Rewind-IL: Online Failure Detection and State Respawning for Imitation Learning

Gehan Zheng[1], Sanjay Seenivasan[1,2], Matthew Johnson-Roberson[1], Weiming Zhi[1,3,4]

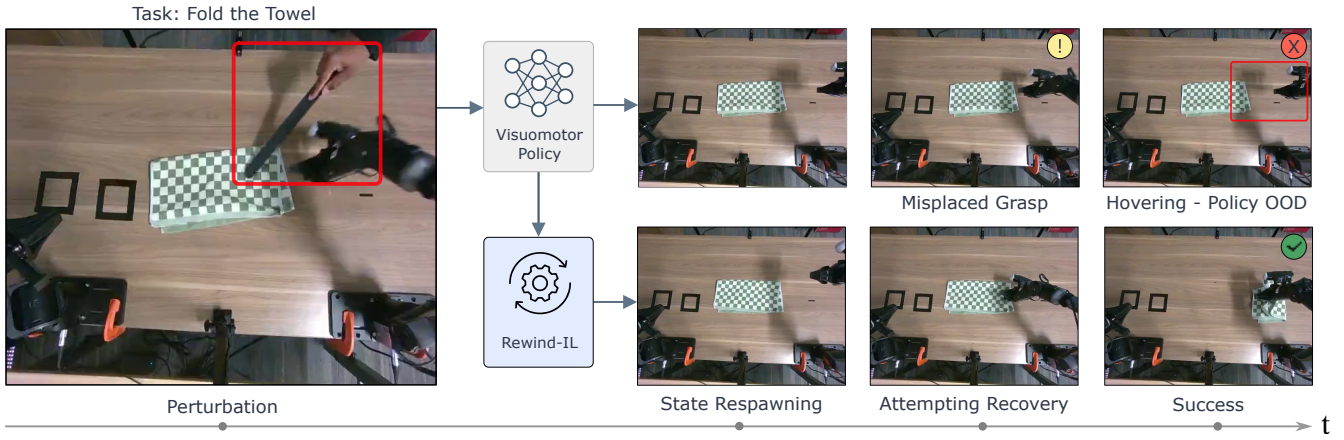

Fig. 1: *Rewind-IL* enables visuomotor robot policies to efficiently predict and recover from task failures. When the baseline policy fails, the framework rewinds to a prior state to reattempt the task successfully.

***Abstract*— Imitation learning has enabled robots to acquire complex visuomotor manipulation skills from demonstrations, but deployment failures remain a major obstacle, especially for long-horizon action-chunked policies. Once execution drifts off the demonstration manifold, these policies often continue producing locally plausible actions without recovering from the failure. Existing runtime monitors either require failure data, over-trigger under benign feature drift, or stop at failure detection without providing a recovery mechanism. We present Rewind-IL, a training-free online safeguard framework for generative action-chunked imitation policies. Rewind-IL combines a zero-shot failure detector based on Temporal Inter-chunk Discrepancy Estimate (TIDE), calibrated with split conformal prediction, with a state-respawning mechanism that returns the robot to a semantically verified safe intermediate state. Offline, a vision-language model identifies recovery checkpoints in demonstrations, and the frozen policy encoder is used to construct a compact checkpoint feature database. Online, Rewind-IL monitors self-consistency in overlapping action chunks, tracks similarity to the checkpoint library, and, upon failure, rewinds execution to the latest verified safe state before restarting inference from a clean policy state. Experiments on real-world and simulated long-horizon manipulation tasks, including transfer to flow-matching action-chunked policies, demonstrate that policy-internal consistency coupled with semantically grounded respawning offers a practical route to improved reliability in imitation learning. Supplemental materials are available at https://sjay05.github.io/rewind-il.**

I. Introduction

Imitation learning (IL) has become a practical paradigm for teaching robots complex manipulation skills directly from expert demonstrations, particularly in settings where specifying rewards or controllers by hand is difficult [1], [2], [3]. Recent visuomotor policy architectures have substantially improved long-horizon performance by predicting temporally extended action sequences rather than single-step controls [4], [5]. Despite these advances, action-chunked imitation policies can remain brittle at deployment time. Small execution errors, perception shifts, object motion, or missed contacts can push the robot into states that are not well covered by the demonstration distribution. Once this occurs, the policy may continue producing actions that are locally consistent with its internal rollout while no longer making task progress. In practice, this often leads to failure modes such as dropped objects, misaligned placements, or incomplete subtask transitions, after which the robot is unable to recover and the episode terminates unsuccessfully.

Reliable deployment requires answering two questions online: *when* to intervene and *where* to return. Prior runtime monitors identify anomalous states but typically stop at failure flagging without providing recovery. Methods that adapt action generation at test time still assume the policy can continue from the current state, forgoing explicit state restoration. We introduce Rewind-IL, a training-free online safeguard that separates failure detection from recovery target selection, grounding both in signals inherent to the trained policy and demonstration data. For detection, Rewind-IL monitors internal self-consistency via *Temporal Inter-chunk Discrepancy Estimate (TIDE)*: the disagreement between the current action chunk and the plan predicted one step prior. A sharp rise signals that the policy is reconsidering

[1]College of Connected Computing, Vanderbilt University, USA.
[2]School of Computer Science, University of Waterloo, Canada.
[3]School of Computer Science, The University of Sydney, Australia.
[4]Australian Centre for Robotics, The University of Sydney, Australia.

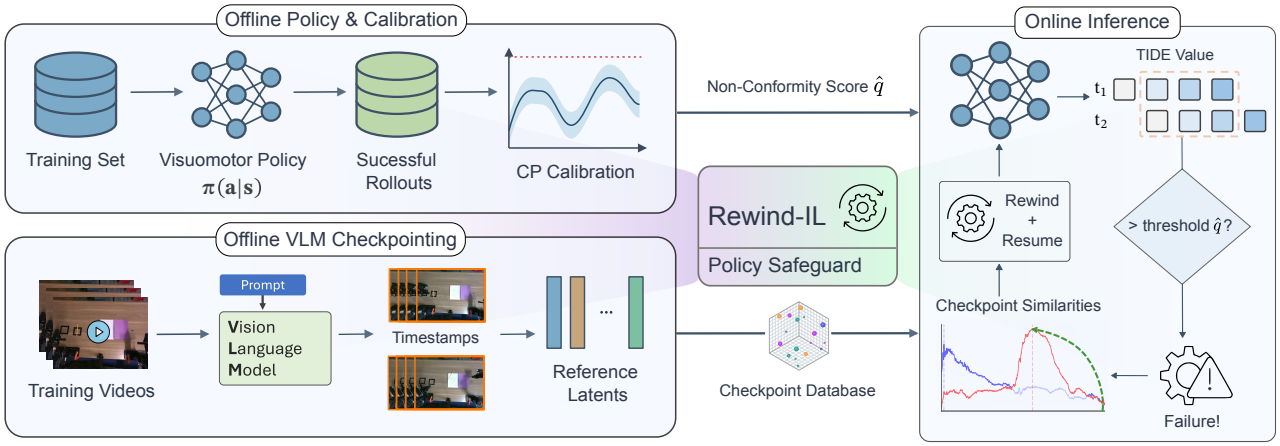

Fig. 2: Overview of the *Rewind-IL* framework. **(Left) Offline Staging:** Successful policy rollouts are used to construct a viable CP threshold for failure detection. Concurrently, a VLM extracts meaningful keyframes from demonstration videos to generate a checkpoint database. **(Right) Online Policy Deployment:** TIDE flags failures, and the policy returns to a checkpointed state.

its near-future plan after an unexpected state. This signal is calibrated with split conformal prediction, yielding a threshold derived solely from successful rollouts.

For recovery, Rewind-IL constructs an offline checkpoint database of semantically verified intermediate states. A VLM identifies recovery timestamps from training demonstrations (e.g., completed grasps, subgoal transitions), and the frozen policy encoder extracts compact latent templates at those frames. Online, Rewind-IL tracks cosine similarity between the current observation embedding and each template, snapshotting the action at peak similarity. Upon failure, the system replays the latest peaked checkpoint action, clears stale policy memory, and restarts inference cleanly. The result is a lightweight safeguard requiring no retraining, no failure data, and no auxiliary controller, built entirely from signals available in standard imitation learning pipelines.

We evaluate Rewind-IL on real-world bimanual manipulation tasks and in simulation. TIDE achieves strong failure-detection performance, and the full Rewind-IL framework substantially improves task success both under natural execution failures and adversarial disturbances.

Concretely, the contributions of this paper include:

1) *Rewind-IL*, an online safeguard framework designed to efficiently detect task failures over generative imitation learning polices based on action chunking;
2) *Temporal Inter-chunk Discrepancy Estimate (TIDE)*, a policy self-consistency signal calibrated by conformal prediction for zero-shot failure detection, together with a VLM-guided checkpoint construction pipeline for selecting semantically meaningful recovery targets;
3) We demonstrate on real-world and simulated long-horizon manipulation tasks that Rewind-IL markedly improves robustness and task success, including under adversarial disturbances.

II. Related Work

Failure Detection in Generative and Action-Chunked Policies: Action-chunked policies such as ACT [4] and Diffusion Policy [5] are prone to compounding errors under distribution shift [6], motivating a line of work on runtime OOD detection. Architecture-specific approaches extract uncertainty from diffusion denoising [7], [8] or exploit internal distillation signals [9], but are tightly coupled to their respective model families. Diffusion-based OOD detection has also been extended to $\mathbb{SE}(3)$ pose sequences [10], though this targets pose-level anomalies rather than policy-level failures in action-chunked settings. FIPER [11] and FAIL-Detect [12] take more general stances, combining encoder-level OOD scores with conformal prediction [13], yet still rely on static reference distributions that misclassify benign feature drift as failures. Sentinel [14] partially addresses this by pairing a temporal-consistency detector with online VLM queries. In contrast, TIDE requires no additional training and is broadly applicable across the action-chunked architectures evaluated here: it monitors the self-consistency of overlapping action chunks and calibrates thresholds via split conformal prediction, with observed generalization to flow-matching policies.

Error Recovery and State Backtracking: Traditional reactive mechanisms, e.g. CBFs [15], heuristic recoveries [16], and fixed-pose resets [17], either stop the robot or return it to a hardcoded geometric state, without regard for semantic task context. CycleVLA [18] goes further by using a VLM to detect failure and trigger subtask backtracking with MBR decoding at retry, but incurs costly online inference and backtracks only at coarse subtask granularity. Rewind-IL instead respawns to a fine-grained, *semantically verified* intermediate state identified entirely offline, enabling recovery at negligible online cost.

Semantic Verification and VLM-Guided Monitoring: EVE [19], FOREWARN [20], and CoVer-VLA [21] integrate VLMs directly into the control loop to verify or score candidate action chunks at every step, achieving strong semantic grounding at the cost of prohibitive inference latency. Rewind-IL avoids this trade-off by restricting VLM use to an *offline* checkpoint identification phase; online monitoring degrades to a lightweight cosine similarity search over a frozen-encoder feature database, preserving semantic rigor at the high frequencies required for closed-loop control.

III. PRELIMINARIES

Behavior Cloning Setup: Imitation learning aims to learn a control policy directly from expert demonstrations. In this work, we consider the standard behavior cloning (BC) setting, where a dataset of N demonstrations is given by $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_N\}$, and each trajectory τ_i is a sequence of observation and action pairs, $\tau_i = \{(o_1, a_1), (o_2, a_2), \ldots, (o_{T_i}, a_{T_i})\}$. Here, $o_t \in \mathcal{O}$ denotes the robot observation, which may include images and proprioception and $a_t \in \mathcal{A}$ denotes the corresponding expert action.

In behavior cloning (BC), the goal is to learn a parametric policy π_θ that maps observations to actions by supervised learning over the demonstration data. In the standard single-step setting, the policy predicts the next action from the current observation as $\pi_\theta(a_t \mid o_t)$, and is trained by minimizing the negative log-likelihood of expert actions:

$$\mathcal{L}(\theta) = \mathbb{E}_{(o,a)\sim\mathcal{D}} \left[-\log \pi_\theta(a \mid o) \right]. \tag{1}$$

Although simple and effective, BC is sensitive to distribution shift at deployment: small execution or perception errors can move the robot away from states covered by the demonstrations, after which errors may accumulate and lead to unrecoverable failures [6].

Action-Chunked Behavior Cloning: In action-chunked behavior cloning, the policy predicts a short horizon of future actions rather than a single action at each timestep. Given the current observation o_t, the policy outputs $\hat{\mathbf{a}}_{t:t+K-1} = \pi_\theta(o_t)$ where $\hat{\mathbf{a}}_{t:t+K-1} = \{\hat{a}_t, \hat{a}_{t+1}, \ldots, \hat{a}_{t+K-1}\}$ is a sequence of K actions. The policy is trained to match expert action chunks $\mathbf{a}_{t:t+K-1}$ using a supervised objective of the form

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\hat{\mathbf{a}}_{t:t+K-1} - \mathbf{a}_{t:t+K-1}\| \right]. \tag{2}$$

Compared with single-step BC, predicting action chunks can improve temporal consistency and better capture short-horizon structure in expert behavior, especially in long-horizon manipulation tasks. Action-Chunking with Transformers (ACT) is a representative model of this class of policies [4]. In practice, overlapping chunk predictions are often temporally aggregated at inference time to produce smoother control commands.

Conformal Prediction: CP is a distribution-free framework that quantifies uncertainty by constructing prediction sets with finite-sample coverage guarantees [13]. Given nonconformity scores $\mathcal{S}_{cal} = \{s_1, \ldots, s_n\}$ on exchangeable calibration samples and a target error rate α, split CP sets the threshold

$$\hat{q} = \text{Quantile}\left(\mathcal{S}_{cal}, \frac{\lceil (n+1)(1-\alpha) \rceil}{n} \right) \tag{3}$$

so that the prediction set $\mathcal{C}(X_{n+1}) = \{Y : s(X_{n+1}, Y) \leq \hat{q}\}$ contains the true label with probability at least $1-\alpha$, i.e. $\mathbb{P}(Y_{n+1} \in \mathcal{C}(X_{n+1})) \geq 1-\alpha$.

IV. REWIND-IL

A. Overview

Rewind-IL is a training-free online safeguard framework for generative action-chunked robot policies that provides two capabilities: (1) zero-shot *real-time failure detection* based on the internal self-consistency of the policy's action predictions, and (2) *state respawning* that physically returns the robot to a semantically verified safe intermediate state when a failure is flagged. The framework is broadly illustrated in Figure 2. In this paper, we instantiate Rewind-IL with ACT as baseline visuomotor policy; the framework is designed to apply broadly to action-chunked policies.

Algorithm 1: Rewind-IL Online Inference Loop

Input: Policy π_θ, templates $\{\hat{\mathbf{t}}_k\}_{k=1}^{K}$, threshold $\hat{q}$, peak gap Δ_{peak}
Initialize template slot trackers
for *each timestep* $t = 0, 1, 2, \ldots$ **do**
  Observe o_t; run $\pi_\theta(o_t) \to (a_t, \tilde{\mathbf{A}}_t, \mathbf{f}_t)$
  $\text{TIDE}_t \leftarrow$ compute_tide$(\tilde{\mathbf{A}}_t, \hat{\mathbf{A}}_{t-1})$
  update_slots$(\mathbf{f}_t, a_t)$
  if $\text{TIDE}_t > \hat{q}$ ***and*** $\exists k : \text{peaked}_k$ **then**
    $k^* \leftarrow$ latest peaked slot
    Execute recovery action $a^*_{k^*}$
    Reset ensembler and action queue
  else
    Execute normal action a_t
  end
end

Calibration Phase (Offline): After training, the policy is rolled out to obtain N successful episodes which form a calibration set. At every frame *Temporal Inter-chunk Discrepancy Estimate* (TIDE) values are recorded to form a calibration corpus from which a statistically motivated failure threshold $\hat{q}$ is derived via split conformal prediction [13].

Checkpoint Database Construction (Offline): Independently, a vision-language model (VLM) inspects the visual observations $\mathbf{I}_t$ from each training episode and identifies a fixed set of K semantically meaningful recovery timestamps per episode. The frozen policy is then run on those frames to extract compact latent feature vectors. A kernel density estimation (KDE) based template selection step identifies the most representative per-slot embedding across episodes. All templates are stored in a *checkpoint feature database*.

Inference Phase (Online): At each timestep the postprocessor *(i)* computes the TIDE signal and checks whether it exceeds $\hat{q}$, *(ii)* measures the cosine similarity of the current latent feature against every template in the checkpoint database, and *(iii)* updates per-slot running-maximum bookkeeping and action snapshots. When a failure is detected and a *peaked* slot is available, *Rewind-IL* replays the corresponding action snapshot, clears all policy caches, and restarts inference from a clean state. Algorithm 1 summarizes the online loop and Figure 1 depicts an example rollout.

B. Error Flagging via TIDE

Temporal Inter-chunk Discrepancy Estimate: A baseline visuomotor policy's temporal ensembler [4] maintains an aggregated action plan $\hat{\mathbf{A}}_{t-1} \in \mathbb{R}^{B\times T\times D}$ assembled from prior inference steps, where B is batch size, T is the overlap length with the new chunk, and D is the action dimension. At step t the policy produces a fresh chunk $\tilde{\mathbf{A}}_t \in \mathbb{R}^{B\times T'\times D}$ with $T' \geq T$. We align the first T steps and compute the mean squared discrepancy:

$$\mathrm{TIDE}_t = \frac{1}{BDT} \sum_{b,d,\tau} \left(\hat{A}^{(b)}_{t-1,\tau,d} - \tilde{A}^{(b)}_{t,\tau,d} \right)^2 \tag{4}$$

This signal measures how much the policy's current belief about the near future diverges from what it predicted one step ago. The temporal ensembler in ACT [4] is predicated on the assumption that overlapping chunk predictions are mutually consistent under nominal execution: it is precisely this consistency that makes averaging across chunks meaningful. TIDE formalizes this implicit design assumption as an explicit runtime criterion.

Proposition 1 (Temporal consistency)**.** *Let $\mathcal{M}$ be the manifold of states covered by training demonstrations, and let o_t denote a compact observation (e.g. proprioceptive state or encoder latent). Assume π_θ is locally L-Lipschitz on $\mathcal{M}$ (i.e. $\|\pi_\theta(o) - \pi_\theta(o')\| \le L\|o - o'\|$ for $o, o' \in \mathcal{M}$) and approximately unimodal on $\mathcal{M}$. If $o_t \in \mathcal{M}$ and $\|o_t - o_{t-1}\|_2 \le \epsilon$, then $\mathrm{TIDE}_t \lesssim L^2\epsilon^2$. When $o_t \notin \mathcal{M}$, the local Lipschitz constant may grow sharply or a mode switch may occur, causing TIDE_t to substantially exceed this bound.*

The bound follows because aligning the first T steps makes both tensors predictions for the same future window, so the Lipschitz condition directly gives $\|\tilde{\mathbf{A}}_t[:T] - \hat{\mathbf{A}}_{t-1}\| \lesssim L\epsilon$; squaring yields $\mathrm{TIDE}_t \lesssim L^2\epsilon^2$. Locality relaxes the impractical requirement of a finite global Lipschitz constant; unimodality excludes intent switches on strongly multimodal tasks (rare within a single manipulation phase); $\lesssim$ accounts for $\hat{\mathbf{A}}_{t-1}$ being a smoothed aggregate, whose averaging further suppresses nominal variation.

Failure detection via conformal prediction: A failure is flagged when:

$$\mathrm{is\_failing}(t) = \mathbf{1}\big[\,\mathrm{TIDE}_t > \hat{q}\,\big]. \tag{5}$$

The threshold $\hat{q}$ is determined offline using split conformal prediction [13], following the approach of [12]. Let $\mathcal{S}_{\mathrm{cal}} = \{s_1, \ldots, s_n\}$ be the per-frame TIDE values collected from calibration rollouts on held-out *successful* episodes (trimming Δ boundary frames per episode to exclude transient startup and completion artifacts). The threshold is set at the empirical $(1-\alpha)$-quantile. We use $\alpha = 0.001$ in all experiments. Figure 3 illustrates how TIDE is used to flag a execution failure.

C. Offline Checkpoint Database Construction

The checkpoint database is built on the training dataset and the trained policy. It is the output of two sequential stages: semantic identification of safe recovery timestamps by a VLM, and extraction of compact latent feature templates at those timestamps.

VLM-guided safe timestamp identification: For each episode in the training dataset, we render a short video clip and query a VLM (e.g., Gemini 3.1 Pro) to identify timesteps at which the robot reaches an unambiguous, task-consistent intermediate state suitable for recovery, for instance immediately after a grasp closes, or at the transition between sub-goals. The VLM returns an ordered list of K per-episode timestamps $\{t_1^{(e)}, \ldots, t_K^{(e)}\}$, which we call *slot timestamps*. This step is fully offline and model-agnostic: manual annotation is a viable substitute, as each episode requires identifying only K salient frames.

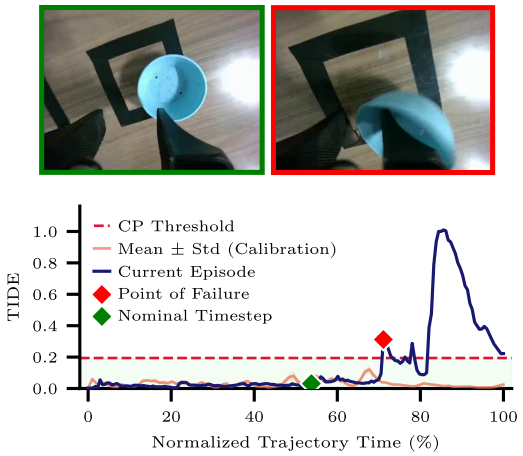

Fig. 3: Illustration of nominal timestep (shown in green) below TIDE threshold versus first point of failure (shown in red).

Policy feature extraction: At each verified checkpoint frame $(e, t_k^{(e)})$ we perform a single forward pass of the frozen policy and capture the observation-conditioned part of the policy encoder's output, which is a token sequence $(f_0, f_1, \ldots, f_{L-1}) \in \mathbb{R}^{L \times d}$. We mean-pool the observation-conditioned tokens:

$$\mathbf{f}_{\mathrm{enc}} = \frac{1}{L} \sum_{i=0}^{L-1} f_i \ \in \mathbb{R}^d. \tag{6}$$

This d-dimensional vector summarizes the policy's semantic interpretation of the current observation.

KDE-based representative template selection: For slot k, the cross-episode feature vectors $\{\mathbf{f}_k^{(e)}\}_{e=1}^{E}$ form a point cloud in $\mathbb{R}^d$. A simple mean $\bar{\mathbf{f}}_k = \frac{1}{E}\sum_e \mathbf{f}_k^{(e)}$ blurs together distinct execution trajectories and may not correspond to any real robot state. Instead, we select the single most *representative* episode embedding using kernel density estimation.

Since the feature vectors are high-dimensional, we first apply PCA retaining a fraction $r = 0.95$ of the total variance, projecting the cloud to $d_{\mathrm{pca}} \ll d$ dimensions. We then fit a Gaussian KDE with the Silverman bandwidth [22]:

$$h = \bar{\sigma} \cdot E^{-1/(d_{\mathrm{pca}}+4)}, \qquad \bar{\sigma} = \frac{1}{d_{\mathrm{pca}}} \sum_{j=1}^{d_{\mathrm{pca}}} \sigma_j, \tag{7}$$

where σ_j is the empirical standard deviation of the j-th PCA dimension. The template for slot k is the *original*-space feature of the episode with the highest estimated log-density in PCA space:

$$\mathbf{t}_k = \mathbf{f}_k^{(e^*)}, \quad e^* = \arg\max_e \ \hat{p}\big(\mathrm{PCA}(\mathbf{f}_k^{(e)})\big). \tag{8}$$

Geometrically, e^* is the episode whose checkpoint lies closest to the mode of the demonstration cluster, the most *typical* example rather than a potentially non-existent centroid. This mirrors the mode-seeking interpretation of mean-shift clustering [23] and produces sharper, more discriminative matching targets than a mean template. All slot templates concatenated into the array and stored in the checkpoint database for live matching.

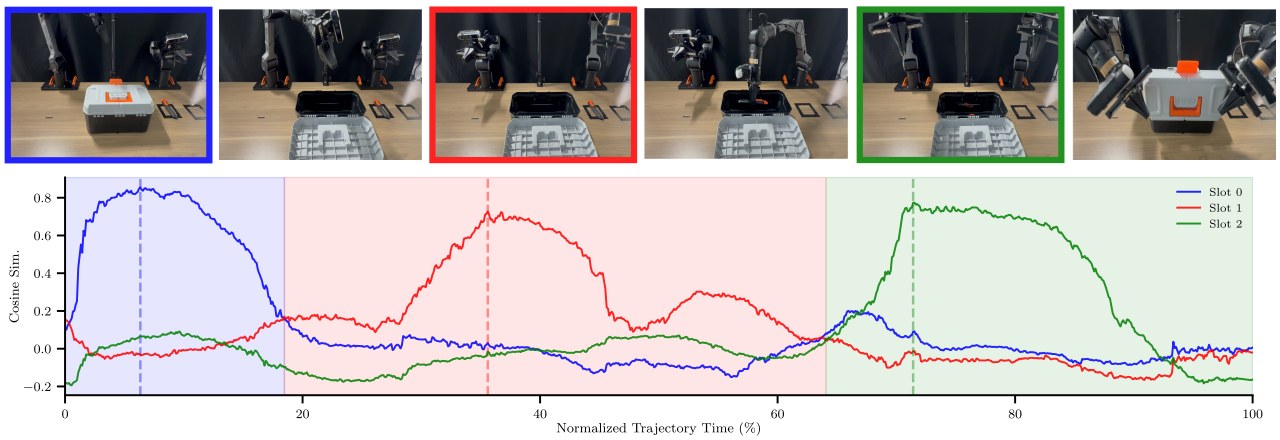

Fig. 4: Visualization of online similarity tracking for *Toolbox and Knife* task. The similarity curves denote the per-slot cosine similarity to reference checkpoint latent features. For each subtask, the respective chosen checkpoint is highlighted above.

D. Online Similarity Tracking and State Respawning

During inference, Rewind-IL runs two parallel bookkeeping processes: a per-slot similarity tracker that continuously monitors how close the current robot state is to each offline checkpoint, and a recovery executor that activates when the TIDE criterion fires.

Per-slot cosine similarity: At each step, a forward hook captures the policy's intermediate feature vector $\mathbf{f}_t$ (using the same extraction procedure as Eq. (6). The cosine similarity to each pre-normalised template is computed via a single batched dot product:

$$s_k(t) = \hat{\mathbf{f}}_t \cdot \hat{\mathbf{t}}_k, \qquad k = 1, \ldots, K, \tag{9}$$

where $\hat{\mathbf{f}}_t = \mathbf{f}_t / \|\mathbf{f}_t\|_2$. When all templates are pre-normalised and stacked as rows of $\hat{\mathbf{T}} \in \mathbb{R}^{K \times d}$, (9) reduces to the single matrix-vector product $\mathbf{s}(t) = \hat{\mathbf{T}}\,\hat{\mathbf{f}}_t$, making it efficient even for large K.

Peak detection and action snapshotting: Each slot k maintains a running maximum similarity and the action snapshot at which it was achieved:

$$\begin{aligned} s_k^{\max}(t) &= \max_{t' \le t} s_k(t'), \\ a_k^*(t) &= a_{t_k^*}, \quad t_k^* = \arg\max_{t' \le t} s_k(t'). \end{aligned} \tag{10}$$

Slot k is declared *peaked* once its similarity has failed to improve for more than Δ_{peak} consecutive steps:

$$\text{peaked}_k(t) = \mathbf{1}\big[t - t_k^* > \Delta_{\text{peak}}\big]. \tag{11}$$

Intuitively, a peaked slot means the robot has already passed through (or close to) the corresponding VLM-verified safe state and is now moving away from it. The *latest peaked slot* $k^* = \arg\max_{k:\,\text{peaked}_k(t)} t_k^*$ is the furthest confirmed safe waypoint the robot has traversed, the natural target for respawning. Figure 4 depicts the peaks across a task episode using per-slot cosine similarity.

Recovery and respawning: When $\text{is\_failing}(t) = 1$ and at least one slot is peaked, Rewind-IL executes the following protocol:

1. **Retrieve.** The recovery action $a_{\text{rec}} = a_{k^*}^*$ is fetched from the latest peaked slot k^*. This is the action that was executed when the robot's feature representation was most similar to the VLM-verified safe template $\hat{\mathbf{t}}_{k^*}$.
2. **Respawn.** The robot executes a_{rec} to physically steer toward the recovered state.
3. **Reset memory.** The temporal ensembler buffer and the action execution queue are cleared, invalidating stale plans that led to the failure. Slot data is *intentionally preserved* (i.e., $s_k^{\max}$ and a_k^* are not zeroed) so that repeated failures within the same episode consistently respawn.

This design cleanly separates *when* to recover, governed by the CP-calibrated TIDE threshold, from *where* to recover, governed by the offline VLM-verified feature library, and can be viewed as an *implicit manifold projection*: respawning to a VLM-verified checkpoint returns the system to a point on $\mathcal{M}$, after which the locally Lipschitz policy resumes reliable execution.

V. Empirical Evaluations

We evaluate Rewind-IL across six bimanual manipulation tasks on a real dual-arm robot and three tasks in the RoboCasa simulation environment [25] (as depicted in Figure 8) with ACT [4] as the baseline visuomotor action-chunked policy. Videos of all tasks are provided in the supplementary video. Our evaluation is organized around three central inquiries:

1) **Detection quality.** Does TIDE reliably distinguish genuine task failures from normal execution, compared to other OOD methods?
2) **Recovery effectiveness.** Does Rewind-IL's checkpoint-respawning loop improve task success rates under both natural policy failures and adversarial disturbances?
3) **Flow-Matching Policies**: Does the framework transfer to flow-matching action-chunked policies?

A. Setup and Metrics

Real World Setup: All real-world evaluations are conducted on a bimanual system featuring AgileX Piper 6-DoF manipulators with parallel grippers, operating at 30 Hz. We benchmark our framework across six tasks ranging from precise multi-step coordination (Cup & Box, Pencil & Notebook, Box & Wrench, Drawers & Hammer, Toolbox & Knife) to deformable object manipulation (Folding Towel). The base ACT models are trained on 50 expert demonstrations for 100k iterations on NVIDIA GeForce RTX 5090 GPUs.

Task	FAIL-Detect [12]			RND [24]			Clustering OOD			Mahalanobis			TIDE (Ours)		
	TPR	TNR	Acc.	TPR	TNR	Acc.	TPR	TNR	Acc.	TPR	TNR	Acc.	TPR	TNR	**Acc.**
Pick and Place	1.00	0.57	0.79	1.00	0.93	0.96	0.75	0.50	0.63	1.00	0.69	0.84	1.00	1.00	**1.00**
Pencil and Notebook	1.00	1.00	1.00	1.00	0.07	0.54	1.00	0.00	0.50	1.00	0.21	0.61	1.00	1.00	**1.00**
Box and Wrench	0.83	0.64	0.74	0.83	0.93	0.88	1.00	0.50	0.75	0.83	0.07	0.45	1.00	0.93	**0.96**
Drawers and Hammer	1.00	1.00	1.00	1.00	1.00	1.00	0.17	0.93	0.55	0.33	0.86	0.60	1.00	1.00	**1.00**
Toolbox and Knife	0.83	1.00	0.92	1.00	0.93	0.96	0.67	0.36	0.51	0.67	0.29	0.48	1.00	1.00	**1.00**
Folding Towel	1.00	0.08	0.54	0.75	0.33	0.54	0.75	0.50	0.63	1.00	0.00	0.50	0.50	0.92	**0.71**
Average	0.94	0.72	0.83	0.72	0.46	0.59	0.72	0.47	0.60	0.81	0.35	0.58	0.92	0.98	**0.95**

TABLE I: Failure detection performance on six standard ACT tasks. We report TPR (↑), TNR (↑), and Balanced Accuracy (↑). Highlighted columns denote TIDE.

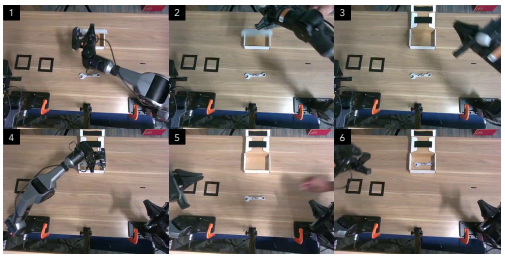

Fig. 5: Rewind-IL on *Box and Wrench* (lifting the top of the box and placing the wrench inside).

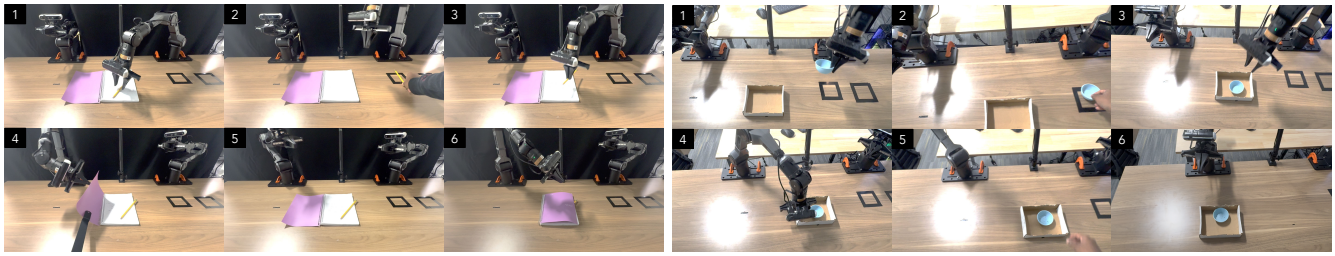

Fig. 6: Rewind-IL with perturbations and recovery on *Pencil and Notebook* (bookmarking the page with a pencil and flipping over the front cover) and *Cup and Box* (placing a cup into a tray and moving both to the side of the table by grasping the tray).

All ACT models incorporate joint torque feedback $\boldsymbol{\tau}_t \in \mathbb{R}^n$ as an additional proprioceptive input (Force-Aware ACT [26], [27]), augmenting the standard observation space to $\hat{\mathbf{a}}_{t:t+K} = \pi_\theta(\mathbf{I}_t, \mathbf{j}_t, \boldsymbol{\tau}_t)$; this torque-augmented representation has been shown to improve performance in bimanual manipulation tasks [26], [27].

Evaluation Protocols:

- *Detection.* Evaluated on unperturbed rollouts labelled post-hoc; thresholds calibrated on a disjoint held-out set of 10 successful episodes with no failure data.
- *Recovery (+ Perturb).* An operator adversarially intervenes during every rollout (e.g., nudging objects, reopening closed drawers) until a sub-task visibly fails, then stops. Without recovery, this drops ACT success rates to 15–25%. This setup is directly analogous to the disturbance protocol of MoE-DP [28], where objects are reset after successful grasps to force re-attempt.

Baselines for failure detection: We compare TIDE against four post-hoc baselines that utilize successful calibration rollouts to detect OOD states:

- **FAIL-Detect** [12]: A recent state-of-the-art (SOTA) sequential failure detector that distills policy inputs and outputs into specific scalar signals (e.g., latent features) and flags failures using conformal prediction.
- **RND / FIPER** [11], [24]: We evaluate Random Network Distillation (RND) applied to the policy's encoder embeddings. This serves as a direct proxy for the core mechanism of FIPER [11], a SOTA failure detector, adapted here to handle deterministic policies.
- **Clustering-Based OOD**: The OOD score is the minimum Euclidean distance to 64 K-means centroids on PCA-project ACT calibration embeddings [12].
- **Mahalanobis Similarity**: Fits a single Gaussian to the PCA-projected calibration embeddings and reports the Mahalanobis distance [14].

All baselines use CP-based thresholding (Section III) with $\alpha = 0.001$. We report Balanced Accuracy $= (\text{TPR} + \text{TNR})/2$ for detection and task completion rate for recovery.

B. Evaluating TIDE as a Failure Detector

Performance comparison: TIDE achieves perfect balanced accuracy (1.00) on four tasks and near-perfect on one more (Box & Wrench, 0.96), averaging 0.95 across all six, an absolute improvement of +0.35, +0.37, +0.36, and +0.12 over Clustering OOD (0.60), Mahalanobis (0.58), RND (0.59), and the best baseline FAIL-Detect (0.83), respectively (see Table I for the full per-task TPR/TNR breakdown).

Why baselines fail: Simpler embedding-based baselines (Clustering OOD and Mahalanobis) fail fundamentally due to severe over-triggering, misclassifying natural embedding drift as failures (e.g., Clustering TNR $= 0.00$ on Pencil & Notebook; Mahalanobis TNR $= 0.00$ on Folding Towel). Recent state-of-the-art methods like RND and FAIL-Detect also struggle due to their underlying reliance on fixed reference distributions. RND suffers from severe false positives on specific tasks, yielding a TNR of just 0.07 on Pencil & Notebook and dragging its overall average accuracy down to 0.59. While FAIL-Detect offers a more robust formulation and performs best among the baselines (average accuracy 0.83), it remains highly vulnerable to over-triggering when the robot encounters safe but novel variations, as evidenced by its near-zero TNR (0.08) on the deformable Folding Towel task. Unlike these methods that measure post-hoc distance from a static historical dataset, TIDE monitors *intra-policy self-consistency*. By dynamically tracking the agreement of the policy's own action-chunk predictions, TIDE is substantially more robust to the benign representation drift that plagues static distribution matching. Consequently, TIDE maintains strong performance across all tasks (average accuracy 0.95), and even on the challenging Folding Towel task it largely avoids over-triggering (TNR $= 0.92$) where baseline metrics break down severely.

TABLE II: Task success rates (%, 20 rollouts each). "Perturb" indicates whether disturbance was applied. Bold marks the best result within each setting.

Task	Perturb	Method	
		ACT	ACT + Rewind-IL
Cup and Box	×	80	**90**
	✓	15	**85**
Pencil and Notebook	×	70	**85**
	✓	25	**80**
Box and Wrench	×	65	**75**
	✓	15	**80**
Drawers and Hammer	×	55	**75**
	✓	20	**75**
Toolbox and Knife	×	70	**90**
	✓	15	**85**
Folding Towel	×	60	**65**
	✓	20	**55**
Average	×	66.7	**80.0**
	✓	18.3	**76.7**

TABLE III: Task success rates (%, 50 rollouts each) in RoboCasa.

Task	ACT	ACT + Rewind-IL
Close Toaster Oven Door	55	**70**
Open Stand Mixer Head	60	**80**
Close Fridge	55	**70**

TABLE IV: FM vs FM + Rewind-IL success rates (%).

Task	Perturb	Method	
		FM	FM + Rewind-IL
Cup and Box	×	75	**90**
	✓	20	**90**
Pencil and Notebook	×	70	**90**
	✓	30	**65**
Toolbox and Knife	×	65	**75**
	✓	10	**70**
Average	×	70.0	**85.0**
	✓	20.0	**75.0**

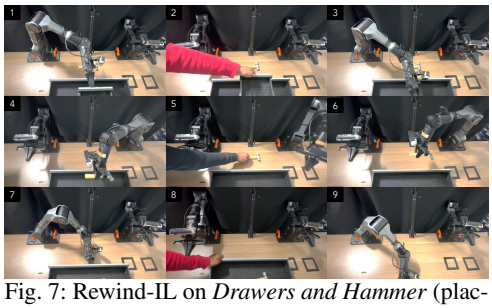

Fig. 7: Rewind-IL on *Drawers and Hammer* (placing an hammer inside and closing drawers).

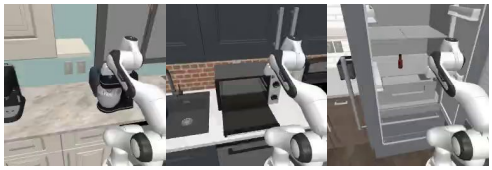

Fig. 8: Task layouts for RoboCasa environment.

C. Task Success Rate: Ablation and Main Results

Detection and respawning are not independently ablatable: detection alone reduces to episode termination (the ACT baseline), and respawning requires a detection trigger to activate. The detection ablation is reported in Table I; Tables II–III measure the integrated benefit.

Effect of Rewind-IL: Tables II and III show results for the real-world and simulation conditions respectively. ACT + Rewind-IL consistently outperforms the plain ACT baseline across all nine tasks, with improvements of $+5$ to $+20$ percentage points (pp) in the real world and $+15$ to $+20$ pp in simulation. The largest gains occur on the *Drawers & Hammer* ($+20$ pp) and *Toolbox & Knife* ($+20$ pp) tasks, which require precise multi-step grasp sequences where a single slip compounds into irrecoverable failure, exactly where checkpoint respawning is most effective (Figs. 5 and 7). In simulation the consistent $+15$ pp lift across all three RoboCasa tasks demonstrates that the benefit is not an artifact of our specific hardware setup.

Robustness under adversarial disturbance: The "Perturb" conditions apply the adversarial disturbance protocol to *every* rollout, providing a worst-case stress test of recovery. Without recovery, ACT + Perturb succeeds only 15–25% of the time, and the disturbance reliably prevents task completion. Coupling disturbance with Rewind-IL (ACT + Perturb + Rewind-IL) recovers most of this loss: five of six tasks reach 75–85%, approaching the performance of the undisturbed ACT + Rewind-IL condition. This demonstrates that Rewind-IL can handle externally-induced failures just as effectively as naturally-occurring ones: the checkpoint database provides a grounded recovery state regardless of failure cause (Fig. 6). The exception is Folding Towel (55%): limited demonstrations (50) leave the policy under-trained, producing inherently inconsistent predictions that weaken TIDE's failure discrimination and reduce recovery frequency.

Repeated disturbance resilience: The + Perturb comparison applies a single adversarial disturbance per-rollout, since methods without recovery fail on the first intervention and further disturbances add no comparative signal. In practice, Rewind-IL places no architectural limit on the number of recovery cycles within an episode. Moreover, because the slot tracker dynamically follows the current scene's similarity profile, clearing the previously peaked slot when a newer slot surpasses it, the system is not constrained to return to the same checkpoint on every recovery. If successive disturbances roll the scene back to progressively earlier states, the tracker naturally selects the most appropriate VLM-verified checkpoint for the current configuration.

Computational overhead: Each pipeline stage is timed over 10 independent CUDA runs; per-stage mean and standard deviation are reported in Table V. The three stages (TIDE scoring, cosine-similarity matching, slot bookkeeping) total under 0.2 ms, less than 1% of ACT inference time, confirming real-time viability.

D. Adaptation to Flow-Matching Robot Policies

Experimental setup: We replace the CVAE decoder in ACT with a flow-matching (FM) action head [29], [30] while keeping the transformer encoder architecture identical; TIDE monitors the FM policy's action chunk predictions without any other modification. Detection results are reported in Table VI; task success results in Table IV.

Detection quality transfers to FM: TIDE achieves near-perfect balanced accuracy (0.97–1.00) across the three FM tasks, averaging an impressive 0.99 (Table VI). RND also performs competitively on these tasks (avg. 0.96), avoiding the severe bimodal collapse observed on standard ACT policies, which suggests that the FM policy's action distribution provides cleaner OOD signals. Notably, while FAIL-Detect struggles on the Toolbox & Knife task (acc. 0.79) due to a lower TPR (0.75), TIDE achieves perfect detection (acc. 1.00; TPR 1.00, TNR 1.00) on the exact same task. This confirms that TIDE's self-consistency metric transfers exceptionally well to deterministic flow-matching architectures.

Rewind-IL delivers consistent gains: The ACT w. FM + Rewind-IL method improves over the FM baseline by $+10$–$+20$ pp across all three tasks (Table IV), reaching 90% on Cup & Box and Pencil & Notebook, and 75% on Toolbox & Knife. Notably, on the first two tasks the absolute performance of ACT w. FM + Rewind-IL (90%) matches

TABLE V: Failure handling pipeline timing (10 samples, CUDA).

Stage	**Mean $\pm$ Std** (10^{-5} **s**)
TIDE Computation	3.7 ± 1.0
Cos-sim Computation	9.6 ± 0.7
Slot Bookkeeping	5.6 ± 0.8
Total	22.0 ± 7.2

TABLE VI: Failure detection on three ACT w/ Flow Matching tasks. TPR (↑), TNR (↑), Balanced Accuracy (↑).

Task	**FAIL-Detect [12]**			**RND [24]**			**Clustering OOD**			**Mahalanobis**			**TIDE (Ours)**		
	TPR	TNR	Acc.	TPR	TNR	Acc.	TPR	TNR	Acc.	TPR	TNR	Acc.	TPR	TNR	**Acc.**
Pick and Place	1.00	0.93	0.97	1.00	1.00	1.00	0.25	0.88	0.56	1.00	0.63	0.81	1.00	1.00	**1.00**
Pencil and Notebook	1.00	0.93	0.97	1.00	0.93	0.97	1.00	0.13	0.57	1.00	0.53	0.77	1.00	0.93	**0.97**
Toolbox and Knife	0.75	0.83	0.79	1.00	0.83	0.92	0.13	0.33	0.23	0.88	0.33	0.60	1.00	1.00	**1.00**
Average	0.92	0.9	0.91	1.00	0.92	0.96	0.46	0.45	0.45	0.96	0.50	0.73	0.92	0.92	**0.99**

or exceeds the best ACT condition (Table II), suggesting that flow-matching and recovery are complementary: FM provides richer action distributions that improve nominal execution, while Rewind-IL handles the residual failures.

Robustness under disturbance: Under adversarial disturbance, ACT w. FM + Perturb + Rewind-IL achieves 90% on Cup & Box, 70% on Toolbox & Knife, but only 65% on Pencil & Notebook. The gap suggests that the FM policy's tighter action distributions leave less margin for recovery on geometrically demanding tasks: once disturbed, those tasks require a more precise return to the checkpoint state than the respawned action can reliably achieve. Flow-matching trains a deterministic transport map concentrated around the training demonstration manifold; in OOD states induced by adversarial perturbation, this concentration reduces coverage of the action space and limits the policy's ability to synthesize the precise corrective motions to re-establish the checkpoint configuration, a limitation less pronounced in CVAE-based ACT, whose stochastic latent space provides broader action diversity. Notably, undisturbed ACT w. FM + Rewind-IL achieves 90% on all three tasks, suggesting that Rewind-IL's core recovery benefit holds broadly across the tested architectures when failures arise from the policy's own limitations rather than external intervention.

VI. Conclusions and Future Work

We introduce Rewind-IL, an online safeguard framework designed for zero-shot failure detection and recovery for action-chunked imitation learning policies. This is enabled by leveraging temporal inter-chunk discrepancy estimates as a lightweight failure signal and combining it with semantically grounded checkpoint selection for recovery. Avenues for future research include (i) fusing TIDE with complementary signals to improve detection accuracy when policy-intrinsic biases suppress the discrepancy signal; (ii) dynamic slot-count mechanisms for variable-subtask settings; (iii) collision-free motion generation [31] for respawning trajectories; and (iv) scene-state verification before committing to recovery.

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, 2020.

[2] W. Zhi, T. Lai, L. Ott, and F. Ramos, "Diffeomorphic transforms for generalised imitation learning," in *Learning for Dynamics and Control Conference, L4DC*, 2022.

[3] W. Zhi, T. Zhang, and M. Johnson-Roberson, "Learning from demonstration via probabilistic diagrammatic teaching," in *IEEE International Conference on Robotics and Automation*, 2024.

[4] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, 2023.

[5] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[6] S. Ross, G. Gordon, and D. 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, PMLR, 2011.

[7] Z. He, Y. Cao, and M. Ciocarlie, "Uncertainty Comes for Free: Human-in-the-Loop Policies with Diffusion Models," 2025.

[8] A. Rosasco, F. Ceola, G. Pasquale, and L. Natale, "KDPE: A Kernel Density Estimation Strategy for Diffusion Policy Trajectory Selection," 2025.

[9] Z. Song, Y. Yang, Z. Zhou, H. Yu, X. Zheng, Y. Wang, and R. Xiong, "Structurally-Fused Goal-Aware Network Distillation for Anomaly Detection in Diffusion Policy," in *2025 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 95–102, 2025.

[10] H. Cheng, T. Zheng, Z. Ma, T. Zhang, M. Johnson-Roberson, and W. Zhi, "DOSE3: diffusion-based unified out-of-distribution detection on $\mathbb{SE}(3)$ trajectories," *IEEE Robotics Autom. Lett.*, vol. 11, no. 2, pp. 1706–1713, 2026.

[11] R. Römer, A. Kobras, L. Worbis, and A. P. Schoellig, "Failure Prediction at Runtime for Generative Robot Policies," 2025.

[12] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. Ambrus, H. Nishimura, and M. Itkina, "Can We Detect Failures Without Failure Data? Uncertainty-Aware Runtime Failure Detection for Imitation Learning Policies," 2025.

[13] A. N. Angelopoulos and S. Bates, "Conformal Prediction: A Gentle Introduction," *Foundations and Trends® in Machine Learning*, vol. 16, no. 4, pp. 494–591, 2023.

[14] C. Agia, R. Sinha, J. Yang, Z.-a. Cao, R. Antonova, M. Pavone, and J. Bohg, "Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress," 2024.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," 2019.

[16] Z. Wang, J. Loo, and D. Hsu, "Fare: Failure Resilience in Learned Visual Navigation Control," 2025.

[17] B. Santhanam, A. Mitrevski, S. Thoduka, S. Houben, and T. Hassan, "Reliable Robotic Task Execution in the Face of Anomalies," *IEEE Robotics and Automation Letters*, vol. 11, no. 1, pp. 314–321, 2026.

[18] C. Ma, G. Yang, K. Lu, S. Xu, B. Byrne, N. Trigoni, and A. Markham, "CycleVLA: Proactive Self-Correcting Vision-Language-Action Models via Subtask Backtracking and Minimum Bayes Risk Decoding," 2026.

[19] Y. Ali, G. Patlin, K. Kothuri, M. Z. Irshad, W. Liang, and Z. Kira, "EVE: A Generator-Verifier System for Generative Policies," 2025.

[20] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy, "From Foresight to Forethought: VLM-In-the-Loop Policy Steering via Latent Alignment," 2025.

[21] J. Kwok, X. Zhang, M. Xu, Y. Liu, A. Mirhoseini, C. Finn, and M. Pavone, "Scaling Verification Can Be More Effective than Scaling Policy Learning for Vision-Language-Action Alignment," 2026.

[22] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Routledge, 2018.

[23] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[24] N. He, S. Li, Z. Li, Y. Liu, and Y. He, "ReDiffuser: reliable decision-making using a diffuser with confidence estimation," in *Proceedings of*

the 41st International Conference on Machine Learning, pp. 17921–17933, JMLR.org, 2024.

[25] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu, "Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots," in *International Conference on Learning Representations (ICLR)*, 2026.

[26] Z. Li, Y. Zhou, R. Qiu, H. Wu, G. Ren, and W. Zhi, "TriPilot-FF: Coordinated Whole-Body Teleoperation with Force Feedback," 2026.

[27] R. Watanabe, M. Alvarez, P. Ferreiro, P. Savkin, and G. Sano, "FTACT: Force Torque aware Action Chunking Transformer for Pick-and-Reorient Bottle Task," 2025.

[28] B. Cheng, T. Liang, S. Huang, M. Shao, F. Zhang, B. Xu, Z. Xue, and H. Xu, "MoE-DP: An MoE-Enhanced Diffusion Policy for Robust Long-Horizon Robotic Manipulation with Skill Decomposition and Failure Recovery," 2025.

[29] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow Matching for Generative Modeling," 2023.

[30] K. Black *et al.*, "pi_0: A Vision-Language-Action Flow Model for General Robot Control," 2026.

[31] W. Zhi, I. Akinola, K. van Wyk, N. Ratliff, and F. Ramos, "Global and reactive motion generation with geometric fabric command sequences," in *IEEE International Conference on Robotics and Automation, ICRA*, 2023.