DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation Supplementary Material

In this supplementary material, we provide more details of the implementation. For more visualization, please see our video and project page: https://yzqin.github.io/dexpoint.

A Environment Details

Object Set. For single-object experiments, we use the single object mesh from the YCB dataset for training and testing. For multi-object experiments, we chooses 10 objects from the ShapeNet dataset [1] for training, and another 40 objects for testing in simulation. The test objects of real-world experiments are described in the main paper.

Object Pre-processing. To use the above object models in the simulation experiments, we apply the object preprocessing. The preprocessing includes two steps: scaling and convex decomposition. The scaling step is to ensure that the object size is within a proper ranger for manipulation. The YCB objects [2] are captured by the real scan, so the scale of object mesh from YCB dataset aligns with the real counterpart and can be used for robot manipulation directly. Different from the YCB dataset, the ShapeNet dataset does not contain any scale information, e.g. the mug in ShapeNet can be larger than the robot. Therefore, we scale each object based on the diagonal length of its bounding box. For each category, we manually select a diagonal length so that all object instances from the category will have the same bounding-box diagonal length after scaling. Besides, we will not use objects with non-manifold geometry to avoid instability in physical simulation. The next step is convex decomposition. We use V-HACD [3] to decompose both YCB object and ShapeNet object into convex parts with default parameters. We will not use the object with more than 40 parts after convex decomposition for both stability and efficiency of the physical simulation.

Reward. As is mentioned in Section 3.2 of the main paper, the overall reward for our task is composed of four parts: reach, contact, lift, and action penalty.

$$\mathcal{R} = w_{\text{reach}} r_{\text{reach}} + w_{\text{contact}} r_{\text{contact}} + w_{\text{lift}} r_{\text{lift}} + w_{\text{penalty}} r_{\text{penalty}}. \tag{1}$$

In the implementation, we set the lift reward as the the difference between object current height and object initial height $r_{\text{lift}} = h_{\text{current}} - h_{\text{init}}$. The action penalty reward $r_{\text{penalty}} = -||a||_2^2$. For reaching reward, it consists of the distance between object and target, the distance between finger tip and the target. The weight for the four reward terms are: $w_{\text{reach}} = 1$, $w_{\text{contact}} = 0.5$, $w_{\text{lift}} = 10$, $w_{\text{penalty}} = 0.01$. Recall that the lift reward r_{lift} is set to 0 if the contact reward r_{contact} is 0. Intuitively, it ensures that the robot can only receive the reward for lifting the object up if the robot hand is in good contact with the object. This design prevents the agent from using a large force to knock the object into the air or using unstable contact to lift the object up.

B Learning Details

RL Training. We use on-policy RL training for setting except the distilling experiments. We select PPO as the RL algorithm to train the point cloud based manipulation policy. The hyper-parameters of the PPO are shown in Table 1.

Network Architecture The RL agent use the PointNet based architecture as the visual backbone. As shown in Figure 3 of the main paper, we first concatenate the visual features from PointNet and the propriception feature from the MLP. This concatenated feature is then shared by both value

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

network and policy network to predict value and action. The details for the network architecture are shown in Table 2.

Parameter	Value
Mini-Batch Size	500
Learning Rate	3e-4
Clip Range	0.8
Horizon	200
Epoch	10
Steps per Iteration	10

Table 1: PPO Parameters.

Module	Architecture	Output Dim
Visual Feature Extractor	PointNet Local Channel: (64, 128, 256)	256
	PointNet Global Channel: (256,)	256
State Feature Extractor	MLP: (64, 64)	64
Actor	MLP: (64, 64)	64
Critic	MLP: (64, 64)	64

Table 2: Network Architecture.

References

[1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[2] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[3] K. Mamou and F. Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *2009 16th IEEE international conference on image processing (ICIP)*, pages 3501–3504. IEEE, 2009.