CHUNK-ALIGNED RECOVERY SYNTHESIS FOR ROBUST GENERATIVE CONTROL POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative control policies, such as diffusion and flow-matching policies, have demonstrated strong performance in behavior cloning from multimodal demonstrations, but remain susceptible to covariate shift: small action errors accumulate over time, driving the policy into states outside the demonstration distribution, where its predictions become unreliable. Existing corrective-augmentation methods address this issue by synthesizing recovery data as isolated transitions. Such transitions are poorly suited to action-chunking policies, which are trained to produce temporally coherent sequences rather than individual corrective steps. We propose ReSynC (**R**eciprocal **Syn**thesis and Cycle **C**onsistency), which synthesizes recovery behavior in a chunk-aligned form: a truncated backward diffusion policy and a backward dynamics model generate a plausible deviation together with its recovery, and this segment replaces the prefix of an expert action chunk, so that the resulting training sample starts from an off-distribution state and ends in expert behavior. Chunk-aligned placement and an expressive backward generator interact: neither provides a consistent benefit without the other. At inference time, ReSynC reuses these dynamics models to score candidate action chunks by their cycle-consistency error and contrastively selects among the low-error candidates. We find that this score reflects whether a candidate remains within the data distribution used to fit the dynamics, rather than how accurate those dynamics are, and that this signal remains reliable under limited demonstrations. Across five simulated manipulation tasks and a physical robot arm, ReSynC improves average robustness under limited demonstrations without additional human supervision or environment interaction.

1 INTRODUCTION

Imitation learning (IL) for robotics often requires modeling diverse and multimodal data distributions (Lin et al., 2024; Urain et al., 2024). Recent advances in generative modeling, particularly diffusion models (Ho et al., 2020; Song et al., 2020) and flow matching (Lipman et al., 2022), have substantially improved IL, enabling robots to tackle increasingly complex and long-horizon manipulation tasks (Chi et al., 2023; Ze et al., 2024; Team et al., 2024; Black et al., 2024; Bjorck et al., 2025). Despite these advances, generative control policies remain vulnerable to states outside the demonstration distribution, often resulting in unpredictable or catastrophic behavior. This fragility can arise from sensor noise, stochastic environments, and external disturbances, which cause small execution errors to accumulate and induce severe covariate shift (Ross et al., 2011; De Haan et al., 2019). Ensuring robustness to such distributional shift is therefore critical for the reliable deployment of generative policies in real-world applications.

Human-in-the-loop approaches, such as DAgger-style (Chernova & Veloso, 2009; Ross et al., 2011; Kelly et al., 2019) and DART-style methods (Laskey et al., 2017; Zhang et al., 2025b), mitigate drift by allowing human experts to intervene and provide corrective feedback during execution. Alternatively, synthetic data augmentation methods exploit task-specific structure to expand the training dataset (Mandlekar et al., 2023; Zhou et al., 2023). While promising, both approaches face practical limitations. Interactive methods are often labor-intensive and constrained by human availability, whereas heuristic-driven augmentation relies heavily on domain-specific assumptions, limiting its generalizability. Achieving scalable and robust policy learning therefore calls for methods that move beyond manual intervention and synthesize recovery behaviors from existing offline demonstrations, without requiring additional human supervision or environment interaction.

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.

To this end, we propose **R**eciprocal **Syn**thesis and Cycle **C**onsistency (ReSynC), a unified framework that improves the robustness of generative control policies to inference-time deviations by combining proactive training with reactive inference. During training, we synthesize corrective behaviors using a backward diffusion policy and a backward dynamics model. To preserve the temporal structure expected by the base policy, we use a trajectory-stitching strategy that prepends imagined deviations and their corresponding recovery actions to expert demonstrations. We call this data augmentation procedure *Reciprocal Synthesis*, which explicitly trains the policy to recover from unfamiliar states back toward the expert distribution. At inference time, ReSynC complements this proactive training with a reactive sampling mechanism that helps prevent divergence. Generative policies naturally produce multiple candidate action chunks from different initial noise samples. We leverage the learned forward and backward dynamics models to compute a *Cycle-Consistency* error for each candidate and use contrastive selection to favor candidates with low error over those with high error. This allows the policy to select action chunks that remain consistent with the learned data distribution without requiring additional environment rollouts. We evaluate ReSynC across robotic manipulation tasks and find that it improves average performance over existing data-augmentation and test-time steering baselines, with substantial gains in long-horizon robustness under limited demonstrations.

In summary, our contributions are threefold: (i) We identify why corrective augmentation transfers poorly to chunk-based generative policies: isolated transitions discard temporal structure. (ii) We introduce *Reciprocal Synthesis*, which generates chunk-aligned recovery data using a backward diffusion policy and dynamics model; a factorial analysis shows that expressive generation and chunk-aligned placement are effective only together. (iii) We introduce contrastive *Cycle-Consistency* selection, which reuses these dynamics models at inference time and provides a reliable signal of alignment with the training distribution under limited data.

2 Related Work

Mitigating Covariate Shift in Imitation Learning. Compounding error, driven by covariate shift, is one of the main challenges in behavior cloning, where small deviations accumulate and push the agent into out-of-distribution (OOD) states (Ross et al., 2011; De Haan et al., 2019). When online querying is possible, interactive methods provide corrective labels (Chernova & Veloso, 2009; Ross et al., 2011; Kelly et al., 2019; Laskey et al., 2017; Zhang et al., 2025b). In strictly offline settings, conservatism is instead enforced through penalties based on empirical transition dynamics (Yu et al., 2020; Kidambi et al., 2020; Chang et al., 2021; Srinivasan & Knottenbelt, 2024) or by correcting for stationary distribution shift (Seo et al., 2024). Expressive generative policies capture multimodal behavior (Florence et al., 2022; Chi et al., 2023) but remain vulnerable over long horizons, motivating test-time adaptation and sampling strategies that steer action selection without retraining (Wagenmaker et al., 2025; Römer et al., 2025; Kwok et al., 2025; Qi et al., 2026). Rather than altering the training objective, we select actions at test time using cycle-consistency error from learned forward and backward dynamics, leaving the base policy's weights unchanged.

Data Augmentation in Imitation Learning. Corrective augmentation synthesizes transitions that return deviated states toward the expert distribution. Early work perturbed states with uniform noise (Laskey et al., 2017), which can violate physical constraints in high-dimensional domains (Ke et al., 2021); later methods generate restorative transitions by recombining trajectory segments (Hepburn & Montana, 2024; Wang et al., 2025), modeling backward reachability (Park & Wong, 2022), or exploiting local Lipschitz continuity with trust-region filtering (Ke et al., 2023). In offline RL, ROMI (Wang et al., 2021) and CABI (Lyu et al., 2022) rely on learned reverse dynamics, while RTDiff (Yang & Wang, 2025) and DiffStitch (Li et al., 2024) apply diffusion for reverse generation and trajectory stitching. Unlike prior work, we replace the prefix of an expert chunk with a multi-step recovery segment, matching the sequence format chunk-based policies consume. Moreover, while these methods target reward- or value-based offline RL, we study reward-free imitation learning, where corrective-data placement is consequential. Finally, we reuse the forward and backward dynamics at inference as a verifier; CABI is the closest antecedent, but uses this consistency check to filter imagined rollouts before adding them to the buffer rather than selecting executed actions.

Inference-Time Policy Sampling and Guidance. Test-time computation can improve generative models without retraining (Kwok et al., 2025). In vision and language, steering guides generation

toward high-likelihood regions (Ho & Salimans, 2022), while contrastive decoding reduces hallucination (Leng et al., 2024; Wang et al., 2024; Park et al., 2025). In robotics, the denoising process has been guided by external dynamics (Du & Song, 2025; Wagenmaker et al., 2025), barrier functions (Sun & Song, 2025), or world models (Qi et al., 2026), with candidates assessed using predictive foresight (Qi et al., 2026) or failure classifiers (Römer et al., 2025). Related work on vision-language-action models uses contrastive sampling (Soh & Lim, 2026) and test-time verification (So et al., 2025; Kwok et al., 2025). Such guidance is sensitive to the guidance scale and depends on an accurate world model. We instead score candidate chunks using the cycle-consistency error of the dynamics models already trained in Stage 1, so no model is introduced solely for inference and the denoising process is left untouched.

3 PRELIMINARIES

Behavior Cloning. We model the environment as a Markov decision process (MDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$, governed by transition dynamics $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$. Given a demonstration dataset $\mathcal{D} = \{(s_t, a_t)\}_t$, behavior cloning (BC) learns a policy π_θ that imitates expert behavior by maximizing $\mathbb{E}_{\mathcal{D}}[\log \pi_\theta(A_t \mid S_t)]$. Following Chi et al. (2023), the policy $\pi_\theta(A_t \mid S_t)$ conditions on the most recent T_o states $S_t = \{s_{t-T_o+1}, \ldots, s_t\}$ and predicts T_p future actions $A_t = \{a_t, \ldots, a_{t+T_p-1}\}$, of which the first T_a are executed before replanning. Because $\mathcal{D}$ covers only a subset of $\mathcal{S}$, a policy that deviates from its support encounters states it was never trained on, and the resulting errors accumulate over the remainder of the episode (Ross et al., 2011).

Diffusion Policy. We parameterize π_θ with a diffusion model (Chi et al., 2023), referred to as a diffusion policy (DP). A fixed *forward process* injects noise into an action chunk, and a learned *reverse process*, parameterized by $\epsilon_\theta(A_t^k, S_t, k)$ and conditioned on S_t, recovers the clean chunk.[1] Training minimizes the denoising score matching loss (Ho et al., 2020):

$$\mathcal{L}_{DP}(\theta) = \mathbb{E}_{(A_t,S_t)\sim\mathcal{D},\epsilon\sim\mathcal{N}(0,I),k\sim\mathcal{U}(1,K)}\left[\|\epsilon - \epsilon_\theta(A_t^k, S_t, k)\|^2\right], \tag{1}$$

where $A_t^k = \sqrt{\bar{\alpha}_k} A_t + \sqrt{1-\bar{\alpha}_k} \cdot \epsilon$ is the action chunk noised to diffusion step k under the standard DDPM schedule (Ho et al., 2020). At test time, sampling begins from $A_t^K \sim \mathcal{N}(0, I)$ and denoises to the final action chunk A_t^0.

4 METHOD

Reciprocal **Syn**thesis and Cycle **C**onsistency (ReSynC) has two stages: a training stage in which the base diffusion policy is trained on corrective data synthesized with learned dynamics models, and a test-time stage in which cycle-consistency is used to select among candidate action chunks. Figure 1 gives an overview.

4.1 GENERATIVE BEHAVIOR CLONING WITH RECIPROCAL SYNTHESIS

To endow our base diffusion policy with the ability to recover from execution errors, we augment the training data with synthetic corrective segments. Following prior work (Ke et al., 2021; Park & Wong, 2022; Ke et al., 2023), the goal is to construct transitions that map deviated states back to the expert distribution. These methods were designed for policies that consume individual transitions, and that assumption becomes limiting once the base policy predicts action chunks. Corrective transitions are synthesized independently and inserted uniformly across states in the dataset, so a policy trained on length-H sequences receives chunks whose recovery steps neither follow from one another nor connect to the expert behavior that succeeds them. A second limitation compounds this one. Generating a recovery segment that remains coherent over multiple steps requires a backward model able to represent the several distinct predecessor states consistent with a given anchor; a unimodal Gaussian backward policy averages over these possibilities and produces physically inconsistent segments. The two cannot be addressed separately: as our factorial analysis shows (Table 2),

[1] In this work, we use superscripts ($k \in \{0, 1, \ldots, K\}$) to denote diffusion timesteps and subscripts ($t \in \{0, 1, \ldots, T\}$) to denote trajectory timesteps.

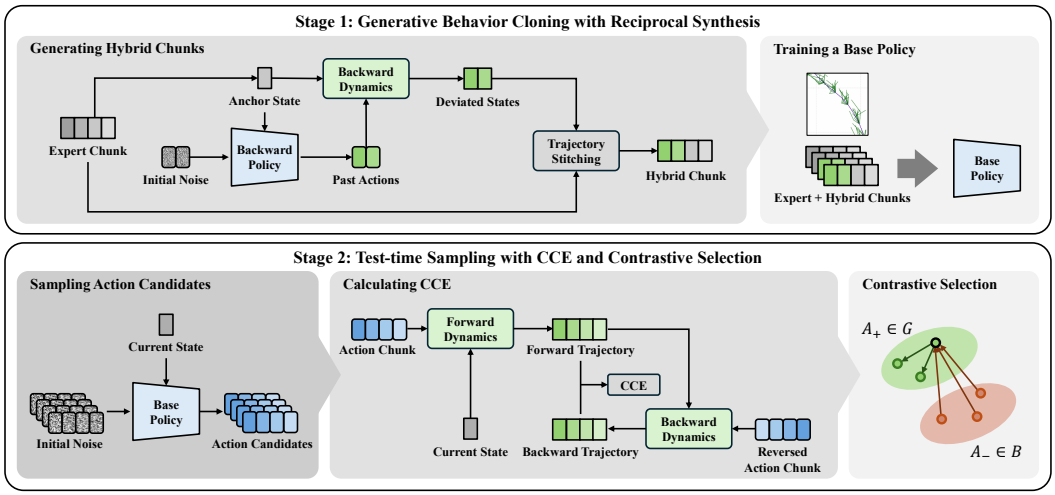


Figure 1: **Reciprocal Synthesis and Cycle Consistency (ReSynC)**: 1) During training, we train a base diffusion policy on expert demonstration data and synthetic corrective data, which is generated using a backward diffusion policy and a backward dynamics model. 2) During inference, ReSynC encourages the agent remain within expert distribution by sampling different action chunk candidates and selecting the candidate that is most likely align with the expert distribution by using cycle-consistency error coupled with contrastive selection.

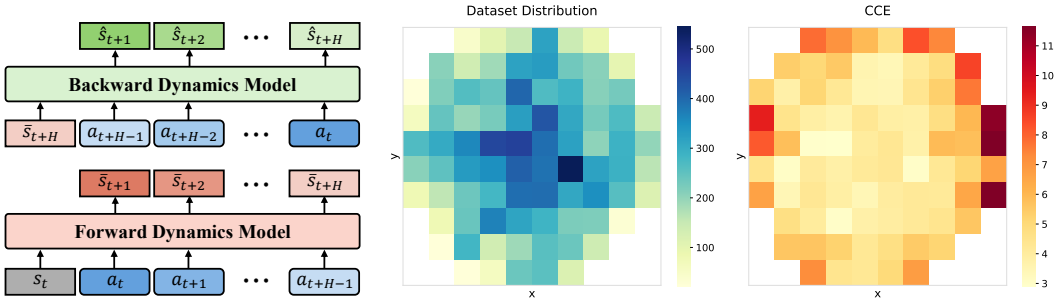
addressing either alone yields little benefit ($+3.4$ and -0.1 points on average), and improvement appears only when coherent placement is combined with an expressive backward generator.

To synthesize plausible and temporally coherent recovery behaviors, we introduce a dual-component backward generation pipeline. First, we parameterize a backward policy $\mu_\omega(\hat{A}_{\text{past}} \mid s^*)$ as a diffusion model, which is trained to predict an anti-chronological sequence of actions $\hat{A}_{\text{past}}$ that immediately precede an expert anchor state $s^* \sim \mathcal{D}$. During action generation, we perform a truncated reverse diffusion process to yield partially denoised actions. This design has a practical benefit: it enables the efficient generation of diverse, perturbed actions while training the base policy, while ensuring that these actions remain close to the valid action manifold of the dataset. Second, a causal Transformer-based backward dynamics model $\mathcal{B}_\psi$ maps these noisy actions into the state space, predicting the corresponding sequence of past deviated states $\hat{S}_{\text{past}} = \mathcal{B}_\psi(s^*, \hat{A}_{\text{past}})$ in anti-chronological order.

During base policy training, we use trajectory stitching to integrate these synthesized segments. Concretely, given a sampled H-length expert chunk $(s_0, a_0, \ldots, s_{H-1}, a_{H-1}) \sim \mathcal{D}$, we designate an early state at index j as the anchor s^* and substitute all preceding states and actions (*i.e.*, from step 0 to $j-1$) with the synthesized $\hat{S}_{\text{past}}$ and $\hat{A}_{\text{past}}$. This yields a hybrid chunk $(\hat{s}_0, \hat{a}_0, \ldots, s_j, a_j, \ldots, s_{H-1}, a_{H-1})$, prepending synthesized recovery actions to the expert trajectory to ensure temporal consistency. The base policy π_θ is then trained on both the original chunk and the hybrid chunk using Eq. (1), thereby encouraging the agent to recover to expert-like states when encountering deviations.

4.2 Test-time Sampling

Cycle-Consistency Error for Uncertainty Estimation. Recent work suggests that the specific initial Gaussian noise sampled during the action generation process of generative control policies, including diffusion (Chi et al., 2023) and flow-matching (Zhang et al., 2025a), can cause the resulting actions to diverge from the expert demonstrator's distribution during inference (Wagenmaker et al., 2025; Soh & Lim, 2026), leading to compounding errors. We therefore introduce the cycle-consistency error (CCE), computed from our learned dynamics, to rank a batch of candidate chunks generated from different initial noise samples. Let $\mathcal{F}_\phi$ denote a forward dynamics model trained to predict future states given a current state s and a sequence of actions A. At inference time, given the current



(a) Dynamics models (b) Estimation of CCE

Figure 2: **Dynamics modeling and Cycle-Consistency Error (CCE) estimation**. (a) The causal Transformer-based forward and backward dynamics models used in ReSynC. (b) Heatmaps illustrating the dataset distribution (left) and the corresponding estimated CCE (right) for the Push-T task. The dataset is projected to 2D using t-SNE. The visualization shows that CCE inversely correlates with data density, which helps penalize OOD states.

state s_t and a candidate action chunk A_t, the CCE is computed as:[2]

$$\mathcal{L}_{\text{CCE}}(A_t) = \frac{1}{T_a} \sum_{i=0}^{T_a-1} \rho^i \|\bar{s}_{t+i} - \hat{s}_{t+i}\|_2. \tag{2}$$

Here, $\rho \in [0, 1]$ is a decay hyperparameter that mitigates the accumulating uncertainty over time. The predicted forward trajectory $(\bar{s}_{t+1}, \ldots, \bar{s}_{t+T_a})$ is obtained from the forward dynamics model $\mathcal{F}_\phi(s_t, A_t)$, and the predicted backward trajectory $(\hat{s}_{t+T_a-1}, \ldots, \hat{s}_t)$ is obtained from the backward dynamics model $\mathcal{B}_\psi(\bar{s}_{t+T_a}, \texttt{rev}(A_t))$, where $\texttt{rev}(\cdot)$ reverses the sequence order. Since both dynamics models are trained on expert demonstrations, a low round-trip error (Guo et al., 2022; Römer et al., 2025) indicates a candidate whose predicted rollout stays close to the demonstration data; we make this interpretation precise below. An example of CCE estimation is illustrated in Figure 2b. While prior work has explored the use of prediction errors from dynamics models as an exploration bonus (Pathak et al., 2017; 2019; Guo et al., 2022) or to encourage pessimism in offline RL (Yu et al., 2020; Kidambi et al., 2020; Srinivasan & Knottenbelt, 2024), our approach repurposes this signal to encourage generative control policies to produce in-distribution actions at inference time.

What CCE measures. Both dynamics models are trained on demonstration data, and the backward model is additionally trained to recover the true past states from the forward model's predicted endpoint (Section A). Round-trip closure is therefore optimized explicitly on the demonstration data and left unconstrained outside this distribution. CCE therefore does not test whether the learned dynamics are accurate; it tests whether a candidate chunk keeps the round trip within the region where closure was optimized. Under this interpretation, CCE is a signal about *support* rather than accuracy, with two consequences for how it should be used. First, a low CCE does not certify that a chunk is safe, only that it resembles the transitions on which the dynamics were fitted; low-CCE out-of-distribution candidates therefore remain possible, motivating the contrastive criterion below. Second, its value as a policy-level OOD signal relies on the dynamics models and the policy sharing the same support. This holds in our setting because both are trained on the same demonstrations, and explains why the mechanism does not require an accurate dynamics model: as reported in Table 13, the correlation between CCE and a model-independent OOD measure is preserved when the dynamics training set is reduced tenfold, and is strongest at the smallest training budgets.

Contrastive Selection. Near the boundary of the demonstration support, a single CCE value can be unreliable in either direction: an in-support chunk may receive a high error, while an out-of-support chunk may receive a low one. Selecting the single lowest-CCE candidate therefore places full weight on a single point estimate. To mitigate this issue, prior work often relies on predictions from ensembles of dynamics models (Yu et al., 2020; Kidambi et al., 2020; Srinivasan & Knottenbelt, 2024). However, this approach may introduce computational overhead during inference. Instead, we draw inspiration from VLM decoding techniques designed to reduce hallucinations (Leng et al.,

[2]For the boundary condition at $i = 0$, we set $\bar{s}_t = s_t$. Additionally, we restrict predictions to the T_a actions that are subsequently executed in the environment.

2024; Wang et al., 2024; Park et al., 2025) and introduce a contrastive selection criterion. Specifically, we divide the generated candidates into two reference sets: a positive set containing the half with the lowest CCE and a negative set containing the half with the highest CCE. We then restrict the final selection to the positive set, choosing the chunk that minimizes its distance to other positive samples while maximizing its distance from negative samples:

$$\mathcal{L}_{\text{contrast}}(A_t) = \frac{1}{|G|} \sum_{A_+ \in G} d(A_t, A_+) - \frac{1}{|B|} \sum_{A_- \in B} d(A_t, A_-), \tag{3}$$

where G and B denote the positive and negative sets, respectively, and $d(\cdot, \cdot)$ is the ℓ_2 distance between the action chunks.

4.3 Implementation Details

During the training stage, we first train the backward policy using the same objective as the base diffusion policy (Eq. 1). However, whereas the base policy is conditioned on a sequence of state histories, the backward policy is conditioned on a single anchor state s^* and predicts a sequence of actions in reverse temporal order. The synthetic data are generated on the fly during base policy training. The forward and backward dynamics models employ a causal Transformer-based architecture (Janner et al., 2021; Jiang et al., 2023) (illustrated in Figure 2a) and are trained jointly using a mean-squared error loss:

$$\begin{aligned} \min_{\psi,\phi} \mathbb{E}[\|\mathcal{F}_\phi(s_t, A_t) - S_t^{\mathcal{F}}\|_2^2 + \|\mathcal{B}_\psi(s_{t+T_a}, \texttt{rev}(A_t)) - S_t^{\mathcal{B}}\|_2^2 \\ + \|\mathcal{B}_\psi(\text{sg}(\bar{s}_{t+T_a}), \texttt{rev}(A_t)) - S_t^{\mathcal{B}}\|_2^2] \end{aligned} \tag{4}$$

where $S_t^{\mathcal{F}} = (s_{t+1}, \ldots, s_{t+T_a})$ and $S_t^{\mathcal{B}} = (s_{t+T_a-1}, \ldots, s_t)$ denote the ground-truth forward and backward state trajectories. The first two terms train the dynamics models using data sampled from the dataset. The third term trains the backward model using a bootstrapped prediction from the forward model, $\bar{s}_{t+T_a}$, where $\text{sg}(\cdot)$ denotes a stop-gradient operation. This explicitly encourages the backward model to learn the true dataset dynamics while remaining robust to potential prediction errors induced by the forward model. As the dynamics model is queried with noisy actions at inference time, we augment the training process by injecting Gaussian noise into the actions, following the same noise schedule of the base policy and sampling noise levels from a geometric distribution.

5 Experiments

In this section, we conduct experiments across five environments to answer the following questions: (1) How does ReSynC improve the robustness of generative control policies compared to baseline methods in a low-data regime? (2) What are the individual contributions of its core components? (3) Can ReSynC be effectively applied to real robotic tasks?

5.1 Simulation Experiments

We evaluate our method on behavioral cloning tasks across five simulated environments, including simple tasks such as Push-T (Florence et al., 2022; Chi et al., 2023), standard benchmarks from Robomimic (Mandlekar et al., 2022), and the long-horizon Kitchen environment (Gupta et al., 2020). Success rate is used as the primary metric for most tasks, while Push-T is evaluated using target area coverage. We train all policies in a limited-demonstration regime, retaining 40% of the original expert demonstrations for Tool-Hang, 5% for Kitchen, and 20% for the remaining tasks. In absolute terms this amounts to 80 demonstrations for Tool-Hang, 28 for Kitchen, 41 for Push-T, and 60 each for Square and Transport. All methods are trained solely from the demonstration dataset without collecting additional data. DP remains a functioning policy at this budget, and the improvement from ReSynC persists when all demonstrations are available (Table 12). We compare ReSynC against the following baselines:

- Diffusion Policy (DP) (Chi et al., 2023): Base diffusion policy trained exclusively on the provided expert demonstrations.
- NoisyBC (Ke et al., 2021): Augments training by injecting state perturbations while retaining original action labels, with noise applied uniformly across dataset states.

Table 1: Average performance across five tasks. We report the mean and standard deviation over three seeds, with each run evaluated on 50 held-out environment initial conditions. The best results for each task are highlighted in bold.

Task	DP	NoisyBC	BMIL	HINT	LPB	ReSynC
Push-T	60.0 ±3.7	68.8 ±3.0	62.0 ±1.6	67.5 ±5.4	62.8 ±2.7	**69.3** ±2.9
Square	68.7 ±6.6	64.7 ±4.7	70.7 ±3.4	56.0 ±5.7	**83.3** ±3.8	77.3 ±3.4
Transport	28.7 ±2.5	35.3 ±0.9	36.0 ±4.9	32.0 ±5.7	28.0 ±1.6	**40.0** ±4.3
Tool-Hang	25.3 ±5.0	10.7 ±3.8	26.7 ±3.4	30.0 ±1.6	28.0 ±4.3	**31.3** ±8.4
Kitchen	78.0 ±1.6	84.7 ±0.9	75.3 ±1.9	54.4 ±1.1	53.9 ±2.2	**90.7** ±2.5

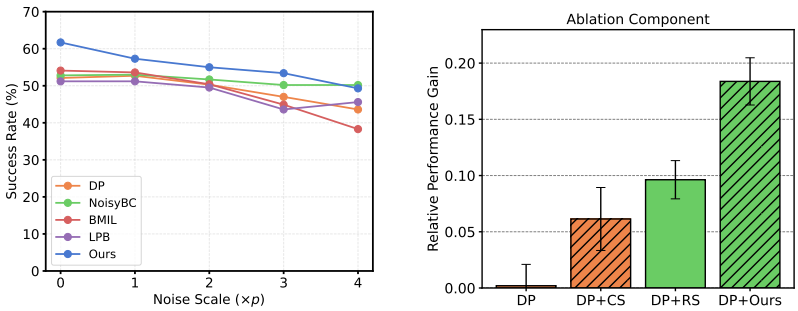
- BMIL (Park & Wong, 2022): Uses a backward policy and a dynamics model to augment the dataset, similar to our approach; however, it relies on a Gaussian model and applies augmentation to arbitrary states in the dataset.
- HINT (Block et al., 2023): Predicts stabilizing low-level controllers from noise-augmented data and injects execution noise to curb compounding errors.
- LPB (Sun & Song, 2025): Employs test-time steering via learned forward dynamics to constrain the policy within the expert distribution, using the same dynamics-model architecture as ours. Following its original protocol, its dynamics model is additionally trained with online environment rollouts, whereas our dynamics models are trained solely on offline demonstration data.

Main Results. Table 1 summarizes the performance of methods. As shown, ReSynC consistently outperforms the base DP across tasks. Notably, the largest gains are observed in long-horizon environments, with an 11.3% improvement on Transport and a 12.7% improvement on Kitchen. ReSynC corrects mild action deviations that would otherwise compound into trajectory failures over extended horizons. Furthermore, on the highly challenging and precision-sensitive Tool-Hang task, simple data augmentation baselines struggle: NoisyBC exhibits severely degraded performance, while BMIL provides only marginal improvements. This is consistent with our claim that temporal consistency matters during training: ReSynC preserves it and performs well on this task. Finally, regarding test-time steering, LPB attains the best result on Square, where it exceeds ReSynC, but does not consistently improve upon DP on the remaining tasks. We do not isolate the cause of this variability; we note only that LPB's dynamics model is trained with additional online environment rollouts under its original protocol, whereas ReSynC uses the offline demonstrations alone and attains a higher average across the five tasks.

Robustness to Perturbation. To investigate how effectively ReSynC mitigates covariate shift at inference time, we simulate actuator noise and external disturbances by introducing temporally correlated action noise during task execution, following So et al. (2025). Figure 3a shows that ReSynC maintains the highest success rate across most noise levels. While baseline methods degrade rapidly as the noise scale increases, ReSynC exhibits only minimal performance degradation. Although NoisyBC matches our method at the largest noise scale, ReSynC does so from a substantially higher base performance.

5.2 Ablation Study

Effects of components. To assess the contribution of each component in ReSynC, we incrementally integrate them into the base diffusion policy. As shown in Figure 3b, each component individually improves performance, while their combination performs best. Applying only contrastive cycle-consistency selection (DP+CS) at inference time improves the average performance from approximately 52% to 55%, indicating that CCE-based filtering effectively removes divergent, OOD action chunks produced by the base policy. In contrast, training with Reciprocal Synthesis augmentation (DP+RS) alone leads to a larger gain, increasing the average success rate to about 57%, demonstrating improved inherent robustness through corrective data augmentation. Combining both components achieves the best performance, exceeding 61% on average and yielding an approximate 10% improvement over the base DP. The combined gain is close to the sum of the individual gains, indicating that the two components address complementary failure modes, with little interaction between them: reciprocal synthesis equips the policy with recovery capabilities, while test-time selection ensures that the most reliable, expert-aligned actions are executed.



(a) Inference-time robustness (b) Component-wise analysis

Figure 3: **Robustness and component analysis.** All results represent metrics aggregated across the five evaluated manipulation tasks. **(a) Inference-time robustness:** Average success rates under varying noise scales p. The scale multiplier is configured as $p \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$ for Push-T and Tool-Hang, and $p \in \{0.0, 0.5, 1.0, 1.5, 2.0\}$ for the remaining tasks. ReSynC degrades gracefully, exhibiting superior resilience to external disturbances compared to baseline methods.

Table 2: **Factorial analysis of Reciprocal Synthesis.** We vary the backward policy family (Gaussian vs. diffusion) and the integration strategy (isolated transitions vs. trajectory stitching), holding inference-time selection fixed to **CS** so that only the training-time augmentation varies. Row 1 recovers BMIL; row 4 is full ReSynC. Mean and standard deviation over three seeds.

Backward policy	Integration	Push-T	Square	Transport	Avg.
Gaussian	Isolated (BMIL)	63.8 ±3.0	64.7 ±4.1	36.7 ±6.6	55.1
Diffusion	Isolated	64.9 ±3.5	70.0 ±3.3	40.7 ±5.0	58.5
Gaussian	Stitched	64.9 ±3.0	58.0 ±7.5	**42.0** ±5.9	55.0
Diffusion	Stitched (ReSynC)	**69.3** ±2.9	**77.3** ±3.4	40.0 ±4.3	**62.2**

What makes Reciprocal Synthesis work. Reciprocal Synthesis combines two design choices: an expressive backward policy, and the stitching of synthesized recovery segments onto expert chunks. To determine which is responsible for the gain, we run a 2×2 factorial over both choices, holding inference-time selection fixed to **CS** so that only the training-time augmentation varies (Table 2). Neither choice suffices on its own. Replacing the Gaussian backward policy of BMIL with a diffusion policy improves the average by $+3.4$ points, while adding stitching to a Gaussian backward policy changes it by -0.1. Only their combination reaches 62.2, a $+7.1$ improvement over BMIL. The two factors interact rather than add: stitching is worth $+3.7$ on a diffusion backward policy but nothing on a Gaussian one, and on Square the joint gain of $+12.6$ far exceeds the -1.4 predicted by summing the individual effects. We attribute this to a structural requirement: coherent stitched recovery presupposes a backward model expressive enough to represent the several distinct predecessor states consistent with a given anchor, which a unimodal Gaussian cannot capture; without one, stitching produces averaged, physically inconsistent transitions that corrupt chunk-based learning. Transport is the exception, where diffusion with stitching (40.0) is statistically indistinguishable from diffusion alone (40.7) and slightly below Gaussian with stitching (42.0); we attribute this to our single-anchor rollout horizon being short relative to the bimanual coordination the task demands.

Inference strategy. To validate the effectiveness of our test-time selection strategy, we investigate several sampling strategies. Specifically, we consider: (i) the reliability of the cycle-consistency error (CCE) as an action quality metric, (ii) the advantage of using distributional consensus over greedy selection, and (iii) the necessity of the contrastive mechanism involving the negative reference set. To this end, we evaluate four distinct selection strategies using an identical pool of $N = 16$ generated candidates: (1) **Max**, which selects the candidate with the highest CCE (*i.e.*, $A^* = \arg\max_{A_t} \mathcal{L}_{\text{CCE}}(A_t)$) as a sanity check; (2) **Min**, which selects the single candidate with the lowest CCE (*i.e.*, $A^* = \arg\min_{A_t} \mathcal{L}_{\text{CCE}}(A_t)$); (3) **Positive Sampling (PS)**, which uniformly samples from the positive set G; and (4) our proposed **Contrastive Sampling (CS)**, which selects the candidate from G that minimizes the distance to other positive samples while maximizing separation from the negative set B.

Table 3: Performance comparison across sampling strategies.

Dataset (% demonstrations)	Max	Min	PS	CS
Push-T (20% demonstrations)	56.7 ±4.8	58.9 ±4.6	60.7 ±3.2	62.0 ±4.2
Square (20% demonstrations)	54.0 ±4.9	59.3 ±8.2	66.7 ±2.5	67.3 ±2.5
Push-T (100% demonstrations)	67.9 ±2.2	84.1 ±2.9	82.9 ±1.5	92.8 ±2.0
Square (100% demonstrations)	57.7 ±7.3	72.3 ±1.7	75.0 ±4.1	83.3 ±2.5

As shown in Table 3, our proposed **CS** consistently outperforms all alternative selection schemes on both tasks and at both demonstration budgets. Regarding the reliability of CCE as an action quality metric, **Min** clears the **Max** sanity check in all four settings, confirming that CCE ordering correlates with action quality. Regarding the advantage of distributional consensus, **PS** uniformly samples from the positive set G rather than deterministically picking the absolute minimum; it improves on **Min** in three of the four settings, and the one exception shows that greedy single-candidate selection is unstable rather than uniformly worse. This is consistent with our core hypothesis: because learned dynamics models can produce artificially low prediction errors for OOD actions in unfamiliar state regions, the single lowest-CCE candidate is not reliably the safest one. Finally, for the necessity of the contrastive mechanism, while **PS** uses the positive set G, it ignores the negative set B. Comparing **PS** against **CS** isolates the contribution of the contrastive penalty: **CS** yields consistent gains, demonstrating that proximity to "good" candidates alone is insufficient for long-horizon robustness; explicitly maximizing separation from the high-error negative set B contributes an additional gain. Further analyses of critical hyperparameters, including the number of candidates N and the positive/negative split ratio w (Table 11), are provided in the Appendix.

Table 4: **Real-robot experiments.** Average success rates (%) over 10 trials per task. The experimental setup involves a 7-DOF Sawyer robotic arm with state acquisition performed via AprilTag markers to ensure precise 6-DoF pose tracking.

Task	**DP**	**HINT**	**LPB**	**Ours**
Push box	20.0	20.0	**50.0**	**50.0**
Lift banana	20.0	30.0	30.0	**70.0**
Avg.	20.0	25.0	40.0	**60.0**

5.3 Real-World Evaluations

To assess sim-to-real transfer, we conduct physical experiments using a 7-DOF Rethink Sawyer robot arm on two tasks: *Push Box*, which requires non-prehensile coordination to maneuver a cube to a target zone, and *Lift Banana*, which involves grasping and elevating a non-primitive object to a specified height. As shown in Table 4, ReSynC attains a 60% average success rate, compared with 20% for DP, 25% for HINT, and 40% for LPB, with the largest margin on the precision-sensitive *Lift Banana* task (70%); on *Push Box* it reaches 50%, matching LPB. Qualitatively, baseline policies frequently fail to recover from minor slippage during the approach phase, whereas ReSynC executes corrective re-grasping, the behavior Reciprocal Synthesis is intended to induce. Further setup details are provided in Appendix A.1.

6 Conclusion

We presented **R**eciprocal **Syn**thesis and Cycle **C**onsistency (ReSynC), which addresses compounding errors in chunk-based generative control policies by synthesizing recovery behavior in a form those policies can consume. The central observation is that corrective data must be temporally aligned with the chunk the policy predicts: substituting a synthesized recovery segment into the prefix of an expert chunk teaches the policy to return to the expert distribution and continue, whereas isolated corrective transitions do not. Our factorial analysis indicates that this alignment and an expressive backward generator are effective only in combination. Across five manipulation tasks in the limited-demonstration regime, ReSynC improves average performance over strong baselines without human intervention or additional environment rollouts, though not uniformly on every task.

AI USE STATEMENT

We used large language models (LLMs) solely to improve the grammar, clarity, and readability of the manuscript. All research ideas, methods, experiments, analyses, and conclusions are the authors' own, and the authors reviewed and take full responsibility for all content of this paper.

REFERENCES

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*, 2024.

Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior. *Advances in Neural Information Processing Systems*, 36:48534–48547, 2023.

Jonathan Chang, Masatoshi Uehara, Dhruv Sreenivas, Rahul Kidambi, and Wen Sun. Mitigating covariate shift in imitation learning via offline data with partial coverage. *Advances in Neural Information Processing Systems*, 34:965–979, 2021.

Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009.

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 (IJRR)*, 2023.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in neural information processing systems*, 32, 2019.

Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance. *arXiv preprint arXiv:2506.13922*, 2025.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pp. 158–168. PMLR, 2022.

Zhaohan Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35: 31855–31870, 2022.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *CoRL*, 2020.

Charles A Hepburn and Giovanni Montana. Model-based trajectory stitching for improved behavioural cloning and its applications. *Machine Learning*, 113(2):647–674, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. *ICLR*, 2023.

Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 6185–6191. IEEE, 2021.

Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. Ccil: Continuity-based data augmentation for corrective imitation learning. *arXiv preprint arXiv:2310.12972*, 2023.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv preprint arXiv:2506.17811*, 2025.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pp. 143–156. PMLR, 2017.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13872–13882, 2024.

Guanghe Li, Yixiang Shan, Zhengbang Zhu, Ting Long, and Weinan Zhang. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. In *International Conference on Machine Learning*, 2024.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv:2210.02747*, 2022.

Jiafei Lyu, Xiu Li, and Zongqing Lu. Double check your state before trusting it: Confidence-aware bidirectional offline model-based imagination. In *Advances in Neural Information Processing Systems*, 2022.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *CoRL*, 2022.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *CoRL*, 2023.

Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, 2011. doi: 10.1109/ICRA.2011.5979561.

Jung Yeon Park and Lawson Wong. Robust imitation of a few demonstrations with a backwards model. *Advances in Neural Information Processing Systems*, 35:19759–19772, 2022.

Yeji Park, Deokyeong Lee, Junsuk Choe, and Buru Chang. Convis: Contrastive decoding with hallucination visualization for mitigating hallucinations in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 6434–6442, 2025.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pp. 5062–5071. PMLR, 2019.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026.

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela P Schoellig. Failure prediction at runtime for generative robot policies. *NeurIPS*, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Seokin Seo, Byung-Jun Lee, Jongmin Lee, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Mitigating covariate shift in behavioral cloning via robust stationary distribution correction. *Advances in Neural Information Processing Systems*, 37:109177–109201, 2024.

Junhyuk So, Chiwoong Lee, Shinyoung Lee, Jungseul Ok, and Eunhyeok Park. Improving generative behavior cloning via self-guidance and adaptive chunking. *arXiv preprint arXiv:2510.12392*, 2025.

Harold Soh and Eugene Lim. Action hallucination in generative visual-language-action models. *arXiv preprint arXiv:2602.06339*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Padmanaba Srinivasan and William Knottenbelt. Offline model-based reinforcement learning with anti-exploration. *arXiv preprint arXiv:2408.10713*, 2024.

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. *arXiv preprint arXiv:2508.05941*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *RSS*, 2024.

Julen Urain, Ajay Mandlekar, Yilun Du, Mahi Shafiullah, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, and Jan Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *arXiv preprint arXiv:2408.04380*, 2024.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *CoRL*, 2025.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. In *Advances in Neural Information Processing Systems*, 2021.

Shuze Wang, Yunpeng Mei, Hongjie Cao, Yetian Yuan, Gang Wang, Jian Sun, and Jie Chen. Robust offline imitation learning through state-level trajectory stitching. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13497–13504. IEEE, 2025.

Xintong Wang, Jingheng Pan, Liang Ding, and Chris Biemann. Mitigating hallucinations in large vision-language models with instruction contrastive decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15840–15853, 2024.

Qianlan Yang and Yu-Xiong Wang. Rtdiff: Reverse trajectory synthesis via diffusion for offline reinforcement learning. In *International Conference on Learning Representations*, 2025.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025a.

Thomas T Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control. *arXiv preprint arXiv:2507.09061*, 2025b.

Allan Zhou, Moo Jin Kim, Lirui Wang, Pete Florence, and Chelsea Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17907–17917, 2023.

A Implementation Details

Base and Backward Policy. We adopt Diffusion Policy (Chi et al., 2023) as the base policy in ReSynC due to its strong capability to model complex robot action distributions. Task-specific and shared hyperparameters are provided in Tables 6 and 7, respectively. The architecture of the backward policy is the same as that of the base policy. The hyperparameters used for training the backward policy are provided in Table 8. Note that, both the base and backward policies are trained with same demonstration dataset. During base policy training, given a batch containing a sequence of states and actions, synthetic corrective transitions are generated as follows. First, we select the state at index j from the sampled batch as the anchor state and define the number of deviation steps as M, *i.e.*, $M = j + 1$. We then generate noisy actions using $K_{\mathcal{B}_\psi}$ denoising steps and use the backward dynamics model to predict the corresponding past states. Finally, we duplicate the original batch and replace the M transitions preceding index j with the M synthesized corrective transitions.

Dynamics Model. The dynamics models are parameterized using a causal Transformer architecture. They receive a concatenation of a state token and action tokens as input and predict the subsequent state tokens. Training hyperparameters are provided in Table 9.

Baselines. For the baselines, we reuse the official implementations whenever applicable. Note that all baselines use Diffusion Policy as the base policy. For baselines that do not originally adopt Diffusion Policy, we reimplement them using the same base policy to ensure a fair comparison. Details of the baseline implementations and hyperparameter search are provided in Table 5.

Table 5: Source code links and hyperparameter/variant search settings for all baselines.

Algorithm	URL	Hyperparameters Tuning
DP	`https://github.com/real-stanford/diffusion_policy`	Default
NoisyBC	Reimplement	The scale noise $\sigma \in \{0.01, 0.02, 0.05, 0.1\}$
BMIL	`https://github.com/jypark0/bmil`	Default
HINT	`https://www.dropbox.com/scl/fi/c519mayngbb8o1zbf7jxz/diffusion_policy.zip?rlkey=t0tazxbtykqkwwmkjhfv2cx1n&e=1&dl=0`	Default
LPB	`https://github.com/zhanyisun/lpb`	Guidance scale $\eta \in \{1, 10, 100, 1000, 10000\}$, OOD threshold $\tau =$ (Average of OOD score)

A.1 Real-World Robot Experiments

To assess the practical applicability of ReSynC, we deploy our framework in a physical robotic manipulation setting.

Table 6: Simulation task-dependent hyperparameters for base diffusion policy training.

Env Name	T_a	T_p	% Demo
Push-T	8	16	20%
Square	8	16	20%
Tool-Hang	15	32	40 %
Transport	8	16	20%
Kitchen	8	16	5 %

Table 7: Shared hyperparameters for base diffusion policy training.

Name	Value
T_o	2
Batch Size	512
Learning Rate	1×10^{-4}
Diffusion Step	100
Training Epochs	1000
Deviation steps M	2
Inference denoising step of μ_ω	2
Number of candidates N	16
Decay factor ρ	{0, 0.3, 0.5, 0.9, 1.0}

Table 8: Hyperparameters for the backward diffusion policy training.

Name	Value
T_o	1
T_a	4
Batch Size	256
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-4}
Diffusion Step	100
Training Epochs	100

Table 9: Hyperparameters for dynamics model training.

Name	Value
History Length	1
Prediction Length	15 (Tool-Hang), 8 (others)
Depth	1
Heads	4
MLP Dim	384
Dropout	0.1
Batch Size	256
Learning Rate	1×10^{-4}
Training Epoch	200
State Noise Augmentation	0.01 (forward model), 0.02 (backward model)

Physical Setup and State Estimation. Our experimental platform consists of a 7-DOF Sawyer robotic arm situated in a structured tabletop workspace (Figure 4-A). To facilitate robust sim-to-real transfer and avoid the high-dimensional complexity of pixel-level control, we adopt a low-dimensional, state-centric observation space. Pose acquisition is managed through the use of AprilTag fiducial markers (Olson, 2011) affixed to the robot's end-effector, the manipulated objects, and a static tabletop reference. Using an external, calibrated RGB camera, a real-time tracking system computes the 6-DoF pose of each entity relative to the global table frame. This configuration ensures that the policy receives informative, low-noise observations that effectively satisfy the Markov property during execution.

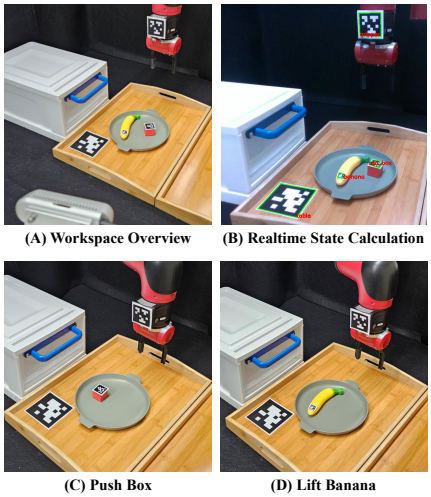


Figure 4: Real-world experimental setup. (A) The overall workspace featuring the 7-DOF Sawyer robotic arm. (B) Real-time state acquisition interface. The tracking software detects AprilTags overlaid on the camera feed to compute the precise 6-DoF coordinates of the robot and target objects relative to the base tag. (C), (D) Task-specific environmental setups for *Push Cube* and *Lift Banana*, respectively.

State and Action Space. The environment is parameterized at each timestep t by a 10-dimensional state vector $\mathbf{s}_t \in \mathbb{R}^{10}$, defined as $\mathbf{s}_t = [\mathbf{p}_g, \mathbf{p}_o, \mathbf{d}_{rel}, g_s]$. Within this formulation, $\mathbf{p}_g, \mathbf{p}_o \in \mathbb{R}^3$ denote the 3D Cartesian positions of the gripper and target object, respectively, while $g_s \in \{0, 1\}$ signifies the binary status of the gripper. We explicitly include the relative displacement, $\mathbf{d}_{rel} = \mathbf{p}_o - \mathbf{p}_g$, to provide a dense geometric signal that facilitates more efficient convergence during the offline training phase. The corresponding action space is a 5-dimensional continuous vector $\mathbf{a}_t = [v_x, v_y, v_z, \omega_\psi, g_{cmd}] \in \mathbb{R}^5$. This vector encodes the relative Cartesian velocities, the fixed end-effector yaw, and the command for gripper actuation.

Task Specifications. We evaluate our framework on two representative manipulation tasks that challenge both prehensile and non-prehensile coordination. The *Lift Banana* task requires the agent to approach, securely grasp, and elevate a non-primitive object to a specified vertical threshold. Conversely, the *Push Cube* task necessitates maneuvering a cube from a randomized initial configuration to a designated target zone on the table surface. Together, these tasks assess the policy's ability to generalize across different contact dynamics and spatial objectives.

B Additional Results

We present comprehensive analyses of critical hyperparameters of ReSynC in Table 10.

C Limitations and Future Work

While ReSynC improves robustness in the settings we evaluate, several limitations bound the claims we can make.

Inference cost. The inference stage generates multiple candidate action chunks and queries both dynamics models to compute the Cycle-Consistency Error. This adds computational overhead at deployment (Table 10), which may be limiting for systems that require high-frequency per-step replanning. Distilling the selection mechanism into a single forward pass is a natural direction for future work.

Reliance on learned dynamics. The two stages depend on the dynamics models in different ways. Reciprocal Synthesis requires the backward dynamics model to produce physically plausible

Table 10: **ReSynC hyperparameter ablations and inference latency.** We present comprehensive analyses of the key hyperparameters in our framework. Results are averaged over 50 trials and 3 random seeds. **Bold** text denotes the default hyperparameter settings used in the main experiments. The inference latency is also reported, which shows our method only introduce slight overhead.

M	1	2	3	4
Square	72.0	**77.3**	72.0	72.0
Transport	38.7	**40.0**	38.0	38.0
Kitchen	86.0	**90.7**	88.7	84.7

(a) **Deviation steps**.

K_{μ_ω}	1	2	10	50
Square	72.7	**77.3**	68.0	68.7
Transport	43.3	**40.0**	38.0	38.0
Kitchen	88.7	**90.7**	91.3	83.3

(b) **Inference denoising steps**.

ρ	0	0.3	0.5	0.9	1.0
Square	68.7	68.7	**77.3**	70.0	66.7
Transport	32.0	36.0	39.3	40.0	**40.0**
Kitchen	79.3	86.0	80.0	86.0	**90.7**

(c) **CCE decay factor**.

N	1	2	4	8	16	32
Square	71.7	68.7	72.7	70.7	**77.3**	73.3
Transport	35.3	36.7	34.7	39.3	**40.0**	38.0
Kitchen	80.0	82.7	87.3	88.7	**90.7**	92.0

(d) **Number of candidates.**

Method	**Inference latency (ms)**
DP	574.83
LPB	589.22
ReSynC	609.86

(e) **Inference latency**.

Table 11: **Sensitivity to the positive/negative split ratio w.** The total reference-set size is held fixed and only its positive-to-negative composition varies, where w denotes the fraction of candidates assigned to the positive set G. The default $w = 0.5$ is the equal split used in all other experiments. We report the mean and standard deviation over three seeds, with each run evaluated on 50 held-out initial conditions. No monotonic trend is observed across split ratios, indicating some task dependence rather than a uniform sensitivity to imbalance.

Task	$w = 0.9$	$w = 0.7$	$w = 0.5$	$w = 0.3$	$w = 0.1$
Push-T	65.4 $\pm$7.6	65.6 $\pm$4.0	**69.3** $\pm$2.9	66.1 $\pm$6.1	63.6 $\pm$8.0
Square	73.3 $\pm$6.6	72.0 $\pm$3.3	**77.3** $\pm$3.4	68.7 $\pm$0.9	70.7 $\pm$5.7
Transport	**44.0** $\pm$2.8	37.3 $\pm$1.9	40.0 $\pm$4.3	38.0 $\pm$2.8	38.0 $\pm$1.6
Tool-Hang	23.3 $\pm$3.4	25.3 $\pm$0.9	**31.3** $\pm$8.4	22.7 $\pm$2.5	26.0 $\pm$6.5
Kitchen	83.3 $\pm$2.5	88.0 $\pm$1.6	**90.7** $\pm$2.5	85.3 $\pm$0.9	89.3 $\pm$2.5
Average	57.9	57.6	**61.7**	56.2	57.5

Table 12: **ReSynC at the limited and the full demonstration budget.** Both methods are trained on the same data at each budget. The gain from ReSynC does not depend on the base policy being degenerate: DP is a functioning policy at 20% of the demonstrations, and the improvement persists when all demonstrations are available.

	20% demonstrations		100% demonstrations	
Task	DP	ReSynC	DP	ReSynC
Push-T	60.0	**69.3**	98.7	**99.1**
Square	68.7	**77.3**	82.7	**88.7**
Transport	28.7	**40.0**	48.7	**55.3**
Average	52.5	**62.2**	76.7	**81.0**

predecessor states, so the quality of the synthesized data is bounded by that model's accuracy; in

Table 13: **Reliability of CCE as the dynamics training set shrinks.** The base policy is held fixed and only the forward and backward dynamics models are retrained, on the stated fraction of Push-T demonstrations. OOD candidates are defined operationally as the top quartile by k-NN distance to expert state–action chunks, a measure computed from raw data and therefore independent of both dynamics models. Reliability is preserved across a tenfold reduction, and is highest at the smallest budgets, consistent with CCE tracking the match between the dynamics' support and the policy's rather than the accuracy of the dynamics.

Dynamics training data	100%	80%	60%	40%	20%	10%
Demonstrations (absolute)	206	165	124	82	41	21
Spearman ρ(CCE, OOD)	0.44	0.54	0.52	0.56	**0.67**	0.62
AUROC (CCE predicts OOD)	0.70	0.76	0.72	0.72	**0.82**	0.78

highly stochastic environments, or in tasks involving irreversible processes such as cutting, mixing, or breaking objects, accurate backward prediction is difficult and the synthesized recoveries may be unreliable. Contrastive selection depends on the dynamics differently. As discussed in Section 4, a low cycle-consistency error indicates that a candidate resembles the transitions on which the dynamics were fitted; it does not establish that executing the candidate is safe, or that either model is correct in that region, so candidates that are out of distribution yet receive a low CCE remain possible and our results do not rule them out. Reliability therefore depends on the dynamics and the policy sharing a support rather than on the volume of dynamics data: in Table 13 the correlation with an external OOD measure is weakest when the dynamics are trained on the full dataset, which we attribute to models fitted on broader data also closing the round trip outside the policy's support, and training the dynamics on substantially more data than the policy should be expected to weaken the signal. Finally, the benefit of the contrastive term over uniform sampling from the low-error set is small in the limited-demonstration regime that is our primary setting (Table 3) and more pronounced with the full demonstration set; it is best understood as stabilizing selection rather than as the principal source of our gains.

Evaluation scope. Our evaluation measures task success and degradation under injected action noise. These are indirect measures of the recovery behavior that the training stage is designed to induce: we do not directly measure return from a deviated state to the expert distribution, and the corrective re-grasping reported in Section 5 is an observational finding rather than a quantified one. All simulated results use three training seeds with 50 held-out initial conditions per task, and the real-robot results use ten trials per task. Our evaluation is also confined to low-dimensional state observations such as object poses and joint configurations. Extending ReSynC to high-dimensional visual observations is an important direction; because learning dynamics directly in pixel space is difficult, one route is to perform both the stitching and the cycle-consistency computation inside a latent space produced by a pretrained visual encoder.