Supplementary Material

Implementation Details

7.1. Task Construction

Domain Randomization. To enhance the robustness and generalization capability of our system, we implement comprehensive domain randomization strategies during the environment reset phase. The randomization encompasses multiple aspects of the environment:

- Object Mass Randomization: At the beginning of each episode, object masses are randomized by scaling each object's default mass with a random factor sampled from a uniform distribution $U(1, 1.5)$ (units: kg):

$$m_{\text{curr}} = m_{\text{default}} \cdot \alpha, \quad \alpha \sim U(1, 1.5)$$

- Object Position Randomization: Small perturbations are applied to the initial positions of objects to introduce variability (units: meters):

$$\Delta x \sim U(-0.02, 0.02)$$
$$\Delta y \sim U(-0.02, 0.02)$$

- Target Position Randomization: The initial position of the target object within the box is randomized. The random displacements are sampled from (units: meters):

$$\Delta x \sim U(-0.15, 0.15)$$
$$\Delta y \sim U(-0.2, 0.2)$$

  This ensures that the target object can be placed within 70% of the box's area.
- Camera Mount Randomization: During data collection, the camera's mounting position is perturbed with small random displacements (units: meters):

$$\mathbf{p}_{\text{camera}} = \mathbf{p}_{\text{default}} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim U(-0.01, 0.01)^3$$

7.2. RL Training

We employ the Proximal Policy Optimization (PPO) algorithm [41] to train a continuous control policy using an actor-critic architecture. Detailed hyperparameters are provided in Table 5. The policy network is parameterized as a multi-layer perceptron (MLP) with three layers of sizes [1024, 512, 256], utilizing the ELU activation function for improved gradient flow and non-linearity. The standard deviation of the policy distribution is learned via a log-std representation, enabling dynamic adjustment of exploration during training.

To ensure stable and efficient learning, we adopt adaptive learning rate scheduling, starting at 3×10^{-4}. The advantage function is normalized to reduce variance in policy gradient updates, while the Generalized Advantage Estimation (GAE) [46] parameter τ is set to 0.95, striking a balance between bias and variance. Gradient clipping with a norm threshold of 1 is applied to prevent exploding gradients. To constrain policy updates, we employ a PPO clipping range of 0.1, which limits large deviations from the current policy, and enforce a KL divergence threshold of 0.02 to promote conservative updates and prevent policy collapse.

The training runs for a maximum of 50,000 epochs, with model checkpoints saved every 1,000 epochs. The best-performing model is selected based on validation returns and retained after 200 epochs to prevent overfitting. A separate centralized value function is used for advantage estimation, parameterized as an MLP with the same architecture as the policy network. The critic network employs a higher learning rate of 1×10^{-3} to facilitate faster convergence in value estimation, a choice informed by preliminary experiments indicating more stable critic updates with this configuration.

7.3. Baseline Implementation

In our simulation experiments, we compare our method against five baseline approaches. Three of these baselines—*Ours w/o RS*, *Ours w/o* r_{stir}, and *Ours w/o* r_{clean}—are derived by removing specific components from our proposed method. The other two baselines are Visual-based Motion Planning Search (VMP) and Grasp-Pick. VMP is a heuristic motion planning approach that uses target object segmentation masks to guide the robotic hand toward the target object and employs predefined rules for retrieval manipulation. Grasp-Pick involves sequentially grasping and placing objects based on the support relationships within the cluttered scene.

VMP. The VMP system implements a vision-guided manipulation framework for dexterous robotic retrieval tasks in cluttered environments. It integrates visual perception, motion planning, and control execution through a state machine architecture to ensure reliable object manipulation.

The vision module employs a top-down camera with a resolution of 1024×512, capturing RGB, depth, and segmentation maps of the workspace. Target objects are identified using segmentation masks obtained from the segmentation map, with their IDs corresponding to known object labels. The center of the target mask's bounding box is extracted as the 2D image coordinate, which is projected into 3D space using depth data to obtain precise object localization. For motion planning, the robotic arm moves its end effector to

Table 5 | Hyperparameters for PPO Training.

Category	Parameter	Value	Description
Model Architecture	MLP Layers	[1024, 512, 256]	Number of neurons per layer
	Activation Function	ELU	Non-linearity used in the network
Training Parameters	Learning Rate	3×10^{-4}	Step size for policy update
	Discount Factor (γ)	0.99	Reward discounting factor
	GAE Parameter (τ)	0.95	Smoothing factor for GAE
	Entropy Coefficient	0	Weight of entropy regularization
	Gradient Clipping	Norm 1	Prevents gradient explosion
	Clip Range (ϵ)	0.1	PPO clipping threshold
	KL Threshold	0.02	KL divergence threshold for stopping training
	Minibatch Size	512	Batch size for optimization
	Mini Epochs	5	Number of updates per batch
	Horizon Length	8	Number of steps before update
	Max Training Epochs	50,000	Maximum number of training iterations
	Value Learning Rate	1×10^{-3}	Learning rate for value function

the computed 3D coordinate and performs a scrape action to retrieve the object.

When the target object is completely occluded and its segmentation mask cannot be detected, the system employs an exploration strategy by randomly sampling four 3D coordinates within the cluttered bin area. The arm sequentially moves to these coordinates, performing scrape actions to uncover the target object.

Specifically, the entire motion planning and scrape action process employs a four-stage approach to ensure reliable object retrieval.

1. Pre-approach stage: The end-effector moves to a predefined position ($h = 0.5$ m) above the target object. This configuration facilitates subsequent control of the hand to reach any position within the bounding box.
2. Final approach stage: Precise positioning is achieved using visual feedback combined with damped least squares inverse kinematics:

$$\tau = J^T(JJ^T + \lambda I)^{-1}\Delta x$$

   where $\lambda = 0.05$ is the damping parameter, J is the Jacobian matrix, and Δx represents the positional error.
3. Scraping stage: The system executes a periodic motion pattern defined by:

$$x(t) = A\sin(2\pi ft + \phi) + O$$

   where the amplitude $A = 2.0$, frequency $f = 20$ Hz, phase shift $\phi = \pi/4$, and offset $O = 0.5$.
4. Reset stage: The target object has been retrieved, so the robot arm will return the end effector to its initial position.

The control execution module utilizes position-based control for both arm and finger joints. Adaptive damping parameters are applied to ensure stable motion, while joint limits are strictly enforced throughout the execution process: $q_{\min} \le q \le q_{\max}$.

Grasp-Pick. This method relies on support relationships among cluttered objects to guide the grasping sequence. To ensure these assumptions hold, we designed tailored setups for both simulation and real-world experiments.

In simulation, object positions are directly accessible, enabling precise calculation of support relationships. We employ a KD-Tree [47] to organize the coordinates of objects near the target. Based on Euclidean distance, we select the three to five nearest objects, depending on the scenario, and manipulate them sequentially to clear access to the target. While this approach offers computational simplicity, it assumes ideal sensing conditions and may not generalize to more complex spatial arrangements.

For robotic control, we implement damped least squares inverse kinematics:

$$\dot{\mathbf{q}} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \lambda\mathbf{I})^{-1}\dot{\mathbf{x}},$$

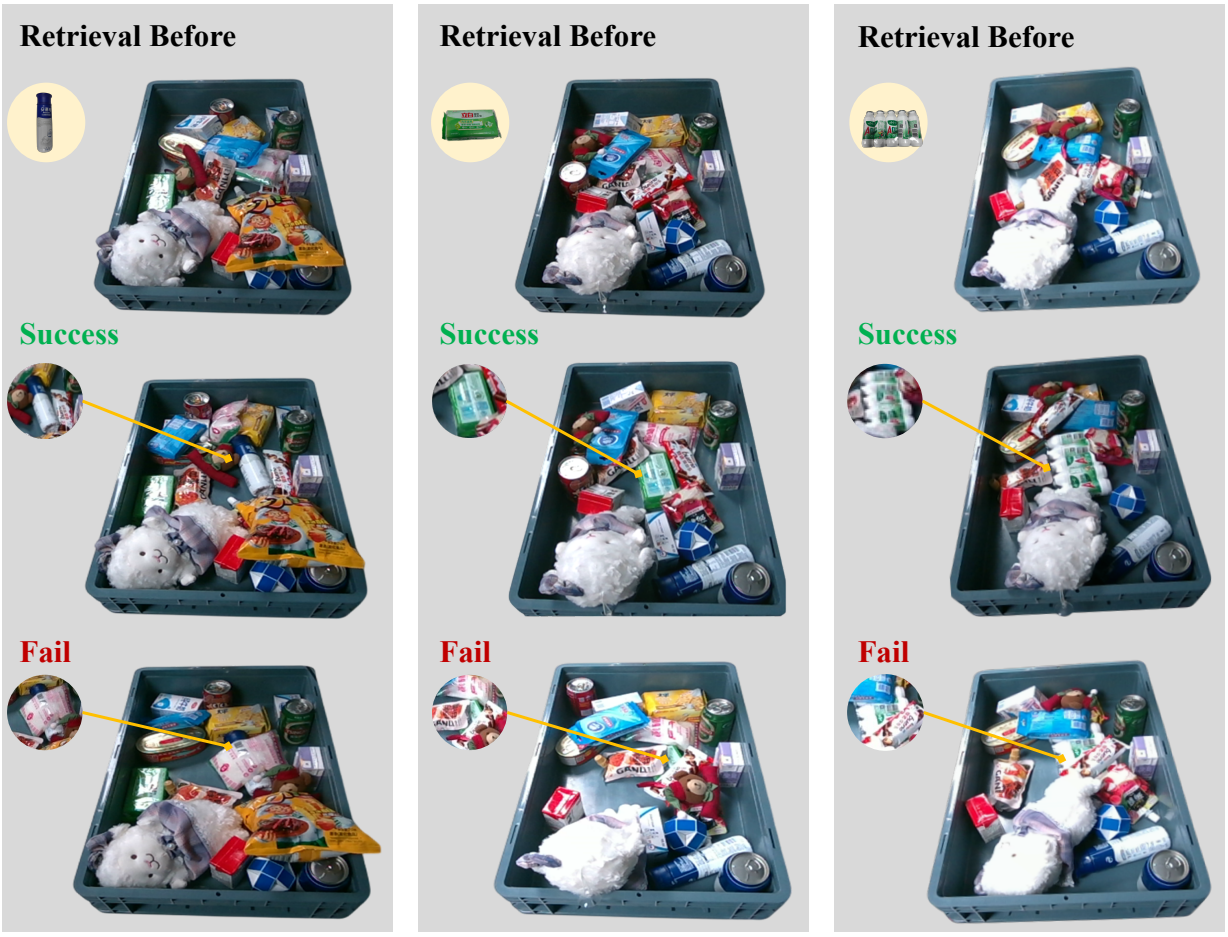

Figure 8 | Examples of successful and failed object retrievals on the real robot.

where λ is the damping coefficient, $\mathbf{J}$ is the Jacobian matrix, and $\dot{\mathbf{x}}$ is the desired end-effector velocity. This formulation offers stable solutions near singularities but may limit the dexterity needed in cluttered environments.

In real-world experiments, sequentially grasping and removing multiple objects in stacked scenes with a dexterous hand remains challenging due to perception and control limitations. To address this, we employ predefined trajectories for each trial, simulating an idealized execution scenario. While this implementation provides an upper-bound estimate of this method's efficiency, it does not reflect the challenges of autonomous execution in unstructured environments.

7.4. Evaluation metric

Exposure Calculation The primary goal of object retrieval is to locate the target object and enhance its visibility within the camera's field of view, facilitating subsequent manipulations. We define *exposure* as the proportion of unobstructed pixels of the target object in the imaging plane. Considering that changes in the object's pose can affect the number of visible pixels, we proceed as follows:

At timestep t, we record the target object's visible pixels p_t^{curr} and its 6D pose. Subsequently, all objects except the target are removed, the target object's recorded 6D pose is reset, and its total visible pixels are recorded as p_t^{all}. The exposure at time t is then computed as:

$$\text{exposure}_t = \frac{p_t^{\text{curr}}}{p_t^{\text{all}}}. \tag{5}$$

Success in Real-World Experiments. To systematically evaluate the success rate of object retrieval on the real robot, we capture images before and after each task using a side-mounted RealSense D435 camera. The success of the retrieval is determined by comparing the exposure of the target object in these images. As illustrated in Figure 8, we present examples of both successful and failed retrieval attempts.

7.5. Sim-to-Real transfer

For real-world deployment, we collect a set of trajectories generated by our RL expert policy. To ensure

Table 6 | Hyperparameters of Distilled Policy.

Category	Parameter	Value	Description
Model Architecture	Input State Dimension	9	Size of input state vector
	Action Dimension	13	Number of output actions
	History Frames	15	Past frames used as input
	Future Action Frames	5	Future actions predicted
	Transformer Hidden Size (d_{model})	384	Hidden layer size
	Number of Attention Heads	6	Transformer attention heads
	Number of Transformer Layers	6	Transformer depth
	Feed-forward Dimension	2048	FFN hidden size
	Dropout Rate	0.15	Dropout probability
Training Parameters	Batch Size	512	Training batch size
	Total Iterations	10,000	Training iterations
	Learning Rate	1e-4	Initial learning rate
	Optimizer	Adam	Optimization algorithm
	Loss Function	Negative Log Product	Loss function used
	Gradient Clip Norm	1.0	Gradient clipping threshold

data quality and consistency, we first select successful trajectories. We then filter out trajectories where the finger's z-coordinate lower 2,cm above the box, a threshold empirically chosen to prevent unstable behavior and reduce the risk of collision with the box during manipulation. To promote generalization, we balance the dataset across various target object positions within the box, ensuring uniform coverage of spatial configurations. This prevents the model from overfitting to specific object placements and enhances its adaptability to unseen scenarios.

The model architecture consists of a state encoder followed by a multi-head self-attention mechanism with six transformer layers, each containing six attention heads. This design captures complex temporal dependencies across historical state sequences of length 15, enabling accurate prediction of future actions over a five-step horizon. The hidden dimension of 384, paired with a feed-forward expansion ratio of 5.33 (2048/384), strikes a balance between model expressiveness and computational efficiency.

To effectively manage the continuous action space inherent in robotic control, we introduce a custom Negative Log Product Loss function, which penalizes trajectory deviations more sensitively than traditional mean squared error. This loss function emphasizes multi-step consistency, enhancing the model's predictive stability. Training is performed using the Adam optimizer with a learning rate of 1×10^{-4} over 10,000 iterations and a batch size of 512. Mixed-precision training accelerates computation without compromising accuracy, while gradient clipping at 1.0 maintains stable learning dynamics. Hyperparameter selection was guided by cross-validation on a held-out dataset to optimize both performance and robustness. Detailed architectural specifications and hyperparameters are provided in Table 6. Despite strong simulation performance, real-world deployment introduces challenges such as sensor noise, domain discrepancies, and dynamic environmental conditions. Our transformer model mitigates these issues by leveraging temporal patterns to predict smooth and consistent actions.