Supplementary Material

Table of Contents

A Technical Appendices and Supplementary Material

Overview The Appendix contains the following content:

1. **Base Policy Implementation Details** (*Section A.1*): Details the implementation of the base policy, including model inputs and outputs, and training hyperparameters.
2. **Residual Policy Implement Details** (*Section A.2*): Describes residual policy implement details and the reward function design for the residual policy across different tasks.
3. **Data Collection and Environment Setup** (*Section A.3*): Outlines the data generation strategy incorporating environment, object, and spatial variations.
4. **Evaluation Test Set and Success Rate Calculation Method** (*Section A.4*): Presents the evaluation test set and the success rate calculation method.

A.1 Base Policy Implementation Details

This section details our base policy implementation, including model inputs and outputs, training hyperparameters and computing resources.

Model Inputs and Outputs. The base policy input state is denoted as $s_t = \{s_t^{\text{vis}}, s_t^{\text{obj}}, s_t^{\text{prop}}\}$, where:

Visual Input s_t^{vis}: For single-arm tasks, the input is a front view image $I_t^f \in \mathbb{R}^{224\times224\times3}$. For dual-arm tasks, the input is a top view image $I_t^t \in \mathbb{R}^{224\times224\times3}$.

Object State s_t^{obj}: In most tasks, the object state is represented by a 13-dimensional vector representing the state of a single manipulated object. However, for pour tasks, two objects are involved, and the object state is represented by a 26-dimensional vector.

Robot Proprioception s_t^{prop}: For single-arm tasks, the proprioception is $s_t^{\text{prop,single-arm}} \in \mathbb{R}^{69}$, including joint positions (19 dimensions), joint velocities (19 dimensions), gripper state (12 dimensions), gripper velocity (12 dimensions), end-effector position (3 dimensions), end-effector orientation (4 dimensions). For dual-arm tasks, the proprioception is $s_t^{\text{prop,dual-arm}} \in \mathbb{R}^{130}$, including joint positions (36 dimensions, 18 per arm), joint velocities (36 dimensions, 18 per arm), gripper states (11 dimensions per gripper), gripper velocities (11 dimensions per gripper), end-effector positions (3 dimensions per arm), and end-effector orientations (4 dimensions per arm).

Output. The action sequence is denoted as $d = (a_t, a_{t+1}, \ldots, a_{t+H})$ where $H = 8$. Each individual action a_t includes: An end-effector 6D pose $a_t^{\text{pose}} \in \mathbb{R}^6$. Target joint angles of hands $a_t^{\text{joint}} \in \mathbb{R}^n$, where $n = 10$ for dual-arm tasks and $n = 7$ for single-arm tasks.

Training Hyperparameters. Table 4 summarizes all hyperparameter used for the base policy training.

Computing Resources. All experiments are conducted on 8 NVIDIA a800 GPUs.

A.2 Residual Policy Implement Details

This section details our residual policy implement details, including policy training and reward design.

Policy Training. We employ the Soft Actor-Critic (SAC) algorithm [51] to train a residual policy that enhances a pre-trained diffusion-based manipulation policy. The residual approach enables efficient learning by leveraging an existing base policy while exploring additional action refinements. Detailed hyperparameters are provided in Table 5. The residual actor network is implemented as a policy decorator that outputs corrections to the base policy's actions, allowing for fine-tuning of manipulation behaviors while maintaining the fundamental skills encoded in the base policy.

To ensure effective learning, we implement a progressive exploration strategy that gradually introduces the residual policy's influence over time. For the first 1,500 timesteps, only the base policy's actions are executed. Between 1,500 and 10,000 timesteps, the probability of including residual actions increases linearly with the global step count, promoting smooth exploration of the action space. All residual actions are scaled by a factor of 0.1 to maintain stability while allowing for meaningful corrections to the base policy. The training architecture features dual soft Q-networks with target networks updated at a rate of $\tau = 0.01$ to provide stable value estimation. The entropy coefficient α is automatically tuned to maintain a target entropy based on the action space dimension, balancing exploration and exploitation. Gradient updates are performed after every 5 environment steps with an updates-to-data ratio of 0.2, resulting in a total of 1 gradient update per environment step. Gradients are clipped with a maximum norm of 10 to prevent unstable updates. The critic networks evaluate the combined actions to assess the overall quality of the agent's behavior, while the actor network operates only on the proprioceptive and object state observations to generate residual corrections. This design allows the residual policy to focus on improving specific aspects of the manipulation task without requiring complete knowledge of the base policy's inner workings. The training process continues for 1.5 million timesteps, with model checkpoints saved every 10 episodes to track progress and enable resumption of training if needed.

Table 4: Hyperparameters for Diffusion Policy Training.

Category	Parameter	Value
General	Action Steps	8
	Observation Steps	1
	Embedding Dimension	768
Network	Transformer Layers	7
	Attention Heads	8
	Attention Dropout	0.1
Vision Encoder	Model Architecture	vit_small_r26_s32_224
	Pretrained	True
	Frozen	False
Diffusion Model	Noise Scheduler	DDIMScheduler
	Train Timesteps	50
	Inference Steps	16
Training	Batch Size	256
	Epochs	200000
	Learning Rate	3.0e-4
Optimization	Weight Decay	1.0e-6
	LR Scheduler	cosine

Table 5: Hyperparameters for SAC Residual Policy Training

Category	Parameter	Value
Network Architecture	Actor Network (MLP Layers)	[256, 256, 256]
	Critic Network (MLP Layers)	[256, 256, 256]
	State Dimension	143
	Action Dimension	34
Training Parameters	Learning Rate	1.0×10^{-4}
	Discount Factor (γ)	0.97
	Tau (τ)	0.01
	Entropy Coefficient (α)	0.2
	Total Timesteps	1,500,000
	Batch Size	1024
	Updates to Data Ratio	0.2
	Learning Starts	300
	Training Frequency	5
	Policy Update Frequency	1
	Target Update Frequency	1
	Max Gradient Norm	10
Residual Strategy	Residual Scale	0.1
	Progressive Exploration	10,000
	Progressive Exploration Threshold	1,500

Reward Design. We carefully design the reward functions to guide the robotic manipulation policies through complex tasks. The reward functions for each task are as follows:

Grasp Task. The reward function for the grasping task encourages precise finger positioning and successful object lifting:

$$r_{\text{grasp}} = \exp\left(-5 \cdot \max\left(\sum_i d_i - 0.05, 0\right)\right) + 100 \cdot \max\left(0.2 - |z_{\text{target}} - z_{\text{current}}|, -0.01\right), \quad (2)$$

where d_i is the distance from the i-th finger (thumb, index, middle) to the object center, $z_{\text{target}} = z_{\text{start}} + 0.2$ is the target height, and z_{current} is the current object height.

Pour Task. The reward function for the pouring task guides the robot through grasping, lifting, and pouring:

$$r_{\text{pour}} = 5.0 \cdot \mathbb{I}(\text{task success}) + 10 \cdot (r_{\text{grasp\_dist}} + r_{\text{lift}}) + 50 \cdot (r_{\text{tilt}} + r_{\text{ball\_bowl}}), \quad (3)$$

where:

- $r_{\text{grasp\_dist}} = 0.5 \cdot \frac{\exp(-8.0 \cdot d_{\text{thumb}}) + \exp(-8.0 \cdot d_{\text{finger}})}{2}$,
- $r_{\text{lift}} = 50 \cdot \max\left(0.08 - |h_{\text{current}} - 0.08|, -0.01\right)$,
- $r_{\text{tilt}} = 0.5 \cdot (1 - \hat{z}_{\text{cup}} \cdot \hat{z}_{\text{up}})$,
- $r_{\text{ball\_bowl}} = 10 \cdot \exp\left(-5.0 \cdot \max\left(d_{\text{ball\_bowl}} - 0.02, 0\right)\right)$.

Lift Task. The reward function for the lift task encourages coordinated grasping and lifting, combining the following components:

$$r_{\text{lift}} = r_{\text{left\_grasp}} + r_{\text{right\_grasp}} + r_{\text{sync}} + r_{\text{lift\_height}} - p_\theta, \quad (4)$$

where:

- $r_{\text{sync}} = 4 \cdot \exp\left(-5 \cdot \max\left(s_{\text{sync}} - 0.2, 0\right)\right)$: Coordination reward based on the sum of average finger distances $s_{\text{sync}} = d_{\text{left}} + d_{\text{right}}$.
- $r_{\text{lift\_height}} = 10 \cdot \min\left(\max\left(\frac{\Delta z}{0.15}, 0\right), 1\right)$: Reward for lifting the object, where Δz is the change in object height (target: 0.15 m).
- $p_\theta = \min\left(5.0, \frac{\theta_{\max} - 30.0}{5.0}\right) \cdot \mathbb{I}(\theta_{\max} > 30.0)$: Penalty for excessive tilt (threshold = 30°).
- $r_{\text{left\_grasp}}$: Reward for left-hand grasping, based on the average distance d_{left} between the left fingers and the object ($\exp(-8 \cdot \max(d_{\text{left}} - 0.08, 0))$).
- $r_{\text{right\_grasp}}$: Reward for right-hand grasping, based on the average distance d_{right} between the right fingers and the object ($\exp(-8 \cdot \max(d_{\text{right}} - 0.08, 0))$).

Handover Task. The reward function for the handover task guides the robot through grasping, lifting, and handover, combining multiple components:

$$\begin{aligned} r_{\text{handover}} = & \alpha_1 \cdot r_{\text{right\_close}} + \alpha_2 \cdot r_{\text{right\_close\_avg}} \\ & + \gamma_1 \cdot \left(\alpha_3 \cdot r_{\text{left\_close}} + \alpha_4 \cdot r_{\text{left\_h}} + \alpha_5 \cdot r_{\text{left\_y}} - \alpha_6 \cdot r_{\text{penalty}}\right) \\ & + \gamma_2 \cdot \left(\alpha_7 \cdot r_{\text{right\_loose}} + \alpha_8 \cdot r_{\text{right\_hand\_open}}\right) \\ & + \gamma_3 \cdot r_{\text{lift}} + \gamma_4 \cdot r_{\text{ori}}, \end{aligned} \quad (5)$$

where:

- $r_{\text{right\_close}}$: Reward for right-hand closing near the object ($-\exp(10 \cdot \max(d_{\text{right}} - 0.08, 0)) + 2$).
- $r_{\text{left\_close}}$: Reward for left-hand closing near the object ($-\exp(6 \cdot \max(d_{\text{left}} - 0.01, 0)) + 2$).
- r_{lift}: Reward for lifting the object ($\min(\max(\frac{h_{\text{current}} - h_{\text{start}}}{0.2}, 0), 1)$).
- r_{ori}: Reward for maintaining the initial object orientation ($-\exp(\|\mathbf{q}_{\text{current}} - \mathbf{q}_{\text{start}}\|_2) + 2$).
- Scaling factors: $\gamma_1 = 3$, $\gamma_2 = 10$, $\gamma_3 = 3$, $\gamma_4 = 3$.

Computing Resources. All experiments in residual policy training are conducted on a single NVIDIA RTX 4090 GPU.

A.3 Data Collection and Environment Setup

This section details our progressive and controlled data collection strategy for generating diverse simulation scenarios. The strategy is structured as follows:

Progressive Data Collection Strategy. We employ a systematic approach to cover environment variations, object variations, and spatial variations in our simulation:

- **Environment Variations**: We randomly sample environments from the available set to introduce diversity in background settings.
- **Object Variations**: We adopt a curriculum-based approach, starting with geometrically similar objects and gradually introducing more challenging ones to ensure a smooth learning curve.
- **Spatial Variations**: We begin generating spatial configurations near the source demonstration scene and progressively extend them to more distant configurations within the manipulation workspace.

Each iteration of the data generation process covers a broader range of variants and presents a higher difficulty level compared to the previous one.

Scenario Sampling Strategy. To ensure comprehensive coverage of generalization factors (i.e., different objects, environments, spatial configurations), we design a scenario sampler. The sampler randomly samples scenarios from the entire set while guaranteeing that all factors are represented. For example, in the second iteration of the pouring task, we sample from 1440 scenarios, and the sampler selects 125 scenarios that cover 12 objects, 12 environments, and 10 spatial configurations. The scenarios are centered around objects, with different environments and spatial combinations.

Trajectory Collection Strategy. We set the number of iterations to $i = \{1, 2, 3\}$. For each task, we generate 20, 100, and 500 trajectories in the three iterations, respectively. Each scenario is used to collect 4 trajectories. We employ a finite mode for data collection, where in each scenario configuration, we set the Try Time to 10 attempts and the Success Threshold to 4 successful trajectories. If the try time exceeds 10 and the number of successful trajectories is less than 4, the scenario is flagged as failed, and we move to the next scenario. After collection, we downsample to the target number of trajectories.

The specific configuration information and the implementation of the scenario sampler can be found in the code.

A.4 Evaluation Test Set and Success Rate Calculation Method

Our evaluation test set consists of two categories for each task:

- T_O^i: The object generalization test set for each round i. This set is designed to evaluate the model's performance on specific objects in a given round.
- T_{OEP}: The comprehensive test set for each task, which includes all objects from the T_O^i sets across all rounds. The specific environments and spatial configurations for T_{OEP} are detailed in the supplementary material (see the code provided).

For the T_O^i test sets, we provide visualizations for each task to illustrate the object generalization scenarios. Below are the figures for each task: Figure 6, Figure 7, Figure 8, and Figure 9.

Grasp Task. The success condition for the grasp task is defined as: the target object must be lifted to a height exceeding 20 cm.

Pour Task. The success condition for the pour task is determined by checking if any ball is inside the bowl. The key evaluation metric is: a ball is considered inside the bowl if its horizontal distance to the bowl is less than 2 cm.

Lift Task. The success condition for the lift task is defined as: the target object must be lifted to a height exceeding 15 cm.

Handover Task. The success condition for the handover task is defined as: the target object must be lifted to a height exceeding 15 cm, with the right hand completely released (distance > 15 cm) and the left hand maintaining a secure grasp (distance < 10 cm) for 10 consecutive steps.

General Failure Condition. For all tasks except the handover task, if the execution time exceeds 600 steps without achieving the success condition, the task is deemed a failure. For the handover task, the maximum allowed execution steps are increased to 800.

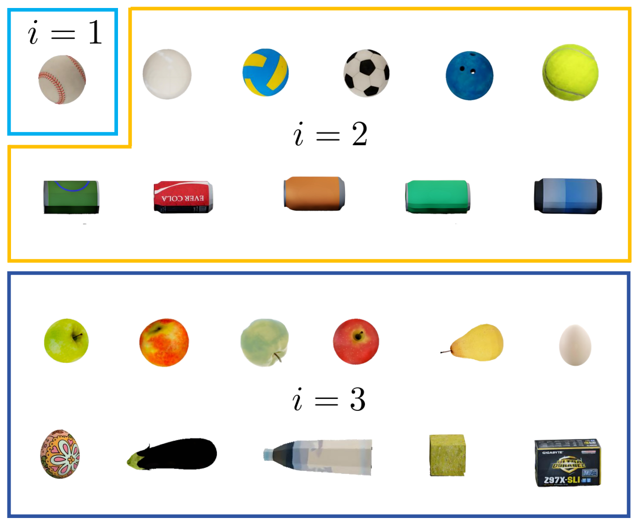

Figure 6: Grasp Task Evaluation Test Set (T_O^i).

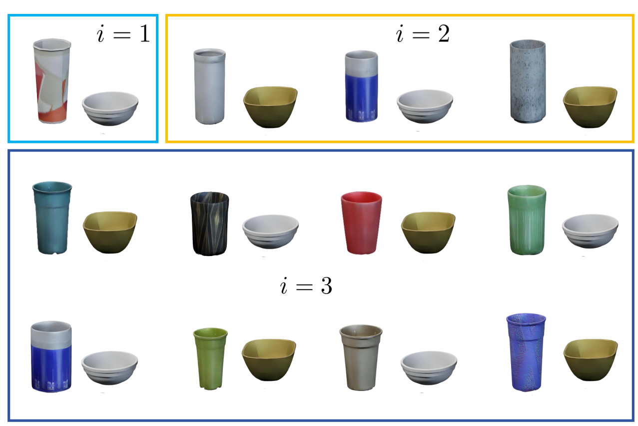

Figure 7: Pour Task Evaluation Test Set (T_O^i).

Figure 8: Lift Task Evaluation Test Set (T_O^i).

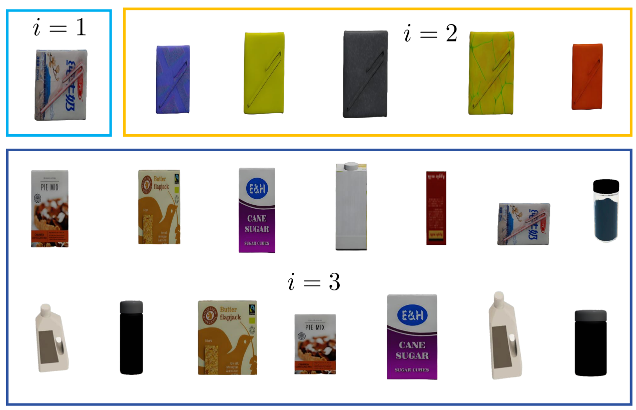

Figure 9: Handover Task Evaluation Test Set (T_O^i).