$\mathcal{D}(\mathcal{R}, \mathcal{O})$ 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 crucial in robotics as the first step in executing complex manipulation tasks. However, quickly obtaining a high-quality and diverse set of grasps remains challenging for dexterous robotic hands due to their high degrees of freedom and the complexities involved in achieving stable, precise grasps. Researchers have developed several optimization-based methods to address this challenge [1, 2, 3, 4, 5]. Some of these methods, however, often focus on fingertip point contact, relying on complete object shape, and require significant computational time to optimize. As a result, data-driven grasp generation methods have gained attention. These methods aim to solve the grasping problem using learning-based techniques. We can broadly categorize them into two types: those that utilize robot-centric representations, such as

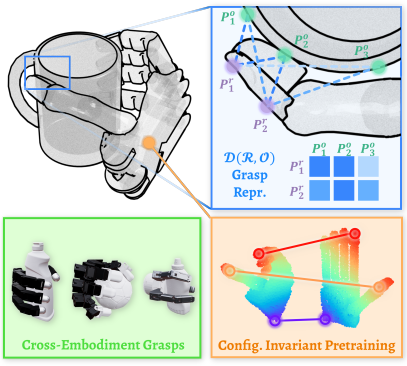

Figure 1: We propose our model that utilizes configuration-invariant pretraining, predicts $\mathcal{D}(\mathcal{R}, \mathcal{O})$ representation, and obtains grasps for cross-embodiment from point cloud input.

	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 [2]	Joint Values	Robot-centric	✓	✗✗	-	✗	✓	✗
UniDexGrasp++ [8]	Joint Values	Robot-centric	✗	✓✓	✗	✓	✓	✗
UniGrasp [9]	Contact Point	Object-centric	✓	✗	✓	✗	✗	✗
GeoMatch [10]	Contact Point	Object-centric	✓	✗	✓	✗	✓	✗
GenDexGrasp [12]	Contact Map	Object-centric	✓	✗	✓	✗	✓	✗
ManiFM [13]	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.

wrist poses and joint values [6, 7, 8], and those that rely on object-centric representations, such as contact points [9, 10, 11] or contact maps [12, 13, 14, 15].

Robot-centric representations (e.g., joint values), as used in methods like UniDexGrasp++ [8], directly map observation to control commands for fast inference but suffer from low sample efficiency and poor generalization across different robot embodiments. The learned mappings are specific to the training data and do not quickly adapt to new robot designs or geometries. Object-centric representations (e.g., key points, contact points, affordances) effectively capture the geometry and contacts of objects, allowing for generalization across different shapes and robots, as demonstrated by methods like UniGrasp [9] and GenDexGrasp [12]. However, these methods are often less efficient as they typically require an additional optimization step—such as solving fingertip inverse kinematics (IK) or fitting the predicted contact maps under penetration-free and joint limit constraints to translate the object-centric representation into actionable robot commands. This optimization process is time-consuming due to its complexity and nonconvexity [16, 17, 18].

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.

Given the point clouds of both an open robotic hand and the object, our network architecture predicts the $\mathcal{D}(\mathcal{R}, \mathcal{O})$ representation. This matrix encodes the relative distances between the point clouds of the object and the robotic hand in the desired grasping pose[19, 20]. Using this representation, we apply a multilateration method [21] to estimate the robot's point cloud at the predicted pose, allowing us to compute the 6D pose of each hand link in the world frame and ultimately determine the joint configurations. To encode robotic hands, we propose a configuration-invariant pretraining method that learns the inherent alignment between various hand configurations, promoting grasp generation performance and cross-embodiment generalization. We validate the effectiveness of our approach through extensive experiments in both simulation and real-world settings. Our model achieves an average success rate of 87.53% in simulation across three dexterous robotic hands and in real-robot experiments, demonstrating its robustness and versatility.

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

Given the object point cloud and the robot hand URDF file, our goal is to generate dexterous and diverse grasping poses that generalize across various objects and robot hands. Fig. 2 provides an overview of our proposed method.

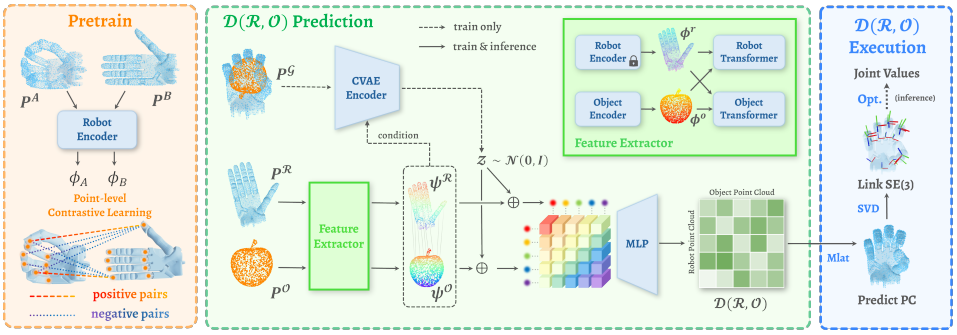

Figure 2: Overview of $\mathcal{D}(\mathcal{R},\mathcal{O})$ framework: We first pretrain the robot encoder with the proposed configuration-invariant pretraining method. Then, we predict the $\mathcal{D}(\mathcal{R},\mathcal{O})$ representation between the robot and object point cloud. Finally, we extract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

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).

2.1 Configuration-Invariant Pretraining with Contrastive Learning

Learning dexterous grasping involves understanding the spatial relationships between the robot hand and the object. The objective is to match the robot hand in a specific configuration with the object. However, this matching process is challenging because the local geometric features of a point in the open-hand configuration may not align with those in the grasp configuration due to huge variations during articulation.

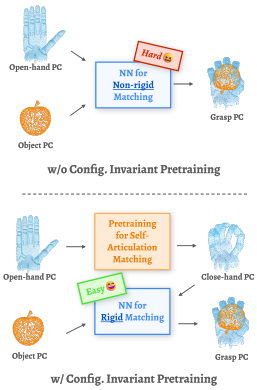

Figure 3: Motivation for configuration-invariant pretraining.

To address this, we break 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. 3, 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 $\boldsymbol{\phi}^\mathcal{A}, \boldsymbol{\phi}^\mathcal{B} \in \mathbb{R}^{N_\mathcal{R}\times D}$, 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:

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

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

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.

2.2 $\mathcal{D}(\mathcal{R}, \mathcal{O})$ 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 $\mathbf{P}^{\mathcal{R}} = \text{FK}\left(q_{init}, \{\mathbf{P}_{\ell_i}\}_{i=1}^{N_\ell}\right) \in \mathbb{R}^{N_{\mathcal{R}} \times 3}$, 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 [22] to better capture local structures and integrate global information (see Appendix). 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 [23] (see Appendix), $g_{\theta_{\mathcal{R}}}(\boldsymbol{\phi}^{\mathcal{R}}, \boldsymbol{\phi}^{\mathcal{O}})$ 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) [24] 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. [25] 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 (see Appendix). 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:

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

2.3 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 [21] 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 [26], this problem has a closed-form solution, and by using the implementation from Eisner et al. [25], 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 $\left\{\mathbf{P}_{\ell_i}^{\mathcal{P}}\right\}_{i=1}^{N_\ell}$, we calculate the 6D pose of each link using rigid body registration techniques:

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

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 [27]:

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

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.

2.4 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:

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

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 $\mathcal{T}^*$ described in 2.3. $\mathcal{L}_{\text{P}}$ 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 $\mathcal{T}^*$ shown in Eqn. 8 is entirely matrix-based, ensuring differentiability for loss backpropagation and computational efficiency.

3 Experiments

In this section, we perform a series of experiments aimed at addressing the following questions (Q1-Q6):

Q1: How successful are our generated grasps?

Q2: Does our unified model train on multi-embodiment outperform models trained on single embodiments?

Q3: How diverse are our generated grasps?

Q4: How well does our pretraining learn configuration-invariant representations, and can this be transferred across different embodiments?

Q5: How robust is our approach with partial object point cloud input?

Q6: How does our method perform in real-world settings?

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 [28]. A simple PD controller is applied to execute the predicted grasps in the simulation. Certain forces are applied sequentially along six orthogonal directions, following the approach in Li et al. [12]. We apply each force for a duration of 1 second. 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.

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

3.2 Dataset

We utilized a subset of the MultiDex dataset [12] (See Appendix 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.

3.3 Overall Performance

Baselines To answer Q1, we present a detailed comparison of $\mathcal{D}(\mathcal{R}, \mathcal{O})$ against DFC [2], GenDexGrasp [12], and ManiFM [13], as shown in Tab. 2. This comparison includes diverse methods to

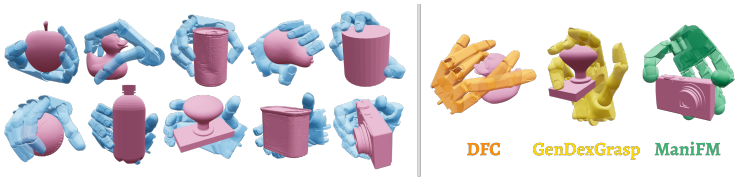

Figure 4: Visualization of all methods.

Method	Success Rate (%) ↑				Diversity (rad.) ↑			Efficiency (sec.) ↓		
	Barrett	Allegro	ShadowHand	Avg.	Barrett	Allegro	ShadowHand	Barrett	Allegro	ShadowHand
DFC [2]	86.30	76.21	58.80	73.77	**0.532**	**0.454**	0.435	>1800	>1800	>1800
GenDexGrasp [12]	67.00	51.00	54.20	57.40	0.488	0.389	0.318	14.67	25.10	19.34
ManiFM [13]	-	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.

address the challenge of cross-embodiment grasping from various perspectives. They were evaluated on 10 previously unseen test objects using the Barrett, Allegro, and ShadowHand robotic hands. 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 supports cross-embodiment grasping but employs a point-contact approach, which was not suitable for training on our dataset that emphasizes surface-contact methods. As a result, we can only evaluate its pretrained model of Allegro Hand for ManiFM.

Our experiments demonstrate that $\mathcal{D}(\mathcal{R}, \mathcal{O})$ significantly outperformed all baselines regarding success rate across the robots by a large margin, highlighting the effectiveness of our approach. For successful grasps of our method, the average displacement remains under 2 mm, with an average rotation below 1°, highlighting the firmness of our generated grasps. Fig. 4 visualizes grasps generated by our method alongside typical failure grasp poses from baselines. DFC often results in unnatural poses. GenDexGrasp struggles with objects of complex shapes, frequently encountering significant penetration issues. Although ManiFM produces visually appealing grasps, its point-contact method lacks stability, lowering its success rate in simulation.

From the first two rows of Tab. 3, we can see a slight improvement in success rates when training across multiple robots compared to training on a single hand, demonstrating the cross-embodiment generalizability of our method (Q2).

Our method significantly improves grasp generation speed. While DFC is slow in producing results and learning-based methods like GenDexGrasp and ManiFM take tens of seconds per grasp due to their complex optimization processes, our approach can generate a grasp within 1 second. This fast computation is crucial for dexterous manipulation tasks.

3.4 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. 5 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 $\mathcal{N}(0, I)$,

Figure 5: 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.

our model can generate multiple grasps in the same direction, addressing Q3. As shown in Tab. 2, the diversity of our method is highly competitive.

3.5 Configuration Correspondence Learning

As described in Sec. 2.1, our proposed configuration-invariant pretraining method learns an inherent alignment across varying robotic hand configurations. To answer Q4, we visualize the learned correspondence in Fig. 6, 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.

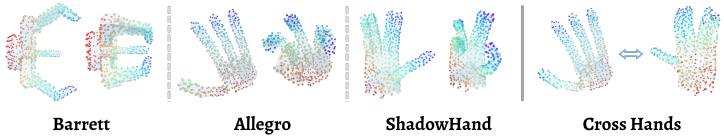

Figure 6: Visualization of the pretrained point matching.

3.6 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 by conducting experiments, removing 50% of the object point cloud in a contiguous region during both training and validation. This setup simulates the incomplete data commonly encountered in practice. As shown in 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.

3.7 Real-Robot Experiments

We conducted real-world experiments with a uFactory xArm6 robot, equipped with the LEAP Hand [29] and the overhead Realsense D435 camera, as illustrated in Fig. 7. As shown in Tab. 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 (Q6). For experiment videos, please visit our website https://drograsp.github.io/.

Figure 7: Real-world experiment setting.

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 4: Real-world experiment results on unseen objects.

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] M. A. Roa and R. Suárez. Grasp quality measures: review and performance. *Autonomous robots*, 38:65–88, 2015.

[2] 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.

[3] S. Chen, J. Bohg, and C. K. Liu. Springgrasp: An optimization pipeline for robust and compliant dexterous pre-grasp synthesis. *arXiv preprint arXiv:2404.13532*, 2024.

[4] A. Patel and S. Song. GET-Zero: Graph embodiment transformer for zero-shot embodiment generalization, 2024. URL https://arxiv.org/abs/2407.15002.

[5] S. Haldar, J. Pari, A. Rai, and L. Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt2: Learning precise manipulation from few demonstrations. *RSS*, 2024.

[18] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. *CoRL*, 2023.

[19] Y. Huang, C. Agia, J. Wu, T. Hermans, and J. Bohg. Points2plans: From point clouds to long-horizon plans with composable relational dynamics. *arXiv preprint arXiv:2408.14769*, 2024.

[20] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

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

[28] 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.

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