \mathbb{R}^{N_\mathcal{R} \times N_\mathcal{O}}$.

Point Cloud Feature Extraction We begin by extracting point cloud embeddings using two encoders, $f_{\theta_\mathcal{R}}(\mathbf{P}^\mathcal{R})$ and $f_{\theta_\mathcal{O}}(\mathbf{P}^\mathcal{O})$, which share the same architecture. Specifically, we use a modified DGCNN [19] to better capture local structures and integrate global information (see Appendix F.1). The robot encoder is initialized with pretrained parameters, using the method described in Sec. 2.1, and remains frozen during training. These encoders extract point-wise features, $\boldsymbol{\phi}^\mathcal{R}$ and $\boldsymbol{\phi}^\mathcal{O}$ from the robot and object point clouds:

$$\boldsymbol{\phi}^\mathcal{R} = f_{\theta_\mathcal{R}}(\mathbf{P}^\mathcal{R}) \in \mathbb{R}^{N_\mathcal{R} \times D},\ \boldsymbol{\phi}^\mathcal{O} = f_{\theta_\mathcal{O}}(\mathbf{P}^\mathcal{O}) \in \mathbb{R}^{N_\mathcal{O} \times D}. \tag{3}$$

To establish correspondences between the robot and object features, we apply two multi-head cross-attention transformers [20], $g_{\theta_\mathcal{R}}(\boldsymbol{\phi}^\mathcal{R}, \boldsymbol{\phi}^\mathcal{O})$ and $g_{\theta_\mathcal{O}}(\boldsymbol{\phi}^\mathcal{O}, \boldsymbol{\phi}^\mathcal{R})$. These transformers integrate the relationships between the two feature sets, embedding correspondence information. This process maps the robot and object features to two sets of correlated features, $\boldsymbol{\psi}^\mathcal{R}$ and $\boldsymbol{\psi}^\mathcal{O}$:

$$\boldsymbol{\psi}^\mathcal{R} = g_{\theta_\mathcal{R}}(\boldsymbol{\phi}^\mathcal{R}, \boldsymbol{\phi}^\mathcal{O}) + \boldsymbol{\phi}^\mathcal{R} \in \mathbb{R}^{N_\mathcal{R} \times D},\ \boldsymbol{\psi}^\mathcal{O} = g_{\theta_\mathcal{O}}(\boldsymbol{\phi}^\mathcal{O}, \boldsymbol{\phi}^\mathcal{R}) + \boldsymbol{\phi}^\mathcal{O} \in \mathbb{R}^{N_\mathcal{O} \times D}. \tag{4}$$

CVAE-based $\mathcal{D}(\mathcal{R}, \mathcal{O})$ Prediction To achieve cross-embodiment grasp diversity, we employ a Conditional Variational Autoencoder (CVAE) [13] network to capture variations across numerous combinations of hand, object, and grasp configurations. The CVAE encoder $f_{\theta_\mathcal{G}}$ takes the robot and object point clouds under the grasp pose $\mathbf{P}^\mathcal{G} \in \mathbb{R}^{(N_\mathcal{R}+N_\mathcal{O})\times 3}$, along with the learned features $(\boldsymbol{\psi}^\mathcal{R}, \boldsymbol{\psi}^\mathcal{O})$, resulting in an input shape of $(N_\mathcal{R} + N_\mathcal{O}) \times (3 + D)$. The encoder outputs the latent variable $z \in \mathbb{R}^d$, set as $d = 64$ in practice. We concatenate z with extracted features $\boldsymbol{\psi}^\mathcal{R}$ and $\boldsymbol{\psi}^\mathcal{O}$, converting the feature to $\widehat{\boldsymbol{\psi}}_i^\mathcal{R}, \widehat{\boldsymbol{\psi}}_j^\mathcal{O} \in \mathbb{R}^{N_\mathcal{O}\times(D+d)}$.

The same kernel function $\mathcal{K}$ as Eisner et al. [14] is adopted, which possesses the properties of non-negativity and symmetry, to predict pair-wise distance $r_{ij} = \mathcal{K}(\widehat{\boldsymbol{\psi}}_i^\mathcal{R}, \widehat{\boldsymbol{\psi}}_j^\mathcal{O}) \in \mathbb{R}^+$ under the grasp pose:

$$\mathcal{K}(\widehat{\boldsymbol{\psi}}_i^\mathcal{R}, \widehat{\boldsymbol{\psi}}_j^\mathcal{O}) = \sigma\left(\frac{1}{2}\mathcal{N}_\theta\left(\widehat{\boldsymbol{\psi}}_i^\mathcal{R}, \widehat{\boldsymbol{\psi}}_j^\mathcal{O}\right) + \frac{1}{2}\mathcal{N}_\theta\left(\widehat{\boldsymbol{\psi}}_j^\mathcal{O}, \widehat{\boldsymbol{\psi}}_i^\mathcal{R}\right)\right), \tag{5}$$

where σ denotes the softplus function, and $\mathcal{N}_\theta$ is an MLP, which takes in the feature of $\mathbb{R}^{N_\mathcal{O}\times(2D+2d)}$ and outputs a positive number. By calculating on all $(\widehat{\boldsymbol{\psi}}_i^\mathcal{R}, \widehat{\boldsymbol{\psi}}_j^\mathcal{O})$ pairs, we obtain the complete $\mathcal{D}(\mathcal{R}, \mathcal{O})$ representation:

$$\mathcal{D}(\mathcal{R}, \mathcal{O}) = \begin{bmatrix} \mathcal{K}(\widehat{\boldsymbol{\psi}}_1^\mathcal{R}, \widehat{\boldsymbol{\psi}}_1^\mathcal{O}) & \cdots & \mathcal{K}(\widehat{\boldsymbol{\psi}}_1^\mathcal{R}, \widehat{\boldsymbol{\psi}}_{N_\mathcal{O}}^\mathcal{O}) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\widehat{\boldsymbol{\psi}}_{N_\mathcal{R}}^\mathcal{R}, \widehat{\boldsymbol{\psi}}_1^\mathcal{O}) & \cdots & \mathcal{K}(\widehat{\boldsymbol{\psi}}_{N_\mathcal{R}}^\mathcal{R}, \widehat{\boldsymbol{\psi}}_{N_\mathcal{O}}^\mathcal{O}) \end{bmatrix}. \tag{6}$$

A.3 Grasp Configuration Generation from $\mathcal{D}(\mathcal{R}, \mathcal{O})$

Given the predicted $\mathcal{D}(\mathcal{R}, \mathcal{O})$, we discuss how to generate the grasp joint values to grasp the object. We first calculate the robot grasp point cloud, then estimate each link's 6D pose based on the joint clouds. The system calculates the joint values by matching each link's 6D pose.

Robotic Grasp Pose Point Cloud Generation For a given point $p_i^\mathcal{R}$, the i-th row of $\mathcal{D}(\mathcal{R}, \mathcal{O})$ denotes the distances from this robot grasp point to all points in the object point cloud. Given the object point cloud, the multilateration method [15] positions the robot point cloud. This positioning technique determines the location of a point $p_i'^\mathcal{R}$ by solving the least-squares optimization problem based on distances from multiple reference points:

$$p_i'^\mathcal{R} = \arg\min_{p_i^\mathcal{R}} \sum_{j=1}^{N_\mathcal{O}} \left(\|p_i^\mathcal{R} - p_j^\mathcal{O}\|_2^2 - \mathcal{D}(\mathcal{R}, \mathcal{O})_{ij}^2\right)^2. \tag{7}$$

As shown in Zhou [21], this problem has a closed-form solution, and by using the implementation from Eisner et al. [14], we can directly compute $p_i'^\mathcal{R}$. Repeating this process for each row of

$\mathcal{D}(\mathcal{R},\mathcal{O})$ yields the complete predicted robot point cloud $\mathbf{P}^{\mathcal{P}}$ in the grasp pose. In 3D space, we can determine a point's position by measuring its relative distances to just three other points. Our $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation provides $N_{\mathcal{O}}(=512)$ relative distances, enhancing robustness to prediction errors.

6D Pose Estimation of Links Directly solving inverse kinematics and getting the joint values from a point cloud is not a trivial task. We first compute the 6D pose of each link in the world frame. As described in Sec. 2.1, we store the point cloud for each link, $\{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}$. Given the predicted grasp point cloud $\{\mathbf{P}_{\ell_i}^{\mathcal{P}}\}_{i=1}^{N_\ell}$, we calculate the 6D pose of each link using rigid body registration techniques:

$$\boldsymbol{\mathcal{T}}^{*}=(\mathbf{x}_i^*,\mathbf{R}_i^*)=\underset{(\mathbf{x}_i,\mathbf{R}_i)}{\arg\min}\,\|\mathbf{P}_{\ell_i}^{\mathcal{P}}-\mathbf{P}_{\ell_i}(\mathbf{x}_i,\mathbf{R}_i)\|^2, \tag{8}$$

where $\mathbf{x}_i$ and $\mathbf{R}_i$ represent the translation and rotation of the i-th link, respectively.

Joint Configuration Optimization After predicting the 6D pose for each link, our objective is to optimize the joint values to align the translation of each link with the predicted result. Starting from an initial value q_{init}, we iteratively solve the following optimization problem using CVXPY [16]:

$$\min_{\delta\boldsymbol{q}}\left(\sum_{i=1}^{N_\ell}\left\|\mathbf{x}_i+\frac{\partial\mathbf{x}_i(\boldsymbol{q})}{\partial\boldsymbol{q}}\delta\boldsymbol{q}-\mathbf{x}_i^*\right\|_2\right)\quad \text{s.t. } \boldsymbol{q}+\delta\boldsymbol{q}\in[\boldsymbol{q}_{min},\boldsymbol{q}_{max}],\ |\delta\boldsymbol{q}|\le\varepsilon_q. \tag{9}$$

In each iteration, the system computes the delta joint values $\delta\boldsymbol{q}$ by minimizing the objective function and updates the joint values as $\boldsymbol{q}\leftarrow\boldsymbol{q}+\delta\boldsymbol{q}$. Here, $\mathbf{x}_i$ represents the current link translation, $[\boldsymbol{q}_{min},\boldsymbol{q}_{max}]$ denotes the joint limits, and $\varepsilon_q=0.5$ is the maximum allowable step size. The optimization process can be efficiently parallelized, typically achieving convergence within one second, even for a 6+22 DoF ShadowHand.

A.4 Loss Function

The training objectives of the whole network include four parts, including the prediction of $\mathcal{D}(\mathcal{R},\mathcal{O})$ and $\boldsymbol{\mathcal{T}}$, the suppression of penetration, and the KL divergence of the CVAE latent variable:

$$\begin{aligned}\mathcal{L}=\;&\lambda_{\mathcal{D}}\mathcal{L}_{\text{L1}}\left(\mathcal{D}(\mathcal{R},\mathcal{O}),\mathcal{D}(\mathcal{R},\mathcal{O})^{\text{GT}}\right)+\lambda_{\mathcal{T}}\frac{1}{N_\ell}\sum_{i=1}^{N_\ell}\mathcal{L}_{\ell_i}\\&+\lambda_{\mathcal{P}}\left|\mathcal{L}_{\text{P}}(\mathbf{P}^{\mathcal{T}},\mathbf{P}^{\mathcal{O}})\right|+\lambda_{KL}\mathcal{D}_{KL}\left(f_{\theta_{\mathcal{G}}}(\mathbf{P}^{\mathcal{G}},\boldsymbol{\psi}^{\mathcal{R}},\boldsymbol{\psi}^{\mathcal{O}})\parallel\mathcal{N}(0,I)\right),\end{aligned} \tag{10}$$

where $\lambda_{\mathcal{D}}$, $\lambda_{\mathcal{T}}$, $\lambda_{\mathcal{P}}$, λ_{KL} are hyperparameters for loss weights.The superscript 'GT' refers to the ground truth annotations. $\mathcal{N}(0,I)$ is a standard Guassian distribution, and $\mathbf{P}^{\mathcal{T}}$ is the robot point cloud under the $\boldsymbol{\mathcal{T}}^{*}$ described in A.3. $\mathcal{L}_{\text{P}}$ computes the sum of the negative values of the signed distance function (SDF) of $\mathbf{P}^{\mathcal{T}}$ to $\mathbf{P}^{\mathcal{O}}$ to penalize any penetration between the robot hand and the object, and $\mathcal{L}_\ell$ computes the difference between two 6D poses:

$$\mathcal{L}_{\ell_i}=\|\mathbf{x}_i^*-\mathbf{x}_i^{\text{GT}}\|_2+\arccos\left(\frac{\text{tr}(\mathbf{R}_i^{*\text{T}}\mathbf{R}_i^{\text{GT}})-1}{2}\right). \tag{11}$$

Notably, the computation from $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation to the 6D pose $\boldsymbol{\mathcal{T}}^*$ shown in Eqn. 8 is entirely matrix-based, ensuring differentiability for loss backpropagation and computational efficiency.

B Dataset Details

We utilized a subset of the MultiDex dataset [7] (See Appendix D.2 for the filtering process). After filtering, 24,764 valid grasps remained. We adopt three robots from the dataset: Barrett (3-finger), Allegro (4-finger), and ShadowHand (5-finger). Each grasp defines its associated object, robot, and grasp configurations. We retain the same training and test dataset splits as in the MultiDex dataset.

B.1 URDF File Preprocessing

To facilitate optimization, we introduce six virtual joints between the world frame and the robot's root link: three prismatic joints representing translation (x, y, z) and three revolute joints representing rotation $(roll, pitch, yaw)$. These virtual joints are incorporated into the robot's URDF file and treated equivalently to other joints to simplify the computation of the Jacobian matrix. Furthermore, virtual links are added to the distal ends of each tip link to address potential errors in the 6D pose during optimization, ensuring consistent constraints across all links despite reduced rotational restrictions.

B.2 Robot Point Cloud Sampling

To extract the stored point clouds $\{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}$ from the URDF file of a specific robot, we first sample 512 points from the mesh of each link. We then apply the Farthest Point Sampling (FPS) algorithm to the complete point cloud, selecting 512 points, denoted as N_R in our method. These point clouds are stored separately for each distinct link.

This process guarantees that, for any joint configuration, our point cloud forward kinematics model, FK $\left(q, \{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}\right)$, can map joint configurations to corresponding point clouds at new poses. This ensures consistent point cloud correspondence across different poses, a key advantage for our pretraining methodology.

B.3 Object Point Cloud Sampling

Starting with the mesh file of an object, we initially sample 65,536 points. For each training iteration, we randomly select 512 points from this set and apply Gaussian noise $\mathcal{N}(0, 0.002)$ for data augmentation. This strategy improves the model's generalization across different object shapes.

C Experiment Details

C.1 Diverse Grasp Synthesis

Grasping diversity includes two key aspects: the wrist pose and the finger joint values. Since the input and grasp rotations in the training data are correspondingly aligned, the model learns to implicitly map these rotations. This alignment enables the model, during inference, to generate appropriate grasps based on the specified input orientation. Fig. 6 illustrates the grasp results for six different input directions, showing that our model consistently produces feasible grasps, demonstrating the controllability of our method. Additionally, by sampling the latent variable $z \in \mathbb{R}^{64}$ from $\mathcal{N}(0, I)$, our model can generate multiple grasps in the same direction. As shown in Tab. 2, the diversity of our method is highly competitive.

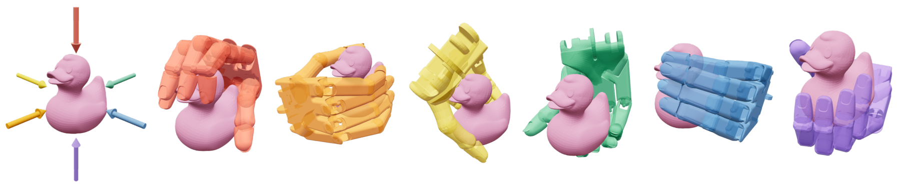


Figure 6: Diverse and pose-controllable grasp generation. The arrow refers to the input palm orientation. Arrows and hands of the same color represent corresponding input-output pairs.

C.2 Configuration Correspondence Learning

As described in Sec. 2.1, our proposed configuration-invariant pretraining method learns an inherent alignment across varying robotic hand configurations. We visualize the learned correspondence in Fig. 7, where each point in the closed-hand pose is colored according to the highest cosine similarity with its counterpart in the open-hand pose. The excellent color matching within the same hand demonstrates that the pretrained encoder successfully captures this alignment. Furthermore, strong matching across different hands highlights the transferability of features. As shown in Tab. 2, removing the pretraining parameters and training the robot encoder directly results in performance degradation across robotic hands, confirming the effectiveness of the pretrained model.

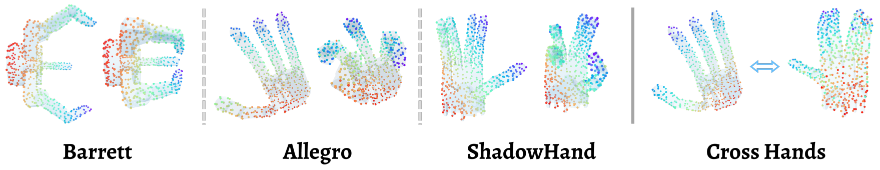


Figure 7: Visualization of the pretrained point matching.

C.3 Grasping with Partial Object Point Cloud Input

A common challenge in real-world experiments is the noise and incompleteness of point clouds from depth cameras. Object-centric methods that rely on full object visibility often suffer performance degradation under such conditions. In contrast, the relative distance feature of $\mathcal{D}(\mathcal{R}, \mathcal{O})$ allows our method to infer the robot point cloud even from partial observation. We validated this approach through experiments. Starting with the mesh of an object, we randomly sample $2 \times N_{\mathcal{O}}$ points. Next, a point is randomly sampled on a unit sphere, and the direction vector r from this point to the origin is computed. For each point in the point cloud, we calculate the dot product between r and the corresponding direction vectors d_i. We then remove half of the points with the smallest dot product values $r \cdot d_i$, leaving a subset of $N_{\mathcal{O}}$ points, which forms the partial object point cloud. This process is used to generate random point clouds during both training and evaluation. This setup simulates the incomplete data commonly encountered in practice. As shown in Fig. 8 and the third row of Tab. 3, even with partial point clouds, our model can successfully predict feasible grasps (Q5), indicating robustness when faced with incomplete input.

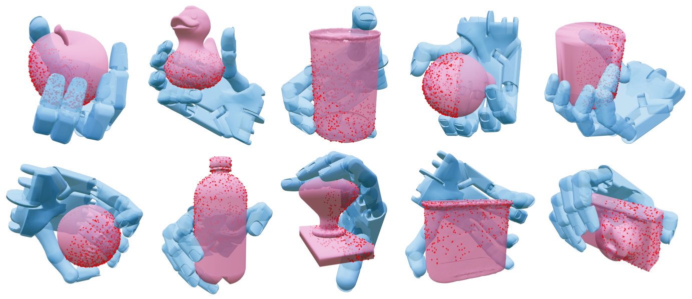


Figure 8: Grasp examples with partial object point clouds. Red points show the observed portion.

C.4 Zero-shot Generalization to Novel Hands Experiment

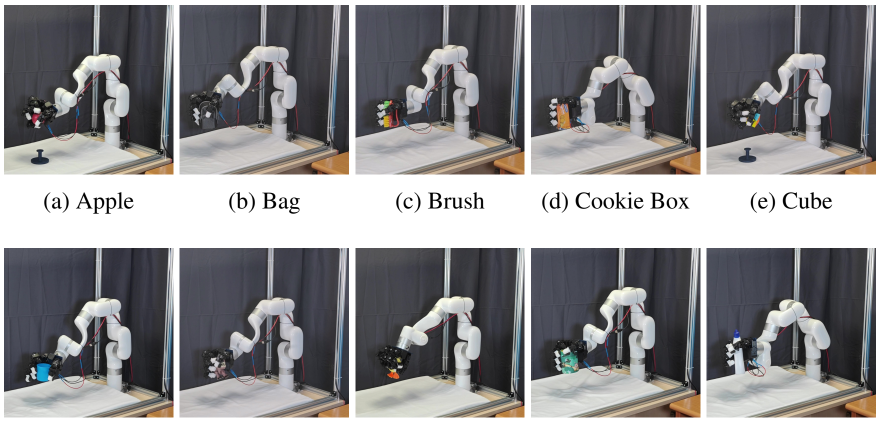
We trained the model separately on each of the three robotic hands and then validated it on the others without further training. As shown in Tab. 4, the results indicate that when transferring from high-DOF hands to low-DOF hands in a zero-shot setting, the model retains a certain level of performance. However, transferring in the opposite direction largely fails. We hypothesize that this difference arises because high-DOF hands have a much more complex configuration space, allowing the model to learn a broader range of articulation-invariant matching tasks, which can still perform well on the simpler articulation-invariant tasks required for low-DOF hands. In contrast, the configuration space of low-DOF hands is relatively simple, and when trained on these hands, the model can only master simple articulation-invariant matching tasks.

C.5 Real-World Experiment Details

Dataset Collection, Pretraining and Training For the real-world experiments, we collected the LEAP Hand dataset and trained a model independently. We initially selected 78 daily objects from the YCB dataset [22] and ContactDB [23], then applied the DFC-based [12] grasp optimization method from [24] to generate 1,000 grasps per object, yielding a total of 78,000 grasps. Following

<table>
<tr><th rowspan="2">Training Robot</th><th colspan="3">Success Rate (%) ↑</th></tr>
<tr><th>Allegro</th><th>Barrett</th><th>ShadowHand</th></tr>
<tr><td>Allegro</td><td>(88.70)</td><td>83.60</td><td>1.10</td></tr>
<tr><td>Barrett</td><td>42.40</td><td>(84.80)</td><td>6.90</td></tr>
<tr><td>Shadowhand</td><td>56.90</td><td>83.70</td><td>(75.80)</td></tr>
</table>

Table 4: Generalization results to novel hands.



(a) Apple (b) Bag (c) Brush (d) Cookie Box (e) Cube

(f) Cup (g) Dinosaur (h) Duck (i) Tea Box (j) Toilet Cleaner

Figure 9: Real-world grasp demonstrations

a dataset filtering process, we obtained 24,656 grasps across 73 objects. The encoder network was first pretrained on the original dataset, and the entire model was then trained on the filtered dataset, as described in Sec. 2.

Real-World Deployment Details We first scanned the objects listed in Tab. 5 using AR Code [25]. After camera intrinsics [26] and extrinsics [27] calibration, we estimated object poses using FoundationPose [28] and sampled point cloud uniformly on their surfaces. In this tabletop grasping setting, only top-down and side grasps are feasible, as other palm orientations would likely collide with the table. To address this, the model took as input the sampled object point clouds and a batch of LEAP Hand point clouds, which corresponded to 32 interpolated hand poses ranging from top-down to right-side orientations, enabled by our palm orientation control functionality. We randomly selected one of the top-5 grasps from the generated batch, ranked according to the same grasp energy calculation used during dataset generation [24]. We then use MPLib [29] for arm motion planning to the desired end-effector pose. A PD controller is applied for grasp execution.

Experiment Result We tested 10 objects with various shapes, performing 10 grasping attempts for each object. The experimental results are shown in Tab. 5 and Fig. 9. Our method achieved an average success rate of **89%** across these 10 objects, demonstrating the effectiveness of our method in dexterous grasping and its generalizability to novel objects.

Apple	Bag	Brush	Cookie Box	Cube	Cup	Dinosaur	Duck	Tea Box	Toilet Cleaner
9/10	10/10	9/10	10/10	9/10	7/10	9/10	8/10	8/10	10/10

Table 5: Real-world experiment results on unseen objects.

D Grasp Controller

To mitigate minor inaccuracies and subtle penetrations commonly found in generative methods, as well as the limitations of directly predicting a static grasp pose—which overlooks the forces exerted on contact surfaces—we developed a heuristic grasp controller to better simulate real-world grasping scenarios. The controller aims to generate a configuration q_{outer} that is farther from the object's center

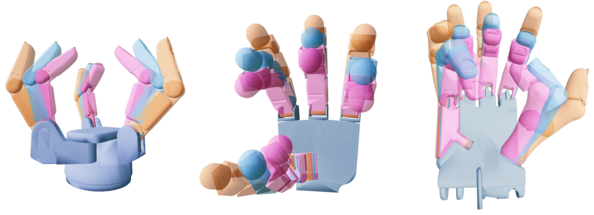


Figure 10: Visualization of the grasp controller's effect: blue indicates the predicted grasp pose, orange represents q_{outer}, and pink represents q_{inner}.

of mass and a configuration q_{inner} that is closer to the center of mass, based on the predicted pose. Fig. 10 illustrates the impact of the grasp controller.

D.1 Evaluation Metric Details

In Isaac Gym, we evaluate the success of a grasp through a two-phase process. First, in the grasp phase, we use the previously described grasp controller to compute q_{outer} and q_{inner}. We set the robot joint position to q_{outer} with a position target at q_{inner}. Then we simulate for 1 second, equivalent to 100 simulation steps for the hand to close and grasp. In the second phase, we apply disturbance forces sequentially along six orthogonal directions, following the method in [7]. These forces are defined as:

$$F_{\pm xyz} = 0.5m/s^2 \times m_{\text{object}} \tag{12}$$

where m_{object} denotes the mass of the object.

Our approach improves upon [7] by introducing a dynamic grasp phase, transitioning the evaluation from static to dynamic, and thereby significantly enhancing the rigor of the evaluation metric. In the original static validation, some grasps could hold objects in unstable positions. By introducing dynamic validation, these unstable grasps are less likely to succeed, resulting in a more stringent and accurate assessment of grasp quality. Moreover, static validation is prone to simulation errors, such as object penetration or robot self-collisions, which can incorrectly classify unstable grasps as successful. The dynamic method alleviates these issues, providing a more robust and reliable evaluation of grasp success.

Fig. 11 illustrates several anomalous grasps that, despite appearing to fail, could still be judged as successful under the static metric. These grasps, either in an unstable state or exhibiting significant self-penetration, are impractical for real-world applications, highlighting the limitations of static validation.

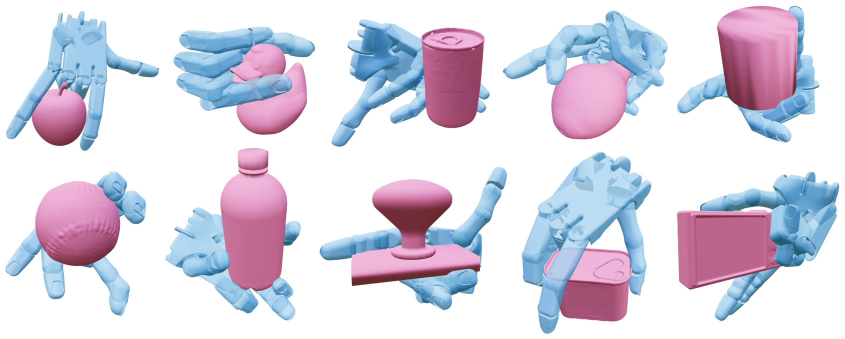


Figure 11: Grasp examples filtered out from the dataset that would otherwise be deemed successful under static metric.

D.2 Dataset Filtering

To address the suboptimal grasp quality, we applied a filtering process to the CMapDataset [7]. Specifically, each grasp in the dataset was evaluated based on the success metrics defined in Appendix D.1. We then store the relative 6D pose and joint values of every successful grasp in the filtered dataset.

E Baseline Description

DFC is an optimization-based approach that searches for feasible grasp configurations through iterative optimization. GenDexGrasp predicts contact heatmaps and uses optimization to determine grasp poses. ManiFM outputs contact points on the object and associated contact forces or post-contact motions for robots to achieve the desired manipulation task.

E.1 DFC [12]

Since DFC is a purely optimization-based method, the speed of generating grasps is particularly slow. Therefore, we evaluate it using the original CMapDataset, which was primarily generated by the DFC method. As the dataset generation process also minimizes the hand prior energy and penetration energy described in [7], and some generated grasps may have already been filtered, the evaluation results are likely better than DFC's actual performance.

E.2 GenDexGrasp [7]

We used the filtered grasp dataset to train the model, where the contact heatmap was generated using the aligned distance as described in the paper. The GenDexGrasp model was trained with default hyperparameters. In Tab. 6, we compared the results of the open-source pretrained model with those of our trained model, demonstrating that our filtered dataset is of higher quality.

Method	**Success Rate (%) ↑**			
	Allegro	Barrett	ShadowHand	Avg.
pretrain	51.00	63.80	44.50	53.10
train	51.00	67.00	54.20	57.40

Table 6: GenDexGrasp Result Comparision

E.3 ManiFM [8]

Due to the unavailability of pretrained models for Barrett and ShadowHand, our evaluation was restricted to the Allegro pretrained model. Considering the fundamental differences between point-contact and surface-contact grasps, we optimized the controller's hyperparameters for improved performance of ManiFM. Nevertheless, despite the seemingly high quality of the generated grasps, the inherent instability of point-contact grasps posed significant challenges in achieving a high success rate during simulation.

E.4 GeoMatch [5]

Although GeoMatch is a keypoint matching-based method that supports cross-embodiment and shares similarities with our approach, we faced challenges in reproducing its results due to the absence of pretrained models and insufficient details regarding the data file formats, which remained unsolved in its repository's issues as well. Consequently, it was not included as a baseline for comparison.

F Network Architecture

F.1 Point Cloud Encoder

To map robot and object features into a shared feature space, enhancing the network's ability to learn correspondences between them, we employed identical architectures for both the robot and object encoders. Our encoder design is based on the DGCNN [19] architecture, as implemented

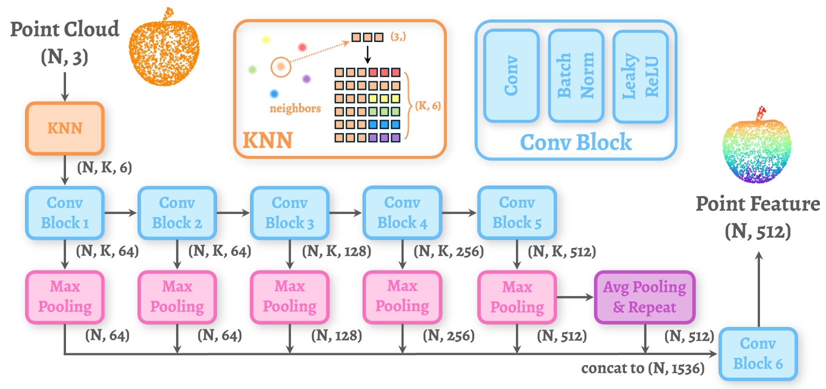


Figure 12: Point cloud encoder architecture.

in [14]. Notably, this implementation omits the original layer-wise re-computation of K-nearest neighbors (KNN) for graph construction, resulting in a "Static Graph CNN". In our setup, K is set to 32, meaning that each point's receptive field is much smaller than the total number of points in the cloud ($N_{\mathcal{R}} = 512$). This constraint limits the ability of the per-point feature extraction process to capture global information, which poses a challenge for the object encoder, as it struggles to learn comprehensive geometric shape features.

We experimented with the original dynamic graph structure, but it led to a decline in pretraining performance. We hypothesize that, for configuration-invariant learning objectives, local structural information in the point cloud is critical, and the network needs to be reinforced to capture this. The dynamic graph structure tends to learn similar structures across different fingers, which, while beneficial for segmentation tasks, is less suited to our specific learning goals. The impact of varying network architectures and feature learning strategies will be further explored in future work.

Consequently, our encoder follows a "Static Graph CNN" architecture with five convolutional layers. After the last convolution, a global average pooling layer generates a global feature concatenated with features from all previous layers. This combined output is passed through a final convolutional layer, projecting into the embedding dimension. The architecture is illustrated in Fig. 12, where the LeakyReLU activation function uses a negative slope of 0.2.

F.2 Cross-Attention Transformer

We followed the architectural design from [14], utilizing a multi-head attention block with 4 heads. The implementation details can be found in the code.

F.3 Kernel MLP

We adopted the same hyperparameters design as [14]. Specifically, the MLP consists of two hidden layers with feature dimensions of 300 and 100, respectively, along with the ReLU activation function.

G Matrix Block Computation

To address the high GPU memory demands of using the MLP kernel function to compute $\mathcal{D}(\mathcal{R}, \mathcal{O})$, we implemented a matrix block computation strategy to optimize memory usage. After experimentation, we ultimately chose to divide the entire matrix into 4×4 blocks for computation, which reduces memory consumption by approximately 34% while maintaining similar computation time.