What Do VLAs Actually Learn through In-Context Failure Conditioning?

Jiajun Liu[1,2,3*] **Jieming Li**[2] **Zi Zhuang**[2] **Hang Yu**[1]
Qingli Chen[1] **Liu Cao**[2,3] **Yingxi Lu**[2] **Ruoqu Chen**[2,3]
Yuhang Cao[2] **Chenyu Zhang**[2,3] **Yankai Lin**[1] **Mengdi Xu**[2,3 †]
[1] Renmin University of China [2] Tsinghua University [3] Shanghai Qi Zhi Institute
jedward.jiajun@gmail.com xumd@tsinghua.edu.cn

Abstract

Vision-language-action (VLA) policies often repeat the same mistakes after failure, as they lack memory of prior attempts. Whether a VLA can learn from its own failure through in-context conditioning, and what information in failure contexts is useful, remains unclear. We study this with RoboRetry, a controlled probe that inserts failure information directly into the policy context window via block-diagonal attention. Across 12 RLBench tasks, we evaluate failure-conditioned retry under a paired protocol and compare it with success-demonstration context under the same retry budget. Our results show that failure-conditioned context improves cumulative retry success over matched blind retry. Our interventions point to two sources of retry lift: a slot-presence effect that provides a structural retry cue, and a content-dependent effect driven primarily by high-quality corrective language. We further identify remaining limits around semantic target/progress re-grounding and contact-precision recovery, where free-form corrective text is not sufficiently actionable for the required recovery. To support further study, we release FailureSlot, a dataset of 1,334 naturally occurring VLA failure trajectories with human-labeled corrective annotations and an 8-category taxonomy. Together, the probe and dataset provide a starting point for studying failure-driven in-context robot learning, where adaptation happens through experience in context rather than through parameter updates.

1 Introduction

In-context learning (ICL), originally developed and widely successful in natural language processing (NLP), has been studied primarily in settings where the context consists of correct demonstrations. In these settings, the attention pathway learns to copy and complete regularities from the context (Olsson et al., 2022; Crosbie and Shutova, 2025), and has been shown to rely more on format than on semantic content (Min et al., 2022). By contrast, the role of failure-conditioned context, particularly when the context contains the model's own prior mistakes, remains largely unexplored.

In robot learning, enabling models to learn from their own failures through in-context conditioning is an appealing goal, as it would allow embodied agents to self-improve in the environment without additional parameter updates. However, existing vision-language-action (VLA) policies are limited in this regard: they either operate with short context windows that capture only a few recent steps (Black et al., 2024; Kim et al., 2024; Pertsch et al., 2025), or rely on conditioning over successful robot or human demonstrations (Jain et al., 2024; Reed et al., 2022). As a result, their rollouts are blind to prior failures. To compensate, most approaches instead route failure signals through auxiliary modules before exposing them to the policy, for example by learning a value function via reinforcement learning (Dai et al., 2025; Lin et al., 2025; Intelligence et al., 2025; Lu et al., 2025a; Syed et al., 2025).

In this work, we ask whether a VLA can use its own failed attempt as in-context evidence for a subsequent retry. We introduce RoboRetry, a controlled probe that inserts a failure slot before the observations from the current rollout and exposes this slot to the policy through block-diagonal attention (Sridhar et al., 2025b). After a failure occurs, RoboRetry populates the slot with failure context captured at the moment of failure, including a keyframe image, proprioceptive state, language description, and action chunk, and then retries the task. Importantly, RoboRetry performs no parameter updates at inference time and introduces no auxiliary recovery module or action interpolation. The policy weights remain fixed, and attention over

[*] Work done during an internship at Tsinghua IIIS and Shanghai Qi Zhi Institute.
[†] Corresponding author.

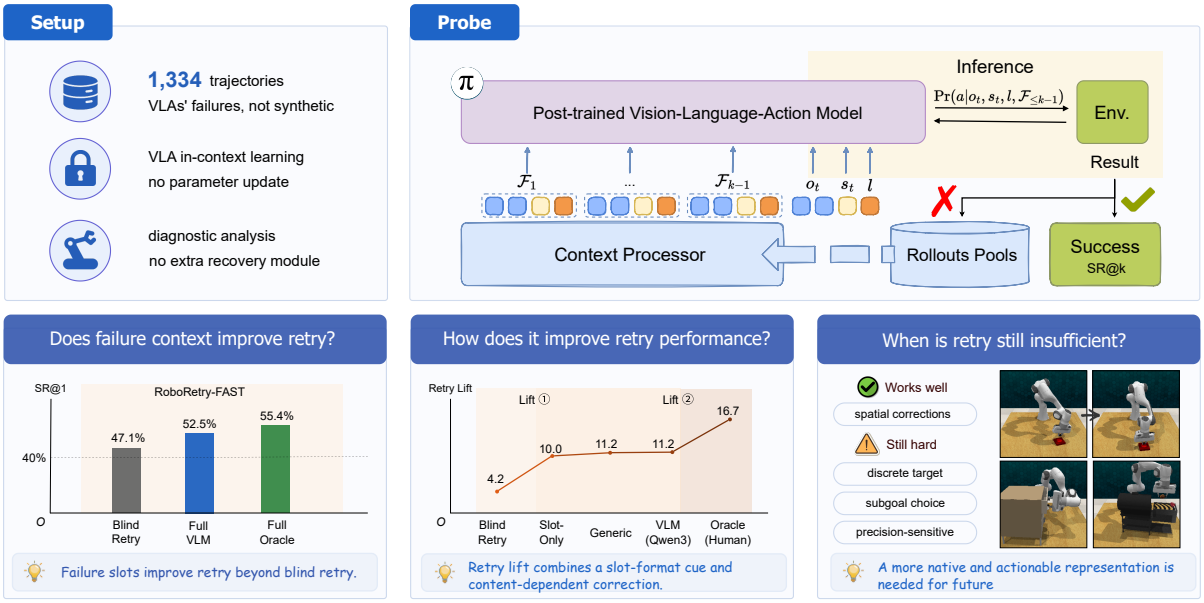

Figure 1: **Overview of RoboRetry.** *Setup:* FailureSlot contains 1,334 naturally occurring VLA failure trajectories, each paired with human Oracle corrective annotation as a usable in-context slot. *Probe:* RoboRetry inserts prior failure slots $\mathcal{F}_{\leq k-1}$ (keyframe image, language, state, and action chunk) into a post-trained VLA context window, with fixed policy weights, no parameter update, and no auxiliary recovery module. We instantiate the probe on Pi0-FAST as RoboRetry-FAST. *Findings:* the evaluation asks whether failure slots improve retry beyond blind retry, how much lift is associated with slot format versus language quality, and when retry remains insufficient.

the failure slot is the only pathway through which the previous attempt can influence subsequent actions. This makes RoboRetry a controlled diagnostic probe for studying failure-conditioned in-context adaptation.

We instantiate RoboRetry on Pi0-FAST (Black et al., 2024; Pertsch et al., 2025) as a probe with 12 RLBench tasks (James et al., 2020). The Pi0-FAST variants are additionally finetuned to process contextual inputs that include failure trajectories. Our analysis addresses three questions. First, does conditioning on a previous failure improve retry performance beyond blind retry? Second, if so, which parts of the multimodal failure slot are actually usable for action generation? Third, where does failure-conditioned retry still fall short?

Our contributions are as follows.

We present a controlled in-context failure-conditioning probe. RoboRetry places a policy's own failed attempt in its native context window, with no auxiliary recovery module, parameter update, or action interpolation. We pair this probe with FailureSlot, a set of 1,334 naturally occurring VLA policy failures with human Oracle corrective descriptions, formatted as context slots and organized under an 8-category taxonomy. (§3)

We show that failure context provides useful retry information. Under a paired retry protocol, failure-conditioned context improves cumulative retry success over matched blind retry. We also include success-demonstration context as a reference under the same retry budget. In our setting, full failure context reaches higher cumulative retry success than this success-context reference, suggesting that episode-specific failure evidence is especially useful for retry.

We analyze what information inside the failure slot is usable. Our interventions suggest that retry lift combines a structural context cue with content-dependent corrective language. Oracle Text carries much of the usable retry signal, while the full slot with failure-moment image, state, and action evidence achieves the strongest overall retry performance. We further identify remaining hard cases, including target/subgoal selection and precision-sensitive contact tasks where free-form text often underspecifies the required action change. (§4.2–§4.3)

2 Related Work

In-context learning and context format. Prior work shows that in-context learning can depend strongly on the structure and distribution of the context, even more than on label semantics (Min et al., 2022; Alazraki et al., 2025). Other work identifies attention-based copy and completion circuits that support learning from demonstrations in context (Olsson et al., 2022; Crosbie and Shutova, 2025; Cho et al., 2024). These studies primarily use correct demonstrations and token-level prediction targets. Our setting asks a different question: what does the same context pathway carry when the context is the model's own prior failure rather than an example to imitate?

Learning from failure. Mistakes can improve model behavior when paired with feedback, verifiers, or suitable supervision (Alazraki et al., 2025; Huang et al., 2023; Tong et al., 2024; Wang et al., 2024). In robotics, failure information is often routed through diagnosis modules, value functions, reinforcement learning, or test-time adaptation before it affects the policy (Dai et al., 2025; Lin et al., 2025; Liu et al., 2023; Zeng et al., 2025; Gu et al., 2025; Syed et al., 2025; Lu et al., 2025a; Intelligence et al., 2025, 2026). Separately, robot ICL work and memory systems study how policies use *success* demonstrations in context (Sridhar et al., 2025b; Wu et al., 2025; Sridhar et al., 2025a; Shi et al., 2025a; He et al., 2026). We remove recovery modules and parameter updates, and expose the failed attempt directly to a fixed VLA through attention.

Robot failure data. Existing robot-failure datasets (Lin et al., 2025; Duan et al., 2024; Pacaud et al., 2025; Grislain et al., 2025; Lu et al., 2025b; Wu et al., 2024; Zeng et al., 2025) provide valuable supervision for failure detection, diagnosis, explanation, QA, or recovery, and many are generated through synthetic perturbations. FAILURESLOT targets a different data unit: each naturally occurring failed VLA rollout is stored as an in-context slot with failure-moment image, proprioceptive state, forward action chunk, and corrective language. This format lets us intervene on structure, modality content, language quality, and episode alignment, and directly test whether prior failure evidence can ground the next action.

3 Experiment Setup

3.1 ROBORETRY architecture

A standard VLA maps the current observation o_t, proprioceptive state s_t, and language instruction l to an action chunk $a_{t:t+H-1}$, $(o_t, s_t, l) \mapsto a_{t:t+H-1}$. Here, H is the action horizon.

ROBORETRY extends this input with k prior failed attempts:

$$a_{t:t+H-1} = \pi(o_t, s_t, l, \mathcal{F}_{0:k-1}),$$

where each failure slot $\mathcal{F}_i = (I_i, d_i, s_i, \bar{a}_i)$ contains a failure keyframe I_i, a language description d_i, a proprioceptive state s_i, and an action chunk $\bar{a}_i$ from the failed attempt. In the main ROBORETRY-FAST probe, the slot is tokenized by the same Pi0-FAST pipeline as the query observation: SigLIP (Zhai et al., 2023; Tschannen et al., 2025) encodes the keyframe image, the language tokenizer encodes the description and state, and FAST (Pertsch et al., 2025) encodes the action chunk. Thus, the failure slot differs from the query in content, not in representation.

We use block-diagonal attention (Sridhar et al., 2025b) so that the query block can attend to previous failure slots while preserving the observation-block structure of Pi0-FAST. Training supervises only the query action tokens. Failure actions are provided as context and are never used as prediction targets, so the training objective remains the original Pi0-FAST action-token prediction loss on the query trajectory.

3.2 Failure dataset and taxonomy

We collect FAILURESLOT by deploying trained baseline VLA policies across the 12 RLBench tasks. The dataset contains 1,334 naturally occurring policy failures. Each record is paired with a human-written *oracle description* from one of seven annotators. The failure keyframe is human-selected as the frame where the task-critical error first becomes visually identifiable; for temporally extended failures, annotators choose the frame closest to the irreversible divergence from a successful execution, and we store the nearby action chunk. So, for each failure, we store the failure keyframe, proprioceptive state, action chunk, and corrective descriptions in the same slot format used by ROBORETRY, making trajectories directly usable for retry context.

Table 1 clarifies the unit of supervision targeted by FAILURESLOT. Existing robot-failure datasets

Table 1: **Comparison with existing robot failure datasets.** *Synth.* = programmatically perturbed; *Nat.* = naturally occurring during policy execution; *ICL* = structured as an in-context failure slot.

Dataset	Src	Env	Ann.	#F	ICL
FailSafe	Synth.	Sim	Labels	131K	✗
AHA	Synth.	Sim	QA	49K	✗
Guardian	Synth.	Both	CoT	37K	✗
RoboFAC	Synth.	Both	QA	8.9K	✗
RoboFAC	Nat.	Real	QA	480	✗
REFLECT	Nat.	Both	LLM	130	✗
FAILURESLOT (Ours)	**Nat.**	**Sim**	**Human+VLM**	**1,334**	✓

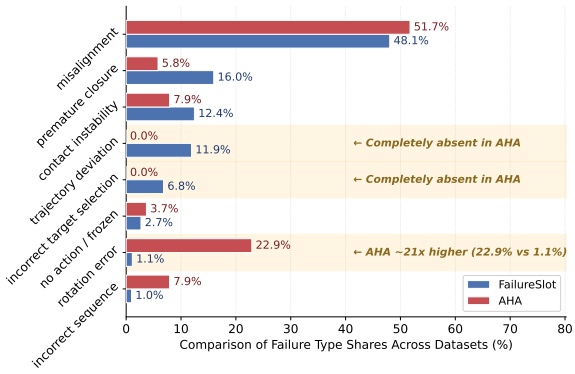

Figure 2: **Naturally occurring VLA failures differ from synthetic perturbations.** Failure-type distributions under the shared 8-category mechanism taxonomy across all available episodes from both FailureSlot and AHA dataset. AHA over-represents rotation errors and omits trajectory-deviation and incorrect-target-selection failures that appear in policy rollouts. Dataset-level positioning is summarized in Table 1.

provide valuable labels, QA pairs, chain-of-thought traces, or recovery descriptions. FAILURESLOT complements them by aligning every failure record with the policy input interface used at retry time. This lets the same bank serve both as a released data resource and as an intervention substrate for testing slot presence, modality content, text quality, and episode alignment while keeping policy weights fixed. The annotation protocol follows this use case: annotators inspect the failed rollout and task instruction, select or verify the task-critical failure moment, then write two fields: what went wrong, and what correction could help the next attempt; Appendix B gives the full record format and annotation instructions.

To characterize the bank, we derive a shared 8-category mechanism taxonomy: misalignment, premature closure, contact instability, trajectory deviation, incorrect target selection, rotation error, incorrect sequence, and no action/frozen. The taxonomy is constructed through LLM-assisted thematic coding followed by manual audit, and is used only for dataset analysis rather than as retry supervision. The collected failures are dominated by misalignment (48.1%), followed by premature closure (16.0%), contact instability (12.4%), trajectory deviation (11.9%), and incorrect target selection (6.8%), showing that the bank covers both coarse spatial errors and contact- or decision-level failures.

We use AHA (Duan et al., 2024) as a synthetic contrast set by mapping both sources into the same taxonomy. Figure 2 reports full-dataset distributions and shows that synthetic perturbations over-represent rotation errors while omitting trajectory-deviation and incorrect-target-selection failures observed in policy rollouts. We therefore use naturally occurring failures as the primary retry evidence base, with AHA serving only as a distributional contrast.

3.3 Training and inference

Our main probe fine-tunes Pi0-FAST (Black et al., 2024; Pertsch et al., 2025) on 12 RLBench tasks (James et al., 2020). During training, the number of failure slots is sampled as $k \sim \text{Uniform}\{0, 1, 2\}$, and the slots are drawn from FAILURESLOT as described in Section 3.2.

We refer to the Pi0-FAST instantiation of this failure-conditioned policy as ROBORETRY-FAST. For comparability, the success-only Pi0-FAST baseline and ROBORETRY-FAST are trained on the same 25 successful demonstrations per task. ROBORETRY-FAST additionally samples in-context failure slots from a 50-failure-per-task bank, with the number of slots sampled as $k \sim \text{Uniform}\{0, 1, 2\}$ during training.

At deployment, each retry tuple is first attempted with k=0, i.e., without any prior-failure context. If the attempt fails, we restore the scene to the same initial state and retry with the accumulated failure context from previous attempts. Across evaluation conditions, we vary the slot contents: the failure description is generated by Qwen3-VL (Bai et al., 2025), written by a human oracle, replaced with generic text, or removed; non-text fields are included or ablated according to the condition. Model weights remain fixed throughout deployment. We therefore attribute retry-time behavioral changes to the provided context, while matched blind and empty-slot controls quantify reset effects

and other residual retry variation.

Full hyperparameters, hardware, and implementation details are provided in Appendix A.

3.4 Evaluation protocol

We evaluate retry performance on fixed retry tuples, each defined by a task, variation, initial scene seed, and checkpoint. SR@k denotes cumulative success within the first k attempts on the same tuple: SR@1 is the first attempt with no prior-failure context, while SR@3 allows two retries after failure. Retry-lift is defined as SR@3 − SR@1. Unless otherwise stated, each condition is evaluated on 20 seeds × 12 tasks, giving 240 paired tuples per condition.

The retry protocol is paired across context conditions. For a given tuple, the policy first attempts the task without a failure slot. If the attempt fails, the scene is restored to the same initial state and the policy retries with the corresponding accumulated context. The policy weights are fixed throughout evaluation; no parameter update, auxiliary recovery module, or action interpolation is introduced at retry time. This design fixes the task, scene seed, checkpoint, and retry budget across conditions, while matched blind and empty-slot controls quantify residual retry and reset effects.

Unless stated otherwise, each table reports results from a pre-specified checkpoint at the matched training step. For uncertainty estimates, we use a task-stratified paired bootstrap: within each task, we resample the evaluated tuples with replacement and then macro-average over the fixed 12-task suite. Appendix F reports 95% confidence intervals for the main contrasts. These controls define the scope of our empirical analysis: the primary comparison is failure-conditioned retry versus matched blind retry, while the remaining context manipulations are used diagnostically to study slot format, language quality, and episode alignment. Code and evaluation scripts are available at `https://github.com/jedward225/RoboRetry`.

4 VLAs Learning from Failure Context

With this protocol, we organize the results around three questions. First, does failure context improve retry beyond simply trying again? Second, which parts of the failure slot account for the improvement? Third, when does failure-conditioned retry remain insufficient?

4.1 Does failure context improve retry?

Takeaways

Failure-conditioned context improves cumulative retry success beyond blind retry. On RoboRetry-FAST, Full VLM and human Oracle failure contexts improve SR@3 over the matched blind-retry baseline by +5.4 and +8.3 points, respectively.

Table 2 compares blind retry with failure-conditioned retry under the paired protocol. Blind retry measures how much success can be recovered by resetting the scene and attempting the task again without exposing the policy to the failed trajectory. Failure-conditioned retry instead inserts the previous failed attempt into the policy context window while keeping the policy weights fixed.

On the main Pi0-FAST probe, the matched RoboRetry-FAST blind-retry baseline improves from 40.8 SR@1 to 47.1 SR@3, showing that some failures are recoverable by another attempt alone. Adding failure context yields larger cumulative success: Full VLM reaches 52.5 SR@3 and Full Oracle reaches 55.4 SR@3, corresponding to $+5.4$ and $+8.3$ points over RoboRetry-FAST blind retry. Thus, the prior failed trajectory provides retry-specific evidence beyond the reset-and-retry effect.

RICL (Sridhar et al., 2025b) is included as a success-context reference. It receives successful demonstrations at each attempt and never observes the failed trajectory, and it contrasts task-level success evidence with episode-specific failure evidence under the same retry budget. In this reference comparison, RICL improves from 22.9 SR@1 to 31.2 SR@3, but remains below the full failure-context rows. This suggests that the immediately preceding failed attempt can provide especially actionable evidence for retry in our setting.

Having established that failure slots improve retry beyond blind retry, we next ask what information inside the slot contributes to the observed changes in behavior.

4.2 What makes a failure slot useful?

Takeaways

Retry lift has two components: a structural cue from the presence of a failure slot, and a content-dependent refinement that is strongest with human corrective text.

Table 2: **Main retry comparison.** SR@k is success within the first k attempts. Δ reports SR@3 relative to the matched RoboRetry-FAST blind-retry SR@3. RICL is a success-context reference rather than a controlled ablation of failure slots.

Policy	Retry context	Image	Lang.	State	Action	SR@1	SR@2	SR@3	Lift	Δ
Pi0-FAST	Blind retry	–	–	–	–	44.2	46.2	48.8	+4.6	n/a
RoboRetry-FAST	Blind retry	–	–	–	–	40.8	45.8	47.1	+6.3	–
RoboRetry-FAST	Empty failure slot	×	×	×	×	40.8	45.8	48.8	+8.0	+1.7
RoboRetry-FAST	Full VLM	✓	VLM	✓	✓	40.8	47.5	52.5	+11.7	+5.4
RoboRetry-FAST	Full Oracle	✓	Oracle	✓	✓	40.8	48.3	55.4	+14.6	+8.3
RICL	Success demos	success demonstrations				22.9	27.9	31.2	+8.3	n/a

Table 3 treats the failure slot as the intervention target. The rows separate five sources of retry signal: slot presence, non-text failure evidence, corrective-text quality, episode-level text alignment, and the sufficiency of text-only context.

Slot presence itself provides a retry cue.

Blind retry obtains a +4.2 lift, whereas an empty failure slot reaches +10.0. This pattern is consistent with format sensitivity in language-model ICL (Min et al., 2022; Alazraki et al., 2025): the presence of a prior-failure slot can act as a structural retry cue even when all slot contents are zeroed.

Non-text failure-moment evidence alone reaches the same 49.6 SR@3 as the empty-slot row, so image, state, and action evidence by themselves do not explain the largest gains.

Corrective language adds content-dependent refinement.

Generic Text replaces the specific corrective description with the sentence "The previous attempt failed," while retaining the same image, state, and action fields. It reaches 50.0 SR@3, compared with 52.5 for VLM text and 55.4 for human Oracle text. Thus, language quality affects retry performance, with human corrective descriptions achieving the strongest performance.

The Fixed-frame VLM row gives the VLM access to the same human-selected failure moment used by the Oracle annotation. It reaches 51.2 SR@3, close to the standard VLM row and still below the Oracle row, suggesting that frame selection alone does not explain the VLM–Oracle gap. Instead, the content of the corrective description appears to matter: our audit in Appendix E shows that VLM descriptions often describe visible symptoms, whereas human annotations more often specify the task-critical corrective change.

Oracle Text Only reaches 53.3 SR@3, close to the full Oracle slot. This shows that high-quality corrective language carries much of the usable retry signal. At the same time, the full Oracle slot achieves the highest overall SR@3, suggesting that episode-aligned image, state, and action evidence can provide additional grounding, although non-text fields alone do not explain the largest gains.

When we retain the current failure's image, state, and action but replace its description with a same-task mismatched Oracle description, SR@3 reaches 52.5: descriptively below the matched Oracle full slot but above the empty-slot control. This suggests that high-quality corrective language is useful as a class of retry evidence and is most effective when aligned with the current failed episode.

Together, these interventions suggest that the failure slot is not a simple modality ranking. Retry behavior reflects a slot-format cue, task-specific corrective language, and episode alignment. In our current probe, human corrective text carries most of the content-dependent signal, while the full multimodal Oracle slot gives the strongest overall performance and current VLM descriptions do not close the numerical gap to human annotations.

4.3 When is failure-conditioned retry still insufficient?

Takeaways

This failure-conditioned retry is most effective when recovery can be expressed as a local spatial correction. It is less reliable when retry requires semantic re-grounding of the intended target or task progress, or when language must specify fine-grained continuous contact adjustments.

We use per-task outcomes as an operating-envelope diagnostic rather than a failure-type taxonomy. Table 4 groups tasks by the kind of recovery signal that appears necessary for retry.

Table 3: **Failure-slot content interventions.** RoboRetry-FAST content sweep under diagnostic protocols (N=240 per row). SR@k is cumulative success within the first k attempts; lift is SR@3−SR@1. SR@1 is reported per row to expose residual first-attempt drift from restored-scene reruns.

Block	Condition	Img	Failure text	State	Act	SR@1	SR@2	SR@3	Lift
No slot	Blind retry	–	–	–	–	42.9	45.8	47.1	+4.2
Slot structure	Empty slot	×	×	×	×	39.6	46.2	49.6	+10.0
Non-text slot	Image+State+Action Only	✓	×	✓	✓	42.1	47.9	49.6	+7.5
Text quality	Image+State+Action + Generic Text	✓	generic	✓	✓	38.8	44.6	50.0	+11.2
Text quality	Image+State+Action + VLM Text	✓	VLM	✓	✓	41.2	47.5	52.5	+11.2
Text quality	Image+State+Action + Fixed-frame VLM Text	✓	VLM+frame	✓	✓	41.7	46.7	51.2	+9.6
Text quality	Full Oracle	✓	oracle	✓	✓	38.8	48.3	55.4	+16.7
Alignment	Image+State+Action + Mismatched Oracle Text	✓	mismatch	✓	✓	43.8	49.2	52.5	+8.8
Text sufficiency	Oracle Text Only	×	oracle	×	×	42.1	47.1	53.3	+11.2

Table 4: **Operating-envelope diagnostics.** Per-task diagnostics from the same 20-seed aggregate as the main Pi0-FAST retry experiments.

Regime	Tasks	Diagnostic pattern
Local spatial correction	`push_button`, `pick_up_cup`, `open_wine_bottle`	Oracle reaches 68.3 SR@3 on average, +33.3 over SR@1 and +16.7 over VLM.
Semantic re-grounding	`push_buttons`, `open_drawer`	Oracle reaches only 35.0 SR@3, +7.5 over SR@1 and +2.5 over VLM, suggesting difficulty revising the grounded target or task progress.
Contact-precision recovery	`meat_on_grill`, `turn_tap`, `pick_and_lift`	Oracle averages 25.0 SR@3 despite context, indicating remaining brittleness in metric pose, timing, and contact geometry.
Ceiling-controlled tasks	`close_box`, `close_microwave`, `take_lid_off_saucepan`	These tasks start at 83.3 SR@1 and reach 90.0 Oracle SR@3, so small lift mainly reflects limited headroom.

The clearest gains occur when the required recovery can be described as a local spatial adjustment. In `push_button`, `pick_up_cup`, and `open_wine_bottle`, the failed attempt usually preserves the intended object. Their recovery mainly requires shifting the approach direction, grasp pose, or contact location. In this regime, Oracle context improves from 35.0 SR@1 to 68.3 SR@3 on average, 16.7 points above VLM text.

A harder pattern requires semantic re-grounding rather than only local motion correction. In `open_drawer`, the instruction specifies one of several semantically distinct targets: the top, middle, or bottom drawer. If the policy commits to the wrong drawer, retry-time language may be insufficient for the action policy to revise the target it has already grounded, causing the policy to fail repeatedly. In `push_buttons`, the task requires the arm to press two buttons in sequence. However, the policy may press the first button and terminate without re-grounding the need to press the next specified button, even when oracle correction is provided. `meat_on_grill` also contains this kind of failure when the policy binds the instruction to the wrong ingredient, specifically, confusing chicken and steak. These cases suggest that language feedback is not always sufficient to update the policy's latent target or progress estimate, even when the correction is semantically clear to a human.

A second hard regime is contact-precision recovery. Here, language can express the intended direction of change, but remains a coarse interface to a continuous recovery action. Tasks such as `meat_on_grill`, `turn_tap`, and `pick_and_lift` require fine control over contact timing, pose, and trajectory geometry. Free-form text can say that the gripper should move lower, rotate more, or maintain contact, but it does not directly specify the metric residual action needed for reliable recovery.

These limits suggest that the current failure slot is not yet action-native enough for all retry regimes. For semantic re-grounding failures, future contexts may need structured target or progress variables that explicitly indicate which drawer, button, ingredient, or subgoal should be pursued next. For contact-precision failures, future contexts may need denser action-grounded signals, such as 3D keypoints, affordance masks, residual action vectors, contact states, or value-ranked candidate trajectories.

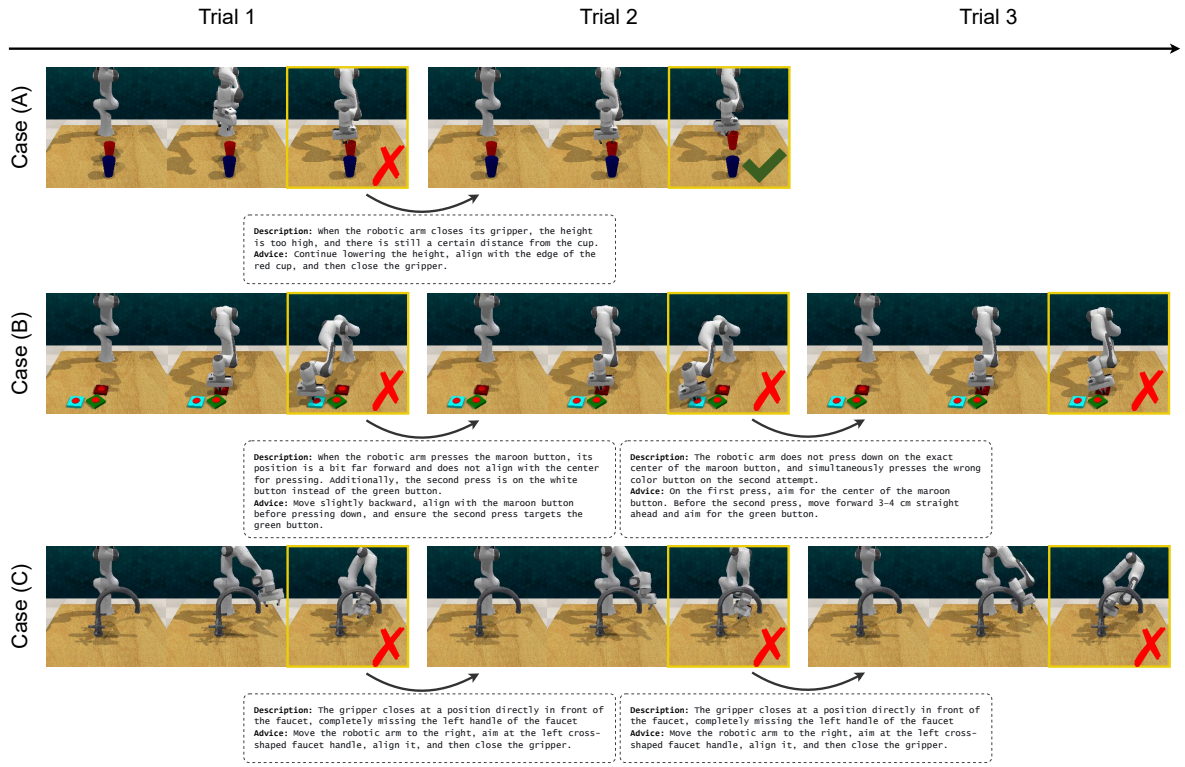

Figure 3: **Operating envelope of failure-conditioned ICL.** Case studies with human Oracle failure descriptions show that failure context can correct coarse spatial errors (Case A), but remains brittle for discrete target/sequence grounding (Case B) and metric/contact precision (Case C).

These limits align with recent VLA evidence that hierarchical robot systems often route complex prompts through simpler intermediate commands for low-level VLA execution, and that task-generalist policies can still grasp visually similar or geometrically biased wrong objects despite language-specified targets (Shi et al., 2025b; Yang et al., 2026).

5 Conclusions

We study prior failures as retry-specific context for VLAs. Under a paired retry protocol, failure-conditioned slots improve cumulative success over blind retry. Our slot-content interventions suggest that the gain is not a simple language-only story: content-free slots provide a retry prior, while human failure annotations add a content-dependent gain. The remaining failures reveal an operating envelope around target identity, sequence-like correction, and precision-sensitive contact, motivating denser context representations for future robot learning.

Limitations

All findings are measured on Pi0-FAST in simulation on tabletop manipulation; whether the retry lift, context-type comparison, and slot-content pattern transfer to other VLA backbones, mobile platforms, bimanual manipulation, or dexterous settings remains open for future work.

A more native context representation (3D visual guidance, a learned latent summarizing the attempted trajectory, or a value function over proposed actions) may provide greater signal for improvement and it is largely unexplored in in-context robot learning at present.

Ethics Statement

This work studies failure-conditioned retry in simulated robotic manipulation. The dataset consists of simulated RLBench trajectories and task-level corrective descriptions. It does not contain images of people, personal information, or annotator identities. Human annotators were informed that their task-level descriptions may be used for research and released in anonymized form with the dataset.

Our experiments are diagnostic and conducted

in simulation for probing the problem. If any further experiments are deployed on real robots, there should be sufficient safety constraints, runtime monitoring, and validation.

AI Assistance Disclosure

We used AI assistants for limited writing support, checklist preparation, and language polishing. All scientific claims, experiments, analyses, citations, and final text were reviewed, verified, and approved by the authors. AI assistants were not used as autonomous authors and did not determine experimental conclusions.

References

Lisa Alazraki, Maximilian Mozes, Jon Ander Campos, Tan Yi-Chern, Marek Rei, and Max Bartolo. 2025. No need for explanations: Llms can implicitly learn from mistakes in-context. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33179–33203.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, and 1 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, and 5 others. 2024. π_0: A vision-language-action flow model for general robot control. *CoRR*, abs/2410.24164.

Hakaze Cho, Mariko Kato, Yoshihiro Sakai, and Naoya Inoue. 2024. Revisiting in-context learning inference circuit in large language models. *arXiv preprint arXiv:2410.04468*.

Joy Crosbie and Ekaterina Shutova. 2025. Induction heads as an essential mechanism for pattern matching in in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5034–5096.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. 2025. Racer: Rich language-guided failure recovery policies for imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15657–15664. IEEE.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. 2024. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*.

Clemence Grislain, Hamed Rahimi, Olivier Sigaud, and Mohamed Chetouani. 2025. I-failsense: Towards general robotic failure detection with vision-language models. *arXiv preprint arXiv:2509.16072*.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. 2025. Safe: Multitask failure detection for vision-language-action models. *arXiv preprint arXiv:2506.09937*.

Yayun He, Zuheng Kang, Botao Zhao, Zhouyin Wu, Junqing Peng, and Jianzong Wang. 2026. Confusion-aware in-context-learning for vision-language models in robotic manipulation. *arXiv preprint arXiv:2603.15134*.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2023. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, and 69 others. 2026. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *CoRR*, abs/2604.15483.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, and 37 others. 2025. $\pi^*_{0.6}$: a VLA that learns from experience. *CoRR*, abs/2511.14759.

Vidhi Jain, Maria Attarian, Nikhil J. Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R. Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, Igor Gilitschenski, Yonatan Bisk, and Debidatta Dwibedi. 2024. Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers. In *Robotics: Science and Systems XX, Delft, The Netherlands, July 15-19, 2024*.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. 2020. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, Proceedings of Machine Learning Research, pages 2679–2713. PMLR.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. 2025. Failsafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*.

Zeyi Liu, Arpit Bahety, and Shuran Song. 2023. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. 2025a. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*.

Weifeng Lu, Minghao Ye, Zewei Ye, Ruihan Tao, Shuo Yang, and Bo Zhao. 2025b. Robofac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 11048–11064.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *CoRR*, abs/2209.11895.

Paul Pacaud, Ricardo Garcia, Shizhe Chen, and Cordelia Schmid. 2025. Guardian: Detecting robotic planning and execution errors with vision-language models. *arXiv preprint arXiv:2512.01946*.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. 2025. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*.

Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. 2022. A generalist agent. *Trans. Mach. Learn. Res.*, 2022.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. 2025a. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. 2025b. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. 2025a. Memer: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. 2025b. Ricl: Adding in-context adaptability to pre-trained vision-language-action models. *arXiv preprint arXiv:2508.02062*.

Shahram Najam Syed, Yatharth Ahuja, Arthur Jakobsson, and Jeff Ichnowski. 2025. Expres-vla: Specializing vision-language-action models through experience replay and retrieval. *arXiv preprint arXiv:2511.06202*.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080.

Michael Tschannen, Alexey A. Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier J. Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. 2025. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *CoRR*, abs/2502.14786.

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *arXiv preprint arXiv:2402.11651*.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, Zhen Zhao, Guangyu Li, Zhao Jin, Lecheng Wang, Jilei Mao, Xinhua Wang, Shichao Fan, Ning Liu, Pei Ren, and 17 others. 2024. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *CoRR*, abs/2412.13877.

Shaokai Wu, Yanbiao Ji, Qiuchang Li, Zhiyi Zhang, Qichen He, Wenyuan Xie, Guodong Zhang, Bayram Bayramli, Yue Ding, and Hongtao Lu. 2025. Dejavu: Post-deployment learning for embodied agents via experience feedback. *arXiv preprint arXiv:2510.10181*.

Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. 2026. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies. *arXiv preprint arXiv:2604.09860*.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. 2025. Diagnose, correct, and learn from manipulation failures via visual symbols. *arXiv preprint arXiv:2512.02787*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986.

A Evaluation and Training Details

Multi-condition same-process evaluation. Multi-condition evaluations run all reported conditions within a single CoppeliaSim process per episode: each (task, episode) `reset()`s once, then tests every condition through scene-state restoration; policy RNG is reset before each condition so first-attempt outputs match across conditions up to residual ODE drift; condition order is shuffled per episode.

Software and implementation. Experiments use RLBench with CoppeliaSim for simulated manipulation. Pi0-FAST experiments are implemented in the OpenPI JAX/Flax codebase. Raw observations use the front and wrist cameras at 256×256 resolution and are resized or processor-normalized to 224×224 before model input. Behavioral t-SNE visualizations use end-effector state concatenated with the forward action chunk as input features. VLM-generated descriptions use the model and prompting setup described in Section 3.

Compute. Main training runs use $8\times$A100 80GB GPUs. Inference is performed on RTX 4090/5090 GPUs. Table 5 reports training steps, batch size, context length, image resolution, and optimizer settings.

Hyperparameters. Trained Pi0-FAST variants share the same optimizer and image-encoder configuration for fair comparison (Table 5).

Table 5: **Training hyperparameters.** Values are shared across Pi0-FAST variants except where noted. The image encoder is frozen.

Parameter	Pi0-FAST variants
backbone	Pi0-FAST
training framework	JAX/Flax fine-tuning
optimizer	AdamW
learning rate	2.5×10^{-5}
LR schedule	linear warmup, cosine decay
warmup steps	1,000
batch size	16 global
training steps	10K for baseline, RICL, and RoboRetry-FAST
`max_token_len`	512 (RoboRetry-FAST); 250 (baseline / RICL)
context observations	2 success-KNN (RICL); max 2 failure slots (RoboRetry-FAST)
cameras	front + wrist; raw 256×256, model input 224×224
state input	proprioceptive state included
image encoder	SigLIP, frozen
hardware	$8\times$A100 80GB

B Failure Dataset and Taxonomy

FAILURESLOT record format. FAILURESLOT is released in the same unit used by the retry probe: an in-context failure slot rather than only a video-level label. Each record is indexed by task, variation, source episode, and a human-verified failure keyframe. The released fields include front and wrist RGB images at the failure keyframe, the initial front image used for retrieval controls, a proprioceptive state chunk, a forward action chunk, and language annotations. The state chunk has shape $H\times 8$ and stores end-effector position, Euler orientation, gripper-open state, and one padding channel. The action chunk has shape $H\times 7$ and stores delta position, delta orientation, and gripper command from the failure keyframe forward. In our released bank, $H=10$.

Taxonomy induction. The 8-category taxonomy used in §3.2 was derived by LLM-assisted thematic coding over the human oracle descriptions, then rule-mapped onto the AHA generator's built-in perturbation labels so both datasets are re-labelled under one scheme. An initial *insufficient force* category proposed by annotators was merged into *contact instability* after inspection: RLBench's gripper is binary (no continuous force control), and the described mechanism is in every case a contact-stability failure misread through an anthropomorphic vocabulary, a reminder that corrective descriptions can carry modality-mismatched language.

Annotation-label convention. On `push_buttons`, our annotators code a wrong-color press as *incorrect target selection* while AHA labels it `wrong_sequence` by generator construction. This annotation-perspective gap leaves the three primary distribution gaps of §3.2 intact.

Behavioral t-SNE. We embed each failure episode into a 150-dimensional behavioral feature vector (end-effector state at the failure keyframe concatenated with the forward action chunk) and project per-task using t-SNE. Figure 4 shows four representative tasks with density contours and mode-collapse hulls marked.

C Task Suite

Table 7 summarizes the 12 RLBench tasks used in our experiments. The tasks cover button pressing, articulated-object interaction, grasping, placement, and contact-sensitive manipulation. We use the same task suite for training, retry evaluation, and failure-bank collection.

Annotation instructions. Annotators were shown the failed rollout, including the failure keyframe and task instruction, and were asked to provide two fields: (i) a concise description of what went wrong, and (ii) corrective advice that could help the policy retry the task. Annotators were instructed to use task-level language, avoid personal information, and avoid offensive content. The annotations describe robot behavior and object interactions only; they do not contain information about the annotators.

Annotator recruitment. Annotations were provided by co-authors and lab members as part of the research project. No external crowdworkers were recruited. Annotators participated voluntarily in their research capacity, and no separate crowdwork-style payment was provided.

D Prompts for Evaluation.

For VLM-generated failure descriptions, the context server sends the failed rollout video and the following prompt template to Qwen3-VL. FAILURESLOT includes annotations from both Qwen3-VL-8B-Instruct and Qwen3-VL-30B-A3B-Instruct using this format. The task description is instantiated from task and variation metadata. In fixed-frame controls, the server additionally inserts the optional human-selected failure-moment sentence shown below. The task instruction is kept separate from the failure-specific text interventions in Table 3.

E VLM–Oracle Description Audit

We audit the evaluation-time annotations to understand why VLM-generated failure text trails human Oracle text in Table 3. Among 276 failure JSON records produced during the Pi0-FAST Oracle evaluation, 275 contain human annotations and 264 contain both a VLM-predicted failure frame and a human-adjusted failure frame. The VLM outputs are often useful at the level of coarse spatial diagnosis, but they tend to select an earlier visible symptom rather than the task-critical failure moment used by the human annotator. At 10 FPS, the median absolute gap between the two selected frames is 22 frames, or about 2.2 seconds; in 235/264 paired records, the VLM frame is more than 10 frames earlier than the human-adjusted frame.

Table 6: **Examples of human Oracle failure annotations.** Text is shortened for space while preserving the correction target and retry cue.

Task	Instruction	Failure description	Recovery advice
`push_buttons`	press maroon, then green	After pressing maroon, the arm reaches the edge of the green button and does not press it.	Aim at the center of the green button and shift slightly backward before pressing.
`pick_and_lift`	pick up the red block	The gripper closes on a pink block instead of the red block.	Move left-back toward the red cube, grasp it, then lift toward the target.
`turn_tap`	turn left tap	The gripper closes while still offset from the left faucet handle.	Move closer in the handle direction, align the gripper, and then grasp firmly before turning.
`meat_on_grill`	put chicken on grill	The arm closes at an incorrect height without approaching the chicken.	Shift toward the chicken, lower the gripper until contact, then grasp before placing it on the grill.

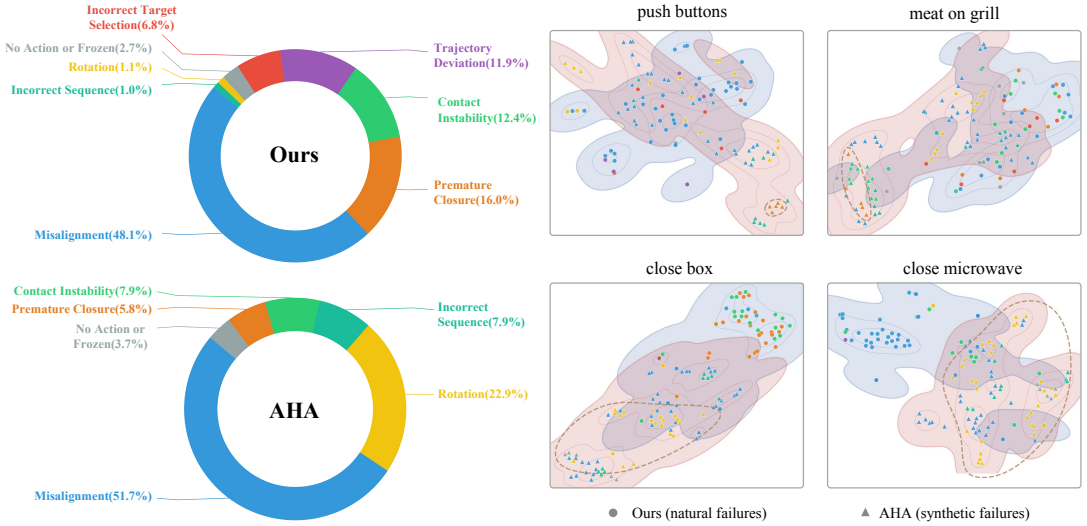

Figure 4: **Failure-distribution and behavior-space comparison.** Left: 8-category distributions for our naturally occurring failures and AHA's synthetic perturbations. Our failures are dominated by misalignment with a long tail of contact-stage modes, while AHA allocates more than a quarter of its mass to rotation errors. Right: behavioral t-SNE on four representative tasks. Ours (circles, blue density) and AHA (triangles, red density) occupy different regions, with divergence concentrated on contact-stage and multi-mechanism tasks.

The same audit also suggests that the gap is not simply a lack of spatial words: the combined VLM description and advice contains coarse spatial terms in 84.7% of annotated records, while the corresponding Oracle text contains them in 68.7%. The difference is instead in temporal and corrective specificity. For example, in a `push_buttons` episode whose instruction is to press the maroon button and then the green button, the VLM selects frame 6 and describes a generic failed green-button push, while the human selects frame 43 after the first subgoal and notes that the arm reaches the edge of the green button rather than its center. This kind of annotation provides a more episode-aligned retry cue.

We further isolate frame selection with a fixed-frame VLM control: before generating the failure description, the VLM is given the human-selected failure moment from the Oracle annotation. This control reaches 51.2 SR@3 with a +9.6 retry lift, in the same range as the full VLM row in Table 3 and below the human Oracle row. Thus, human frame selection creates a sharper diagnostic setting, while the remaining Oracle advantage is associated with more specific and actionable corrective language.

F Bootstrap Confidence Intervals

Table 9 reports 95% confidence intervals for the predeclared paired contrasts used to interpret the main Pi0-FAST results. The bootstrap keeps the task suite fixed, resamples seed/episode tuples with replacement within each task, and macro-averages

VLM failure-description prompt

```
Watch this <duration>s video (recorded at 10 FPS) of a robot arm attempting to: <task and variation description>

The robot FAILED at this task. Analyze the failure and respond in EXACTLY this format:

FAILURE_TIME: <time in seconds, e.g. 8.3>
FAILURE_DESCRIPTION: <1-2 sentences describing the specific error, max 50 words>
RECOVERY_ADVICE: <1-2 sentences, include action type (grasp/push/rotate/align)
and spatial direction (left/right/up/down/closer/further), no exact numbers, max 50 words>

Rules:
<optional fixed-frame instruction>
- FAILURE_TIME: The moment the critical error occurs (e.g. gripper closes on empty air, object slips, arm collides). This
      is usually in the middle or latter part of the video, NOT at the very beginning when the arm is still approaching.
Give one decimal place.
- FAILURE_DESCRIPTION: Be specific - say "gripper closed above the cup rim without making contact" not "failed to grasp".
      Only describe what you see, do not invent objects or actions.
- RECOVERY_ADVICE: Focus on action type (grasp, push, rotate, align) and spatial direction (left, right, higher, lower,
      closer). Do NOT give exact measurements like "move 3cm". Example: "lower the gripper before closing" or "approach from
      the left side".
- Only reference objects actually visible in the video.

Optional fixed-frame instruction:
A human oracle has already selected the failure moment at <frame/time>. Do not search for a different failure time; focus
      your description and recovery advice on this selected moment.
```

Figure 5: **Prompt used for VLM-generated failure descriptions.** The optional fixed-frame instruction is inserted only when a human-selected failure moment is provided.

Table 7: **RLBench task suite.** The same 12 tasks are used for training, retry evaluation, and failure-bank collection.

Task name	Task description
`push_button`	Press the [maroon, green, blue] button.
`push_buttons`	Press [the maroon, the green, the maroon then the green] button.
`meat_on_grill`	Put the [chicken, steak] on the grill.
`pick_up_cup`	Pick up the [red, maroon, lime] cup.
`open_wine_bottle`	Open the wine bottle by removing the bottle cap.
`turn_tap`	Turn on the tap using the left handle
`close_box`	Close the hinged lid of the box
`close_microwave`	Close the microwave door.
`take_lid_off_saucepan`	Grasp and remove a saucepan lid.
`open_drawer`	Grasp and pull the handle of the [bottom, middle, top] drawer.
`lamp_on`	Press the button to turn on the lamp
`pick_and_lift`	Pick up the red block and lift it up to the target.

Table 8: **VLM–Oracle annotation audit.**

Audit quantity	Value
Failure records	276
Human-annotated records	275
Records with paired VLM/Oracle frame	264
Median absolute frame gap	22 frames
Within 10 frames	11.0%
VLM earlier by >10 frames	89.0%
Oracle advice with metric magnitude	30.2%
VLM advice with metric magnitude	0.0%

over tasks, matching the aggregation used in the paper tables. The intervals support the main retry claim: both VLM and Oracle failure contexts improve SR@3 over matched blind retry.

G Per-Task Retry Results

Per task retry results are in Table 10 and Table 11.

Table 9: **Task-stratified paired bootstrap contrasts.** Deltas are SR@3 percentage-point differences. Confidence intervals use 5,000 bootstrap resamples over seed/episode tuples within each task, followed by macro-averaging over the 12-task suite.

Table	Paired contrast	Δ SR@3	95% CI
Table 2	RoboRetry-FAST Full VLM − RoboRetry-FAST Blind	$+5.4$	$[+1.2, +9.6]$
Table 2	RoboRetry-FAST Full Oracle − RoboRetry-FAST Blind	$+8.3$	$[+3.8, +12.9]$
Table 2	RoboRetry-FAST Full Oracle − RoboRetry-FAST Full VLM	$+2.9$	$[-2.1, +7.5]$
Table 2	RoboRetry-FAST Empty failure slot − RoboRetry-FAST Blind	$+1.7$	$[-2.9, +6.7]$
Table 3	Empty slot − Blind retry	$+2.5$	$[-2.1, +7.1]$
Table 3	Image+State+Action Only − Empty slot	$+0.0$	$[-5.0, +4.6]$
Table 3	Oracle full slot − VLM full slot	$+2.9$	$[-2.1, +7.9]$
Table 3	Fixed-frame VLM − VLM full slot	-1.2	$[-5.8, +3.8]$
Table 3	Oracle full slot − Fixed-frame VLM	$+4.2$	$[-1.7, +10.0]$
Table 3	Oracle full slot − Mismatched Oracle	$+2.9$	$[-3.3, +9.2]$
Table 3	Oracle full slot − Oracle Text Only	$+2.1$	$[-2.1, +6.7]$

Table 10: **Per-task SR@3 for Table 2.** Entries are cumulative success within three attempts on each task. The average column matches the SR@3 values in Table 2.

Condition	push_btn	push_btns	meat	pick_cup	wine	turn_tap	close_box	close_mw	lid_off	drawer	lamp	pick_lift	Avg
Pi0-FAST blind retry	60.0	45.0	30.0	55.0	45.0	20.0	65.0	90.0	85.0	25.0	60.0	5.0	48.8
RoboRetry-FAST Blind retry	50.0	40.0	15.0	60.0	25.0	35.0	95.0	95.0	75.0	20.0	45.0	10.0	47.1
RoboRetry-FAST Empty slot	50.0	45.0	20.0	70.0	55.0	20.0	100.0	85.0	80.0	15.0	40.0	5.0	48.8
RoboRetry-FAST Full VLM	50.0	45.0	20.0	65.0	40.0	50.0	90.0	100.0	80.0	20.0	60.0	10.0	52.5
RoboRetry-FAST Full Oracle	65.0	50.0	20.0	80.0	60.0	40.0	95.0	95.0	80.0	20.0	45.0	15.0	55.4
RICL success demos	30.0	25.0	5.0	55.0	50.0	25.0	60.0	75.0	25.0	15.0	10.0	0.0	31.2

Table 11: **Per-task SR@3 for Table 3.** Entries are cumulative success within three attempts for the failure-slot content interventions. The average column matches the SR@3 values in Table 3.

Condition	push_btn	push_btns	meat	pick_cup	wine	turn_tap	close_box	close_mw	lid_off	drawer	lamp	pick_lift	Avg
Blind retry	50.0	40.0	15.0	60.0	25.0	35.0	95.0	95.0	75.0	20.0	45.0	10.0	47.1
Empty slot	45.0	45.0	10.0	65.0	55.0	20.0	95.0	95.0	80.0	10.0	60.0	15.0	49.6
Image+State+Action Only	50.0	40.0	25.0	70.0	20.0	40.0	95.0	95.0	90.0	10.0	55.0	5.0	49.6
Image+State+Action + Generic Text	55.0	45.0	20.0	70.0	25.0	40.0	85.0	95.0	80.0	20.0	60.0	5.0	50.0
Image+State+Action + VLM Text	50.0	45.0	20.0	65.0	40.0	50.0	90.0	100.0	80.0	20.0	60.0	10.0	52.5
Image+State+Action + Fixed-frame VLM Text	50.0	40.0	25.0	65.0	60.0	40.0	90.0	95.0	80.0	10.0	50.0	10.0	51.2
Image+State+Action + Oracle Text	65.0	50.0	20.0	80.0	60.0	40.0	95.0	95.0	80.0	20.0	45.0	15.0	55.4
Image+State+Action + Mismatched Oracle Text	60.0	40.0	30.0	65.0	50.0	40.0	85.0	95.0	85.0	10.0	60.0	10.0	52.5
Oracle Text Only	65.0	40.0	15.0	70.0	50.0	45.0	95.0	90.0	85.0	15.0	55.0	15.0	53.3
Oracle Text+State Only	55.0	45.0	15.0	60.0	45.0	45.0	75.0	90.0	85.0	10.0	55.0	5.0	48.8