Dexterous Grasp Transformer

Supplementary Material

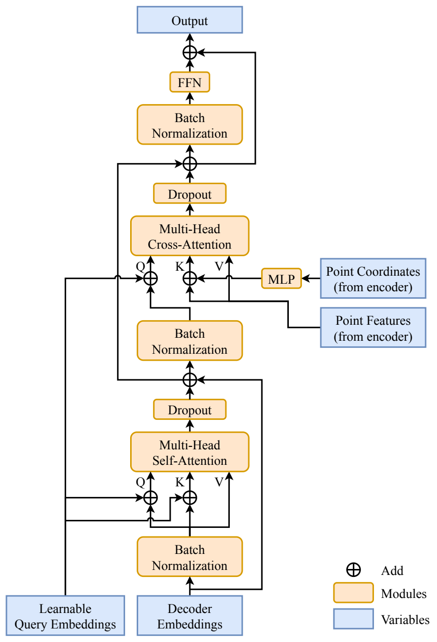
A. Method and Implementation Details

A.1. DGTR Architecture

Encoder. Although the conventional transformer encoder has been empirically proven to be capable of extracting intrinsic features of a point cloud by several works [5, 9, 14, 22], we follow recent works [10, 19, 20] in the field of robotic grasping and adopt a three-layer PointNet++ [15] as the encoder. Concretely, each layer $l_i, i \in \{1, 2, 3\}$ takes as input a point cloud $\mathcal{O}_i \in \mathbb{R}^{M_i \times 3}$ and its features (raw XYZ coordinate for the first layer) $\mathcal{F}_i \in \mathbb{R}^{M_i \times C_i}$ from the previous layer, then down-samples and aggregates the features by the "set-aggregation" operation [15], and finally processes the grouped features by a three-layer perceptron, which contains three $Linear - BatchNorm - ReLU$ blocks. In addition, we adopt kNN instead of "ball query" in the original PointNet++ as the grouping operation with the purpose of adapting to objects of various scales. The hyper-parameters for the encoder are shown in Table S1.

Decoder. We cascade Transformer blocks[18] as our decoder. Generally, the decoder takes the M' points and features from the encoder, as well as N learnable query embeddings $\{q_i\}_{i=1}^{N}$ as input, and extracts N features to predict the grasp poses afterwards. Each decoder layer mainly consists of (1) a self-attention layer for interactions between the decoder embeddings, (2) a cross-attention layer for aggregating the object features from the encoder, and (3) a feed-forward layer. At each decoder layer, the learnable query embeddings are added to the decoder embeddings to form the query and key in the self-attention stage and to form the query in the cross-attention stage. In addition, we encode the M' points $\mathcal{O}' \in \mathbb{R}^{M' \times 3}$ to position embeddings by a multi-layer perceptron (MLP) module, and point-wisely add them to encoder features $\mathcal{F}'$ to form the key in the cross-attention stage. The decoder embeddings and the encoder feature $\mathcal{F}'$ are used to form the value in the self-attention and cross-attention stages, respectively. Figure S1 shows the detail of one decoder layer. The hyper-parameters for the decoder are shown in Table S2.

Prediction Heads. We use three independent MLPs to predict the translations, rotations, and joint angles of the grasps with the final decoder embeddings. Each MLP has the same *Linear-BatchNorm-ReLU-Dropout-Linear* structure, and the hidden layer sizes are all 128. Both the translation and the joint angle predictions are passed through a sigmoid activation to generate a normalized value w.r.t. the limits of each dimension. For the rotation prediction, we normalize the MLP output to obtain the unit quaternions.



Output
FFN
Batch Normalization
Dropout
Multi-Head Cross-Attention
Q
K
V
MLP
Point Coordinates (from encoder)
Point Features (from encoder)
Batch Normalization
Dropout
Multi-Head Self-Attention
Q
K
V
Batch Normalization
Learnable Query Embeddings
Decoder Embeddings
Add
Modules
Variables

Figure S1. Detail of one decoder layer.

Hyper-parameters	$Layer1$	$Layer2$	$Layer3$
Input size	4096 (M)	1024	512
Output size	1024	512	256 (M')
k for kNN	32	16	8
Hidden layer dim. in MLP	(64, 64)	(128, 128)	(256, 256)
Dim. of output features	128	256	512 (C')

Table S1. Hyper-parameters for the DGTR encoder. *Dim.* stands for *Dimension*.

Hyper-parameters	Value
Num. of decoder layers	4
Num. of learnable query embeddings	16 (N)
Dimension in self-attention	256
Dimension in cross-attention	256
Dimension in feed-forward network	256

Table S2. Hyper-parameters for the DGTR decoder.

A.2. Grasp Losses

We provide more complementary details of our grasp losses in the section. To avoid ambiguity, we denote again the i^{th} predicted item as $\mathbf{x}_i$ and the j^{th} ground-truth item as $\hat{\mathbf{x}}_j$ ($\mathbf{x} \in \{\mathbf{g}, \mathbf{t}, \mathbf{r}, \mathbf{q}\}$) in this section.

Translation and Joint Angles. For translations $\mathbf{t}_i \in \mathbb{R}^3$ and joint angles $\mathbf{q}_i \in \mathbb{R}^J$ (J is the number of joints of the dexterous hand), we adopt the smooth $L1$ loss [4] to regress the ground truths:

$$\mathcal{L}_{smooth_{L1}}(\mathbf{x}_i, \hat{\mathbf{x}}_j) = \begin{cases} 0.5 * (\mathbf{x}_i - \hat{\mathbf{x}}_j)^2, & |\mathbf{x}_i - \hat{\mathbf{x}}_j| < 1 \\ |\mathbf{x}_i - \hat{\mathbf{x}}_j| - 0.5, & |\mathbf{x}_i - \hat{\mathbf{x}}_j| \geq 1 \end{cases} \tag{1}$$

Therefore, the translation regression loss is defined as $\mathcal{L}_{trans} = \mathcal{L}_{smooth_{L1}}(\mathbf{t}_i, \hat{\mathbf{t}}_j)$ and the joint angles regression loss is defined as $\mathcal{L}_{joints} = \mathcal{L}_{smooth_{L1}}(\mathbf{q}_i, \hat{\mathbf{q}}_j)$

Hand Chamfer Loss. We obtain the point cloud $\Phi(\mathbf{g}_i)$ and $\Phi(\hat{\mathbf{g}}_j)$ with the method mentioned in the paper. Then we use a chamfer distance the measure the discrepancy between the actual shapes of the predicted and the ground-truth hands. Given two point clouds Γ_i and Γ_j, the chamfer distance [3] between them is defined as:

$$\mathcal{D}_{chamfer}(\Gamma_i, \Gamma_j) = \sum_{x \in \Gamma_i} \underset{y \in \Gamma_j}{min} \|x - y\|_2^2 + \sum_{x \in \Gamma_j} \underset{y \in \Gamma_i}{min} \|x - y\|_2^2. \tag{2}$$

Therefore, the hand chamfer loss is defined as $\mathcal{L}_{chamfer}(\mathbf{g}_i, \hat{\mathbf{g}}_i) = \mathcal{D}_{chamfer}(\Phi(\mathbf{g}_i), \Phi(\hat{\mathbf{g}}_j))$.

Object Penetration Loss. We follow [21] to calculate the penetration depth with the signed distance function from the object point cloud to the hand mesh.

Self-Penetration Loss. Following previous works [20, 23], we apply self-penetration loss to prevent the dexterous hand from penetrating itself:

$$\mathcal{L}_{spen}(\Phi(\mathbf{g}_i)) = \sum_{p \in \Phi(\mathbf{g}_i)} \sum_{q \in \Phi(\mathbf{g}_i)} \mathbb{I}(p \neq q)\, max(\delta - d(p, q), 0). \tag{3}$$

Here $\mathbb{I}(\cdot)$ is the indicator function, and $d(p, q)$ is the $L2$ distance between the point p and q.

The hyper-parameters for the losses are listed in Table S3.

Hyper-parameters	Value
λ_1	10.0
λ_2	10.0
λ_3	10.0
λ_4	1.0
λ_5	10.0
λ_6 in DMT	0.0
λ_6 in SMW	0.0
λ_6 in SMPT	50.0
weight for $\mathcal{L}_{van-dist}$ in SMPT	10.0

Table S3. Hyper-parameters for the grasp losses.

A.3. Optimization.

We optimize the model with a batch size of 64 by the Adam optimizer [8], with a weight decay rate of 5.0×10^{-6}, and a learning rate of 2.0×10^{-4}, which is decayed to 2.0×10^{-5} by the cosine learning rate [12] scheduler.

A.4. Metrics

We explain the detailed settings and the computation process of the metrics in this section.

Mean Q_1. Intuitively, the Q_1 metric reflects the norm of the smallest wrench which can disrupt the stability of a grasp. We follow [20] to set the contact threshold to 1cm and set the penetration threshold to 5mm. Any grasp with its maximal penetration depth greater than 5mm is considered invalid and we set the Q_1 of it to 0 before taking the average.

Maximal penetration depth, which is the maximal penetration depth from the object point cloud to hand meshes.

Grasping success rate in Isaac Gym [13]. Specifically, we apply the predicted hand parameters to ShadowHand [16] and load the hand meshes and the object mesh into Isaac Gym. Following [20], we consider a grasp pose valid if the grasp can hold the object steadily under any one of the six gravity directions.

Occupancy proportions for translation δ_t. To evaluate δ_t, we uniformly sample $\xi = 16$ points $\{\mu_i\}_{i=1}^{\xi}$ on a unit sphere with Fibonacci sampling. The distribution of the sampled points is shown in Figure S2. These vectors serve as grasping direction prototypes and we assign each grasp translation to the nearest direction prototype by comparing the cosine similarity. Concretely, denote the number of grasp translations that are assigned to the i^{th} prototype as ν_i, and the object center is $\mathbf{c}$. Then ν_i is calculated with

$$\nu_i = \sum_{k=1}^{N} \mathbb{I}(\underset{j}{argmax}\, cos(\mu_j, \mathbf{t}_k - \mathbf{c}) = i), \tag{4}$$

where $cos(\cdot, \cdot)$ refers to the cosine similarity, and $\mathbb{I}(\cdot)$ is the indicator function. Finally, δ_t is the proportion of the direction prototypes with samples assigned to them

$$\delta_t = \frac{\sum_{i=1}^{\xi} \mathbb{I}(\nu_i > 0)}{\xi} \tag{5}$$

Occupancy proportions for rotation δ_r and joint angles δ_q. Similar to δ_t, we uniformly divide the ranges of each component of the rotation $\mathbf{r}$ and the joint angles $\mathbf{q}$ into ξ bins, and assign each component of $\mathbf{r}$ and $\mathbf{q}$ into the appropriate bin according to their values. The occupancy proportion for each component is the proportion of the bins with samples assigned to them. To obtain the δ_r and δ_q of an object, we take the average of the components of $\mathbf{r}$ and $\mathbf{q}$ respectively.

A.5. Implementation Details of SOTA Methods

In this section, we present the implementation details of three state-of-the-art methods on the DexGraspNet dataset: DDG [11], GraspTTA [7], and UniDexGrasp [21]. Specifically, we ensure that the parameter settings, data preprocessing methods, and training procedures are consistent

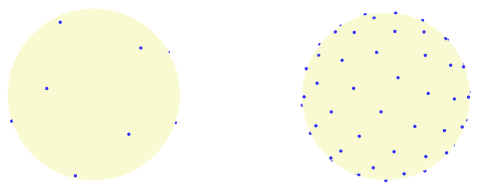


Figure S2. Uniformly sample several points on a unit sphere with Fibonacci sampling.

Num. Point	$Q_1 \uparrow$	$Pen. \downarrow$	$\eta_{np} \uparrow$	$\eta_{tb} \uparrow$
1024	0.0242	0.568	42.84	69.27
512	0.0241	0.557	42.61	69.80
128	0.0278	0.466	52.36	65.10

Table S4. Analysis of different numbers of encoder output points after downsampling. Our adopted configuration is highlighted in gray.

with those described in [20] for DDG and GraspTTA. For UniDexGrasp, we reproduce the grasp generation part while removing the constraints on the desktop plane. Additionally, to ensure a fair comparison, we adjust the number of iterations in TTA to 100 as ours, while keeping all other hyper-parameters consistent with the original version.

B. Additional DGTR Experiments

B.1. Analysis of Architecture

Number of Encoder Output Points. We conduct experiments to analyze the impact of varying the cardinality M' of the encoder output point cloud. Specifically, we utilize the point feature propagation module introduced in [15] to increase the cardinality of the output point cloud. As shown in Table S4, the increase in the number of output points does not enhance the performance. We hypothesize that the integration of the feature propagation module as well as the increased number of encoder features bring additional challenges to the network and the learning process, leading to a slight decrease in performance.

Position Embedding. We explore different position embeddings for the point features from the encoder. As the 3D point coordinates naturally contain position information, we apply various operations directly on the coordinate to generate position embeddings. We experiment with Fourier and Sine encodings as described in [1, 17], as well as a learnable MLP to derive the position encoding. As illustrated in Table Table S5, the result indicates that the model with an MLP for position encoding outperforms the ones using Fourier and sine encodings.

The Number of Decoder Layers. We investigate how the number of transformer decoder layers influences model

Pos. Emb.	$Q_1 \uparrow$	$Pen. \downarrow$	$\eta_{np} \uparrow$	$\eta_{tb} \uparrow$
XYZ $\rightarrow$ Fourier	0.0191	0.605	40.54	68.40
XYZ $\rightarrow$ Sine	0.0225	0.565	43.76	66.84
XYZ $\rightarrow$ MLP	0.0278	0.466	52.36	65.10

Table S5. Analysis of different position embeddings. Our adopted configuration is colored in gray.

Num. Layer	$Q_1 \uparrow$	$Pen. \downarrow$	$\eta_{np} \uparrow$	$\eta_{tb} \uparrow$
2	0.0152	0.642	36.28	68.79
4	0.0278	0.466	52.36	65.10
6	0.0235	0.665	35.20	77.57

Table S6. Analysis of different numbers of decoder layers. Our adopted configuration is highlighted in gray.

λ_6	λ_7	$Q_1 \uparrow$	$Pen. \downarrow$	$\eta_{np} \uparrow$	$\eta_{tb} \uparrow$
30	10	0.0229	0.699	25.51	84.34
50	0	0.0217	0.418	57.24	54.87
50	10	0.0278	0.466	52.63	65.10
50	20	0.0269	0.569	40.15	73.66
50	30	0.0165	0.699	22.07	84.78
70	10	0.0254	0.439	55.75	59.67
90	10	0.0240	0.418	58.24	55.34

Table S7. Analysis of different loss weights in SMPT. λ_6 and λ_7 are the weights of object penetration loss and vanilla distance loss.

α_1	α_2	$Q_1 \uparrow$	$Pen. \downarrow$	$\eta_{np} \uparrow$	$\eta_{tb} \uparrow$
3	3	0.0504	0.340	81.46	67.81
5	3	0.0515	0.241	90.62	64.36
7	3	0.0449	0.190	93.25	59.15
3	5	0.0515	0.421	75.78	69.62
5	5	0.0533	0.319	82.18	71.60
7	5	0.0498	0.265	86.95	65.40
3	7	0.0483	0.480	59.37	80.70
5	7	0.0496	0.387	71.54	74.83
7	7	0.0490	0.319	79.49	68.09

Table S8. Analysis of different loss weights in AB-TTA. α_1 and α_2 are the weights of object penetration loss and generalized tta-distance loss.

performance by replacing our decoder with a two-layer and a six-layer version. As depicted in Table S6, both excessive and insufficient layers can result in a decline in performance. Our model achieves an optimal performance when the number of layers is set to 4. It is important to note that, for the sake of consistency and fair comparison, we keep identical training hyper-parameters across different experiments, though it is possible that different training hyper-parameters may be suitable for different configurations.

B.2. Analysis of Loss Weight

Loss Weight in DSMT. We explore the impact of adjusting the loss weights, object penetration loss weight (denoted as

λ_6), and the vanilla distance loss weight (denoted as λ_7), in the static matching penetration training (SMPT) stage of dynamic-static matching training (DSMT). As illustrated in Table S7, setting λ_6 to 50 and λ_7 to 10 yields the best performance for Q_1 and maintains a balanced performance for η_{np} and η_{tb}.

Loss Weight in AB-TTA. We examine the effects of loss weight adjustments on object penetration loss $\mathcal{L}_{pen}$ (denoted as α_1) and generalized tta-distance loss $\mathcal{L}_{tta-dist}$ (denoted as α_2) in the Adversarial-Balanced Test-Time Adaptation (AB-TTA). The adversarial interplay between these two losses is illustrated in Table S8. When the weight of object penetration loss is increased, the predicted grasps exhibit a higher non-penetration ratio (η_{np}) but a lower torque balance ratio (η_{tb}). Conversely, increasing the weight of generalized tta-distance loss results in a higher η_{tb} but a lower η_{np}.

B.3. Diversity measured by the coverage metric

We investigate the diversity of DGTR predictions using the "coverage" metric [2]. Different from the "occupancy" metrics introduced in Section A.4, "coverage" considers the guidance of GT, while our metrics prioritize "pure diversity". As shown in Table S9, DGTR outperforms the other two models in all three types of "coverage" metric, indicating a better coverage of the ground-truth grasp poses.

Method	$cov_1 \uparrow$	$cov_2 \uparrow$	$cov_3 \uparrow$
UniDexGrasp [21]	10.53	6.62	13.84
Scene Diffuser [6]	43.15	10.93	21.28
DGTR (ours)	**55.95**	**12.89**	**24.34**

Table S9. Coverage (%) with $\epsilon = 1.5$ and $\omega = 5.0$

C. Additional Visualizations

We provide more qualitative results of different objects in Figure S3. Each object is visualized with 4 out of 16 grasp poses. The grasping details in Figure S3 show that our DGTR is capable of generating diverse and high-quality grasp poses in one forward pass. To better demonstrate the diversity of the predicted grasp poses, we visualize all predicted grasp poses of one object in Figure S4.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, 2020. 3

[2] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. A billion ways to grasps - an evaluation of grasp sampling schemes on a dense, physics-based grasp data set. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2019. 4

[3] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017. 2

[4] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2015. 2

[5] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 2021. 1

[6] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[7] Hanwen Jiang, Shaowei Liu, Jiashun Wang, and Xiaolong Wang. Hand-object contact consistency reasoning for human grasps generation. In *Proceedings of the International Conference on Computer Vision*, 2021. 2

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[9] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1

[10] Puhao Li, Tengyu Liu, Yuyang Li, Yiran Geng, Yixin Zhu, Yaodong Yang, and Siyuan Huang. Gendexgrasp: Generalizable dexterous grasping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 1

[11] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Deep differentiable grasp planner for high-dof grippers. *arXiv preprint arXiv:2002.01530*, 2020. 2

[12] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2016. 2

[13] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. 2

[14] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 2017. 1, 3

[16] ShadowHand. Shadowrobot. https://www.shadowrobot.com/dexterous-hand-series/, 2005. 2

[17] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 2020. 3

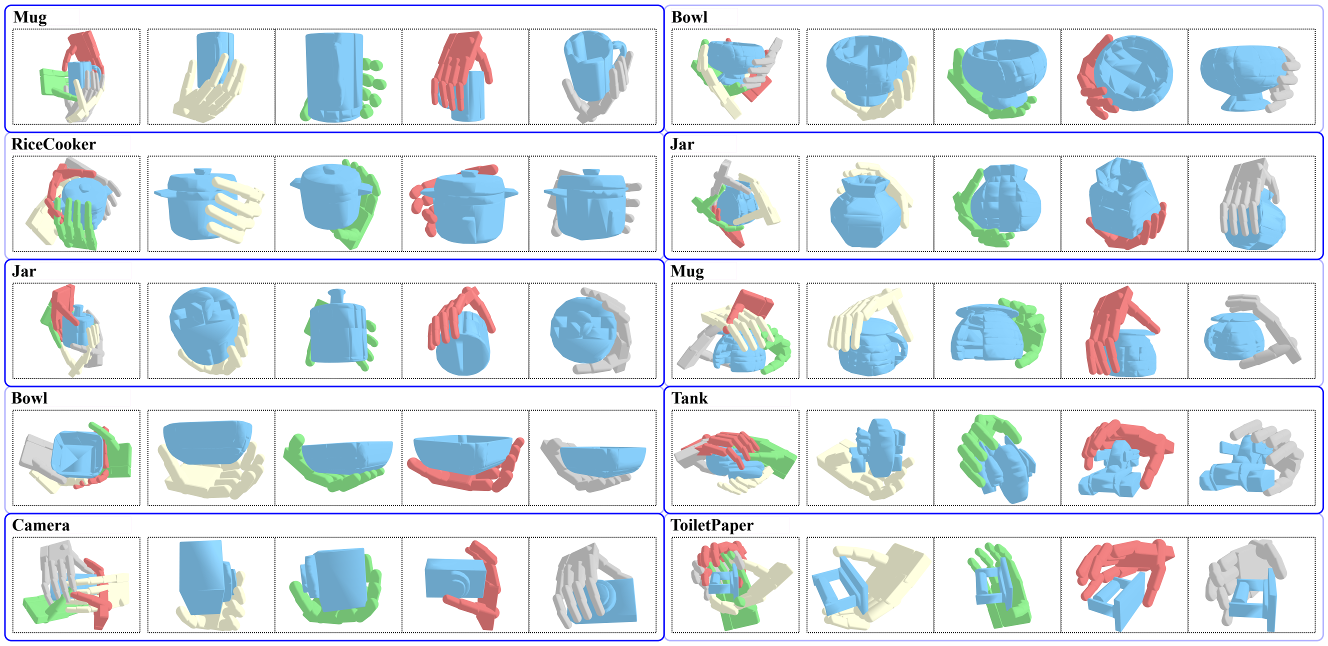


Figure S3. Visualization of predicted grasp poses. We visualize four grasp poses in five images for each object. The first image visualizes all grasps together to demonstrate their global positions. The following four images mainly visualize the details of the grasp poses.

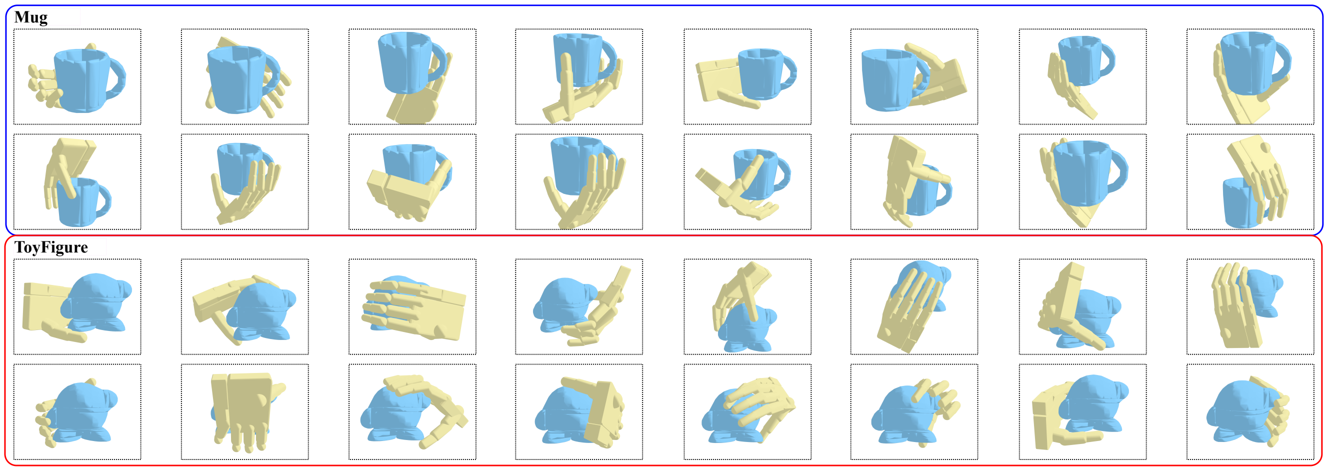


Figure S4. Visualization of all predicted grasp poses of one object. We visualize all predicted grasp poses of one object in two rows, where each image shows one grasp pose. To show the diversity of the predicted grasp poses, the camera views of all images are almost identical.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017. 1

[19] Chenxi Wang, Hao-Shu Fang, Minghao Gou, Hongjie Fang, Jin Gao, and Cewu Lu. Graspness discovery in clutters for fast and accurate grasp detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 1

[20] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 1, 2, 3

[21] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi 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*, 2023. 2, 4

[22] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*,

2021. 1

[23] Tianqiang Zhu, Rina Wu, Xiangbo Lin, and Yi Sun. Toward human-like grasp: Dexterous grasping via semantic representation of object-hand. 2021 ieee. In *CVF International Conference on Computer Vision (ICCV)*, 2021. 2