

APPENDIX

A. Environment Modeling Details

Modeling underactuated joints. Since modeling underactuated joint structure is not directly supported, we approximate the relationship between each pair of actuated and underactuated joints by fitting a linear function $q_u = k \cdot q_a + b$, where q_u is the underactuated joint angle and q_a is the actuated joint angle. Note that parameters k, b are included as tunable parameters to search over using our autotune module detailed in Section III-A.

B. Reward Design Details

We design generalizable rewards based on the principle outlined in Section III-B and list task reward details below.

Both **grasp** and **lift** tasks can be defined with the following goal states: (1) finger contact with the object; (2) the object being lifted up to a goal position. Our reward design can, therefore, follow by combining the contact goal reward and the object goal reward terms:

$$r(s_h, s_o) = r_{\text{contact}}(s_h, s_o) + r_{\text{goal}}(s_o) \quad (3)$$

where s_h includes fingertip positions, s_o includes object center-of-mass position, and all contact marker positions (if any).

Similarly, the **handover** task can be defined with the following goal states: (1) one hand’s finger contact with the object; (2) object being transferred to an intermediate goal position while still in contact with the first hand; (3) the second hand’s finger contact with the object; (4) object being transferred to the final goal position. Due to the hand switching, we introduce a stage variable $a \in \{0, 1\}$ and design the reward as follows:

$$r(s_h, s_o) = (1 - a) \cdot (r_{\text{contact}}(s_{h_A}, s_{o_A}) + r_{\text{goal}}(s_{o_A})) + a \cdot (r_{\text{contact}}(s_{h_B}, s_{o_B}) + r_{\text{goal}}(s_{o_B})) \quad (4)$$

where s_{h_A}, s_{h_B} denote fingertip positions of the engaged hand at each stage, s_o denote object center-of-mass position and desirable contact marker positions (if any) at each stage. At completion of each stage, we also reward policy with a bonus whose scale increases as stage progresses.

C. Policy Training Details

RL implementation. To learn the specialist policies, the observation space includes object position and robot joint position at each time step, and the action space is robot joint angles. We use Proximal Policy Optimization [47] with asymmetric actor-critic as the RL algorithm. In addition to the policy inputs, we provide the following privilege state inputs to the asymmetric critic: arm joint velocities, hand joint velocities, all fingertip positions, object orientation, object velocity, object angular velocity, object mass randomization scale, object friction randomization scale, and object shape randomization scale. Both the actor and critic networks are 3-layer MLPs with units (512, 512, 512).

Domain randomization. Physical randomization includes the randomization of object friction, mass, and scale. We also

TABLE IV: Domain Randomization Setup.

Object: Mass (kg)	[0.03, 0.1]
Object: Friction	[0.5, 1.5]
Object: Shape	$\times \mathcal{U}(0.95, 1.05)$
Object: Initial Position (cm)	$+\mathcal{U}(-0.02, 0.02)$
Object: Initial z-orientation	$+\mathcal{U}(-0.75, 0.75)$
Hand: Friction	[0.5, 1.5]
PD Controller: P Gain	$\times \mathcal{U}(0.8, 1.1)$
PD Controller: D Gain	$\times \mathcal{U}(0.7, 1.2)$
Random Force: Scale	2.0
Random Force: Probability	0.2
Random Force: Decay Coeff. and Interval	0.99 every 0.1s
Object Pos Observation: Noise	0.02
Joint Observation Noise.	$+\mathcal{N}(0, 0.4)$
Action Noise.	$+\mathcal{N}(0, 0.1)$
Frame Lag Probability	0.1
Action Lag Probability	0.1
Depth: Camera Pos Noise (cm)	0.005
Depth: Camera Rot Noise (deg)	5.0
Depth: Camera Field-of-View (deg)	5.0

apply random forces to the object to simulate the physical effects that are not implemented by the simulator. Non-physical randomization models the noise in observation (e.g., joint position measurement and detected object positions) and action. A summary of our randomization attributes and parameters is shown in Table IV.

D. Distillation Details

To learn the generalist policy, we reduce the choices of observation inputs to the robot joint states and selective object states, including 3D object position and egocentric depth view, since privileged information is unavailable for sim-to-real transfer. To more efficiently utilize the trajectory data and improve training stability, for each sub-task specialist policy, we evaluate for 5000 steps over 100 environments, saving trajectories filtered by success at episode reset on the hard disk. We then treat the saved data as “demonstrations” and learn a generalist policy for each task with Diffusion Policies [11].

The proprioception and object position states are concatenated and passed through a three-layer network with ELU activation, hidden sizes of (512, 512, 512), and an output feature size of 64. For depth observations, we use the ResNet-18 architecture [18] and replace all the BatchNorm [22] in the network with GroupNorm [59], following [11]. All the encoded features are then concatenated as the input to a diffusion model. We use the same noise schedule (square cosine schedule) and the same number of diffusion steps (100) for training as in [11]. The diffusion output from the model is the normalized 7 DoF absolute desired joint positions of each humanoid arm and the 6 DoF normalized (0 to 1) desired joint positions of each humanoid hand. We use the AdamW optimizer [25, 34] with a learning rate of 0.0001, weight decay of 0.00001, and a batch size of 128. Following [11], we maintain an exponential weighted average of the model weights and use it during evaluation/deployment.