RESTA: TIME-SCALING AUGMENTATION FOR DATA-EFFICIENT ROBOT IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot manipulation policies trained through imitation learning commonly rely on costly demonstrations recorded in one task direction and at one temporal resolution. We propose **Re**versal and **S**tride-based **T**rajectory **A**ugmentation (RESTA), which exploits time-scaling data augmentation to improve data efficiency through trajectory reversal and temporal resampling. Trajectory-reversal augmentation reverses and relabels demonstrations between paired tasks with opposite objectives, reusing recorded rollouts without additional environment interaction. Because contact-rich manipulation is not exactly reversible, we treat reversal as an approximate paired-task prior rather than a physical symmetry. Temporal-stride augmentation varies observation and action sampling strides and conditions the policy on the resulting offsets, enabling fixed or randomized temporal schedules. Experiments across 16 paired bimanual manipulation tasks show that trajectory reversal is particularly effective in the low-data regime and that temporal-stride augmentation improves success under randomized temporal schedules. With 25 and 50 independently collected demonstrations per task, RESTA achieves 78.4% and 85.2% average success, respectively, matching or exceeding baselines trained with twice as many demonstrations (77.3% and 85.4%). These results show that RESTA can improve data efficiency in robot imitation learning.

1 INTRODUCTION

Imitation learning offers a direct way to teach robots manipulation skills (Zhao et al., 2023; Chi et al., 2023; Ze et al., 2024), but requires demonstrations that are costly to collect: each trajectory needs resetting the scene, executing the task, and checking the result. Since a demonstration also carries temporal structure (i.e., time direction and interval between observations and actions), a set of demonstration may thus provide potential supervision both across task directions (e.g., open vs close door) and across temporal resolutions (e.g., accelerating vs decelerating). This motivates our research question: *Can temporal transformations of existing demonstrations improve imitation learning across task directions and execution schedules, without additional data collection?*

We use *time scaling* to describe transformations of the temporal coordinate used to construct policy-training windows. In physics, mathematics, and computer simulation, time scaling refers to changing the ratio between simulation time and real-world time: it can speed up or slow down a process, or formally run its temporal coordinate backward. In our setting, the two relevant properties are *time reversal* and *frequency change*. Time reversal (Jiang et al., 2025; Qiao et al., 2026; Kim et al., 2026; Barkley et al., 2023) changes the direction of a sequence: for paired tasks such as opening and closing an articulated object, a recorded opening trajectory can provide potential supervision for closing when its observations are reversed and actions are relabeled toward the next robot configuration. This correspondence is approximate and depends on whether the reversed transitions are realizable in the environment. Frequency change, in contrast, preserves temporal order while changing the time intervals between observations or action targets. It can be achieved in several ways, including changing action persistence (Hu et al., 2024), using multi-frequency action chunks (Zhang et al., 2026), or sparsely sampling target sequences (Bai et al., 2026; Park et al., 2025). In this work, we use positive temporal strides to change frequency: sampling a trajectory every step defines the reference speed; increasing the stride samples more widely separated states and corresponds intuitively to accelerating the robot's motion, whereas decreasing it samples more densely and corresponds to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

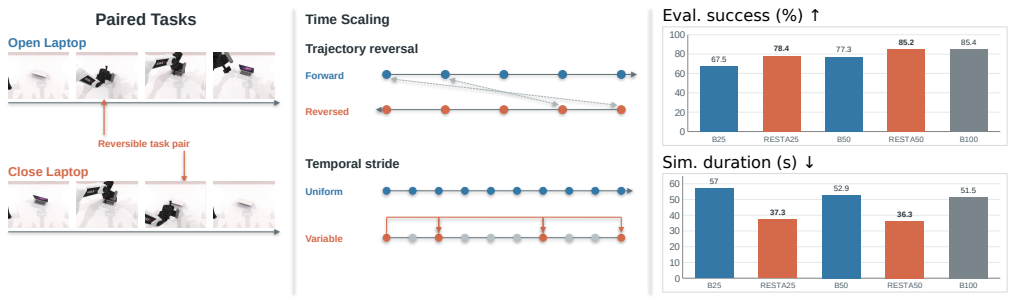


Figure 1: Overview and aggregate results for RESTA. Left: Open Laptop and Close Laptop trajectories illustrate a reversible task pair. Middle: time scaling is illustrated through trajectory reversal, which connects representative states across forward and reversed time, and temporal-stride augmentation, which samples a shared timeline uniformly or variably. Right: bars report mean evaluation success and approximate simulated duration over the 16-task suite; higher success and lower duration are preferred. B_n denotes DP3 trained with n native demonstrations per task, and RESTA_n adds trajectory-reversal and temporal-stride augmentation.

decelerating it. Thus, reversal changes the task direction of a demonstration, while our stride-based augmentation changes its temporal resolution. During training, both can be naturally jointly applied.

To exploit these two properties, we propose **Re**versal and **S**tride-based **T**rajectory **A**ugmentation (RESTA), which combines paired-task trajectory reversal with temporal-stride augmentation and temporal offset conditioning. In more details, reversal uses a trajectory from one task direction to provide additional supervision for its paired direction, reducing the need to collect independent demonstrations for both. We form these examples by reversing the observation sequence and relabeling each action toward the next robot configuration. Temporal-stride augmentation samples histories and action chunks at different positive strides, allowing the same demonstrations to represent varied control frequencies. Because the policy must distinguish these frequencies, we provide the temporal offset, which contains both observation and action offsets, through a lightweight embedding. This conditioning can be adapted to different policy architectures; at inference, different offset schedules set the time intervals between predicted targets, allowing the policy to operate at higher or lower effective frequencies.

Contributions Our contributions can be summarized as follows (Figure 1):

(i) We formulate the imitation-learning training-window problem and its time-scaling view, with each window pairing an observation history with future action targets. This gives a common view of trajectory reversal and positive temporal resampling, exposing two ways to reuse temporal structure in demonstrations (Sections 4.1 and 4.2).

(ii) We introduce paired-task trajectory reversal as a data augmentation (Section 4.3) that creates candidate behavior-cloning examples by reversing observations and relabeling absolute next-configuration targets, without additional environment interaction.

(iii) We introduce temporal-stride augmentation with observation and action offset conditioning (Section 4.4) to vary the frequency in demonstration windows and tell the policy at which time intervals it should represent and predict.

(iv) We evaluate RESTA on sixteen paired bimanual tasks (Section 5.2). At matched data budgets, it achieves higher average success and requires fewer simulator physics steps per successful episode than the baseline, indicating more reliable task completion and shorter simulated execution time. We also compare against larger-data baselines and analyze the contribution of each augmentation.

2 Related Work

We organize the related work around two directions: action-chunk imitation learning and temporal transformations. Within temporal transformations, we distinguish time-reversal methods from approaches that change action frequency. Due to page limit, discussions of action-chunk imitation policies and temporal-frequency methods appear in Section A.

Temporal Symmetry for Robotics Control Time Reversal Deep Reinforcement Learning (TR-DRL) (Jiang et al., 2025) filters dynamics-consistent reversible transitions, augments replay data, and shapes rewards using reversal; Time-Symmetric Data Augmentation (Barkley et al., 2023) also constructs reverse-time transitions for off-policy RL and studies failures from irreversible dynamics. Other uses include symmetry-guided quadruped gait transitions (Ding et al., 2024), self-supervised manipulation planning (Nair et al., 2020), and reverse curricula that alter the state distribution (Florensa et al., 2017; Tao et al., 2024). We instead construct behavior-cloning examples from paired manipulation demonstrations. Recent work also studies demonstration reversal. Reverse Playback Motion (Kim et al., 2026) has an operator execute a motion from its goal state, then reconstructs a forward trajectory by reversing recorded end-effector poses. We reverse observations and relabel targets in an existing expert dataset, without additional execution or collection. Reverse to Advance (Qiao et al., 2026) iteratively uses environment-collected easy- and hard-task rollouts, retaining segments with kinematic and critic-guided advantage filters; a learned dynamics filter appears only in a comparison baseline. Our method uses fixed expert trajectories without policy rollouts, resets, or new data, and also augments temporal strides and observation offsets.

3 Background

The background covers the foundations of action-chunk imitation learning: the Markov decision-process formulation and offline imitation learning (Sections 3.1 and 3.2), followed by 3D Diffusion Policy (DP3) as the action-chunk policy backbone (Section 3.3).

3.1 Markov Decision Processes

We use a finite-horizon Markov decision process (MDP) to model robot manipulation. It is specified by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \rho, r, T)$, with full robot–environment state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel P, initial-state distribution ρ, reward r, and horizon T. At the start of each episode, the initial state is sampled as $s_0 \sim \rho$, where ρ is the distribution over robot–environment states induced by the reset procedure. At step t, state $s_t \in \mathcal{S}$ evolves as $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ under action $a_t \in \mathcal{A}$. Actions specify absolute joint-position and gripper targets. A supplied low-level controller converts each target into the robot commands executed over the next transition, so its tracking behavior is included in the transition kernel P. The objective is to reach the success set $\mathcal{G} \subseteq \mathcal{S}$ within T steps. Episodes terminate at first success or the horizon, with reward one for success and zero otherwise (provided by reward function r). The policy uses sensory observations o_t in a length-L history $h_t = (o_{t-L+1}, \ldots, o_t)$ instead of full states.

3.2 Imitation Learning

Offline imitation learning fits a policy to N expert trajectories without rewards or a transition model. Let $\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}$, where $\tau^{(n)} = (o_0, a_0, \ldots, a_{T_n-1}, o_{T_n})$ has T_n transitions. At timestep t, the policy receives the history $h_t = (o_{t-L+1}, \ldots, o_t)$ and predicts an H-action chunk. A training window is the pair (h_t, A_t^*), where $A_t^* = (a_t^*, \ldots, a_{t+H-1}^*)$; valid windows satisfy $t \geq L-1$ and $t + H - 1 < T_n$.

At deployment, the history has length L; before L observations are available, the earliest observation is repeated to left-pad it, and then the most recent L observations are retained. The policy executes part of each predicted chunk, collects new observations, and replans. Behavior cloning (BC) fits the policy to these expert windows with the objective

$$\mathcal{L}_{\mathrm{IL}}(\theta) = \mathbb{E}_{(h_t, A_t^*) \sim \mathcal{D}} \left[\ell(\pi_\theta(\cdot \mid h_t), A_t^*) \right], \tag{1}$$

where $\pi_\theta(\cdot \mid h_t)$ is the predictive distribution and ℓ measures its disagreement with the expert chunk and the expectation ranges over valid windows. Here, windows use consecutive trajectory samples; our method later varies their temporal offsets in Section 4.2.

3.3 3D Diffusion Policy (DP3)

DP3 (Ze et al., 2024) represents $\pi_\theta(A_t \mid h_t)$ through conditional denoising. An encoder e_ϕ with parameters ϕ combines point-cloud features and proprioception across h_t into $c_t = e_\phi(h_t)$. In our action-chunk setting, the policy predicts H absolute joint-position and gripper targets from this observation history. Let $\psi(o_t)$ extract the recorded joint positions and gripper configuration in the controller's target representation. For consecutive samples, the supervised action chunk is $(\psi(o_{t+1}), \ldots, \psi(o_{t+H}))$, with $t + H \le T$. A supplied low-level controller drives the robot toward each predicted target; it is not learned, and its tracking behavior is included in the transition model. The objective is to learn action-chunk predictions from demonstrations that enable successful closed-loop task execution. For a normalized expert chunk x_0, diffusion step n produces

$$x_n = \sqrt{\bar{\alpha}_n}\, x_0 + \sqrt{1 - \bar{\alpha}_n}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{2}$$

where x_n is the noisy chunk, $\bar{\alpha}_n$ is cumulative signal retention, ϵ is Gaussian noise, and I is the identity covariance. Training learns denoising conditioned on c_t and n using the objective

$$\mathcal{L}_{\mathrm{DP3}}(\theta; x_0, c_t) = \mathbb{E}_{n,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_n, c_t, n)\|_2^2\right], \tag{3}$$

where ϵ_θ predicts the noise added at diffusion step n. Inference iteratively converts noise into an action chunk.

4 RESTA

We learn action-chunk policies from expert demonstrations using two augmentations based on the time-scaling framework introduced below. After defining the control problem and paired-task setting in Section 4.1, we introduce the time-scaling framework in Section 4.2, then construct reversed demonstrations to share supervision between task directions in Section 4.3. We then vary the temporal offset of training windows and condition the policy on that offset in Section 4.4. Section 4.5 brings these steps together for training and inference.

4.1 Problem Formulation

We consider learning robot control from a fixed dataset $\mathcal{D}$ of expert demonstrations. The action-chunk policy and its target representation are defined in the DP3 background; here we specify the paired-task setting used for trajectory reversal. The tasks are organized into pairs with expert datasets $\mathcal{D}^+$ and $\mathcal{D}^-$, such as opening and closing an articulated object. We construct reverse-task supervision by reversing recorded trajectories and relabeling each action toward the next configuration in the reversed sequence. This is an offline augmentation procedure, not an assertion that every reversed transition is physically realizable.

4.2 Time Scaling

Time scaling changes how a demonstrated trajectory is indexed to construct a training window (h_t, A_t), a finite portion of the trajectory used to form one training example. These windows appear in Equation (1). Let $\xi \in [0, T]$ denote the source coordinate along the original trajectory, with T its final source-coordinate value. For a window with transformed sampling instants $0 = v_0 < v_1 < \cdots < v_N = \mathcal{V}$, where $\mathcal{V}$ is the transformed window endpoint, define the mapping $\xi(v)$ from transformed position to source-coordinate location by

$$\xi(v_0) = \xi_0, \qquad \xi(v_k) = \xi(v_{k-1}) + \lambda_k(v_k - v_{k-1}), \qquad \lambda_k \neq 0, \tag{4}$$

where ξ_0 is the source-coordinate location at the beginning of the window. The local scaling factor λ_k may vary across steps. The window is valid when $0 \le \xi(v_k) \le T$ for every k. The sign of λ_k determines the local traversal direction, while $|\lambda_k|$ determines the source-coordinate spacing for that step. Constant scaling is the special case $\lambda_k = \lambda$ for all k. For positive scaling, a larger value

of λ_k advances farther along the demonstrated trajectory per unit transformed time and therefore corresponds to accelerating along the trajectory; a smaller positive value corresponds to decelerating. When the scale varies across a window, the low-level controller must track these changing effective rates.

Reversal uses the constant schedule $\lambda_k = -1$ for all k, with $\xi_0 = T$ and $\mathcal{V} = T$, so that $\xi(v) = T - v$. Let $\mathbf{s}(\xi)$ denote the full state trajectory. Physical reversal then gives $\mathbf{s}^{\mathrm{rev}}(v) = \mathcal{R}\big(\mathbf{s}(\xi(v))\big) = \mathcal{R}\big(\mathbf{s}(T - v)\big)$, where $\mathcal{R}$ preserves positions and negates velocities. For the absolute targets in Section 3.1, reversed demonstrations require actions relabeled toward preceding source configurations, rather than simply reversed action arrays. This provides reverse-task supervision for paired tasks such as opening and closing, but contact, friction, delays, and grasp or release events can make some reversed transitions infeasible.

Positive scaling preserves order. For uniform recording frequency f_0 and period $\Delta t_0 = 1/f_0$, sampling v every Δt_0 with $\lambda = g + 1$ skips $g \in \{0, 1, 2, \ldots\}$ source samples. The resulting source-time interval and sampling frequency are

$$\Delta t_g = (g + 1)\Delta t_0, \qquad f_g = \frac{f_0}{g + 1}. \tag{5}$$

Variable strides give a piecewise time transformation. Thus, reversal changes the task direction represented by supervision, while resampling changes its temporal resolution.

Idealized transition reversibility. Using the transition model in Section 3.1, write P^+ and P^- for the paired task directions. Following the definition of full time-reversal symmetry in Jiang et al. (2025), an idealized reversibility condition would require that for each transition (s_t, a_t, s_{t+1}), there exists a reverse action $\hat{a}_t \in \mathcal{A}$ satisfying

$$P^+(s_{t+1} \mid s_t, a_t) = P^-(\mathcal{R}s_t \mid \mathcal{R}s_{t+1}, \hat{a}_t). \tag{6}$$

This equality describes an idealized condition for exact transition reversibility; our method does not assume that it holds for every transition in the demonstrations. We nevertheless use reversed trajectories and their relabeled targets directly for training, without segment-level feasibility filtering. This simple augmentation is effective in our experiments (Sections B.2 and 5.2), while filtering infeasible segments is a promising direction for future work.

Strided target prediction. Building on the DP3 action-chunk formulation in Section 3.3, we now allow the observation history and target samples within a chunk to use positive adjacent offsets rather than the default consecutive spacing. Consider a discrete demonstration indexed by $t = 0, \ldots, T$, where T is the final trajectory index. For an anchor t, we randomly sample observation indices $0 \le i_1 < \cdots < i_L = t$ and target indices $t < j_1 < \cdots < j_H \le T - 1$ according to the selected observation and action strides. Consecutive sampling is the special case $i_\ell = t - L + \ell$ and $j_k = t + k$. The resulting observation history and action targets are $(o_{i_1}, \ldots, o_{i_L})$ and $(a_{j_1}, \ldots, a_{j_H}) = (\psi(o_{j_1+1}), \ldots, \psi(o_{j_H+1}))$, respectively, where each action target is the configuration of the next observation.

In the next two subsections, we explain how to exploit time scaling in imitation learning.

4.3 Trajectory Reversal Augmentation

Trajectory reversal augments each task's expert dataset with demonstrations from its paired task using the offline construction in Section 4.1. The construction reverses the observation order and relabels each action to target the next configuration in the reversed sequence.

Reversed demonstrations. Given a source trajectory $\tau = (o_0, a_0, \ldots, a_{T-1}, o_T)$ with T transitions, we reverse the observation order:

$$\tilde{o}_t = o_{T-t}, \qquad t = 0, \ldots, T. \tag{7}$$

This applies the reversal case $\lambda = -1$ of Equation (4) to the recorded point clouds and robot configurations.

Using the configuration extractor ψ defined in Section 3.3, the action at each reversed observation targets the next robot configuration in the reversed sequence:

$$\tilde{a}_t = \psi(\tilde{o}_{t+1}) = \psi(o_{T-t-1}), \qquad t = 0, \dots, T-1. \tag{8}$$

The targets are thus obtained directly from recorded proprioception.

Let Rev denote observation reversal together with this action relabeling. Applying it in both directions yields two augmented datasets, which can be used to train for their respective tasks:

$$\widetilde{\mathcal{D}}^{+} = \mathcal{D}^{+} \cup \mathrm{Rev}(\mathcal{D}^{-}), \qquad \widetilde{\mathcal{D}}^{-} = \mathcal{D}^{-} \cup \mathrm{Rev}(\mathcal{D}^{+}). \tag{9}$$

The two datasets reuse the recorded demonstrations without extra environment interaction. We next describe how to sample training windows from these trajectories at different temporal spacings.

4.4 Temporal-Stride Augmentation

For either an original or a reversed trajectory, consecutive training samples correspond to $g = 0$ in Equation (5). Temporal-stride augmentation also samples windows with $g \geq 1$, so that the policy learns to predict chunks of more widely spaced joint targets. We apply the positive time scaling in Section 4.2 to vary both observation and action spacing while keeping the history length L and action horizon H fixed.

Training windows. Let $\mathcal{B}$ be the set of allowed positive integer strides. For each training window anchored at index t, we independently and randomly sample the observation and action offset vectors. Let $\boldsymbol{\delta}^o = (\delta^o_1, \dots, \delta^o_{L-1}) \in \mathcal{B}^{L-1}$ and $\boldsymbol{\delta}^a = (\delta^a_1, \dots, \delta^a_H) \in \mathcal{B}^H$. These are adjacent offsets: each entry is the gap between neighboring samples, with the first action gap measured from the anchor. They determine the time indices through

$$i_L = t, \;\; i_\ell = i_{\ell+1} - \delta^o_\ell \;\; (\ell = 1, \dots, L-1), \quad j_0 = t, \;\; j_k = j_{k-1} + \delta^a_k \;\; (k = 1, \dots, H), \tag{10}$$

Since all adjacent offsets are positive, these indices satisfy

$$0 \leq i_1 < \cdots < i_L = t = j_0 < j_1 < \cdots < j_H \leq T-1. \tag{11}$$

Thus, $\boldsymbol{\delta}^o$ and $\boldsymbol{\delta}^a$ directly specify the adjacent temporal spacing of the observation history and action chunk. Writing $\eta = [\boldsymbol{\delta}^o, \boldsymbol{\delta}^a]$ for this adjacent-offset schedule, we construct

$$h^\eta_t = (o_{i_1}, \dots, o_{i_L}), \qquad A^\eta_t = (a_{j_1}, \dots, a_{j_H}). \tag{12}$$

These offsets describe spacing in the recorded sequence. For uniformly timed recordings, their relation to sampling frequency follows Equation (5).

Temporal-offset conditioning. Temporal-offset conditioning consists of two offset vectors: the adjacent spacing $\boldsymbol{\delta}^o$ in the observation history and the adjacent spacing $\boldsymbol{\delta}^a$ in the future action chunk. We collect them in the schedule $\eta = [\boldsymbol{\delta}^o, \boldsymbol{\delta}^a]$. Since these two offset vectors are sampled independently, the observation history alone does not determine the temporal spacing of the action targets. We therefore provide the full schedule η to the policy by augmenting the DP3 condition $e_\phi(h^\eta_t)$ with a learned embedding of η. Let g_ω denote a learnable function with parameters ω that encodes the normalized adjacent observation and action offsets. The resulting condition and policy become:

$$c^\eta_t = \left[e_\phi(h^\eta_t), g_\omega(\eta)\right], \qquad \pi_\theta(A^\eta_t \mid h^\eta_t, \eta), \tag{13}$$

where brackets denote concatenation and θ collects the trainable policy parameters.

Inference schedules. Because the policy is conditioned on temporal offsets, inference must specify the spacing of the future targets before each prediction. At replanning cycle q, the scheduling rule σ selects $\boldsymbol{\delta}^a_q \in \mathcal{B}^H$ according to a fixed, randomized, or context-dependent strategy. If t_q is the current anchor, the target indices satisfy

$$j_{q,0} = t_q, \qquad j_{q,k} = j_{q,k-1} + \delta^a_{q,k}, \qquad k = 1, \dots, H. \tag{14}$$

For $q = 0$, there is no previous action chunk, so the history is initialized with configured observation offsets $\boldsymbol{\delta}^o_0 = \boldsymbol{\delta}^{o,\mathrm{init}}$ (consecutive observations correspond to $\boldsymbol{\delta}^{o,\mathrm{init}} = (1, \dots, 1)$), and the initial schedule is $\eta_0 = [\boldsymbol{\delta}^{o,\mathrm{init}}, \boldsymbol{\delta}^a_0]$. For $q \geq 1$, the current observation offsets are determined by the

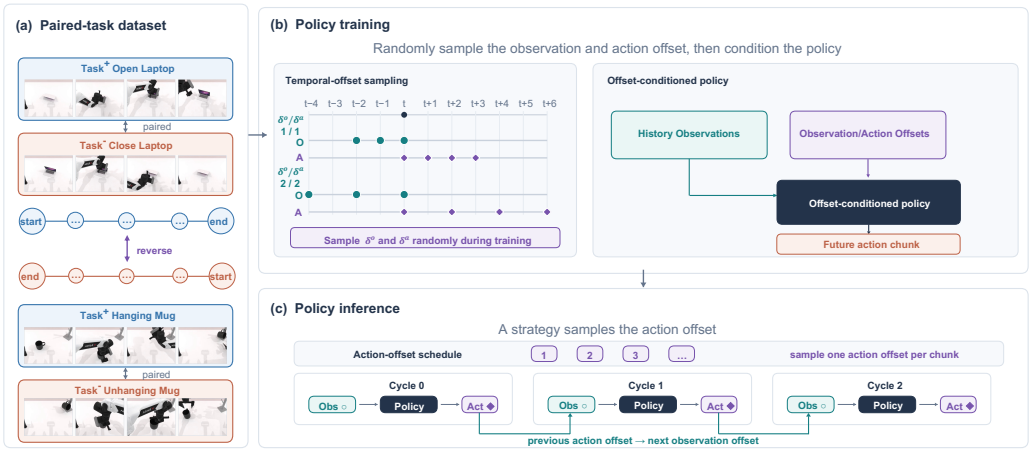


Figure 2: Overview of RESTA. (a): paired task directions provide demonstrations that are reversed and relabeled to construct augmented datasets. (b): during training, sampled temporal offsets determine the observation and action samples and condition the policy. (c): at inference, a schedule selects action offsets, and the executed action chunk determines the observation spacing for the next policy cycle.

spacing of the previous action chunk. Although observations arrive one at a time, the retained history represents the gaps induced by the last chunk, so

$$\boldsymbol{\delta}_q^o = \left(\delta^a_{q-1,H-L+2}, \ldots, \delta^a_{q-1,H}\right). \tag{15}$$

For example, with $L = 3$, $H = 6$, and $\boldsymbol{\delta}^a_{q-1} = (2, 1, 3, 2, 1, 3)$, the transition from cycle $q-1$ to cycle q gives $\boldsymbol{\delta}^o_q = (1, 3)$. The schedule $\eta_q = [\boldsymbol{\delta}^o_q, \boldsymbol{\delta}^a_q]$ then conditions the prediction via Equation (13). We study the choice of σ in the action-offset ablation in Section 5.3.

4.5 RESTA Algorithm

The two augmentations enter training in sequence. We first form $\widetilde{\mathcal{D}}^+$ and $\widetilde{\mathcal{D}}^-$ using Equation (9). We then sample temporal schedules when constructing windows from these datasets, so both original and reversed demonstrations provide supervision at different spacings.

For direction $\kappa \in \{+, -\}$, sample a trajectory τ from $\widetilde{\mathcal{D}}^\kappa$, a valid anchor t, and a schedule η. We construct h^η_t and A^η_t using Equation (12) and their condition c^η_t using Equation (13). Using the conditional denoising loss defined in Equation (3), the training objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau,t,\eta}\left[\mathcal{L}_{\mathrm{DP3}}(\theta; A^\eta_t, c^\eta_t)\right]. \tag{16}$$

This applies the conditional denoising objective in Equation (3) to the augmented windows sampled under the imitation-learning dataset. At inference, the scheduling rule σ selects $\boldsymbol{\delta}^a_q$, which determines the future action indices through Equation (14). We obtain the observation offsets from the preceding chunk using Equation (15). The resulting schedule conditions the next action prediction together with the current observation history. The appendix pseudocode in Algorithm 1 summarizes both stages, and Figure 2 shows the method overview.

5 Experimental Results

We evaluate RESTA on paired bimanual manipulation tasks from the RoboTwin benchmark (Mu et al., 2025; Chen et al., 2025). We first describe the benchmark, training protocol, and evaluation metric in Section 5.1. Section 5.2 compares RESTA with DP3 at the same and twice the demonstration budget. Section 5.3 examines action-offset selection at inference, randomized observation offsets during training, and the individual effects of trajectory reversal and temporal-stride augmentation.

Table 1: Main results on all 16 paired tasks at 3,000 training epochs. B_n is DP3 trained on n native trajectories. $RESTA_n$ is DP3 trained on the same n native trajectories and their offline time-reversed counterparts, with temporal-stride augmentation and observation-offset conditioning. The subscript counts independently collected trajectories; B_{2n} is DP3 trained on $2n$ native trajectories. Bold text with blue shading marks the best result at each current budget; B_{100} is a reference upper bound and is not highlighted. Values are success rates (%). Higher is better.

Task	B_{25}	$RESTA_{25}$	B_{50}	$RESTA_{50}$	B_{100}
Place Fan	23	**43**	34	**55**	65
Remove Fan	87	**96**	92	**99**	99
Place Object Stand	43	**58**	61	**79**	80
Remove Object Stand	64	**75**	**82**	76	87
Place Empty Cup	47	**77**	77	**91**	93
Remove Empty Cup	91	**97**	90	**100**	100
Hang Mug	19	**25**	24	**36**	31
Unhang Mug	**79**	72	**96**	85	94
Rotate QR Code	55	**80**	69	**88**	75
Rotate QR Code Back	99	**100**	**100**	99	98
Handover Block	86	**92**	89	**96**	97
Handover Block Back	99	**100**	97	**99**	99
Open Laptop	70	**93**	85	**97**	86
Close Laptop	**73**	69	88	**92**	88
Open Microwave	72	**96**	80	**98**	92
Close Microwave	73	**82**	73	**74**	83
Average(%)	67.5	**78.4**	77.3	**85.2**	85.4

5.1 Experimental Setup

We select eight forward tasks from RoboTwin 2.0 and construct a reverse-task counterpart for each one using the trajectory-reversal procedure. This gives sixteen paired tasks in total; their complete list is provided in Section B.1. We use DP3 as the baseline policy and keep the observation and action interfaces fixed across methods. All models are trained for 3,000 epochs and evaluated for 100 episodes per task. We report task success rates and their average over the sixteen tasks. The model, optimization, and augmentation settings are summarized in Table 3 in the appendix. We count only independently collected demonstrations toward each data budget; reverse trajectories are generated offline. Unless otherwise noted, methods that use temporal-stride augmentation and observation-offset conditioning are evaluated with the `rand` action schedule: for each prediction chunk, one action gap is sampled from $\{1, 2, 3\}$ and applied uniformly across the chunk, with the corresponding observation offsets supplied to the policy.

5.2 Main Results

Evaluation Success Rate Table 1 compares the baseline and RESTA at two collection budgets. Our goal is to test whether $RESTA_n$ can improve over the same-budget baseline B_n while approaching the performance of the larger-data reference B_{2n}. Across the full task suite, the resulting relationship is $B_n < RESTA_n \approx B_{2n}$. In practical terms, RESTA can reach the performance of a policy trained with $2n$ native demonstrations while collecting only n trajectories and deriving the additional reverse trajectories offline. For a paired task, demonstrations collected in one direction can therefore provide supervision for the opposite direction without a second data-collection run. We count only independently collected demonstrations toward the data budget; reverse trajectories are generated offline.

Execution-Step Analysis Success rate alone does not reveal the execution cost of successful episodes. We report both policy actions and simulator physics steps until success. Physics steps count each simulator update, including the low-level interpolation used to execute an action. The simulator advances by $1/250$ s per physics step, or 4 ms, so dividing the count by 250 gives the simulated duration of an episode. This duration is a closer proxy for robot execution time than the

Table 2: Average execution steps over the sixteen paired tasks. Each task mean uses all successful episodes. RESTA uses the `rand` offset schedule. Lower is better.

Measure	B_{25}	RESTA_{25}	B_{50}	RESTA_{50}	B_{100}
Policy actions	217.8	**100.3**	209.2	**99.3**	196.4
Physics steps	14251.3	**9337.0**	13223.3	**9063.5**	12879.4

policy-action count, although it does not include policy inference or other system overhead and is not a hardware timing measurement. For each task, we average over successful episodes and then average the task means across the sixteen tasks. In these units, the mean simulated duration is 57.0 s for B_{25} and 37.3 s for RESTA_{25}, and 52.9 s for B_{50} and 36.3 s for RESTA_{50}; B_{100} averages 51.5 s. Thus, RESTA reduces this duration by 19.7 s and 16.6 s at the two matched budgets, respectively. Table 2 reports average policy-action and physics-step counts; per-task results are provided in Tables 8 and 9.

5.3 Ablation Study

We examine how to select action offsets at inference, whether to randomize observation offsets during training, and how each augmentation contributes at different demonstration budgets.

Action-offset Selection At $n = 50$, we evaluate the same TSAO checkpoint under seven inference schedules. The fixed schedules `offset1`, `offset2`, and `offset3` use action gaps of one, two, and three, respectively. The `rand` schedule samples one gap per chunk and applies it to every target, whereas `rand_vector` samples each gap independently. The voting schedules `vote3` and `vote11` select among candidate action chunks. This ablation provides the full analysis of action-schedule selection; definitions of all seven inference schedules are provided in Section B.3. At $n = 50$, `rand` has the highest average success among the seven evaluated schedules (83.2%), so we use it for RESTA and the TSA–TSAO comparison. Full per-task schedule results are in Table 6.

Randomized Observation Offsets We compare temporal-stride augmentation with consecutive training observations (TSA) against its extension with randomized observation offsets and the corresponding conditioning (TSAO). Both variants randomize action gaps during training and use `rand` at inference. TSAO exposes the policy to observation histories with different temporal spacings, matching the variation induced by previously executed action chunks. TSAO improves over TSA at $n = 25$ (74.6% vs. 72.4%) and at $n = 50$ (83.2% vs. 80.8%), supporting the benefit of training on varied observation histories and conditioning on their offsets. The full per-task comparison is provided in Table 7.

Effects of the Proposed Techniques We compare trajectory reversal, temporal-stride augmentation, and their combination at matched collection budgets. TR_n uses n native and n time-reversed trajectories, whereas TSAO_n uses n native trajectories with temporal-stride augmentation and observation-offset conditioning, evaluated with the selected `rand` schedule. $RESTA_n$ combines trajectory reversal with this temporal-stride and observation-offset augmentation. The full per-task results and averages are provided in Table 5. TSAO_n is the stronger aggregate component under the randomized inference schedule, consistently improving average success across all evaluated collection settings. TR_n can be more helpful on individual paired tasks whose demonstrations transfer well across directions. Combining the two gives the strongest current-budget result and approaches the larger-data reference without additional independent collection.

6 Conclusion and Limitations

RESTA combines offline paired-task reversal, temporal strides, and temporal offset conditioning. Across 16 simulated RoboTwin tasks, it raises success from 67.5% to 78.4% at 25 demonstrations and from 77.3% to 85.2% at 50, nearing DP3's 85.4% at 100 while shortening episodes (Tables 1 and 2). Simulation-only tests and infeasible reversals limit our evidence; future work includes automatic task-pair discovery.

AI USE STATEMENT

We used generative AI tools to assist with drafting and revising selected manuscript text; writing code and running experiments; generating tables from the experimental results; and designing and editing the teaser figure, including its figure-production code. AI tools were not used to fabricate or alter experimental measurements. We reviewed AI-assisted code, experiment outputs, tables, text, and visual materials, and checked reported values against the experiment results. The authors take responsibility for the final content of this work, including text, claims, code, tables, and figures produced with AI assistance.

REFERENCES

Ranga Shaarad Ayyagari, Anurita Ghosh, and Ambedkar Dukkipati. Temporal abstraction in reinforcement learning with offline data. *arXiv preprint arXiv:2407.15241*, 2024. URL https://arxiv.org/abs/2407.15241.

Jiaqi Bai, Jindou Jia, Yuxuan Hu, Gen Li, Xiangyu Chen, Tuo An, Kuangji Zuo, and Jianfei Yang. Flash: Efficient visuomotor policy via sparse sampling. *arXiv preprint arXiv:2605.15492*, 2026. URL https://arxiv.org/abs/2605.15492.

Brett Barkley, Amy Zhang, and David Fridovich-Keil. An investigation of time reversal symmetry in reinforcement learning. *arXiv preprint arXiv:2311.17008*, 2023. URL https://arxiv.org/abs/2311.17008.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2023. URL https://arxiv.org/abs/2303.04137.

Jiayu Ding, Xulin Chen, Garret E. Katz, and Zhenyu Gan. Towards dynamic quadrupedal gaits: A symmetry-guided RL hierarchy enables free gait transitions at varying speeds. *arXiv preprint arXiv:2403.10723*, 2024. URL https://arxiv.org/abs/2403.10723.

Mingjie Dong, Hanwei Ruan, Zeyu Wang, Chenyang Sun, Shiping Zuo, Yifeng Chen, Jianfeng Li, and Mingming Zhang. Toward physician-level performance in robot-assisted ankle rehabilitation via imitation learning with empirical and temporal adaptation. *IEEE Transactions on Robotics*, 41:5360–5377, 2025. doi: 10.1109/TRO.2025.3600162.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, 2017. URL https://arxiv.org/abs/1707.05300.

Jianshu Hu, Paul Weng, and Yutong Ban. State-novelty guided action persistence in deep reinforcement learning, 2024. URL https://arxiv.org/abs/2409.05433.

Yunpeng Jiang, Jianshu Hu, Paul Weng, and Yutong Ban. Time reversal symmetry for efficient robotic manipulations in deep reinforcement learning. In *Advances in Neural Information Processing Systems 38*, pp. 158404–158435. Neural Information Processing Systems Foundation, Inc., 2025. doi: 10.52202/085713-4778. URL https://doi.org/10.52202/085713-4778.

Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://arxiv.org/abs/2105.06411.

Hyunjun Kim, Jaeseog Won, Hong-ryul Jung, Donghoon Kim, Minsuk Ko, and Hyungpil Moon. Reverse playback motion: A simple way for anyone to obtain high-quality demonstration data. *International Journal of Precision Engineering and Manufacturing*, 2026. doi: 10.1007/s12541-026-01589-3. URL https://doi.org/10.1007/s12541-026-01589-3.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 27649–27660, 2025.

Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. Time reversal as self-supervision. In *2018 IEEE International Conference on Robotics and Automation*, 2020. doi: 10.1109/ICRA40945.2020.9196862. URL https://arxiv.org/abs/1810.01128.

Hyeonjun Park, Daegyu Lim, Seungyeon Kim, and Sumin Park. Proleptic temporal ensemble for improving the speed of robot tasks generated by imitation learning. *Journal of Korea Robotics Society*, 20(1):61–68, 2025. doi: 10.7746/jkros.2025.20.1.061. URL https://arxiv.org/abs/2410.16981.

Qiyuan Qiao, Ge Yuan, Can Wang, and Dong Xu. Reverse to advance: Teleoperation-cost effective hard policy learning from reversed easy tasks. *arXiv preprint arXiv:2607.13455*, 2026. URL https://arxiv.org/abs/2607.13455.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020. URL https://arxiv.org/abs/2006.12862.

Andrea Tagliabue and Jonathan P. How. Efficient deep learning of robust policies from MPC using imitation and tube-guided data augmentation. *IEEE Transactions on Robotics*, 40:4301–4321, 2024. doi: 10.1109/TRO.2024.3431988.

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demonstration efficiency in reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2405.03379.

Haoxu Wang and David Meger. Robotic object manipulation with full-trajectory gan-based imitation learning. In *2021 18th Conference on Robots and Vision (CRV)*, pp. 57–63, 2021. doi: 10.1109/CRV52889.2021.00016.

Hongmin Wu, Wu Yan, Zhihao Xu, Shuai Li, and Xuefeng Zhou. Learning robot anomaly recovery skills from multiple time-driven demonstrations. *Neurocomputing*, 464:522–532, 2021. doi: 10.1016/j.neucom.2021.08.036.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. URL https://arxiv.org/abs/2107.09645.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2403.03954.

Jiyao Zhang, Zimu Han, Junhan Wang, Xionghao Wu, Shihong Lin, Jinzhou Li, Hongwei Fan, Ruihai Wu, Dongjiang Li, and Hao Dong. Hipolicy: Hierarchical multi-frequency action chunking for policy learning. In *Computer Vision – ECCV 2026*, 2026. doi: 10.1007/978-3-032-37447-9_20. URL https://arxiv.org/abs/2604.06067.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. URL https://arxiv.org/abs/2304.13705.

A Additional Related Work

Action-Chunk Policies for Imitation Learning Action-chunk policies predict multiple future targets for receding-horizon execution. Action Chunking with Transformers (ACT) (Zhao et al., 2023) predicts chunks from visual observations, while Diffusion Policy (Chi et al., 2023) models action sequences with conditional denoising diffusion. 3D Diffusion Policy (DP3) (Ze et al., 2024) conditions this formulation on compact point-cloud and proprioceptive features. We use DP3 as our backbone for its geometric and proprioceptive inputs, keeping its architecture fixed to isolate our data augmentations: reversing demonstrations across paired task directions and sampling observation–action windows at varied temporal strides and offsets.

Other methods improve demonstration efficiency or robustness by learning from a single demonstration (Johns, 2021), synthesizing trajectories with GANs (Wang & Meger, 2021), or perturbing MPC-generated data (Tagliabue & How, 2024). DrQ-v2 and DrAC augment visual-control data or select augmentations automatically in reinforcement learning (Yarats et al., 2021; Raileanu et al., 2020). These methods alter observations, state–action coverage, or trajectory content; our method instead changes the temporal indices used to form imitation-learning windows.

Temporal Frequency and Multi-Rate Action Generation Temporal adaptation has been studied for robot-assisted rehabilitation imitation (Dong et al., 2025), time-driven demonstrations for robot anomaly recovery (Wu et al., 2021), and temporal abstraction in offline RL (Ayyagari et al., 2024). HiPolicy (Zhang et al., 2026) predicts chunks at multiple frequencies, fuses frequency-aligned histories, and selects execution frequency using an entropy-based rule. FLASH (Bai et al., 2026) combines sparse temporal sampling with a continuous Legendre-polynomial action representation for long-horizon flow matching; Proleptic Temporal Ensemble (Park et al., 2025) changes inference-time temporal aggregation to accelerate execution. These methods address different tasks or alter policy representations and inference. Our generic data-level approach changes training-window sampling and supplies the temporal offsets to an existing action-chunk policy, without frequency-specific branches or an entropy-based selector; it improves results under fixed and randomized schedules.

B Additional Experimental Details

This section reports the DP3 configuration and the temporal-offset settings used in our experiments. All methods use DP3 as the policy backbone; the table separates its model and optimization settings from offsets used by augmented variants.

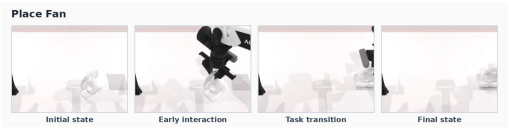
For completeness, Algorithm 1 gives the combined training and inference procedure.

B.1 Task Suite

The experimental suite contains sixteen RoboTwin bimanual manipulation tasks. We select the forward tasks from RoboTwin 2.0 and construct the reverse task for each pair by applying the trajectory-reversal procedure to its recorded demonstrations. Figure 3 shows expert demonstrations for all sixteen task directions; each sequence contains four frames from the initial state to the final state. The task-pair correspondences are listed in Table 4.

B.2 Effects of the Proposed Techniques

Table 5 reports per-task and average results for the data-efficiency comparison. The effects of the two augmentations vary across tasks. At $n = 25$, trajectory reversal improves Place Empty Cup from 47% to 83%, whereas TSAO reaches 71%. For Open Laptop, TSAO reaches 89%, compared with 81% for trajectory reversal and 70% for the baseline. At $n = 50$, trajectory reversal improves Place Fan from 34% to 58%, but reduces Unhang Mug from 96% to 76%. These task-level differences help explain why reversal yields little aggregate gain at this budget.



Place Fan. Place the fan at its target.



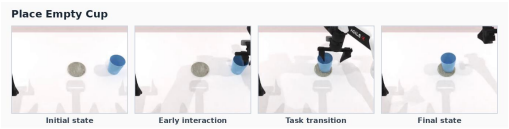
Remove Fan. Move the fan away from its target.



Place Object Stand. Place the stand at its target.



Remove Object Stand. Move the stand away from its target.



Place Empty Cup. Place the cup at its target.



Remove Empty Cup. Move the cup away from its target.



Hang Mug. Hang the mug on the fixture.



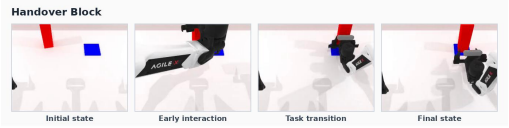
Unhang Mug. Remove the mug from the fixture.



Rotate QR Code. Rotate the board to its target orientation.



Rotate QR Code Back. Return the board toward its initial orientation.



Handover Block. Transfer the block between grippers.



Handover Block Back. Transfer the block in reverse.



Open Laptop. Raise the laptop display.



Close Laptop. Lower the laptop display.



Open Microwave. Open the microwave door.



Close Microwave. Close the microwave door.

Figure 3: Four-frame expert demonstrations for all sixteen tasks, arranged as eight paired rows with the two task directions side by side.

Table 3: Model, optimization, and temporal-offset settings. The settings are shared across data budgets; augmentation is enabled only for the corresponding method variants.

Component	Setting
Inputs	1,024 point-cloud points; 14-D robot state and action
Chunking	3 observation steps; horizon 8; predict and execute 6 action targets
DP3 backbone	FiLM-conditioned U-Net with channel widths 512, 1024, and 2048; 128-D point-cloud feature
Diffusion	100 training steps with the squared-cosine schedule; 10 DDIM inference steps
Optimizer	AdamW; learning rate 10^{-4}; weight decay 10^{-6}; betas $(0.95, 0.999)$
Learning-rate schedule	Cosine decay with 500 warm-up steps
Batching	Batch size 256; one gradient-accumulation step
Training	3,000 epochs; seed 0; 2% validation split; EMA enabled with maximum decay 0.9999
Training action strides	For temporal-stride variants, independently sample each of the 6 action gaps uniformly from $\{1, 2, 3\}$
Training observation offsets	TSAO and RESTA independently sample 2 observation skip counts from $\{0, 1, 2\}$; TSA retains consecutive observations
Offset conditioning	TSA conditions on 12 action-offset values (6 gaps and 6 cumulative gaps); TSAO and RESTA add the 2 observation skips for a 14-D input
Inference schedule	The `rand` schedule samples one action gap uniformly from $\{1, 2, 3\}$ per chunk and applies it to all 6 targets; TSAO and RESTA derive the next observation spacing from the executed action gaps
Evaluation	100 episodes per task; report success rate and average task-level success

Algorithm 1 RESTA algorithm

Require: Paired datasets $\mathcal{D}^+, \mathcal{D}^-$, history length L, action horizon $H \geq L$, stride set $\mathcal{B}$

procedure TRAIN($\kappa \in \{+, -\}$)
  Form $\widetilde{\mathcal{D}}^{\kappa}$ using Equation (9)
  while training **do**
    Sample $\tau \in \widetilde{\mathcal{D}}^{\kappa}$ and valid anchor t
    Randomly sample $\boldsymbol{\delta}^o \in \mathcal{B}^{L-1}$ and $\boldsymbol{\delta}^a \in \mathcal{B}^H$ independently
    Construct η, h_t^η, and A_t^η using Equations (10) and (12)
    Compute c_t^η using Equation (13)
    Update θ with Equation (16)
  end while
end procedure
procedure INFER(π_θ, σ)
  Initialize history, its observation offsets, and $q \leftarrow 0$
  while task execution continues **do**
    Select adjacent action offsets $\boldsymbol{\delta}_q^a \in \mathcal{B}^H$ using σ
    Compute target indices from $\boldsymbol{\delta}_q^a$ using Equation (14)
    Set $\eta_q \leftarrow [\boldsymbol{\delta}_q^o, \boldsymbol{\delta}_q^a]$
    Predict an action chunk conditioned on history and η_q
    Execute the chunk and retain the last L observations
    $q \leftarrow q + 1$
    Update observation offsets using Equation (15)
  end while
end procedure

B.3 TEMPORAL SCHEDULE SELECTION

The sweep compares seven schedulers for six-target chunks, with adjacent action gaps from $\{1, 2, 3\}$. Their cumulative sums give target distances from the anchor while keeping chunk length

Table 4: Forward tasks selected from RoboTwin 2.0 and their constructed reverse-task counterparts. Direction: not applicable; this table lists task correspondences.

Task pair	Forward task	Reverse task
Fan manipulation	Place Fan	Remove Fan
Object-stand manipulation	Place Object Stand	Remove Object Stand
Empty-cup manipulation	Place Empty Cup	Remove Empty Cup
Mug manipulation	Hang Mug	Unhang Mug
QR-code manipulation	Rotate QR Code	Rotate QR Code Back
Block handover	Handover Block	Handover Block Back
Laptop manipulation	Open Laptop	Close Laptop
Microwave manipulation	Open Microwave	Close Microwave

Table 5: Per-task data-efficiency comparison at 3,000 training epochs. B_n is DP3 trained with n native trajectories per task. TR_n adds n time-reversed trajectories, while TSAO_n uses only the n native trajectories with temporal-stride augmentation and observation-offset conditioning. $RESTA_n$ combines trajectory reversal with this augmentation. TSAO and $RESTA_n$ use the `rand` inference schedule. B_{100} is the larger-data reference and is not highlighted. Bold text with blue shading indicates the best result at each current budget. Values are success rates (%). Higher is better.

Task	B_{25}	TR_{25}	TSAO_{25}	$RESTA_{25}$	B_{50}	TR_{50}	TSAO_{50}	$RESTA_{50}$	B_{100}
Place Fan	23	34	30	**43**	34	**58**	38	55	65
Remove Fan	87	82	**100**	96	92	85	**100**	99	99
Place Object Stand	43	**61**	50	58	61	72	67	**79**	80
Remove Object Stand	64	67	68	**75**	82	80	**85**	76	87
Place Empty Cup	47	**83**	71	77	77	74	85	**91**	93
Remove Empty Cup	91	94	95	**97**	90	98	**99**	**100**	100
Hang Mug	19	**27**	21	25	24	28	28	**36**	31
Unhang Mug	79	77	**82**	72	**96**	76	**96**	85	94
Rotate QR Code	55	55	65	**80**	69	78	79	**88**	75
Rotate QR Code Back	99	98	**100**	**100**	**100**	**100**	99	99	98
Handover Block	86	**92**	86	**92**	89	94	**97**	96	97
Handover Block Back	99	**100**	98	**100**	97	97	98	**99**	99
Open Laptop	70	81	89	**93**	85	81	**97**	**97**	86
Close Laptop	73	61	**79**	69	88	73	93	92	88
Open Microwave	72	83	86	**96**	80	73	95	**98**	92
Close Microwave	73	77	74	**82**	73	69	**75**	74	83
Average(%)	67.5	73.2	74.6	**78.4**	77.3	77.2	83.2	**85.2**	85.4

fixed. The fixed schedules `offset1`, `offset2`, and `offset3` repeat gaps of one, two, and three, spanning 6, 12, and 18 steps. `rand` samples one gap per chunk, whereas `rand_vector` samples each of the six gaps independently.

The voting schedulers evaluate candidate gap vectors and execute the candidate chunk closest to the mean predicted action chunk. `vote3` uses

$$\mathcal{S}_3^{\mathrm{vote}} = \{(1,1,1,1,1,1), (2,2,2,2,2,2), (3,3,3,3,3,3)\}, \tag{17}$$

and `vote11` expands this set with eight candidates:

$$\begin{aligned}\mathcal{S}_{11}^{\mathrm{vote}} = \mathcal{S}_3^{\mathrm{vote}} \cup \{&(1,2,3,1,2,3), (3,2,1,3,2,1),\\ &(1,3,2,1,3,2), (2,1,3,2,1,3),\\ &(1,1,2,2,3,3), (3,3,2,2,1,1),\\ &(1,2,2,3,3,2), (3,2,2,1,1,2)\}.\end{aligned} \tag{18}$$

The eight added `vote11` vectors test reordered three-gap cycles, progressive coarsening or refinement, and nonmonotonic spacing. The first six span 12 steps, matching `offset2`; the final two

Table 6: Per-task success rates for the inference-schedule sweep in the $n = 50$ setting, where n is the number of native trajectories per task. The baseline is DP3; the remaining columns use TSAO and the indicated inference schedule. Bold and blue shading indicate the best result for each task or average across both blocks. Values are success rates (%). Higher is better.

Task	B_{50}	offset1	offset2	offset3
Place Fan	34	33	34	32
Remove Fan	92	**100**	**100**	**100**
Place Object Stand	61	66	**67**	63
Remove Object Stand	82	78	84	82
Place Empty Cup	77	79	85	**87**
Remove Empty Cup	90	97	97	98
Hang Mug	24	**35**	27	27
Unhang Mug	**96**	96	96	95
Rotate QR Code	69	71	80	74
Rotate QR Code Back	**100**	**100**	99	**100**
Handover Block	89	95	93	93
Handover Block Back	97	97	99	**100**
Open Laptop	85	86	92	96
Close Laptop	88	92	**93**	92
Open Microwave	80	80	96	96
Close Microwave	73	**89**	73	71
Average(%)	77.3	80.9	82.2	81.6

Task	rand	rand_vector	vote3	vote11
Place Fan	**38**	36	33	35
Remove Fan	**100**	**100**	**100**	**100**
Place Object Stand	**67**	**67**	66	65
Remove Object Stand	**85**	81	81	**85**
Place Empty Cup	85	85	**86**	85
Remove Empty Cup	**99**	97	98	98
Hang Mug	28	**30**	21	26
Unhang Mug	**96**	**96**	95	90
Rotate QR Code	79	78	79	**83**
Rotate QR Code Back	99	98	98	99
Handover Block	**97**	93	94	92
Handover Block Back	98	99	98	98
Open Laptop	**97**	95	92	92
Close Laptop	**93**	91	92	**93**
Open Microwave	95	94	**98**	96
Close Microwave	**75**	70	**75**	73
Average(%)	**83.2**	81.9	81.6	81.9

span 13 and 11 steps. Together with the three constant vectors, they form a compact, non-exhaustive subset of the 3^6 possible schedules. The executed action gaps set the next cycle's observation offsets. Comparing vote3 and vote11 tests agreement-based selection, not task success directly; individual vote11 candidates were not ablated.

The preferred fixed gap varies by task: changing it from one to three raises Open Laptop success from 86% to 96%, but lowers Close Microwave from 89% to 71% (Table 6). At $n = 50$, rand has the highest average success (83.2%), versus 81.9% for rand_vector, and is used as the default (Table 6). This favors varying the scale between chunks while keeping regular spacing within each chunk in this evaluation. vote11 reaches 81.9%, up from 81.6% for vote3, but remains below rand.

Table 7: Per-task effect of temporal-stride augmentation and observation-offset conditioning at two trajectory budgets. The TSA and TSAO entries are the selected `rand` inference results from the corresponding budget-specific evaluations. For each $n \in \{25, 50\}$, B_n is the baseline, TSA_n uses temporal-stride augmentation with consecutive observations, and TSAO_n additionally conditions on observation offsets. Bold text with blue shading indicates the best result within each budget for each task or average across tasks. Values are success rates (%). Higher is better.

Task	B_{25}	TSA_{25}	TSAO_{25}	B_{50}	TSA_{50}	TSAO_{50}
Place Fan	23	18	**30**	34	30	**38**
Remove Fan	87	**100**	**100**	92	**100**	**100**
Place Object Stand	43	**55**	50	61	**67**	**67**
Remove Object Stand	64	66	**68**	82	81	**85**
Place Empty Cup	47	69	**71**	77	80	**85**
Remove Empty Cup	91	**95**	**95**	90	97	**99**
Hang Mug	19	17	**21**	24	22	**28**
Unhang Mug	79	79	**82**	**96**	95	**96**
Rotate QR Code	55	61	**65**	69	76	**79**
Rotate QR Code Back	99	96	**100**	**100**	99	99
Handover Block	**86**	85	**86**	89	93	**97**
Handover Block Back	99	**100**	98	97	**99**	98
Open Laptop	70	87	**89**	85	90	**97**
Close Laptop	73	67	**79**	88	88	**93**
Open Microwave	72	**87**	86	80	**95**	**95**
Close Microwave	73	**76**	74	73	**81**	75
Average(%)	67.5	72.4	**74.6**	77.3	80.8	**83.2**

B.4 Temporal-stride Ablation Details

The action schedule also determines the temporal spacing represented by the next observation history, as specified in Equation (15). This motivates training with varied observation spacings and conditioning on their offsets: the policy must interpret histories produced by different preceding action schedules. Table 7 compares TSA and TSAO using the selected `rand` schedule at both trajectory budgets. TSAO raises average success from 72.4% to 74.6% at $n = 25$ and from 80.8% to 83.2% at $n = 50$. The improvement is not uniform across tasks or budgets, supporting the combined use of randomized observation spacings and offset conditioning on average; this comparison does not isolate their individual effects.

B.5 Execution-Step Results

The following tables provide per-task execution-step measurements for the baseline and RESTA checkpoints, including policy actions and simulator physics steps. RESTA uses the `rand` schedule; values are means over ten successful episodes per task.

Table 8: Mean policy actions per successful episode for baseline and RESTA checkpoints on the 16-task paired suite. RESTA uses the `rand` schedule; each task mean is based on all successful episodes. For each task and for the average across tasks, bold text with blue shading marks the lower value within each $n = 25$ and $n = 50$ baseline/RESTA pair. Lower is better.

Task	B_{25}	RESTA_{25}	B_{50}	RESTA_{50}	B_{100}
Place Fan	131.5	**71.9**	143.8	**68.7**	139.3
Remove Fan	140.1	**70.4**	135.9	**65.9**	117.9
Place Object Stand	132.9	**86.7**	129.9	**64.9**	130.9
Remove Object Stand	159.2	**60.9**	131.8	**63.2**	112.7
Place Empty Cup	149.6	**63.0**	139.8	**58.0**	144.0
Remove Empty Cup	135.5	**63.4**	142.1	**65.9**	119.3
Hang Mug	306.1	**143.2**	333.0	**151.6**	267.6
Unhang Mug	335.5	**156.5**	310.4	**172.7**	289.9
Rotate QR Code	163.7	**105.8**	290.9	**71.7**	130.3
Rotate QR Code Back	151.3	**75.7**	275.1	**68.9**	133.7
Handover Block	270.8	**128.3**	144.1	**137.6**	236.0
Handover Block Back	295.0	**133.5**	148.2	**132.0**	255.5
Open Laptop	114.0	**60.5**	158.1	**67.5**	117.7
Close Laptop	172.4	**101.5**	128.6	**84.5**	149.1
Open Microwave	624.6	**186.5**	527.5	**215.7**	588.8
Close Microwave	203.2	**97.8**	208.1	**100.1**	209.3
Average	217.8	**100.3**	209.2	**99.3**	196.4

Table 9: Mean simulator physics steps per successful episode for baseline and RESTA checkpoints on the 16-task paired suite. Physics steps include low-level interpolation. RESTA uses the `rand` schedule; each task mean is based on all successful episodes. For each task and for the average across tasks, bold text with blue shading marks the lower value within each $n = 25$ and $n = 50$ baseline/RESTA pair. Lower is better.

Task	B_{25}	RESTA_{25}	B_{50}	RESTA_{50}	B_{100}
Place Fan	10734.4	**7908.4**	10897.0	**7406.1**	10494.5
Remove Fan	11734.1	**5995.5**	9287.3	**6789.6**	9386.8
Place Object Stand	10198.1	**8883.6**	9574.5	**6259.4**	9130.2
Remove Object Stand	13009.2	**6246.6**	9436.0	**6097.9**	8730.4
Place Empty Cup	9728.8	**5878.8**	8935.4	**5460.3**	8629.3
Remove Empty Cup	9251.7	**5984.6**	9447.8	**6354.0**	8864.5
Hang Mug	23211.5	**15110.3**	23636.2	**15185.0**	21226.0
Unhang Mug	24263.4	**16084.7**	23053.7	**18252.1**	22690.2
Rotate QR Code	14794.1	**12491.0**	18528.5	**8239.4**	11635.6
Rotate QR Code Back	13078.9	**8950.1**	18666.3	**8260.9**	12341.6
Handover Block	19377.1	**12323.4**	**11801.3**	13266.0	17585.8
Handover Block Back	19259.8	**12444.9**	12257.0	**12084.2**	17652.0
Open Laptop	7905.4	**5171.2**	8361.7	**5915.5**	7781.5
Close Laptop	12146.0	**8367.9**	9881.8	**7105.3**	12231.5
Open Microwave	18580.9	**10211.3**	16921.9	**10920.0**	16856.2
Close Microwave	10746.9	**7339.9**	10886.1	**7420.7**	10833.7
Average	14251.3	**9337.0**	13223.3	**9063.5**	12879.4