UniGraspTransformer: Simplified Policy Distillation for Scalable Dexterous Robotic Grasping

Supplementary Material

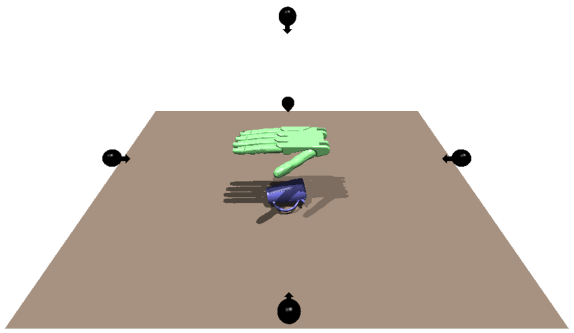

Figure 1. Illustration of the simulation environment.

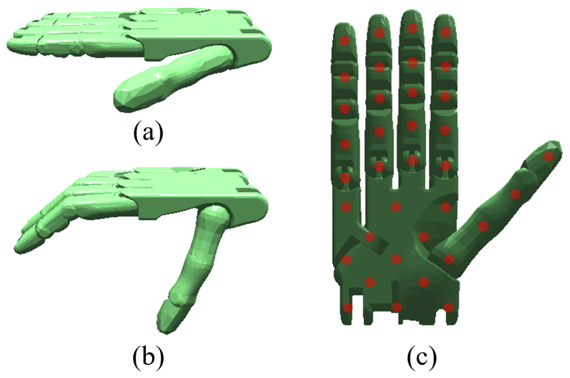

Figure 2. Shadow Hand poses. (a) Initial pose at the first frame. (b) Pre-contact opening pose used in dedicated policy training. (c) 36 selected hand points for computing hand to object distance.

A. Implementation Details

A.1. Environment Setup

Initialization. We use Isaac Gym 3.0 [3] to build simulation environments, each containing a table (brown), an object placed on top (blue), a controllable Shadow Hand (green) [9], and five surrounding cameras (black), as illustrated in Figure 1. The system's origin is defined at the center of the table, where all objects are initially placed. The Shadow Hand is positioned 0.2 meters above the table center, with the goal located 0.3 meters above the table center.

For each object utilized in our project, we randomly drop it onto the table with arbitrary rotations to generate a dataset comprising 12K static tabletop poses. This dataset is divided into three subsets for specific purposes: 10K poses for dedicated policy training, 1K poses for offline trajectory generation, and 1K poses for evaluation.

Task Definition. The objective is to develop a robust universal policy capable of controlling the Shadow Hand [9] to grasp and transport a diverse range of tabletop objects to a designated midair goal position. Each grasping consists of 200 execution steps and is deemed successful if the positional difference between the object and the goal remains within a predefined threshold by the end of the sequence.

Observation Space. At each simulation step, the observation space of state-based UniGraspTransformer includes a 167-d proprioceptive state of the hand, a 24-d representation of the hand's previous action, a 16-d object state, a 128-d object visual feature, a 36-d hand-to-object distance, and a 29-d time embedding, as detailed in Table 1 of the main paper. During dedicated RL policy training, the 128-d object visual feature is excluded to enhance training efficiency. For vision-based UniGraspTransformer training, the original 16-d object state is replaced by the center position of the partial object point cloud (3-d) and its three principal component axes (3×3-d). Additionally, we compute 36 distances between 36 selected points on the Shadow Hand and the partial object point cloud, as illustrated in Figure 2(c).

Action Space. The action space comprises motor commands for 24 actuators on the Shadow Hand. The first 6 actuators manage the wrist's position and orientation through applied forces and torques, while the remaining 18 actuators control the positions of the finger joints. The action values are normalized to a range of -1 to 1 according to the specifications of the actuators.

Camera Setup. Following a similar approach to UniDexGrasp++ [11], five RGBD cameras are mounted around the table, as illustrated in Figure 1. The cameras are positioned relative to the table center at coordinates (0.0, 0.0, 0.55), (0.5, 0.0, 0.15), (-0.5, 0.0, 0.15), (0.0, 0.5, 0.15), and (0.0, -0.5, 0.15), with their focal points aligned at (0, 0, 0.15). In the vision-based setting, the depth images captured by these cameras are fused to generate a scene point cloud, from which the partial point cloud of the object is segmented.

A.2. Dedicated Policy Training

PPO. Proximal Policy Optimization [8] is a widely used model-free, on-policy reinforcement learning algorithm that simultaneously learns a policy and estimates a value function. We utilize PPO to train dedicated RL policies for each of the 3,200 objects. Both the policy and value networks are implemented as 4-layer MLPs with hidden dimensions of $\{1024, 1024, 512, 512\}$. At each simulation step, the policy network takes the current observation as input and outputs a 24-d action, while the value network predicts a 1-d value. The simulation environment then executes the action, updates the observation, and calculates the corresponding reward. The policy and value networks are updated every 16 simulation steps using the collected observations, actions, values, and rewards. Each dedicated RL policy is trained on an NVIDIA V100 GPU for a total of 10,000 update iterations, taking approximately 3 hours to complete.

Reward Function. The reward function described in Eq.(1) of the main paper comprises five components: R_d, R_o, R_l, R_g, and R_s. These reward components are governed by a contact flag f_c, which indicates whether the hand is in contact with the object.

The distance reward R_d penalizes the average Chamfer Distance between the hand points H_i and the object point cloud P_{obj}, promoting contact and encouraging the hand to maintain a secure grasp on the object's surface. Specifically, 36 points $\{H_i\}_{i=1}^{36}$ are selected on the Shadow Hand for this calculation, as illustrated in Figure 2(c).

$$R_d = -\omega_d \frac{1}{36} \sum_{i=1}^{36} ChamferDistance(H_i, P_{obj}), \quad (1)$$

where the reward weight ω_d is set to 1.0.

The contact flag f_c is set to 1 if the average Chamfer Distance between the hand points and the object point cloud falls below a predefined threshold $\lambda_c = 0.06$. This is determined as follows:

$$f_c = \mathbb{1}[\frac{1}{36} \sum_{i=1}^{36} ChamferDistance(H_i, P_{obj}) < \lambda_c], \quad (2)$$

where $\mathbb{1}[\cdot]$ denotes the indicator function.

Inspired by DexGraspNet [12], before the contact is established, the opening reward R_o penalizes deviations of the current hand pose q from a predefined opening pose q_{open}, as depicted in Figure 2(b). This encourages the hand to remain open until it makes contact with the object. The reward is calculated as:

$$R_o = -\omega_o \parallel q - q_{open} \parallel_2, \quad (3)$$

where the reward weight ω_o is set to 0.1.

Once contact is established, the rewards R_l, R_g, and R_s are introduced to guide the grasping process:

- Lift reward (R_l): This reward encourages the hand to perform a lifting action a_z along the z axis:

  $$R_l = \omega_l(1 + a_z), \quad (4)$$

  where ω_l is set to 0.1.
- Goal reward (R_g): This reward penalizes the Euclidean distance between the object center position x_{obj} and the target goal position x_{goal}:

  $$R_g = -\omega_g \parallel x_{obj} - x_{goal} \parallel_2, \quad (5)$$

  where ω_g is set to 2.0.
- Success reward (R_s): This reward provides a bonus when the object successfully reaches the goal position, defined by a threshold $\lambda_g = 0.05$:

  $$R_s = \omega_s \mathbb{1}[\parallel x_{obj} - x_{goal} \parallel_2 < \lambda_g], \quad (6)$$

  where ω_s is set to 1.0.

A.3. Grasp Trajectory Generation

Our 3,200 dedicated RL policies achieve an average success rate of 94.1% across all 3,200 training objects. For each object, we randomly initialize it in diverse poses and apply its corresponding RL policy to generate $M = 1000$ successful trajectories, which are used for offline training of the UniGraspTransformer model. Each trajectory, $\mathcal{T} = \{(S_1, A_1), \ldots, (S_t, A_t), \ldots, (S_T, A_T)\}$, consists of a sequence of steps. Here, A_t represents robotic action at timestep-t, and S_t captures the environment state, including proprioception (167-d), previous action (24-d), object state (16-d), hand-object distance (36-d), and time embedding (29-d), as detailed in Table 1 of the main paper. Additionally, we save the complete object point cloud (1024×3-d) and the partial object point cloud (1024×3-d), which are used to generate object features to train the state-based and vision-based versions of the UniGraspTransformer model.

A.4. Point Cloud Encoder Training

S-Encoder. To train our S-Encoder, we use a dataset consisting of 3,200 point clouds of seen objects, denoted as $\{P_i\}_{i=1}^{3200}$, where each P_i represents the canonical point cloud of a specific object. During each training iteration, a batch of 100 object point clouds is randomly sampled from this dataset. For each point cloud in the batch, indexed by j ($j = 1, 2, \ldots, 100$), the centroid $\mathbf{c}_j$ is subtracted to center the point cloud, followed by the application of a random rotation matrix R_j. The resulting transformed point cloud is expressed as $\hat{P}_j = R_j\,(P_j - \mathbf{c}_j)$, which serves as input to the S-Encoder.

The S-Encoder, as part of an encoder-decoder framework [5], processes $\hat{P}_j$ to produce a latent feature z_j. This latent feature is then passed to the decoder, which reconstructs the point cloud, yielding $\tilde{P}_j$. The model is trained by minimizing the reconstruction loss $\mathcal{L}_{CD}$, defined as the Chamfer Distance between the original transformed point cloud $\hat{P}_j$ and its reconstruction $\tilde{P}_j$:

$$\mathcal{L}_{CD} = ChamferDistance(\hat{P}_j, \tilde{P}_j). \quad (7)$$

The S-Encoder is trained for 800K iterations on an NVIDIA A100 GPU. After training, the state-based object features are generated by encoding the complete object point clouds using the trained S-Encoder.

V-Encoder. The V-Encoder is trained using a knowledge distillation approach, leveraging the pre-trained S-Encoder

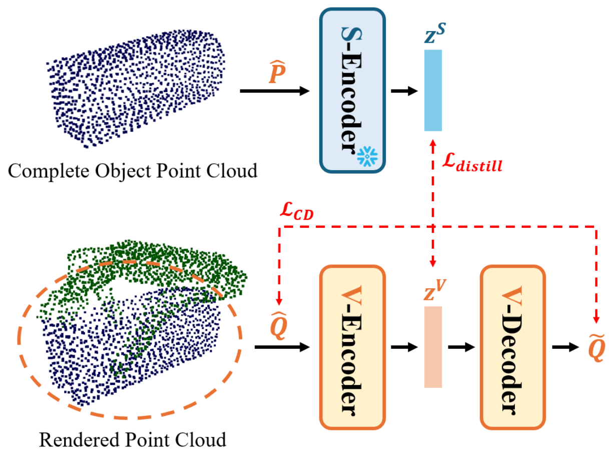

Figure 3. V-Encoder training with distillation.

and the grasp trajectories $\mathcal{T}$ generated by the dedicated RL policies, as illustrated in Figure 3. In each training iteration, a batch of 100 steps is randomly sampled from the generated trajectories. Both the complete object point cloud P_t and the partial point cloud Q_t are centered by subtracting their mean positions. The centered complete point cloud $\hat{P}_t$ is passed through the pre-trained S-Encoder (with frozen weights), producing a latent feature z_t^S. Simultaneously, the centered partial point cloud $\hat{Q}_t$ is fed to the V-Encoder, which outputs both a latent feature z_t^V and a reconstructed point cloud $\tilde{Q}_t$.

The V-Encoder is optimized using two loss functions:

- Feature Distillation loss ($\mathcal{L}_{distill}$): This L2 loss measures the difference between the latent features produced by the S-Encoder and the V-Encoder:

$$\mathcal{L}_{distill} = \| z_t^S - z_t^V \|_2 \tag{8}$$

- Reconstruction loss ($\mathcal{L}_{CD}$): This is the Chamfer Distance between the centered partial point cloud $\hat{Q}_t$ and its reconstruction $\tilde{Q}_t$:

$$\mathcal{L}_{CD} = ChamferDistance(\hat{Q}_t, \tilde{Q}_t), \tag{9}$$

The total loss for training the V-Encoder is defined as:

$$\mathcal{L} = \omega_{CD}\mathcal{L}_{CD} + \omega_{distill}\mathcal{L}_{distill}, \tag{10}$$

where the weights are set to $\omega_{CD} = 1.0$ and $\omega_{distill} = 0.1$. The V-Encoder is trained on an NVIDIA A100 GPU for 800K iterations. After training, the vision-based object features are generated by encoding the partial object point clouds using the trained V-Encoder.

A.5. UniGraspTransformer Training

Input Types. The UniGraspTransformer is trained using the generated grasp trajectories and encoded object features under two configurations, as outlined in Table 1:

Input of UniGraspTransformer	
State-Based	Vision-Based
Proprioception (167)	Proprioception (167)
Previous Action (24)	Previous Action (24)
Object State (16)	Object State* (12)
Object Feature (128)	Object Feature* (128)
Hand-Obj. Dist. (36)	Hand-Obj. Dist.* (36)
Time (29)	Time (29)

Table 1. Input types for state-based and vision-based UniGraspTransformer, organized into six groups.

- State-Based Setting: The complete object point clouds are assumed to be perfectly accurate and are encoded using the S-Encoder. Object states, including positions, rotations, and velocities, are directly accessible, as detailed in Table 1 of the main paper.
- Vision-Based Setting: Partial object point clouds are reconstructed and segmented from depth data captured by five cameras mounted above and around the table. These object features are encoded using the V-Encoder, and object states are estimated rather than directly accessed.

The key differences between the inputs for the state-based and vision-based UniGraspTransformer are:

- For the object state representation, the vision-based setting uses the center of the partial object point cloud (3-d) as the object position and three principal component axes (9-d) to represent object orientation.
- The object feature is derived from the partial object point cloud and encoded using the V-Encoder in the vision-based setting.
- The hand-object distance is computed using the partial object point cloud in the vision-based setting.

Training Process. Each trajectory step consists of six observation groups, as detailed in Table 1, paired with a ground truth action A_t. The UniGraspTransformer processes these observations as follows: (1) The six observation groups are converted into six 256-dimensional tokens using individual single-layer MLPs; (2) These tokens are passed through 12 self-attention layers [10], producing six refined 256-dimensional features; (3) The six features are concatenated into a single 1536-dimensional representation, which is then processed by a 4-layer MLP to predict the final 24-d action P_t. The model is optimized using a single L2 loss, defined as: $\mathcal{L} = ||A_t - P_t||_2$.

Training is conducted on a dataset of 3,200 objects with 3.2 million trajectories, using a batch size of 800 trajectories (each with 200 steps) over 100 epochs. The process is carried out on 8 NVIDIA A100 GPUs and takes approximately 70 hours to complete. The average L2 loss at convergence is around 0.015.

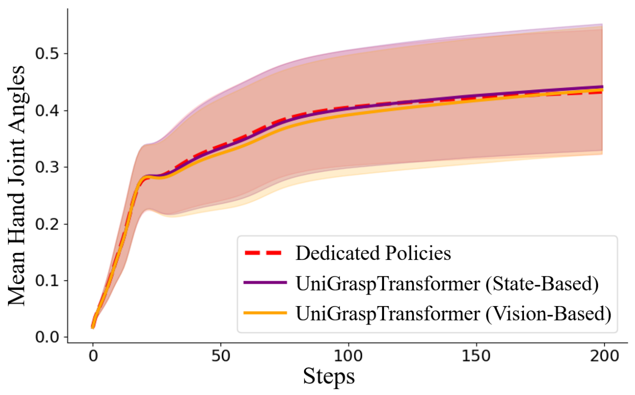

Figure 4. Quantitative analysis of grasp pose diversity.

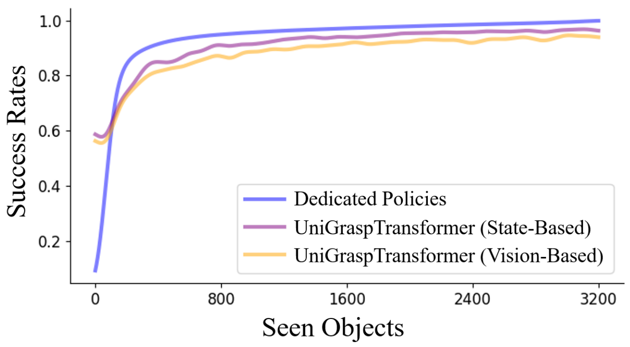

Figure 5. Success rates across seen objects.

B. Experiment Details

B.1. Baseline Methods

The implementation of baseline methods listed in Table 2 of the main paper is outlined below. Additional details can be found in UniDexGrasp++[11].

PPO. This reinforcement learning baseline directly trains a state-based universal model using PPO with all training objects. The vision-based universal policy is derived from the state-based policy through distillation using DAgger [7].

DAPG. Demo Augmented Policy Gradient (DAPG) [6] is a widely used imitation learning method that leverages expert demonstrations to reduce RL sampling complexity. In this baseline, grasp trajectories generated via motion planning serve as demonstrations to train a state-based deep RL policy. The vision-based universal policy is then distilled from the state-based policy using DAgger [7].

ILAD. ILAD [13] enhances the generalization capabilities of DAPG [6] by introducing an imitation learning objective focused on the object's geometric representation. In this baseline, a pipeline similar to DAPG [6] is implemented.

GSL. Generalist-Specialist Learning (GSL) [2] begins by training a generalist policy using PPO over the entire task space. Specialists are then fine-tuned to master each subset of the task space. The final generalist is trained using DAPG [6], leveraging demonstrations generated by the trained specialists.

UniDexGrasp. UniDexGrasp [14] decomposes the grasping task into two stages: static grasp pose generation followed by dynamic grasp execution via goal-conditioned reinforcement learning. First, an IPDF-based [4] grasp pose generator is trained using all training objects. An Object Curriculum Learning protocol is then applied, starting reinforcement learning with a single object and gradually incorporating similar objects to train a state-based universal policy. Finally, DAgger [7] is used to distill the state-based universal policy into a vision-based universal policy.

UniDexGrasp++. UniDexGrasp++ [11] builds on the Generalist-Specialist Learning framework by integrating geometry-based clustering during specialist training, where each specialist focuses on a group of geometrically similar objects. Additionally, it introduces a generalist-specialist iterative process in which specialists are repeatedly trained from the generalist, followed by generalist distillation.

C. More Analysis

From Dedicated to Universal. Our 3,200 dedicated RL policies achieve an average success rate of 94.1% across all 3,200 training objects. In comparison, the UniGraspTransformer achieves success rates of 91.2% (88.9%) on 3,200 seen objects, 89.2% (87.3%) on 140 unseen objects from seen categories, and 88.3% (86.8%) on 100 unseen objects from unseen categories under the state-based (vision-based) settings, respectively.

As depicted in Figure 4, the UniGraspTransformer effectively replicates the grasping trajectories generated by the dedicated RL policies through offline distillation. While there is a minor performance drop from 94.1% to 91.2% (88.9%) in the state-based (vision-based) setting, as illustrated in Figure 5, the model demonstrates robust generalization and efficiency.

Qualitative Results. The progressive online distillation approach [2] employed in UniDexGrasp++[11] results in a universal policy that tends to grasp different objects using similar poses. In contrast, our UniGraspTransformer, utilizing a larger model and an offline distillation framework, demonstrates the ability to grasp objects of various shapes with a wide range of diverse poses. This increased diversity in grasping strategies is further highlighted in Figure 6.

Real-World Deployment. We extend the deployment of our vision-based UniGraspTransformer to a real-world environment using the Inspire Hand [1], which features six active DoFs for its fingers. The training process remains identical to that used for the Shadow Hand. Demonstration videos showcasing grasping across 12 distinct objects are provided in the supplementary materials.

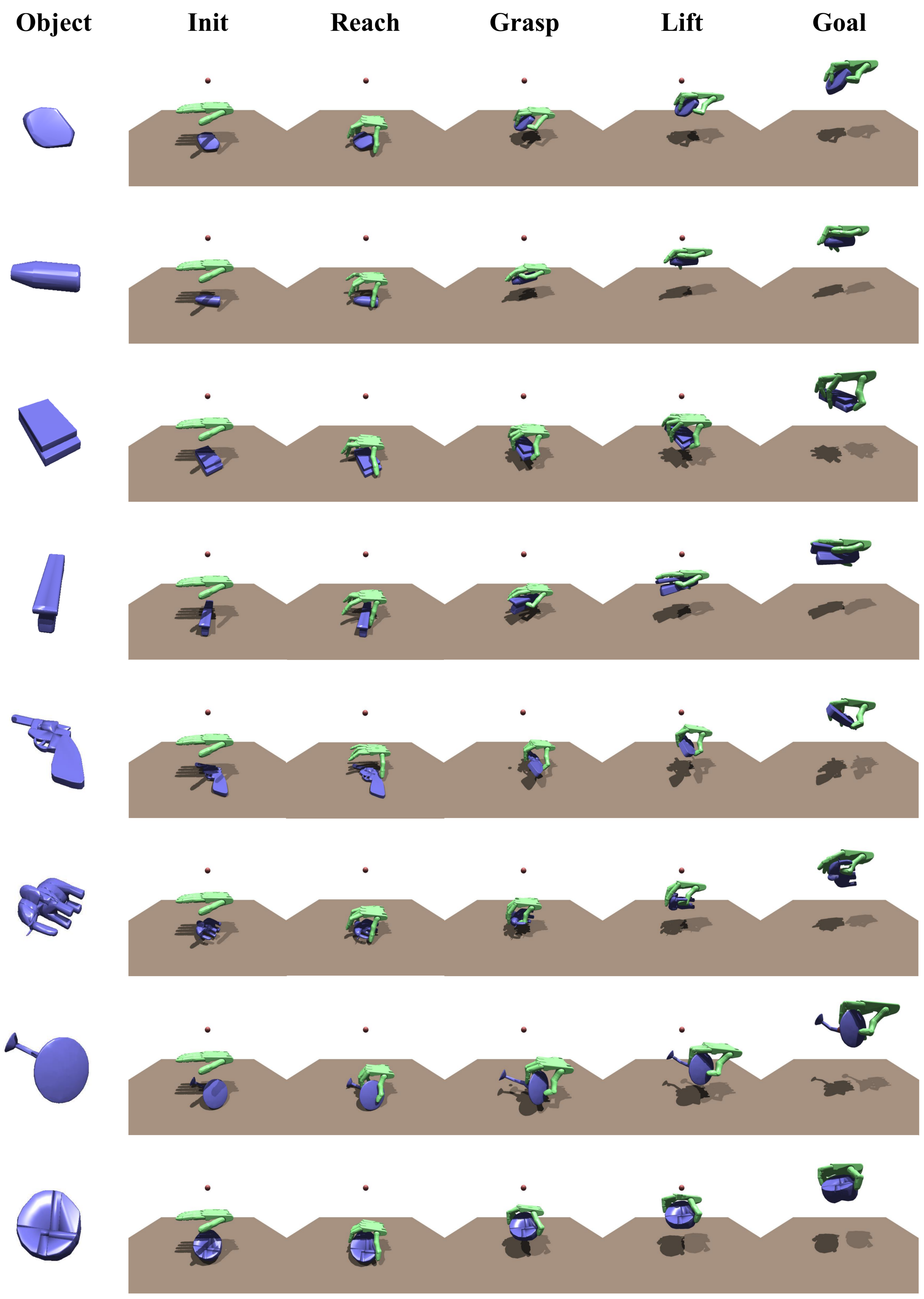

Figure 6. Qualitative analysis of the grasp pose diversity achieved by UniGraspTransformer.

References

[1] InspireRobots. https://inspire-robots.store/collections/the-dexterous-hands, 2016. 4

[2] Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, and Hao Su. Improving policy optimization with generalist-specialist learning, 2022. 4

[3] 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. 1

[4] Kieran Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold, 2022. 4

[5] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2

[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, 2018. 4

[7] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. 4

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 1

[9] ShadowRobot. https://www.shadowrobot.com/dexterous-hand-series, 2005. 1

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 3

[11] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning, 2023. 1, 4

[12] 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, 2023. 2

[13] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance, 2022. 4

[14] 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. 4