A Data Augmentation Loop

A.1 Pseudo Code

Algorithm 1 DATA AUGMENTATION LOOP

```
Require: Human data $D = (G, a^W)$, training set $D_t \in D$, high-level planner $\pi^H$, low-level
    controller $\pi^L$, augmentation iteration $L$, data augment function $L_{aug}()$, wrist pose trajec-
    tories $G_t = (\boldsymbol{g}_t, \boldsymbol{g}_{t+1}, ..., \boldsymbol{g}_{t+T-1})$, goal trajectory of the object and its geometric features
    $\boldsymbol{a}_t^W = (\boldsymbol{a}_t^W, \boldsymbol{a}_{t+1}^W, ..., \boldsymbol{a}_{t+T-1}^W)$.
 1: Initialize $\pi^H$, $\pi^L$, $D_t = \{\}$.
 2: for iteration $m = 0,1,...,L$ do
 3:     while until convergence of $\pi^H$ do
 4:         Generate augmented data $L_{aug}(D)$
 5:         Append into training set $D_t \leftarrow D_t + L_{aug}(D)$
 6:         Train $\pi^H$ on $D_t$
 7:     end while
 8:     while until convergence of $\pi^L$ do
 9:         Train $\pi^L$ on $(\pi^H(G), G)$
10:     end while
11:     Rollout success trajectories $D_s^t = (\boldsymbol{a}_t^W, G_t)$ with $\pi^L$
12:     Append into human data $D \leftarrow D_T + D_s^t$
13: end for
```

A.2 Detail of the Data Augmentation Loop

Below are the details for each augmentation. The unit of length is centimeters and the unit of angle is degrees.

- Random the object's mesh scales with a small scale:
  - The scale of the width of the manipulated object ranges from 0.9 to 1.1.
  - The scale of the length of the manipulated object ranges from 0.9 to 1.1.
  - The scale of the height of the manipulated object ranges from 0.9 to 1.1.
- Random the object's initial pose with a small scale:
  - The x-coordinate of the manipulated object ranges from -0.02 to 0.02.
  - The y-coordinate of the manipulated object ranges from -0.02 to 0.02.
  - The manipulated object's z-axis Euler degree ranges from 0 to 30.
- Modify the goal trajectories of the object with waypoint interpolation:
  - The x-coordinate of the goal trajectories position is added by ranges from -0.02 to 0.02.
  - The y-coordinate of the goal trajectories position is added by ranges from -0.02 to 0.02.
  - The z-coordinate of the goal trajectories position is added by ranges from -0.02 to 0.02.

B Detail Implementation of RL in Simulation

B.1 Observation Space

Table.5 gives the specific information of the observation space.

B.2 Reward Design

Denote the $\hat{\boldsymbol{g}}_i^R$, $\hat{\boldsymbol{g}}_i^T$ and $\hat{g}_i^J$ is the current 3D translation, 3D rotation and joint angle of the object respectively, the desired object 3D rotation $\boldsymbol{g}_i^R$, the desired object 3D translation $\boldsymbol{g}_i^T$, and the desired

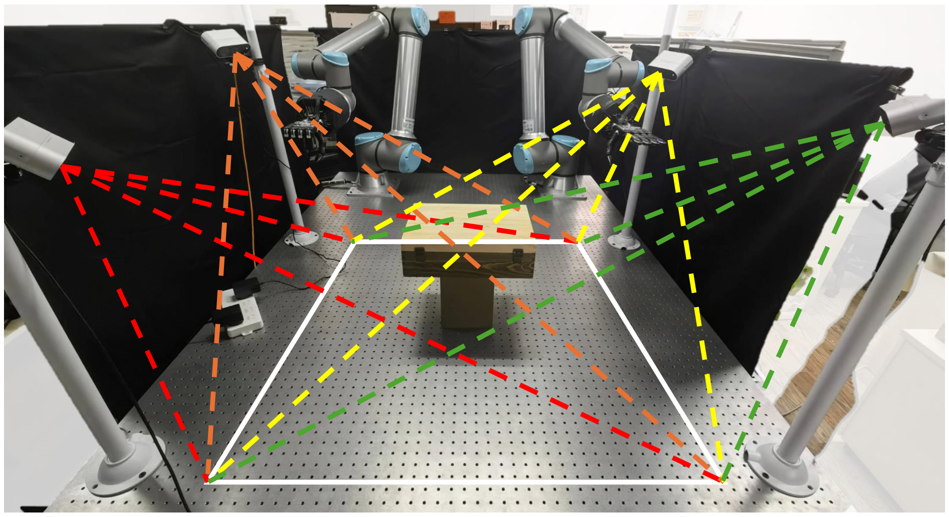

Figure 2: Setup of the cameras.

Index	Description
0 - 60	right arm-hand dof position, velocity
60 - 120	left arm-hand dof position, velocity
120 - 133	right hand end-effector position, velocity, linear velocity, angle velocity
133 - 146	left hand end-effector position, velocity, linear velocity, angle velocity
146 - 159	object base position, rotation, linear velocity, angle velocity
159 - 172	articulated object top part position, rotation, linear velocity, angle velocity
172 - 185	articulated object bottom part position, rotation, linear velocity, angle velocity
185 - 187	object dof position, velocity
187 - 257	desired object motion trajectory $G=(\boldsymbol{g}_t,\boldsymbol{g}_{t+1},...,\boldsymbol{g}_{t+T})$
257 - 397	sequence of 6-DoF wrist actions $(\boldsymbol{a}_t^W,\boldsymbol{a}_{t+1}^W,...,\boldsymbol{a}_{t+T}^W)$ generated by high-level planner
397 - 462	right hand fingertip pose, linear velocity, angle velocity
462 - 527	left hand fingertip pose, linear velocity, angle velocity

Table 5: Observation space of our framework in simulation.

object joint angle g_i^J. λ_1, λ_2 and λ_3 is the hyperparameters to balance the weight of each component of the reward.

The reward function is defined as:

$$r_t = \exp^{-(\lambda_1 * \|\boldsymbol{g}_t^R - \hat{\boldsymbol{g}}_t^R\|_2 + \lambda_2 * \|\boldsymbol{g}_t^T - \hat{\boldsymbol{g}}_t^T\|_2 + \lambda_3 * \|g_t^J - \hat{g}_t^J\|_2)} \tag{1}$$

where $\lambda_1 = 20$, $\lambda_2 = 1$, and $\lambda_3 = 5$.

We use an exponential map in the reward function, which is an effective reward shaping technique used in the case to minimize the distance, introduced by [64, 65]. To improve the calculation efficiency, we use quaternion to represent the object orientation. The angular position difference is then computed through the dot product between the normalized goal quaternion and the current object's quaternion.

C Detail Implementation in Real-World

C.1 Perception

Our perception setup is shown in Figure 2. We arranged 4 identical Femto Bolt cameras around the table and face towards the object. We use FoundationPose [66] to estimate the articulated object pose. To remove the abnormal results, we compare each pose to the desired pose and remove the pose if the error is smaller than a threshold (5 centimeters in translation and 0.5 radians in orientation). Finally, we average the rest of the poses as our observation for the policy. If none of the poses is smaller than the threshold, we continue to use the pose from the previous frame.

C.2 Policy Distillation

We use the DAgger [67] algorithm for policy distillation. Table.6 gives the specific information of the observation space of the distilled policy.

Index	Description
0 - 24	right hand dof position
24 - 48	left hand dof position
48 - 55	right hand end-effector position, rotation
55 - 62	left hand end-effector position, rotation
62 - 69	articulated object top part position, rotation
69 - 76	articulated object bottom part position, rotation
76 - 77	object dof position
77 - 147	desired object motion trajectory $G = (\boldsymbol{g}_t, \boldsymbol{g}_{t+1}, ..., \boldsymbol{g}_{t+T})$
147 - 287	sequence of 6-DoF wrist actions $(\boldsymbol{a}_t^W, \boldsymbol{a}_{t+1}^W, ..., \boldsymbol{a}_{t+T}^W)$ generated by high-level planner

Table 6: Observation space of our framework in the real-world.

C.3 Resets

In the real-world evaluation, we used a trajectory from the ARCTIC dataset as our goal trajectory. In terms of resets in our real-world experiments, we pose the object within 3 centimeters and 0.5 radians of the initial pose of the goal trajectory during resets. We evaluated 20 times and reported the completion rate.

C.4 Evaluation

When evaluating real-world experiments, we take the same metric (completion rate) as in simulation. It will fail if the tolerance is exceeded(5 centimeters in translation, 2.5 centimeters in object's longest dimension multiplied by rotation angle, and 0.5 radians in joint angle), and record the completion rate.

D Hyperparameters of the PPO

Table.7 gives the hyperparameters of the PPO.

Hyperparameters	Value
Num mini-batches	4
Num opt-epochs	5
Num episode-length	8
Hidden size	[1024, 1024, 512, 256]
Clip range	0.2
Max grad norm	1
Learning rate	3.e-4
Discount (γ)	0.998
GAE lambda (λ)	0.95
Init noise std	0.8
Desired kl	0.02
Ent-coef	0

Table 7: Hyperparameters of PPO.

E Domain Randomization

Isaac Gym provides lots of domain randomization functions for RL training. We add the randomization for all the tasks as shown in Table. 8 for each environment. we generate new randomization every 1000 simulation steps.

Parameter	Type	Distribution	Initial Range
Robot			
Mass	Scaling	uniform	[0.5, 1.5]
Friction	Scaling	uniform	[0.7, 1.3]
Joint Lower Limit	Scaling	loguniform	[0.0, 0.01]
Joint Upper Limit	Scaling	loguniform	[0.0, 0.01]
Joint Stiffness	Scaling	loguniform	[0.0, 0.01]
Joint Damping	Scaling	loguniform	[0.0, 0.01]
Object			
Mass	Scaling	uniform	[0.5, 1.5]
Friction	Scaling	uniform	[0.5, 1.5]
Scale	Scaling	uniform	[0.95, 1.05]
Position Noise	Additive	gaussian	[0.0, 0.02]
Rotation Noise	Additive	gaussian	[0.0, 0.2]
Observation			
Obs Correlated. Noise	Additive	gaussian	[0.0, 0.001]
Obs Uncorrelated. Noise	Additive	gaussian	[0.0, 0.002]
Action			
Action Correlated Noise	Additive	gaussian	[0.0, 0.015]
Action Uncorrelated Noise	Additive	gaussian	[0.0, 0.05]
Environment			
Gravity	Additive	normal	[0, 0.4]

Table 8: Domain randomization of all the tasks.

F Goal Representation

Our framework requires the full 6D pose of the object trajectory at each time-step. For downstream tasks, we are able to get the trajectories in many ways. For example, we can specify a few key poses of the object based on the task, and use linear interpolation to generate the pose trajectories between the key poses. Another solution is to use some object motion synthesis methods in the field of graphics to generate the 6D pose of the object trajectory, such as [68]. We design an additional experiment to prove that our method can work with this setup. Taking a task "lifting up the box in the air and dropping it down" as an example, we can just simply manually define the pose of the box in the air and the landing point after picking it up as the key poses, then interpolate it as our goal trajectory, and train the robot to manipulate the object to follow the goal trajectory. The results are shown in Table 9 and the snapshot are shown in Figure 3. The results show that we can complete the task under this setting, and shows that our framework is extensible.

	Box	Manually Designed Trajectory
Our	$100_{\pm 0.0}$	$100_{\pm 0.0}$

Table 9: Results for the goal representation experiments.

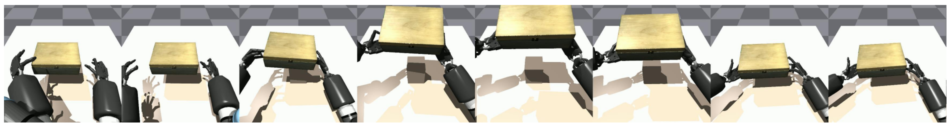

Figure 3: Snapshots of the goal representation experiment.

G Time-indexed of the Goal Trajectory

For the time-indexed of the goal trajectory, we randomize the sampling of input trajectories with various time gaps during training. We added an additional experiment to show that varying the time gap does not significantly affect our performance, indicating that our approach maintains generalization despite these temporal constraints. Using Coffee Maker as an example, we interpolated the goal trajectories provided by ARCTIC datasets to varying levels, and tested the completion rate of our policy on it. Origin represents the original goal trajectory from ARCTIC, and Skip 2 and Skip 1 represent skipping 2/1 poses between every two poses on the basis of origin goal trajectory. Inter 2 and Inter 1 represent insertion of 2/1 pose between every two poses according to the linear interpolation method on the basis of origin goal trajectory. Our policy generates robot actions conditioned on a sequence of object goal poses, with the time gaps defined by the distance between each consecutive goal pose. In this experiment, we randomly use time gaps of 0 to 3 units when sampling the object trajectory to test whether the policy is sensitive to these time gaps (Random Sample). The results is shown in Table 10, the performance of the trained policy will not vary greatly.

	Skip 2	Skip 1	Origin	Inter 1	Inter 2	Random Sample
Our	85.7$_{\pm 7.5}$	86.5$_{\pm 3.4}$	86.1$_{\pm 5.5}$	85.4$_{\pm 8.1}$	87.2$_{\pm 4.8}$	86.2$_{\pm 6.2}$

Table 10: Results for the time-indexed experiments.

H Model-base Baselines

We add a baseline of the model-based method in the simulation, the result is shown in Table 11. We use the sample-based model predictive control method that is modified from [69] as our low-level controller in our tasks. Our method outperforms the sample-based model predictive control by 46.3% on average.

	Box	Coffee Maker	Espresso Machine	Ketch	Micro-wave	Mixer	Note-book	Scis-sors	Laptop
MPC	31.6±1.5	22.8±0.7	30.4±2.3	24.8±1.3	28.6±1.9	26.8±0.5	19.2±2.0	23.6±0.8	24.0±1.4
Ours	**100**±0.0	**86.1**±5.5	**81.1**±8.6	**41.2**±13.3	**100**±0.0	**57.6**±4.9	**38.7**±3.3	**41.4**±14.9	**100**±0.0

Table 11: Results for the experiments of using one policy per object.