UniDexGrasp: Universal Robotic Dexterous Grasping via Learning Diverse Proposal Generation and Goal-Conditioned Policy

Supplementary Material

Yinzhen Xu*[1,2,3], Weikang Wan*[1,2], Jialiang Zhang*[1,2], Haoran Liu*[1,2], Zikang Shan[1], Hao Shen[1], Ruicheng Wang[1], Haoran Geng[1,2], Yijia Weng[4], Jiayi Chen[1], Tengyu Liu[3], Li Yi[5], He Wang†[1,2]

[1] Center on Frontiers of Computing Studies, Peking University [2] School of EECS, Peking University
[3] Beijing Institute for General AI [4] Stanford University [5] Tsinghua University

https://pku-epic.github.io/UniDexGrasp/

Abstract In this supplementary material, we provide our dataset generation method in Section A, details about our method, baselines, and implementation in Section B, details about the experiments in Section C, and more quantitative and qualitative experiment results, in Section D. For more visualization of our generated dataset and grasping demonstrations, please refer to the supplementary video.

A. Dataset Generation

In order to train our vision model and RL policy to be universal and diverse in the table-top setting, we need a dexterous grasping dataset that provides numerous object instances and holds diverse grasping labels. Moreover, each grasp should correspond to a physically plausible tabletop scene that is free of any penetration. We synthesized this dataset using a similar method from [21].

Object Preperation Our object dataset is composed of 5519 object instances in 133 categories selected from ShapeNet, [8], and [20]. Each object instance is canonicalized into a unit sphere, then re-scaled by each factor in $\{0.06, 0.08, 0.1, 0.12, 0.15\}$. We decompose our meshes into convex pieces using [22].

Grasp Generation Our table-top scene starts with a flat plane which overlaps with the $z = 0$ plane of the world reference frame. For each generation environment, we randomly select an object from the pool of our training instances, randomly rotate it, then let it fall onto the plane from a high place. Next, we randomly initialize a dexterous gripper in an area above the object, and let it face the object. Then, the initial gripper pose is optimized into a plausible grasp in a 6000-step optimization process, guided by an energy function. Finally, the object's pose and the gripper's translation, rotation, and joint angles are saved for further validation.

Figure 1. Our object dataset contains more than five thousand objects from various categories. These are the visualization of some decomposed meshes.

Energy Function We base our energy function on [9], and modified it to suit the table-top setting. It is composed of the following terms. 1) E_{fc}: A differentiable force closure estimator that encourages physical stability; 2) E_{dis}: Attraction energy to ensure contact; 3) E_{pen}: Repulsion energy to elliminate penetration; 4) E_{tpen}: L1 energy that keeps the gripper above the table; 5) E_{joints}: Regularization term to enforce joint limits; 6) E_{spen}: Self penetration energy inspired by [25]. The total energy is a linear combi-

nation of the six terms.

$$E = E_{\text{fc}} + w_{\text{dis}}E_{\text{dis}} + w_{\text{pen}}E_{\text{pen}} + w_{\text{tpen}}E_{\text{tpen}} + w_{\text{joints}}E_{\text{joints}} + w_{\text{spen}}E_{\text{spen}} \quad (1)$$

Grasp Validation We filter our generated dataset by physical stability and penetration depth. A grasp is considered physically stable if it can resist gravity in all 6 axis-aligned directions in the IsaacGym simulator. Moreover, we discard grasps that penetrate the object for more than 1mm. We ran 1000 generations for each object instance, and harvested 1.12 million valid grasps in total. This table-top dataset features stability and diversity, which empowers our models to learn object-agnostic dexterous grasping in a table-top scene.

Hand SDF Calculation The signed distance function from the object point cloud to the hand mesh is needed when calculating the penetration energy. However, this forms a batched points-to-mesh distance calculation problem with different meshes, which is hard to compute. So we add some tricks to speed up this calculation. First, we use the collision mesh of the Shadowhand, which is composed of articulation of simple primitives namely boxes and capsules. Second, for a batch of object point clouds, we consider each link of the hand respectively. Third, we use forward kinematics to transform the object point clouds into the link's local reference frame. This operation turns the problem into a points-to-mesh distance calculation with a single mesh. The meshes for boxes are simple, so we use Kaolin [11] to compute points-to-mesh distances. As for capsules, the signed distance functions can be defined analytically. Finally, for each point, we take the minimal signed distance across all links to get the signed distance from that point to the complete hand collision mesh.

B. Method and Implementation Details

B.1. Goal Proposal Generation

B.1.1 Details about Our Method

GraspIPDF: The original IPDF [10] is a probabilistic model over $\mathbf{SO(3)}$ that maps a pair of input and rotation, *i.e.*, $(\mathcal{X}, R)$, where the input $\mathcal{X}$ in IPDF is an image, to an unnormalized joint log probability that indicates the likelihood of this pair. Following this work, we implement our GraspIPDF as a function $f(X_0, R) : \mathbb{R}^{N\times 3} \times \mathbf{SO(3)} \to \mathbb{R}$. We choose PointNet++ [13] to extract a global feature from the input point cloud X_0, and concatenate it with rotation representation using the same positional encoding in [10]. The concatenated feature is processed by an MLP with layer sizes $(256, 256, 256)$ to output $f(X_0, R)$ as formulated in the paper.

GraspGlow: We use Glow [6] to implement normalizing flow in this part. Glow implements its bijective transformation $f : \mathbb{R}^{3+K} \to \mathbb{R}^{3+K}$ as the composition of several blocks, where each block consists of three parts: actnorm, 1×1 convolution, and affine coupling layer.

Our implementation of actnorm and 1×1 convolution is similar to the implementation in Glow, and the only difference is that our flow is used to transform a single vector of $\mathbb{R}^{3+K}$ instead of an image. Actnorm is a linear function $f_{\text{actnorm}}(x) = x/\sigma_\theta + \mu_\theta$ where $\mu_\theta, \sigma_\theta \in \mathbb{R}^{3+K}$ are initialized using the first batch's mean and standard deviation and then optimized with gradient descent like other parameters. 1×1 convolution, which can be written as $f_{\text{conv}}(x) = Wx$ where W is a $(3+K)\times(3+K)$ invertible matrix, is a linear transformation used as a generalization of a permutation operation. To constraint W to be invertible, W is parametrized with LU decomposition $W = PL(U+S)$ where P is a random orthogonal matrix fixed in the training process, L is a lower triangular matrix with ones on the diagonal, U is an upper triangular matrix with zeros on the diagonal, and S is a diagonal matrix whose diagonal elements are ensured to be positive using $\exp$. We modify the affine coupling layer in a similar way to ProHMR [7], which can be described as follows:

$$\begin{aligned}(x_1, x_2) &= \text{split}(x)\\ (\log s, b) &= \text{NN}(x_2, c)\\ s &= \exp(\log s)\\ y_1 &= x_1\\ y_2 &= s \odot x_2 + b\\ f_{\text{coup}}(x) &= \text{concat}(y_1, y_2)\end{aligned}$$

where x is the input of the transformation, $c \in \mathbf{R}^c$ is the feature extracted by PointNet, x_1 is the first half dimensions of x and x_2 is the last half dimensions.

In GraspGlow, we compose 21 blocks described above, and the NNs in each block's coupling layer has 2 residual blocks containing MLPs with two layers, and the number of hidden dimensions is 64. The activation function is ReLU and we use batch normalization in MLPs and the probability of dropout is 0.5. For more details, we refer the reader to ProHMR [7]'s code as we use their implementation of conditional Glow.

ContactNet: The ContactNet has 2 independent PointNet [12] modules respectively for the canonicalized object point cloud $\tilde{X}$ and the hand point cloud X_H sampled from the hand mesh constructed by the forward kinematics. The global feature from the hand is broadcast and concatenated to the per-point feature of the object point cloud. Afterward, the feature goes through an MLP with layer sizes $(1024, 512, 512, 256, 256, 128, 128, 10)$ to output the 10-bin-discretized contact map per-point prediction. The discretization is found to greatly boost the robustness of our ContactNet.

B.1.2 Details about Baselines

GraspTTA: GraspTTA [5] proposed a two-stage framework to synthesize grasps for MANO [15]. They designed a CVAE and a ContactNet to perform the tasks. During training, the CVAE learns to reconstruct the grasp dataset, and the ContactNet learns a mapping from the object and hand point cloud to the object's contact map. During testing, in the first stage, the CVAE takes the object point cloud's feature as a condition, samples a latent vector from the Gaussian distribution, and outputs the hand rotation, translation, and parameters. They use these to reconstruct the hand mesh. In the second stage, the ContactNet takes the object and hand point cloud, and predicts a target contact map on the objects. The consistency energy is defined as the MSE between the actual contact map and the target contact map predicted by the contact net. Using this energy, the hand is optimized toward the target in a test-time adaptation process. Note that in their work, the target contact map is recalculated in every optimization step. We also use test-time adaptation in our pipeline, but only calculate the target contact map in the first iteration to save time.

DDG: Deep Differentiable Grasp (DDG) [8] takes 5 depth images of the object and regresses the translation, rotation, and joint angles of the ShadowHand. The learning process is divided into two stages. In the first stage, only a min-of-N loss is used for grasp pose regression. In the second stage, other loss functions are added to encourage contact, avoid penetration, and improve grasp quality. It is important to note that, the original method doesn't take the table as input. They assume that all depth images are taken in the object reference frame, and for each set of depth images, 100 ground-truth grasps are required to define the min-of-N regression loss. However, if we change this setting and take the depth images in the table reference frame, then only 15 ground-truth grasps are available for each set of images on average. This is because when we synthesized data for each object, the table planes are randomly chosen in each generation process. So in this experiment, we preserved their original settings without the table. In evaluation, we don't filter grasps that have large table penetration depth for this method, which makes the problem easier, and prove that our method still out-perform theirs.

ReLie: ReLie [2] proposes a general way to perform normalizing flow on Lie groups using Lie algebra, and in $\mathbf{SO(3)}$ this equals to using the axis-angle representation $\mathbf{v} = \theta\mathbf{n}$ where θ is the angle of the rotation and that $\mathbf{n}$ is a unit vector representing the axis of rotation. In their implementation, the normalizing flow is performed on $\mathbb{R}^3$ and then the samples are transformed using $\tanh(\cdot)$ and multiplied by r, so that the length of the sampled vector is less than r, and at last, the transformed samples are converted to rotation using the exponential map. Note that for surjectivity, they sacrifice invertibility and set r to 1.6π. In our experiment, we add those three dimensions to GraspGlow so that the hand root rotation, translation, and joint angles can be sampled jointly. The problem with this method is that it suffers from discontinuity of the axis angle representation and that the lack of bijectivity is also harmful to the learning process.

ProHMR: ProHMR [7] proposes to use a 6D representation of $\mathbf{SO(3)}$, which is the first two columns of the rotation matrix, to avoid discontinuity. In their method, normalizing flow is performed on $\mathbb{R}^6$ and the samples are projected to the manifold of $\mathbf{SO(3)}$ afterward. In our experiment, we add those six dimensions to GraspGlow similar to the baseline of ReLie, and to make the samples close to the $\mathbf{SO(3)}$ manifold, we also add the orthogonal loss following ProHMR. The problem with this method is that the projection is an infinite-to-one mapping so that the probability of a specific rotation is intractable, so theoretically the normalizing flow can place infinite probability to the $\mathbf{SO(3)}$ manifold without learning distribution on $\mathbf{SO(3)}$ to get infinitely low NLL.

B.2. Goal-conditioned Dexterous Grasping Policy

B.2.1 Details about Our Method

As we introduced in Sec. 3.3.2, we use PPO to update the teacher policy. We also adopt the technique in ILAD [23] which jointly learns object geometric representation by updating the PointNet using behavior cloning objective during RL policy training. We further propose three important techniques including state canonicalization, object curriculum learning, and joint training object classification to update the PointNet in our network.

Reward Function: To ensure proper interaction between the robot hand and the object and encourage the robot to grasp the object according to the input grasping goal pose, we define a novel goal-conditioned reward function. Since we aim to solve the dexterous manipulation problem with pure RL, the reward design is crucial. Note that all the ω_{**} here are hyper-parameters.

The goal pose reward r_{goal} encourages the robot hand to reach the input grasping goal pose. It measures the weighted sum of distances between current robot joint angles q_j and goal joint angles q_j^g, and the distance between hand root pose (t_h^{obj}, R_h^{obj}) in the object reference frame

and the goal hand root pose (t_h^g, R_h^g) :

$$r_{\text{goal}} = -\omega_{g,q} \sum_{j=1}^{J} |q_j - q_j^g| - \omega_{g,t} \|t_h^{obj} - t_h^g\|_2 - \omega_{g,R} L_{rot} \tag{2}$$

$M_{R_h^{obj}}$ and $M_{R_h^g}$ are matrices of the object relative hand rotation and goal hand rotation. The L_{rot} here stands for the axis angle from goal hand rotation to object relative hand rotation, and is formulated as follows:

$$L_{rot} = \text{acos}(0.5(\text{trace}(M_{R_h^{obj}} M_{R_h^g}^\top) - 1)) \tag{3}$$

The reaching reward r_{reach} encourages the robot fingers to reach the object. Here, $\mathbf{x}_{\text{finger}}$ and $\mathbf{x}_{\text{obj}}$ denote the position of each finger and object:

$$r_{\text{reach}} = -\omega_r \sum \|\mathbf{x}_{\text{finger}} - \mathbf{x}_{\text{obj}}\|_2 \tag{4}$$

The lifting reward r_{lift} encourages the robot hand to lift the object when the fingers are close enough to the object and the robot hand is considered to reach the target goal grasp pose. f is a flag to judge whether the robot reaches the lifting condition: $f = \mathbf{Is}(\sum_{j=1}^{J} \omega_{g,j} \|\mathbf{x}_j^{obj} - \mathbf{x}_{g,j}^{obj}\|_2 < \lambda_{f_1}) + \mathbf{Is}(\sum \|\mathbf{x}_{\text{finger}} - \mathbf{x}_{\text{obj}}\|_2 < \lambda_{f_2}) + \mathbf{Is}(d_{\text{obj}} > \lambda_0)$. Here, $d_{\text{obj}} = \|\mathbf{x}_{\text{obj}} - \mathbf{x}_{\text{target}}\|_2$, where $\mathbf{x}_{\text{obj}}$ and $\mathbf{x}_{\text{target}}$ are object position and target position. a_z is the scaled force applied to the hand root along the z-axis ($\omega_l > 0$).

$$r_{\text{lift}} = \begin{cases} \omega_l * (1 + a_z) & \text{if } f = 3 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The moving reward r_{move} encourages the object to reach the target and it will give a bonus term when the object is lifted very close to the target:

$$r_{\text{move}} = \begin{cases} -\omega_m d_{\text{obj}} + \frac{1}{1+\omega_b d_{obj}} & \text{if } d_{\text{obj}} < \lambda_0 \\ -\omega_m d_{\text{obj}} & \text{otherwise} \end{cases} \tag{6}$$

Finally, we add each component and formulate our reward function as follows:

$$r = r_{\text{goal}} + r_{\text{reach}} + r_{\text{lift}} + r_{\text{move}} \tag{7}$$

Details of Object Curriculum Learning: For OCL (Object Curriculum Learning) experiments in Sec. 4.3 in the main paper, here is the detail of the curriculum.

For 1-stage OCL we randomly choose three different categories' objects and do three experiments. Each time first we train on one object and then train on all the categories.

For 2-stage OCL we randomly choose three categories and do three experiments. Each time first we train on one object from each category, then train on the category where the object is from, and last train on all the categories.

For 3-stage OCL, we do two experiments. The 3 representative categories we use in the first experiment are (toy car, bottle, and camera). The 3 representative categories we use in the second experiment are the (light bulb, cereal box, and toy airplane). We first train on one object, then train on the category of the object, then train on 3 representative categories, and last train on all the categories.

Network Architecture: The MLP in teacher policy $\pi_{\mathcal{E}}$ and student policy $\pi_{\mathcal{S}}$ consists of 4 hidden layers (1024, 1024, 512, 512). The network structure of the PointNet in both the $\pi_{\mathcal{E}}$ and $\pi_{\mathcal{S}}$ is (1024, 512, 64). We use the exponential linear unit (ELU) [1] as the activation function.

B.2.2 Details about Baselines

MP (Motion Planning) We use cross-entropy method (CEM) [17] for motion planning given target hand joint positions j^g computed from the target goal hand grasp label $\boldsymbol{g}$ using forward kinematics. The goal is to find a robot hand action sequence $a_1, ..., a_K$ which generates a robot hand joint position sequence $\boldsymbol{j}_0^r, ..., \boldsymbol{j}_K^r$ and the last robot hand joint positions $\boldsymbol{j}_K^r$ reaches $\boldsymbol{j}^g$. Followed by [23], the objective of the motion planning is $\min_{a_1,...a_K} \|\boldsymbol{j}_K^r - \boldsymbol{j}^g\|^2 + \lambda \|\mathbf{x}_{\text{obj}}^1 - \mathbf{x}_{\text{obj}}^K\|^2$, where $\mathbf{x}_{\text{obj}}^1$ and $\mathbf{x}_{\text{obj}}^K$ are object poses at time step 1 and K. This objective function encourages the robot hand to reach the goal hand grasp label as well as prevents the object from moving during the process. We use model predictive control (MPC) to execute the planned trajectories sampled from CEM process until the objective is below a threshold δ . Once the process ends, we lift the robot's hand to see whether the object falls down.

PPO PPO [18] is a popular model-free on-policy RL method. We use PPO together with our designed goal-conditioned reward function as our RL baseline.

DAPG Demo Augmented Policy Gradient (DAPG) [14] is a popular imitation learning (IL) method that leverages expert demonstrations to reduce sample complexity. Followed by ILAD [23], we use motion planning to generate demonstrations from our goal grasp label dataset. We use our designed goal-conditioned reward function in this method.

ILAD ILAD [23] is an imitation learning method that improves the generalizability of DAPG. ILAD proposes a novel imitation learning objective on top of DAPG and it jointly learns the geometric representation of the object using behavior cloning from the generated demonstrations during policy learning. For this method, we use the same

Parameters	Description
$\boldsymbol{q} \in \mathbb{R}^{18}$	joint positions
$\dot{\boldsymbol{q}} \in \mathbb{R}^{18}$	joint velocities
$\boldsymbol{\tau}_{\text{dof}} \in \mathbb{R}^{24}$	dof force
$x_{\text{finger}} \in \mathbb{R}^{3\times5}$	fingertip position
$\alpha_{\text{finger}} \in \mathbb{R}^{4\times5}$	fingertip orientation
$\dot{x}_{\text{finger}} \in \mathbb{R}^{3\times5}$	fingertip linear velocities
$\omega_{\text{finger}} \in \mathbb{R}^{3\times5}$	fingertip angular velocities
$F_{\text{finger}} \in \mathbb{R}^{3\times5}$	fingertip force
$\tau_{\text{finger}} \in \mathbb{R}^{3\times5}$	fingertip torque
$\boldsymbol{t} \in \mathbb{R}^{3}$	hand root global transition
$R \in \mathbb{R}^{3\times3}$	hand root global orientation
$\boldsymbol{a} \in \mathbb{R}^{24}$	action

Table 1. Robot state definition.

generated demonstrations as in DAPG and use our designed goal-conditioned reward function.

IBS-Grasp IBS-Grasp [19] propose an effective representation of the grasping state called Interaction Bisector Surface (IBS) characterizing the spatial interaction between the robot hand and the object. The IBS representation, together with a novel vector-based reward and an effective training strategy, facilitates learning a strong control model of dexterous grasping with good sample efficiency and cross-category generalizability. It uses SAC [4]to train the policy. Note that we evaluate the baseline using the official code which uses the Pybullet simulator because it's hard to do the proposed fast IBS approximation in Isaac gym. Additionally, it cannot train under the goal-conditioned setting using the proposed reward function.

C. Experiment Details

C.1. Environment Setup

State Definition The full state of the teacher policy $\mathcal{S}^{\mathcal{E}} = (\boldsymbol{s}_t^r, \boldsymbol{s}_t^o, X^O, \boldsymbol{g})$. The full state of the student policy $\mathcal{S}^{\mathcal{S}} = (\boldsymbol{s}_t^r, X_t, \boldsymbol{g})$. The robot state $\boldsymbol{s}_r$ is detailed in Tab. 1 and the object oracle state $\boldsymbol{s}_o$ includes the object pose, linear velocity, and angular velocity. For the pre-sampled object point cloud X^O, we sample 2048 points from the object mesh. For the scene point cloud X_S, we only sample 1024 points from the object and the hand to speed up the training.

Action Space The action space is the motor command of 24 actuators on the robotic hand. The first 6 motors control the global position and orientation of the robotic hand and the rest 18 motors control the fingers of the hand. We normalize the action range to (-1,1) based on actuator specification.

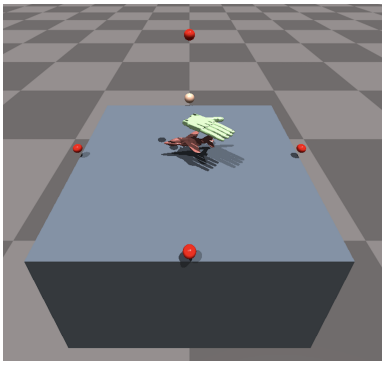

Figure 2. **Camera positions**

Camera Setup We placed five RGBD cameras, four around the table and one above the table, as shown in Fig.2. The origin of the system is the center of the table. The positions of the five cameras are: ([0.5, 0, 0.05], [-0.5, 0, 0.05], [0, 0.5, 0.05], [0, -0.05, 0.05], [0, 0, 0.55]) and all their focus point is [0, 0, 0.05].

C.2. Training Details

(1) For the goal proposal generation part, we use Adam optimizer to train GraspIPDF, with a learning rate of 10^{-3} and a batch size of 16. The loss Curve converges in 24 hours. In the training process of GraspGlow, we use Adam as our optimizer and the learning rate is 10^{-3}. The experiment is done on an NVIDIA RTX A5000, and the batch size is set to 64 in the first stage and 32 in the second stage with 8 samples for each object. The training process consists of 160k iterations in the first stage and 8k iterations in the second stage, and it needs one day in total. ContactNet also uses Adam, with a learning rate of 10^{-3} and a batch size of 128, and the normalization factor β is set to 60. This module needs 8 hours of training. The hyperparameters for end-to-end training and test-time adaptation are in Tab. 2.

(2) For the goal-conditioned dexterous grasping policy part, we use PPO [18] to learn $\pi_{\mathcal{E}}$ and then distill to $\pi_{\mathcal{S}}$ using DAgger [16]. Since some of the object meshes in the dataset are too large for dexterous grasping, we filter out some large-scale object instances. In the end, we obtain a train set of 3200 object instances and a test set of 241 object instances for the grasping execution experiments. The hyperparameters for the experiments are in Tab. 3.

C.3. Metric Details

C.3.1 Metrics for Goal Proposal Generation

For the goal proposal generation part, we introduce seven metrics to evaluate grasp quality and two metrics to measure the diversity of our generated results.

Mean Q_1 Q_1 [3] is defined as the minimal wrench needed

Hyperparameter	Value
λ_{cmap}	0.02
λ_{pen}	500
λ_{tpen}	50
λ_{spen}	10
$\lambda_{\text{cmap}}^{\text{TTA}}$	0.07
$\lambda_{\text{pen}}^{\text{TTA}}$	10000
$\lambda_{\text{tpen}}^{\text{TTA}}$	1000
$\lambda_{\text{spen}}^{\text{TTA}}$	10
step size	0.001

Table 2. **Hyperparameters for end-to-end training (upper half) and test-time adaptation (lower half).**

Hyperparameter	Value
Num. envs (Isaac Gym)	1024
Env spacing (Isaac Gym)	1.5
Num. rollout steps per policy update	8
Num. batches per agent	4
Num. learning epochs	5
Episode length	200
Discount factor	0.96
GAE parameter	0.95
Entropy coeff.	0.0
PPO clip range	0.2
Learning rate	0.0003
Value loss coeff.	1.0
Max gradient norm	1.0
Initial noise std.	0.8
Desired KL	0.16
Clip observations	5.0
Clip actions	1.0
$\omega_{g,q}$	0.1
$\omega_{g,t}$	0.6
$\omega_{g,R}$	0.1
ω_r	0.5
ω_l	0.1
ω_m	2
ω_b	10

Table 3. **Hyperparameters for grasping policy.**

to make a grasp unstable. This metric is only well defined when the grasp has exact contact and doesn't penetrate the object, which is impossible for vision-based methods. So we relaxed the contact distance to 1cm. Moreover, if a grasp penetrates the table for more than 1cm, or has an object penetration depth larger than 5mm, then this grasp will be considered invalid, so its Q_1 will be manually set to zero.

Object Penetration Depth We define object penetration depth as the maximal penetration from the object's point cloud to the hand mesh. This is calculated using the tricks we introduced in Sec. A.

Rotation Standard Deviation This metric evaluates the standard deviation of rotation by first calculating the chordal L2 mean of rotation samples ($\underset{R}{\text{argmin}} \sum_{i=1}^{n} (||R - R_i||_F^2)$, and then calculate the standard deviation between the rotation samples and the mean. This metric is used to show the diversity of rotation samples from GraspIPDF.

Translation and Joint Angles Standard Deviation (conditional) These two metrics evaluate the standard deviations of translation and joint angles, given a sampled rotation from GraspIPDF, and are used to show the diversity of translation and joint angles in the grasp samples from GraspGlow.

Keypoint Standard Deviation This metric evaluates the average standard deviation of 15 joint (keypoint) positions of the robotic hand. This metric is used to show the diversity of our grasp proposals generated by the whole grasp proposal generation pipeline.

Log-likelihood We evaluate the log-likelihood of the ground truth grasping rotation, translation, and joint angles predicted by the model, and this metric is used to show how well the model fits the ground truth distribution. Note that for our model, we calculate $p(R, t, \theta|X)$ as $p(R|X) \cdot p(t, \theta|R^{-1}X)$.

C.3.2 Metrics for Goal-conditioned Grasp Execution

For the goal-conditioned dexterous grasping policy part, we introduce metrics to measure the success rate of grasping, as well as how strictly our policy follows the specified grasping goal.

Simulation Success Rate We define the success rate as the primary measure of the grasping policy. The target position of the object is 0.3m above its initial position. The task is considered successful if the position difference between the object and the target is smaller than 0.05m at the final step of one sequence.

MPE (cm) This metric is used to measure the mean joint position error between the joint position $\boldsymbol{j}^r$ of the exact grasping pose and the joint angels $\boldsymbol{j}^g$ computed from the input goal hand grasp label $\boldsymbol{g}$ using forward kinematics. J is the number of joints. Note we only calculate the MPE for the success grasp.

Method	goal-conditioned						non-goal-conditioned		
	Train		Test				Train	Test	
			unseen obj seen cat		unseen cat			unseen obj seen cat	unseen cat
	succ↑	MPE (cm)↓	succ↑	MPE (cm)↓	succ↑	MPE (cm)↓	succ↑	succ↑	succ↑
MP	0.12	**1.2**	0.02	**1.8**	0.02	**1.8**	/	/	/
PPO [18]	0.14	4.4	0.11	4.9	0.09	5.8	0.24	0.21	0.17
DAPG [14]	0.13	8.0	0.13	7.4	0.11	9.1	0.21	0.15	0.10
ILAD [23]	0.25	5.1	0.22	5.3	0.20	5.6	0.32	0.26	0.23
IBS-Grasp [19]	/	/	/	/	/	/	0.57	0.54	0.54
Ours (teacher)	**0.74**	3.5	**0.71**	3.9	**0.67**	4.5	**0.79**	**0.74**	**0.71**
Ours (student)	0.68	3.8	0.64	4.3	0.60	4.7	0.74	0.69	0.65

Table 4. **Results on dexterous grasping policy.** We use bold to indicate the best metric and underline to indicate the second-best metric. Note that for MP (Motion planning), "train" means optimizing on our synthetic ground truth grasp dataset and "test" means optimizing on the predicted grasp from our vision pipeline.

Method	Log-likelihood
ReLie [2]	-1.540
ProHMR [7]	-1.710
ours (R + GL)	**10.908**

Table 5. **Comparison on Log-likelihood of ground truth grasps.** R: GraspIPDF, GL: GraspGlow. Note that the outputted probability of flow in baselines on $\mathbf{SO(3)}$ is unnormalized and that we generate uniform grids using the method described in [24] and approximate the normalizing constant similar to IPDF [10].

$$e_{\text{mpe}} = \frac{1}{J} \sum \|\boldsymbol{j}^r - \boldsymbol{j}^g\|_2 \tag{8}$$

D. Additional Results and Analysis

This section contains extended results of the experiment depicted in Sec. 4.

D.1. Goal-conditioned Dexterous Grasping Policy Results

Goal-conditioned vs. Non-Goal-conditioned We also conducted experiments in a non-goal-conditioned setting, which is the task of grasping objects alone. We compare our results with baselines described in Sec. B.2.2. We add MP and IBS-Grasp only in supplementary because MP cannot perform non-goal-conditioned tasks and IBS-Grasp does not have a goal-conditioned setting. To perform non-goal-conditioned tasks using our method, we simply remove the goal input and goal reward in RL training. The results are shown in Tab. 4. In non-goal-conditioned settings, our teacher policy has the highest success rate across training and all testing data sets.

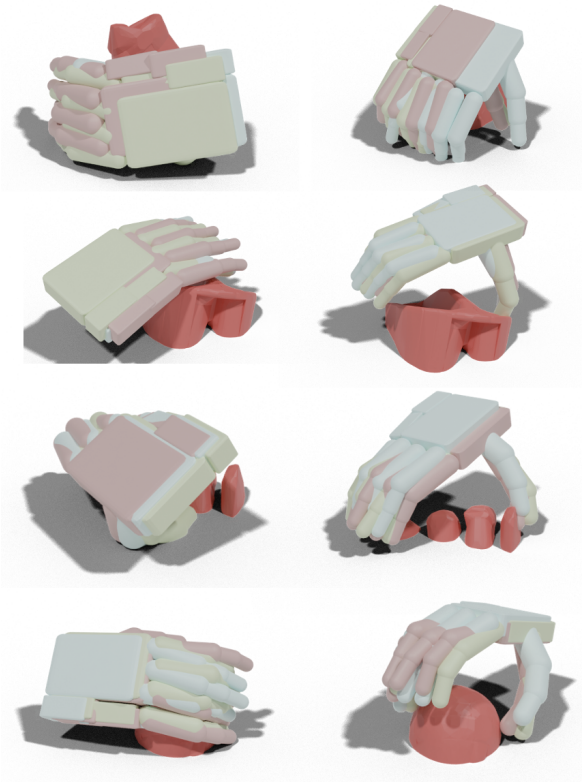

Figure 3. **Diverse grasp proposals**. Here we show the diversity of our grasping pose samples. On the same row, the objects are the same, and in the same picture, the hand root rotation is also the same. It can be shown that both the rotation samples from GraspIPDF and the translation and joint angles samples from GraspGlow have high diversity.

Analysis of Quantitative Grasping Results The metric MPE in Tab. 4 measures the deviation between each

method's interaction ending grasp and input goal grasp, as defined in Sec. C.3. The results show that except for the motion planning, our method has the lowest MPE among all the RL-based methods. Though the MPE of motion planning is the lowest, it has the lowest success rate, so it is unreliable. Especially, since our generated grasping proposal on unseen object categories is noisy, simply motion planning to the goal position cannot grasp the object firmly. The gaps between hand and object in generated data lead to the minimization of the MPE of MP but this kind of low MPE is helpless since it cannot seam the gap. On the contrary, our method can modify the generated noisy grasp goal and make the grasp possible.

Additional Qualitative Grasping Results We provide a qualitative demonstration of the diverse grasp proposals in Fig. 3. Comparing the two images of each row, one can see the diverse rotation predictions of GraspIPDF. The different hands in each image demonstrate the diverse translation and articulation predictions of GraspGlow. We also provide additional qualitative grasping results in Fig. 4. The left part of the figure is the visualization of the goal hand-grasping pose. For each row, the right part is the generated grasping sequence using the left part as the grasping goal, and we select four representative stages as pictures. From top to bottom, the four featured objects are a bottle, a camera, a toy dog, and a headphone. All the objects here are in the test data set, and the grasping goals are selected from our generated grasping proposals.

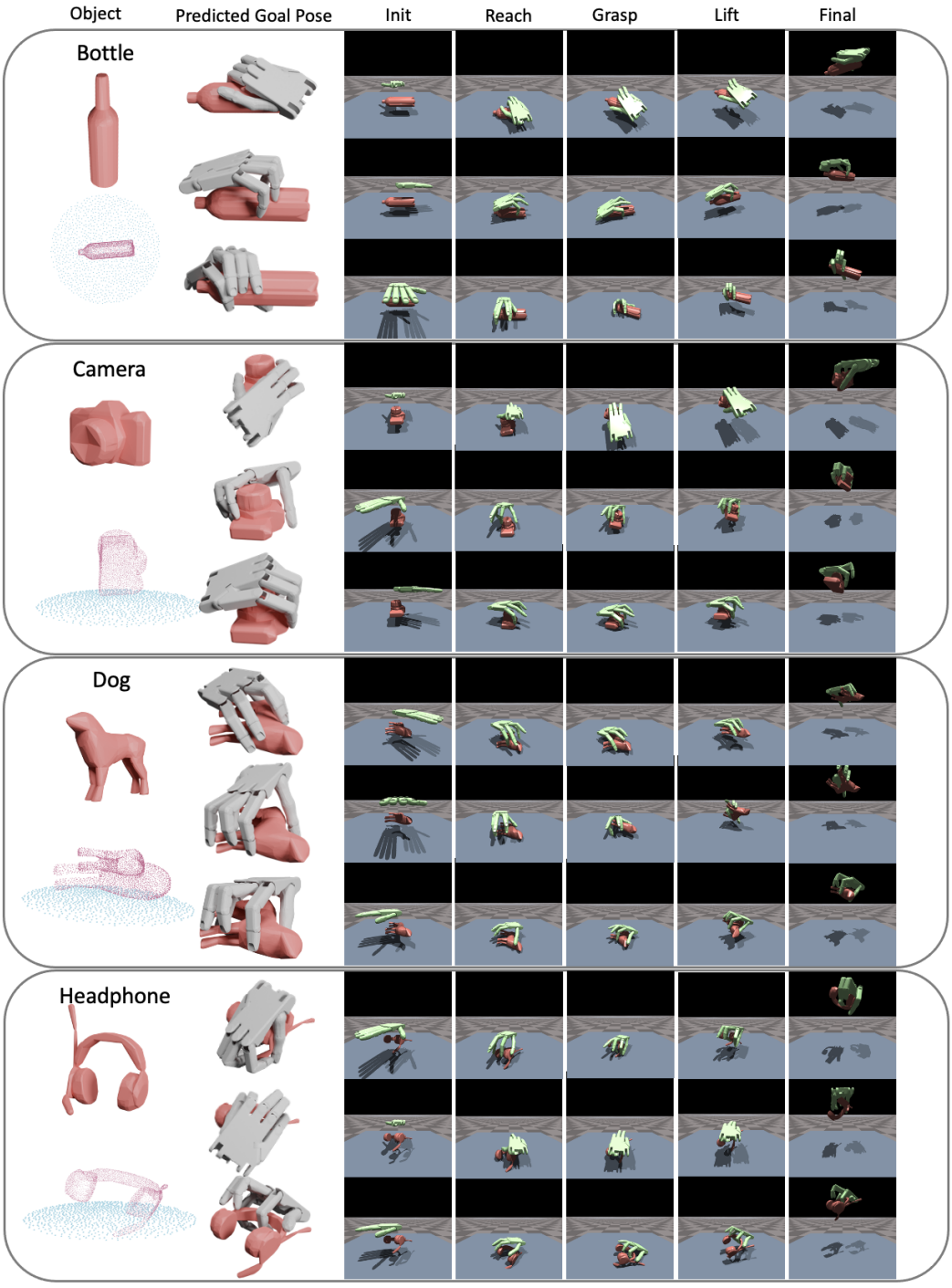

Figure 4. **Qualitative Grasping Results**. The left side includes objects and corresponding grasping poses generated using our method, and the right side is the policy-generated grasping sequence using the left corresponding part as the grasping goal. Object categories from top to bottom: bottle, camera, toy dog, and a headphone.

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 4

[2] Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3244–3253. PMLR, 2019. 3, 7

[3] Carlo Ferrari and John F Canny. Planning optimal grasps. In *ICRA*, volume 3, page 6, 1992. 5

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 5

[5] 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. 3

[6] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018. 2

[7] Nikos Kolotouros, Georgios Pavlakos, Dinesh Jayaraman, and Kostas Daniilidis. Probabilistic modeling for human mesh recovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11605–11614, 2021. 2, 3, 7

[8] 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. 1, 3

[9] Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. Synthesizing diverse and physically stable grasps with arbitrary hand structures by differentiable force closure estimation. *CoRR*, abs/2104.09194, 2021. 1

[10] Kieran A Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7882–7893, 2021. 2, 7

[11] Krishna Murthy Jatavallabhula, Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. *arXiv e-prints*, pages arXiv–1911, 2019. 2

[12] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 2

[13] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2

[14] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017. 4, 7

[15] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM TOG*, 2017. 3

[16] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 5

[17] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999. 4

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4, 5, 7

[19] Qijin She, Ruizhen Hu, Juzhan Xu, Min Liu, Kai Xu, and Hui Huang. Learning high-dof reaching-and-grasping via dynamic representation of gripper-object interaction. *arXiv preprint arXiv:2204.13998*, 2022. 5, 7

[20] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012. 1

[21] 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. *arXiv preprint arXiv:2210.02697*, 2022. 1

[22] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *arXiv preprint arXiv:2205.02961*, 2022. 1

[23] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. *arXiv preprint arXiv:2204.02320*, 2022. 3, 4, 7

[24] Anna Yershova, Swati Jain, Steven M. LaValle, and Julie C. Mitchell. Generating uniform incremental grids on so(3) using the hopf fibration. *The International Journal of Robotics Research*, 29(7):801–812, 2010. PMID: 20607113. 7

[25] Tianqiang Zhu, Rina Wu, Xiangbo Lin, and Yi Sun. Toward human-like grasp: Dexterous grasping via semantic representation of object-hand. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15741–15751, 2021. 1