Supplementary Material for UniDexGrasp++: Improving Dexterous Grasping Policy Learning via Geometry-aware Curriculum and Iterative Generalist-Specialist Learning

Abstract In this supplementary material, we present a comprehensive description of our method, baselines, and implementation in Section A. We also provide detailed information about our experiments in Section B, including the experimental setup and training details. Additionally, we report more comprehensive quantitative results in Section C. For more visualization of our pipeline and grasping demonstrations, please refer to the supplementary video.

A. Method and Implementation Details

We formalize our whole pipeline method in Algorithm A.

A.1. Details about Our Method

Algorithm A UniDexGrasp++

Require: Task Space $\mathbb{T}$, K State-based Specialists $\{SS^j\}$, a State-based Generalist SG, K Vision-based Specialists $\{VS^j\}$, a Vision-based Generalist VG

$\{\mathcal{C}_l\} \leftarrow$ ***GeoCurriculum***$(\mathbb{T})$ for object curriculum.
Geometry-aware task curriculum learning to train SG_0.
for $i = 1, 2, \ldots$ **do**:
  Initialize specialist $SS_i^j = SG_i$
  $\{c_j\} \leftarrow$***GeoClustering***$(\mathbb{T})$
  Online assign tasks that are nearest to c_j to specialist SS_i^j and train SS_i^j ▷ RL
  if $\{SS_i^j\}$ are optimal **then break**
  else
    Distill $\{SS_i^j\}$ to SG_{i+1} ▷ DAgger-based Distillation
  end if
end for
Distill $\{SS_i^j\}$ to VG_0 ▷ DAgger-based Distillation
for $i = 1, 2, \ldots$ **do**:
  Initialize specialist $VS_i^j = VG_i$
  $\{c_j\} \leftarrow$ ***GeoClustering***$(\mathbb{T})$
  Online assign tasks that are nearest to c_j to specialist VS_i^j and train VS_i^j ▷ RL
  Distill $\{VS_i^j\}_{i=1}^K$ to VG_{i+1} ▷ DAgger-based Distillation
  if VG_{i+1} is optimal **then break**
  end if
end for

Details of *GiGSL*: During the state-based policy learning stage, we terminate training when the success rate of the current policy SS_n is only marginally better than the previous policy SS_{n-1} (by less than 0.5%). At this point, we distill SS_n to the vision-based policy. In the vision-based policy learning stage, we stop training when the success rate of the current policy VG_n is only marginally better than the previous policy VG_{n-1} (by less than 0.5%). We then use VG_n as our final policy.

Details of *AutoEncoder*: We train the point cloud 3D autoencoder using the point cloud $\{P_{t=0}^{(k)}\}_{k=1}^{N\text{sample}}$ of the initialized objects in the sample tasks $\{\tau^{(k)}\}_{k=1}^{N\text{sample}}$. The autoencoder follows an encoder-decoder structure. The encoder $\mathcal{E}$ encodes $P_{t=0}^{(k)}$ and outputs the encoding latent feature $z^{(k)} = \mathcal{E}(P_{t=0}^{(k)})$. The decoder $\mathcal{D}$ takes $z^{(k)}$ as input and generates the point cloud $\hat{P}_{t=0}^{(k)}$.

$$z^{(k)} = \mathcal{E}(P_{t=0}^{(k)}) \tag{1}$$

$$\hat{P}_{t=0}^{(k)} = \mathcal{D}(z^{(k)}) \tag{2}$$

The model is trained using the reconstruction loss $\mathcal{L}_{\text{AE}}$, which is the Chamfer Distance between $P_{t=0}^{(k)}$ and $\hat{P}_{t=0}^{(k)}$.

$$\mathcal{L}_{\text{AE}} = \text{ChamferDistance}(P_{t=0}^{(k)}, \hat{P}_{t=0}^{(k)}) \tag{3}$$

Details of *GeoCurriculum*: In our implementation, we choose $N_{\text{level}} = 4$ and use a 4-stage *GeoCurriculum* to train, where the task number is 1-300-900-N_{train}. We also compare different N_{level} and the result can be found in Sec. C

A.2. Details about Baselines

PPO Proximal Policy Optimization (PPO) [8] is a popular model-free on-policy RL method. We adopt PPO as our RL baseline.

DAPG Demo Augmented Policy Gradient (DAPG) [6] is a popular imitation learning (IL) method that leverages expert demonstrations to reduce sample complexity.

Following the approach of ILAD [10], we generate demonstrations using motion planning.

ILAD ILAD [10] is an imitation learning method that enhances the generalizability of DAPG. It introduces a novel imitation learning objective on top of DAPG, which jointly learns the geometric representation of the object using behavior cloning from the generated demonstrations during policy learning. We use the same generated demonstrations as in DAPG in this method.

GSL Generalist-Specialist Learning (GSL) [3] is a three-stage learning method that first trains a generalist using RL on all environment variations, then fine-tunes a large population of specialists with weights cloned from the generalist, each trained using RL to master a selected small subset of variations. Finally, GSL uses these specialists to collect demonstrations and employs DAPG for the IL part to train a generalist. For a fair comparison, we adopt PPO [8] for the RL part and DAPG [6] for the IL part in our implementation.

UniDexGrasp UniDexGrasp [11] is a two-stage learning method. In the first state-based stage, they propose Object Curriculum Learning (OCL), which starts RL with one object and gradually incorporates similar objects from the same or similar categories into the training to obtain a state-based teacher policy. Once they obtain this teacher policy, they use DAgger [7] to distill it to a vision-based policy.

B. Experiment Details

As described in Sec.4, we use PPO [8] in *GeoCurriculum* learning stage to get the first generalist SG_1, and in specialist learning stage $\{\text{SS}_i\}$, $\{\text{VS}_i\}$ to train these specialist. In the generalist learning stages $SG_i (i > 1)$ and VG_i, we employ our proposed DAgger-based policy distillation. Note that we freeze the vision-backbone in the $\{\text{VS}_i\}$ learning stage.

B.1. Environment Setup

State Definition The full state of the state-based policy is denoted as $\mathcal{S}_t^{\mathcal{S}} = (R_t, O_t, P_{t=0})$, while the full state of the vision-based policy is represented as $\mathcal{S}_t^{\mathcal{V}} = (R_t, P_t)$. The robot state R_r is detailed in Table A, and the object oracle state O_t includes the object pose (3 degrees of freedom for position and 9 degrees of freedom for rotation matrix), linear velocity, and angular velocity. To accelerate the training process, we sample only 1024 points from the object and the hand in the scene point cloud P_t.

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 A. Robot state definition.

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

Camera Setup Similar to UniDexGrasp [11], we employ a setup consisting of five RGBD cameras positioned around and above the table, as shown in Fig. A. The system's origin is located at the center of the table, and the cameras are positioned at ([0.5, 0, 0.05], [-0.5, 0, 0.05], [0, 0.5, 0.05], [0, -0.05, 0.05], [0, 0, 0.55]), with their focal points set to [0, 0, 0.05]. We fuse the partial point clouds generated by the five cameras to one scene point cloud P_t.

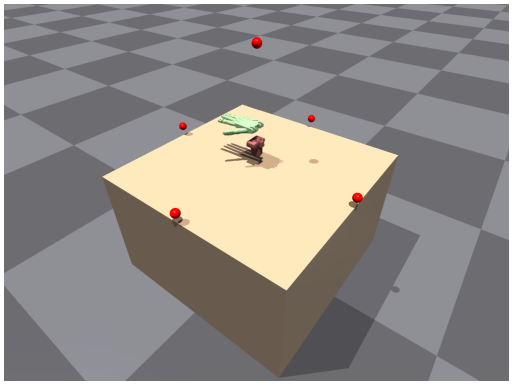

Figure A. **Camera positions**

Reward Function: We use the non-goal-conditioned reward version in UniDexGrasp [11], and we formalize it as follows (All the ω_* here are hyper-parameters same with UniDexGrasp.):

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. f is a flag to judge whether the robot reaches the lifting condition: $f = \mathbf{Is}(\sum \|\mathbf{x}_{\text{finger}} - \mathbf{x}_{\text{obj}}\|_2 < \lambda_{f_1}) + \mathbf{Is}(d_{\text{obj}} > \lambda_0)$. 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 ($\omega_l > 0$).

$$r_{\text{lift}} = \begin{cases} \omega_l * (1 + a_z) & \text{if } f = 2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

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 closely 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:

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

B.2. Training Details

Network Architecture: The MLP used in the state-based policy $\pi_{\mathcal{E}}$ and the vision-based policy $\pi_{\mathcal{S}}$ consists of 4 hidden layers (1024, 1024, 512, 512). We use the exponential linear unit (ELU) [1] as the activation function. The network structure of the PointNet in the autoencoder is (1024, 512, 64). We use the PointNet + Transformer backbone in [4] as our vision backbone, where we use different PointNets [5] to process points having different segmentation masks (robot, object, entire point cloud). There's also an additional MLP to output a 256-d hidden vector for the robot state alone. All the features from the MLP and PointNets are fed into a Transformer [9]. The output vectors are passed through global attention pooling to extract a representation of dimension 256, which is then provided into a final MLP with layer sizes [256, 128, feature_dim] to output a visual feature, that is then concatenated with the robot state.

Hyperparameters of Training: The hyperparameters in our experiments are listed in Tab.B.

Training time: The experiment is done on four NVIDIA RTX 3090 Ti. The training process consists of 20,000 environment steps in the first stage of *GeoCurriculum* and 15,000 environment steps (for every single policy) in other stages. It needs two days in total.

Hyperparameter	Value
Num. envs (Isaac Gym, state-based)	1024
Num. envs (Isaac Gym, vision-based)	32
Env spacing (Isaac Gym)	1.5
Num. rollout steps per policy update (PPO)	8
Num. rollout steps per policy update (DAgger)	1
Num. batches per agent	4
Num. learning epochs	5
Buffer size (DAgger)	2000
Episode length	200
Saturation threshold of policy iteration	0.005
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
N_{sample}	270,000
N_{train}	3200
N_{clu}	20
ω_r	0.5
ω_l	0.1
ω_m	2
ω_b	10

Table B. **Hyperparameter for grasping policy.**

C. Additional Results and Analysis

This section contains extended results of the experiment depicted in Sec. 5.

More ablation on *GeoCurriculum*. We do additional ablation experiments on the effectiveness of *GeoCurriculum*, and the results are presented in Table C. Specifically, we compare our proposed *GeoCurriculum* approach with not using any curriculum learning and with the object-curriculum-learning (OCL) method proposed in [11]. Our findings indicate that curriculum learning is essential for achieving success in the challenging dexterous grasping task with large variations in object instances and their initial poses. Moreover, we observed that our *GeoCurriculum* approach, which considers the geometric similarity of different objects and poses, outperforms the OCL method, which only considers the category label of objects. In addition, we do an ablation study on the number of curriculum learning stages. For the 3-stage *GeoCurriculum*, the task number is 1-100-N_{train}; for the 4-stage *GeoCurriculum*, the task number is 1-300-900-

N_{train}; and for the 5-stage *GeoCurriculum*, the task number is 1-20-100-1000-N_{train}. We compared the performance of SG_1 for all the experiments. Since the performance of the 5-stage *GeoCurriculum* is similar to that of the 4-stage *GeoCurriculum*, we choose the 4-stage in our main experiment for simplicity.

Model	Train(%)	Test(%)	
		Uns. Obj. Seen Cat.	Uns. Cat.
No Curriculum	30.5	23.4	20.6
OCL[11]	79.4	74.3	70.8
GeoCurriculum (3)	81.3	75.6	73.3
GeoCurriculum (4)	82.7	**76.8**	**74.2**
GeoCurriculum (5)	**82.9**	76.4	74.0

Table C. **Ablation study on *GeoCurriculum*.** OCL refers to the Object Curriculum Learning proposed in [11]. The numbers in brackets represent the number of stages for curriculum learning.

More ablation study on *iGSL* For the policy distillation method used in iterative Generlist-Specilist Learning (*iGSL*), we compare our DAgger-based policy distillation with several popular imitation learning methods, including Behavior Cloning (we also add a value function learning to make the process iterative), GAIL [2] and DAPG [11]. We use *GeoCurriculum* for all the methods and compare the performance of SG_{n+1}. Tab.D shows the results, which demonstrate that our DAgger-based policy distillation method significantly outperforms other methods. Notably, our method uses the teacher checkpoint, while other methods only use the demonstrations from the teacher.

Model	Train(%)	Test(%)	
		Uns. Obj. Seen Cat.	Uns. Cat.
BC + Value	12.4	8.6	8.4
GAIL[2]	30.7	26.9	26.0
DAPG[11]	61.4	52.6	47.9
Ours	**87.9**	**84.3**	**83.1**

Table D. **Ablation study on the policy distillation method.**

More ablation study on *GiGSL* We provide more ablation results of *GiGSL*. First, we do ablation experiments on the cluster number N_{clu} in *GeoClustering*. We compare the performance of the final vision-based policy VG_n for different N_{clu}. The results are in Tab.E which show that increasing N_{clu} beyond a certain point does not improve performance and may even decrease it.

Then, we compare our *GeoClustering* with random clustering and category label-based clustering (we evenly divide all the categories into N_{clu} parts for a fair comparison). In category class-based clustering, we pre-train a classification task on all the objects and their initial poses. We then use the feature of the second-to-last layer for clustering and concatenate this feature to the robot state and object state in the state-based policy learning. We compare the performance of the final vision-based policy VG_n for different methods. The results are shown in Tab.F.

Model	Train(%)	Test(%)	
		Uns. Obj. Seen Cat.	Uns. Cat.
0 (No specialist)	77.4	72.6	68.8
10	80.3	74.9	75.2
20	**85.4**	**79.6**	**76.7**
50	77.2	71.2	69.9

Table E. **Ablation study on the cluster number.**

Model	Train(%)	Test(%)	
		Uns. Obj. Seen Cat.	Uns. Cat.
Random	77.0	71.9	68.2
Category Label.	79.7	73.9	74.1
Ours	**85.4**	**79.6**	**76.7**

Table F. **Ablation study on the pre-trained autoencoder.** The features from the encoder are used in *GeoClustering* in the state-based setting.

More Results on Meta-World Here we show additional results on Meta-World [12], a popular multi-task policy learning benchmark. The MT-10 task consists of 10 diverse and challenging tasks, such as opening a door or picking up objects, that require a wide range of skills and abilities. The MT-50 task set is an extension of the MT-10 task set and includes 50 additional tasks that are even more complex and diverse. The results in Tab.G demonstrate that our proposed technique, ***iGSL***, performs well on the multi-task MT-10 & MT-50 and outperforms the baseline methods.

	PPO[8]	GSL[3]	Ours
MT-10 (%)	58.4±10.1	77.5±2.9	**80.3±0.5**
MT-50 (%)	31.1±4.5	43.5±2.2	**45.9±1.7**

Table G. **Addtional Experiment in Meta-World.**

Additional Qualitative Grasping Results We show more qualitative results in Fig.B and Fig.C. In Fig.B, we provide more results about our *GeoClustering* in the vision-based policy learning stage. The vision-based policy VG_1 utilizes its vision backbone to extract visual features of the tasks for clustering. Due to the vision-based clustering be-

ing task-aware, we also show the grasping poses of these tasks. The results in Fig.B demonstrate that our approach can cluster tasks based on the object geometry, pose features, and corresponding grasping strategy of the generalist policy. In Fig.C, we provide several grasping trajectories for different objects with different initial poses.

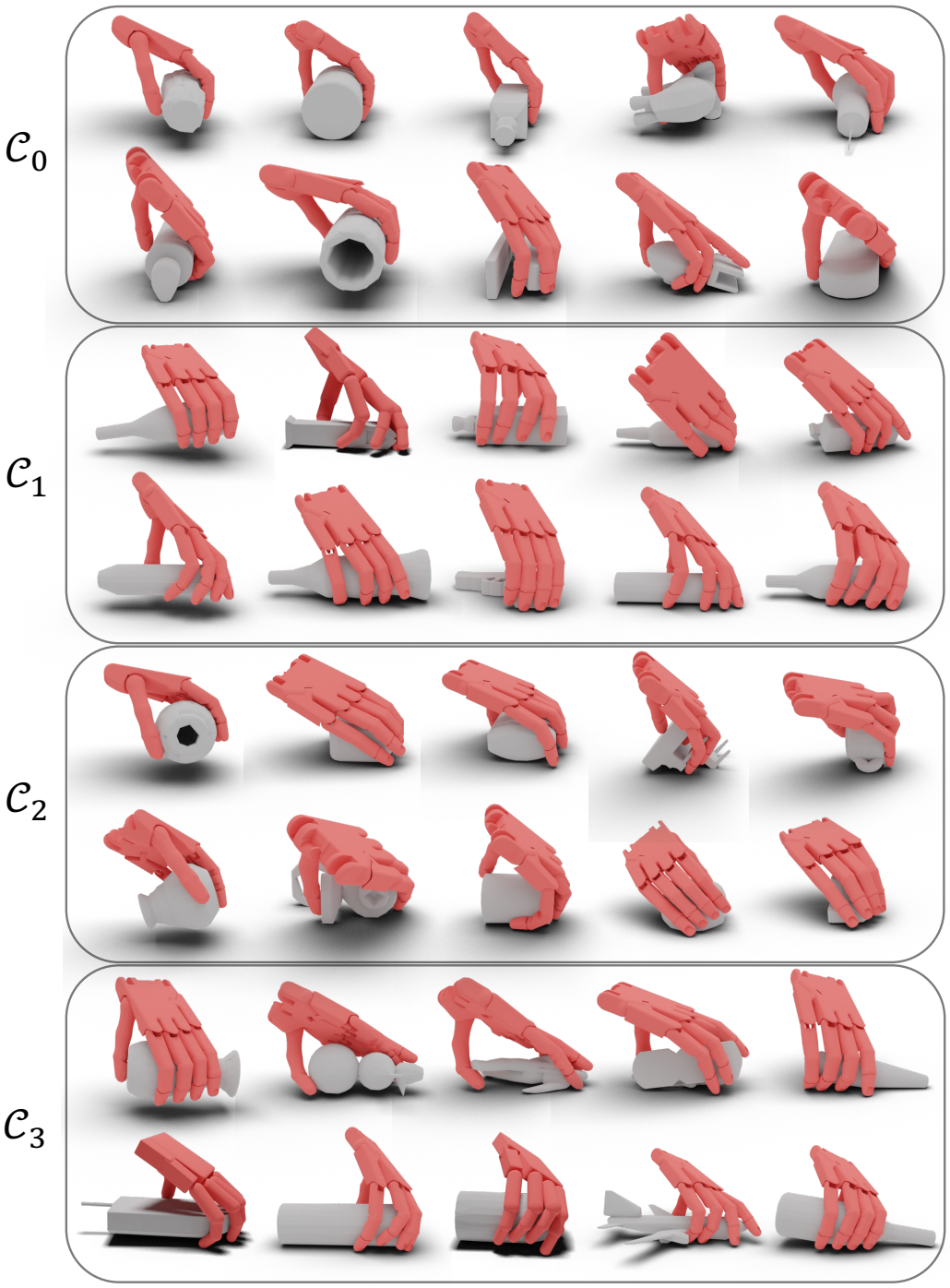

Figure B. **Qualitative Grasping Results**. For each of the 4 clusters, we visualize 10 tasks and their corresponding grasping poses of the policy. The clusters are generated by our *GeoClustering* in the vision-based policy learning stage.

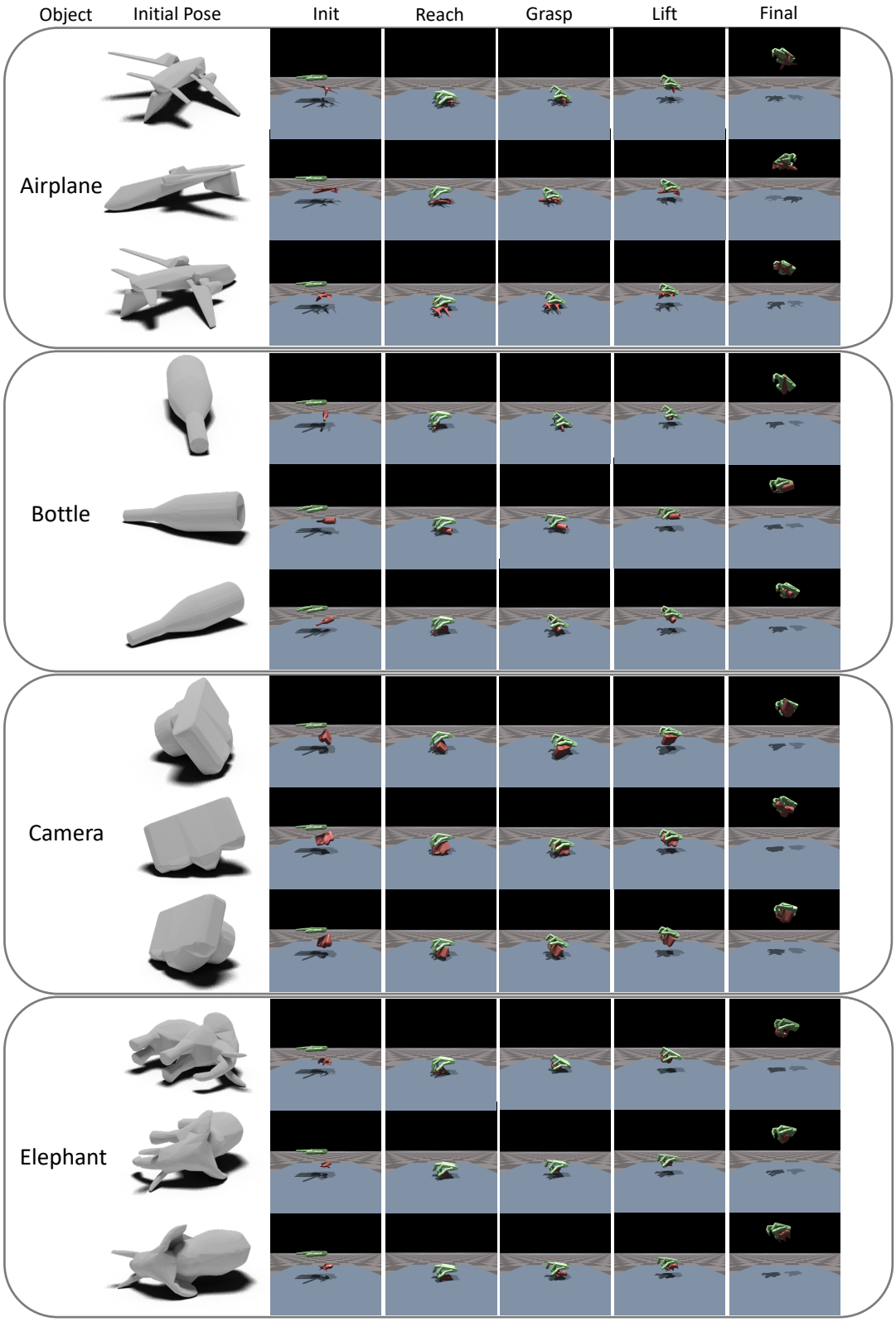

Figure C. **Qualitative Grasping Trajecoties**. We provide several grasping trajectories for different objects with different initial poses.

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. 3

[2] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016. 4

[3] Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, and Hao Su. Improving policy optimization with generalist-specialist learning. In *International Conference on Machine Learning*, pages 10104–10119. PMLR, 2022. 2, 4

[4] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021. 3

[5] 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. 3

[6] 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. 1, 2

[7] 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. 2

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 1, 2, 4

[9] 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*, 30, 2017. 3

[10] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. *arXiv preprint arXiv:2204.02320*, 2022. 2

[11] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, Tengyu Liu, Li Yi, and He Wang. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy, 2023. 2, 3, 4

[12] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020. 4