6 Appendix

6.1 Hindsight Goal Auxiliary Training Algorithm

Algorithm 1 describes our training procedure with hindsight goals. The loss functions for the critic and the actor with hindsight labels are

$$\begin{aligned} L_\phi &= \frac{1}{2}(Q_\phi(s,a) - y)^2 + L_{\text{AUX}}(\hat{g}, g_\phi) \cdot r_T \\ L_\theta &= \lambda L_{\text{BC}}(a^*, a_\theta) + (1-\lambda) L_{\text{DDPG}}(s, a_\theta) + L_{\text{AUX}}(\hat{g}, g_\phi) \cdot r_T, \end{aligned} \tag{5}$$

where $r_T \in \{0, 1\}$ is the binary reward for the last state in the episode. Therefore, we only have the goal auxiliary losses for successful episodes during fine-tuning on unknown objects. We fine-tune the pretrained actor-critic network on the dataset $\mathcal{D} = \mathcal{D}_{\text{expert}} \cup \mathcal{D}_{\text{hindsight}}$. Note that we cannot collect extra data for $\mathcal{D}_{\text{dagger}}$ and $\mathcal{D}_{\text{ddpg}}$ with unknown objects, but we can potentially use the ones that are already collected.

Compared with the common relabeling strategy for goal-conditioned policies [30, 31, 32], Algorithm 1 does not modify the reward of an episode. While using every hindsight goal might allow the robot to explore diverse goals, it hurts grasping performance if we use unsuccessful grasp poses as goals. Although goal-conditioned policies are useful for explorations and long-horizon tasks, we do not treat grasping an arbitrary object as a simple goal-reaching task since solving the grasp detection problem is a challenge. The policy will ignore contacts during grasping and will also be affected by the grasp quality at test time.

Algorithm 1: Hindsight Goal Auxiliary Training (policy π_θ, dataset $\mathcal{D}_{\text{hindsight}}$, critic Q_ϕ)

```
for $i = 0, ..., N$ do
    Execute $\pi_\theta$ in environment for an episode $\tau$
    Log trajectory $\tau = (s_0, a_0, r_0 ..., s_T, a_T, r_T)$
    Compute $\hat{g}_t$ from $\tau$ and augment goals to the dataset $\mathcal{D}_{\text{hindsight}}$
    Optimize the $Q_\phi, \pi_\theta$ with Eq. (5)
end
```

6.2 Task and Environment Details

To better utilize the 3D information and resolve ambiguities of the current view due to local geometry for grasping static objects, we aggregate point clouds as the gripper moves in time for our state representation. Fig. 6 illustrates our point aggregation procedure.

In order to aggregate 3D points from different time steps, we need to transform them into a common coordinate system. We choose the coordinate of the robot base to aggregate the 3D points. Specifically, at time t, let $X_t^C \in \mathbb{R}^{n_t \times 3}$ denote the 3D point cloud of the target object from the wrist camera, where n_t represents the number of points. Since the camera is attached to the robot hand, we first transform X_t^C to the end-effector frame denoted as X_t using the camera extrinsics. Through forward kinematics, we can compute the transformation of the end-effector in the robot base as $\mathcal{T}_t \in \mathbb{SE}(3)$. Then we can transform the point cloud to the robot base by $\mathcal{T}_t^{-1}(X_t)$. Finally, our state representation at time t is $s_t = \mathcal{T}_t\big(\cup_{i=0}^{t} \mathcal{T}_i^{-1}(X_i)\big)$, i.e., aggregating the point clouds in the robot base up to time t and then transforming the points to the end-effector frame at time t. Notably, using 3D points suffers less from the sim-to-real gap and has a much better sample efficiency than image-based methods, in Fig. **??**.

At time step t, we sample m_t points among the n_t observed points uniformly to limit the number of points. Note that a closer view of the object leads to a larger foreground mask. Therefore, n_t increases as t goes from 0 to T. In order to balance points sampled from different time steps, we aggregate observed point clouds in an exponential decaying schedule. Specifically, $m_t = \lceil \alpha^t n_t \rceil$, where $\alpha = 0.95$. The final network input contains 1024 sampled points from the state s_t.

The expert planner is tuned to achieve around 95% success rates on both YCB and ShapeNet objects. Compared to the YCB dataset which composes of objects that are manipulated in daily life, some

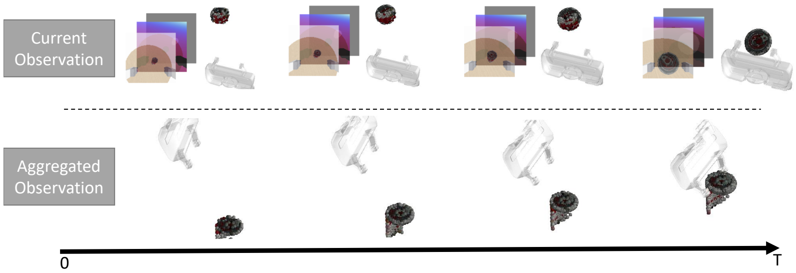

Figure 6: Illustration of our point aggregation procedure. We show the RGB, depth, foreground mask and point cloud at time t, and the aggregated point cloud up to time t.

objects in the ShapeNet repository such as Shelves, Desks, and Beds do not resemble real-world tabletop objects. Therefore we resize them to be graspable by the Panda gripper and use the same object physics (friction and mass) for all the objects [7, 48]. We randomize initial states ρ_0 from a uniform pose distribution facing toward the target object, in a distance from 0.2 meter to 0.5 meter. Compared to the restricted top-down grasping, the diversity of the initial arm configurations and object shapes and poses forces the policy to learn grasping in the 6D space. When images are used, we apply domain randomization during training by randomly changing the textures.

We have also tried using images from an overhead static camera as input and joint positions/velocities as the actions. For 6D grasping, we believe that using the ego motion from a wrist camera generalizes better to unseen objects in different backgrounds and suffers less from occlusions during grasping. Moreover, the point cloud representation is suitable for sim-to-real transfer since it has no reliance on color information and camera parameters, and often shows robustness to depth noises from sensors. The aggregated point cloud also allows action space to align with the observation space and uses history observations when the current sensory feedback is limited.

6.3 Network Architecture

The feature extractor backbone for point clouds is PointNet++ [40] with a model size around 10M. In our ablation studies, the backbone network for images is ResNet18 [45] with a much larger model size of 45M. The remaining timestep $(T-t)$ is concatenated with the state feature to provide information about the horizon. The actor network and the critic network are both implemented by a multi-layer perceptron with three fully-connected layers using the ReLU activation function. We use Adam [49] as the optimizer for training and decrease the learning rate in a piece-wise step schedule. We use separate feature extractors for the actor and the critic. For fine-tuning with hindsight goals, we fix the weights of the feature extractor backbones, and only fine-tune the actor-critic.

6.4 Training Details

The dataset $\mathcal{D}_{\text{expert}}$ contains 50,000 interaction steps in successful episodes from expert demonstrations and extra 120,000 failure data points in the case of offline RL in our ablation studies. These data contain a few rollouts for each object to resemble the real-world setting where the demonstration data has a fixed size. Our online setting has around 3 million interaction steps. The replay buffer contains $2 \cdot 10^6$ transitions for points and $2 \cdot 10^5$ transitions for images due to the memory limit.

During each iteration of the DDPG training, we sample 14 parallel rollouts of the actor to collect experiences. We then perform 20 optimization steps on a mini-batch of size 250 which has 70% of the data from the on-policy buffer and the rest from expert buffer. We add decaying uniform noises from 3 centimeters, 8 degrees to 0.5 centimeter, 1 degree for the exploration on translation and rotation, respectively. During expert rollouts, we randomly apply motion perturbations to increase the diverse state space explorations. The balancing ratio of the BC loss and the DDPG loss is $\lambda = 0.2$ and the MDP discount factor is $\gamma = 0.95$. We adopt the TD3 algorithm [41], and the policy network is updated once every two Q-function updates. The target network of the actor and the first target network of the critic are updated every iteration with a Polyak averaging of 0.999. The second target network used in Q-learning uses $3,000$ update step delay [4]. In the point matching loss function for gripper poses $L_{\text{POSE}}(\mathcal{T}_1, \mathcal{T}_2)$ in Eq. (2), X_g contains 6 keypoints on the finger tips, finger bases, and the palm base of the gripper. Although action and grasp predictions use different

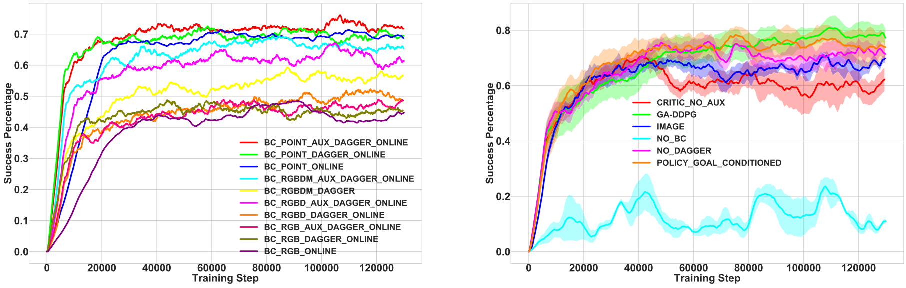

Figure 7: Left) Learning curves of different BC models. Right) Learning curves of different RL models. The mean and the 1 standard deviation range are plotted for each model.

Method	Grasp-Translation (m)				Grasp-Rotation (°)				Grasp-Point-Distance (m)			
	ShapeNet		YCB		ShapeNet		YCB		ShapeNet		YCB	
	Min	Mean	Min	Mean	Min	Mean	Min	Mean	Min	Mean	Min	Mean
Goal-Auxiliary	0.015	0.045	0.020	0.047	6.621	17.989	7.610	19.465	0.023	0.065	0.028	0.069
Goal-Conditioned	0.016	0.039	0.025	0.045	6.987	17.638	7.821	19.864	0.025	0.056	0.033	0.071

Table 5: Evaluation on grasp prediction performance

rotation representations, the same loss function is used to compute L_{BC} and L_{AUX}. Fig. 7 shows the learning curves of several BC models and RL models.

6.5 Grasp Prediction Performance

Since our method involves grasping goal prediction, we evaluate the grasp detection quantitatively and qualitatively. In this experiment, we train a policy network using BC on the offline dataset and use point clouds as our input.

Table 5 shows the grasp prediction results. We use the target goals from the expert planner as the ground truth, and compute translation errors in meters, rotation errors in degrees and the point distances in meters as used in the loss function in Eq. (2). We compute both the mean error and the minimum error along the trajectories from the expert.

In Table 5, "Goal-Auxiliary" indicates our policy network with auxiliary goal prediction using the aggregated point cloud representation. "Goal-Conditioned" indicates a separate network trained for grasp prediction only, which is used for goal-conditioned models. We can see that the performance of the two networks are quite similar, which demonstrates that our training is able to optimize the auxiliary task well and the grasp prediction objective aligns well with the policy learning. Overall, jointly training the policy and grasp prediction does not affect the grasp prediction performance. We also observe that the grasp prediction error decreases as the camera gets closer to the object in our closed-loop setting. Note that the regression errors are affected by the discrete nature of the pre-defined grasp set and the multi-modality, especially for certain symmetric objects such as bowls. Fig. 8 illustrates some grasp prediction examples on ShapeNet objects.

6.6 Closed-Loop 6D Grasping of Moving Objects in Simulation

In this experiment, we move the target in simulation by continually resetting its pose in the first 12 environment steps. We perturb the object pose on a table with uniform translation noises (3 centimeters maximum) and uniform z-axis rotation noises (15 degrees maximum) per step. Our experiment shows that our fine-tuned policy can still achieve performance at 90.6% success rate in these dynamic scenarios. The policy is robust even if we do not aggregate points for moving objects.

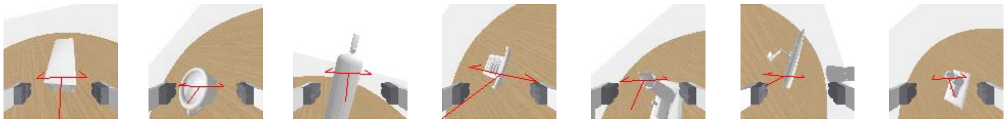

Figure 8: Grasp prediction examples on several ShapeNet objects. The red fork denotes the gripper pose predicted from our policy network.

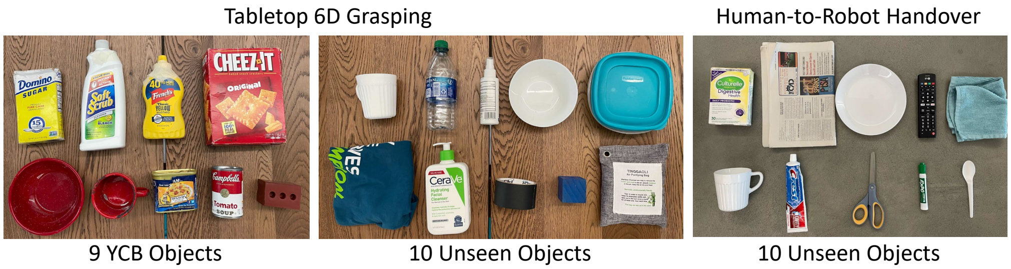

Figure 9: Objects used in our real-world experiments.

	Fixed initial joint			Varying initial joint		
Object	Policy	Planning	Combined	Policy	Planning	Combined
Cracker Box	3/5	3/5	5/5	3/5	3/5	3/5
Sugar Box	5/5	5/5	5/5	5/5	3/5	4/5
Tomato Soup Can	2/5	2/5	2/5	3/5	3/5	3/5
Mustard Bottle	4/5	5/5	5/5	3/5	2/5	4/5
Meat Can	5/5	5/5	3/5	3/5	5/5	3/5
Bleach Cleanser	4/5	4/5	4/5	5/5	3/5	4/5
Bowl	4/5	5/5	5/5	4/5	2/5	4/5
Mug	3/5	5/5	3/5	1/5	3/5	4/5
Foam Brick	5/5	4/5	5/5	4/5	2/5	5/5
Unseen	8/10	5/10	8/10	7/10	7/10	7/10
All	43/55	43/55	45/55	38/55	33/55	41/55
Success Rate (%)	78.2	78.2	**81.8**	69.1	60.0	**74.5**

Table 6: Detailed results of the real-world tabletop grasping for single objects on a table. The numbers of successful grasps among trials and the overall success rates are presented.

6.7 Details on Real-world Experiments

Fig 9 shows the objects used in our tabletop grasping and handover experiments. Table 6 presents the details of our tabletop experiments for 9 YCB objects and 10 unseen objects in the real world. From the table, we can see that the policy cannot handle "Cracker Box", "Tomato Coup Can" and "Mug" very well due to the incorrect contact modeling in the PyBullet simulator.

Table 7 shows the detailed handover experiments for the 10 objects in our system evaluation, where we handed each object five times for three different locations and two moving scenarios. Fig 10 lists all the handovers of the 10 objects in our user study for six participants.

6.8 Design Limitations

The underlying assumption for hindsight goals in our method is that the object pose does not change dramatically due to robot interaction, which we find quite applicable for precision grasping. It also requires minimal camera drifts that often holds with industry-grade robot manipulators. We also assume the target is specified with a separate image segmentation module that can lead to delays. While using end-effector delta pose as action makes the policy learning easier by removing some of

	Left	Middle	Right	Left→Right	Right→Left	Success Rate (%)
Medicine Box	1	1	1	1	0	80
Newspaper	1	1	1	1	0	80
Plate	1	1	1	0	1	80
Mug	0	1	1	1	1	80
Remote	1	1	1	0	0	60
Toothpaste	1	1	1	1	0	80
Scissors	1	1	1	0	1	80
Towel	1	1	1	0	0	60
Pen	1	1	1	1	0	80
Spoon	1	1	1	1	1	1
Success Rate (%)	90	100	100	60	40	78

Table 7: Handover success rate with varied handover locations and moving objects.

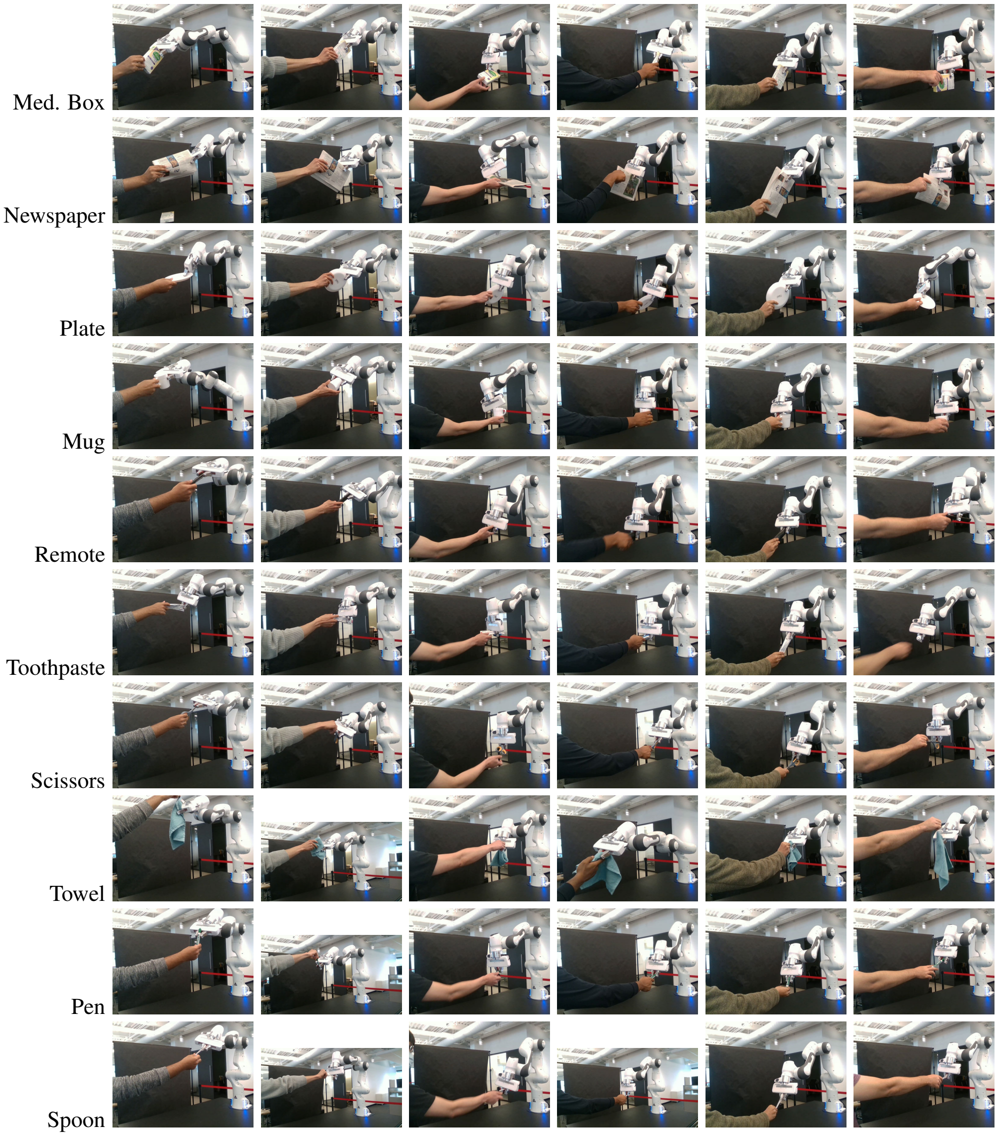

Figure 10: 10 Household objects used in the user study with 6 participants. Our approach is able to adapt to various ways of holding objects. Please zoom in for the best view.

Figure 11: The policy needs to learn complex contacts between the robot gripper with different objects in simulation. So it suffers from the domain transfer from simulation to the real world.

the complexity of the configuration space, it is less suitable for high-frequency control domains with configuration-space constraints such as joint limits and hard collision instances. In the simulator, our value function in the actor-critic can be used to improve the policy in situations where experts rarely visit, for instance, contact-rich scenarios (Fig. 11) or environments with different physics parameters. However, when deploying the learned policy to the real world, we still observe the sim-to-real domain gap in contact modeling.