2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

CorrectManip: A Data-Driven Closed-Loop Framework for Autonomous Skill Learning with Failure Recovery

Shiwen Li[1,3*], **Zhen Yang**[1,4*†], **Zuofu Wang**[1†], **Bin Ma**[1†],
Tengju Ye[2], **Shiyu Zhao**[1], **Junbo Chen**[2‡], **Kaicheng Yu**[1,3‡]
[1]Autolab, Westlake University, [2]Udeer.ai, [3]Auwomo.ai, [4]The University of Hong Kong
yang.zhen@connect.hku.hk, junbo@udeer.ai
{lishiwen, kyu}@westlake.edu.cn

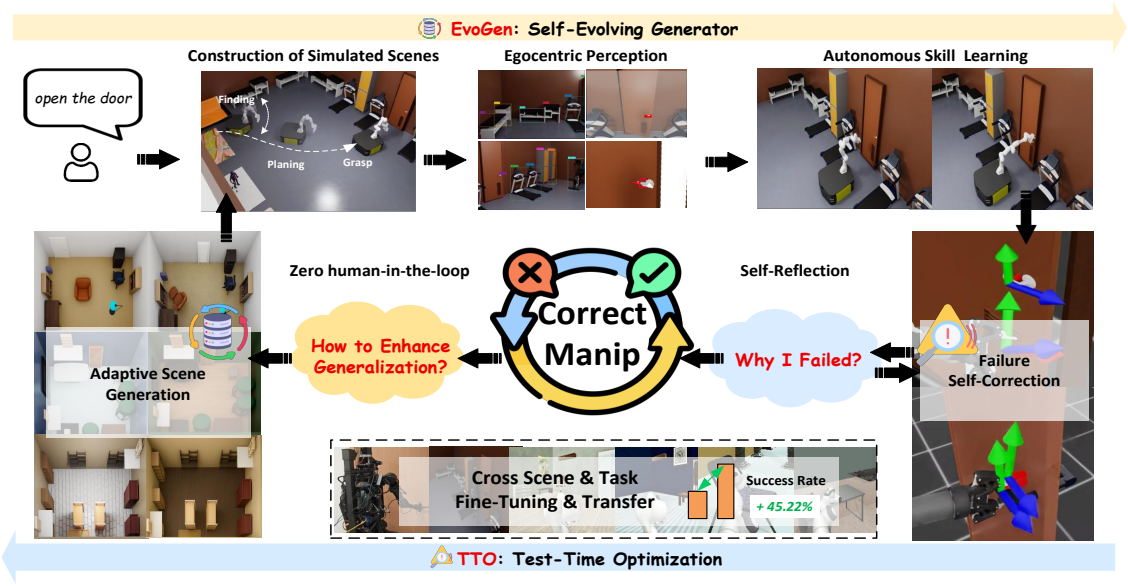

Fig. 1. **Overview of CorrectManip framework.** A data-driven closed-loop framework for autonomous robotic skill learning. Starting from a task goal (*e.g.*, open the door), CorrectManip integrates 1) **EvoGen**, a self-evolving generator that constructs diverse simulation scenes for egocentric perception and automated skill learning, and 2) **TTO**, a test-time optimization module that performs failure analysis, self-correction, and online adaptation. By iteratively reflecting on failures and adapting training conditions without human supervision, CorrectManip systematically enhances policy generalization, achieving an average improvement of 45.22% over baseline methods.

***Abstract*—Simulation-based training offers an efficient paradigm for robotic skill learning, providing scalable data generation while reducing reliance on costly hardware trials and manual data collection. However, existing methods that rely on handcrafted scenarios fail to fully cover the complexity of open-world variations and neglect the critical insights offered by inevitable failures in unseen environments. As a result, current policies struggle to achieve robust generalization, hindering deployment in open-world settings. This highlights the need for a continuous learning framework that enables robots to reflect on failures and iteratively refine policies in a targeted way. In this paper, we propose CorrectManip, a novel data-driven closed-loop framework that enables the policy to continuously improve performance in unseen environments by learning from failures. Existing methods remain confined to single-loop adaptation, addressing policy errors in static environments or indiscriminately scaling data without targeting failure modes, CorrectManip closes the loop both at the policy recovery and environment generation: EvoGen, a self-evolving generator, and TTO, a test-time optimization module. EvoGen adaptively generates training data to strengthen policy performance, while TTO analyzes execution failures to provide fine-grained optimization signals. Together, TTO exposes policy weakness and EvoGen converts them into task-relevant training data, forming a closed feedback loop that drives continual policy improvement and stronger generalization. Extensive experiments across diverse tasks demonstrate that CorrectManip improves the average success rate in unseen environments by 45.22% over baseline methods. These results validate the complementary roles of TTO and EvoGen in enhancing generalization. Furthermore, we showcase sim-to-real transfer ability on Unitree H1 and Unitree G1. Demos are available here.**

I. Introduction

Robotic skill learning has witnessed remarkable progress in recent years, driven by advances in Large Language Models (LLMs) [1], [2] and large-scale simulation environ-

*Equal contribution.
†Work done during their visiting at Autolab, Westlake University.
‡Co-corresponding authors.

ments [3]–[5]. These developments have allowed robots to acquire complicated skills from large amounts of simulated data [6]–[8]. However, as robots transition from controlled laboratories to open-world environments, the demand for robust generalization across unseen environments, object configurations, and task variations becomes critical [9], [10].

Current data generation approaches for policy learning fall into three categories: 1) curated task-specific environments built by experts [11], [12], 2) scaling human demonstration datasets via augmentation or generative expansion [13], [14], and 3) generative simulation methods that synthesize large amounts of data without human demonstrations [15], [16]. Despite progress, these methods struggle to generalize beyond their training distribution, leading to performance degradation in unexpected environments. This generalization gap remains a key barrier to reliable open-world deployment.

To mitigate this gap, existing methods have primarily focused on scaling up data diversity to broaden training coverage [5], [17], enhancing adaptation through domain randomization [18], [19], and policy adaptation via fine-tuning [20], [21]. Although these approaches have advanced the state-of-the-art, they remain fundamentally limited. In particular, 1) they indiscriminately scale data diversity through offline learning, open-loop generation, which merely expands the amount of data without effectively improving its quality, 2) they lack the ability to autonomously analyze failure causes and fine-tune the policy by optimized training configurations. Crucially, these methods lack mechanisms to iteratively refine data distributions, which constrains sustained generalization improvements in unseen environments. Our core insight is that robust generalization requires a closed-loop process that iteratively reflects failure during policy execution and effectively improves the targeted generalization on identified weakness.

To this end, we introduce **CorrectManip**, a closed-loop framework for skill learning that autonomously generates diverse scenes, identifies policy failure modes, and leverages self-reflective refinement to continually improve generalization. These capabilities are realized through two key components: a self-evolving scene generator (EvoGen) and a test-time optimization module (TTO), as illustrated in Fig. 1. Unlike previous works [15], [16], which passively rely on offline diversity without dynamic feedback, or lack mechanisms to leverage experience from failures [22], [23]. CorrectManip enables adaptive scene generation, fine-grained failure analysis, and online policy improvement within a unified framework. We conduct extensive experiments and preliminary real-world validation, demonstrating its sim-to-real transfer potential. Ablation studies further highlight the critical roles of EvoGen and TTO, confirming their effectiveness in enabling closed-loop optimization and continual improvement of policy generalization.

Our contributions are summarized as follows:

1) We introduce CorrectManip, a closed-loop framework that unifies adaptive scene generation and test-time optimization to continuously improve policy generalization.
2) We propose EvoGen, a self-evolving data generator that generates data conditioned on failures discovered during policy evaluation.
3) We propose TTO, a test-time optimization module that identifies failure modes and provides fine-grained correction to inform policy fine-tuning.
4) Extensive experiments show that CorrectManip achieves an average success rate of 85.98% across diverse tasks, significantly outperforming the baselines by 45.22%. We further demonstrate sim-to-real performance on real-world platforms, including Unitree H1 and Unitree G1.

II. Related Work

Embodied AI benchmarks. Various frameworks and benchmarks have been developed for robotic manipulation [6]–[8]. While existing frameworks provide structured environments, they often rely on manually designed scenarios and are limited in scene diversity, restricting their scalability in real-world applications. Some frameworks like RoboGen [15] and RoboCasa [5] enable automated scene generation; however, they typically assume ideal perception from simulators and neglect realistic perception errors. Recent advances in Vision-Language-Action (VLA) models bridge semantic reasoning and control via large-scale pretraining [24], [25]. While these end-to-end methods show promise, they face challenges that include dependence on extensive demonstrations and slow inference speeds that hinder real-time deployment [26]–[28]. In contrast, our CorrectManip framework integrates automated scene generation with failure recovery in a closed-loop process, leveraging visual feedback for policy correction, and systematically reducing reliance on human demonstrations to achieve robust generalization.

Closed-loop frameworks. Closed-loop frameworks improve policy robustness by continuously monitoring and correcting robot actions during execution [29]–[33]. CLOVER introduces a text-conditioned video diffusion model combined with feedback-driven control, mitigating error accumulation through real-time adjustments [33]. [30] demonstrates how feedback mechanisms, such as image registration, allowed robots to maintain task objectives and retry tasks when necessary. The Inner Monologue framework leverages LLMs to optimize decision-making processes, using environmental feedback to improve reasoning capabilities [31]. However, previous works remain constrained to the current environment and focus only on reactive correction. In contrast, our CorrectManip not only applies closed-loop optimization in the current environment but also actively generates failure-targeted new scenarios based on task feedback, closing the loop across both policy and environment dimensions.

III. Method

A. The Overview of CorrectManip Framework

CorrectManip is a closed-loop skill learning framework that automatically enhances policy generalization across diverse configurations by learning from interaction failures

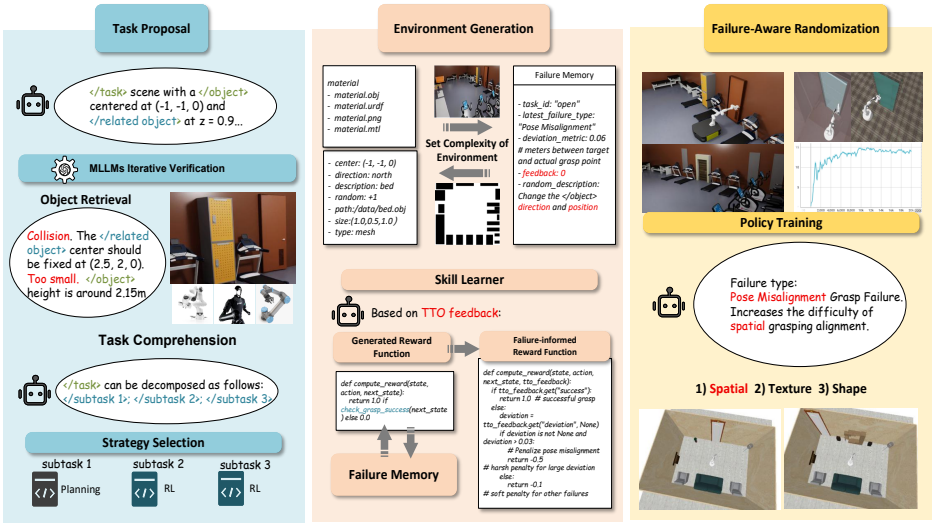

Fig. 2. **Illustration of EvoGen.** The self-evolving generator in CorrectManip. 1) **Task Proposal**: an LLM-driven process verifies feasibility through iterative object retrieval and task decomposition. 2) **Environment Generation**: constructs environments with configurable complexity, informed by a growing failure memory. 3) **Failure-Aware Randomization**: leverages TTO feedback to identify failure types and update reward functions to guide policy learning.

(Fig. 2). Unlike traditional data augmentation strategies that indiscriminately increase data diversity, CorrectManip explicitly captures the underlying failure modes of the current policy and actively generates targeted training configurations to address these specific shortcomings. Formally, given a user-defined task $\mathcal{T}$, CorrectManip operates in iterative cycles. We first decompose $\mathcal{T}$ into a set of subtasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ to localize evaluation and failure discovery. CorrectManip then proceeds with two synergistic modules:

1) **EvoGen** automates task proposal, environment generation, and policy learning in a self-evolving way. For $\mathcal{T}$, EvoGen proposes a set of training environments $\{E_1, E_2, \ldots, E_m\}$ and either initializes the policy set $\{\pi_0, \pi_1, \ldots, \pi_n\}$ or updates $\{\pi_i^{(k)}\}_{i=1}^n$ at iteration k, based on failure memory $\mathcal{M}^{(k-1)}$ accumulated from the previous iteration. The adaptive generation process is formally expressed as

$$\{\mathcal{T}_i, \{E_j\}_{j=1}^m, \pi_i^{(k)}\} \leftarrow \text{EvoGen}(\mathcal{T}, \mathcal{M}^{(k-1)}). \quad (1)$$

By leveraging failure feedback from TTO, EvoGen refines the environment complexity and training configurations, allowing the policy to more effectively target identified failure modes.

2) **TTO** focuses on action correction and failure discovery during test time. It incorporates an action supervisor that provides action-level correction a_t^v to the initial policy action a_t. Failure discovery detects and categorizes a failure case f during execution, analyzes discrepancies between predicted and observed outcomes. Detected failures are aggregated into failure memory:

$$\mathcal{M}^k \leftarrow \mathcal{M}^{k-1} \cup \{f\}. \quad (2)$$

Using insights derived from failure memory, the initial reward function R_{init} is refined to a failure-informed reward function R_{opt}, which guides adaptive generation and training reconfiguration in the next iteration.

These two modules operate in an iterative loop: EvoGen generates adaptive environments based on failure modes, while TTO discovers failures and refines policy at test time.

B. EvoGen: Self-evolving Generator

Task proposal. Given the task description $\mathcal{T}$, LLMs and VLMs automatically execute the following process: $\mathcal{T}$ is decomposed into subtasks; for each subtask, select appropriate strategy, generate initial reward functions and the corresponding training scripts. We query GPT-4o [1] to generate diverse scene configurations, ranging from large exhibition halls to narrow corridors. We incorporate elements such as wall decorations and pillars, and introduce varied object placements, including floor-placed items, ceiling-suspended elements and wall-mounted furniture. For object asset retrieval, we employ CLIP [34] and SBERT [35] to encode textual and visual features, and use GPT-4o to generate unique identifiers (UIDs) for each object. These UIDs are then used to index and retrieve objects from a large-scale 3D asset dataset. The semantic similarity between the retrieved annotations and BLIP-2 image features is computed to select the most suitable objects. To ensure physical plausibility, GPT-4o and Gemini-Pro [2] are used to iteratively verify the layout and size of objects. The system dynamically corrects unreasonable arrangements to maintain realism and task feasibility.

Environment Generation. We categorize scenes into three levels of complexity based on the spatial occupancy rate: *spacious*, characterized by minimal interactive interference; *moderate*, with a balanced distribution of objects; and *compact*, where objects are densely arranged, making navigation and manipulation more challenging. We reserve 30% free

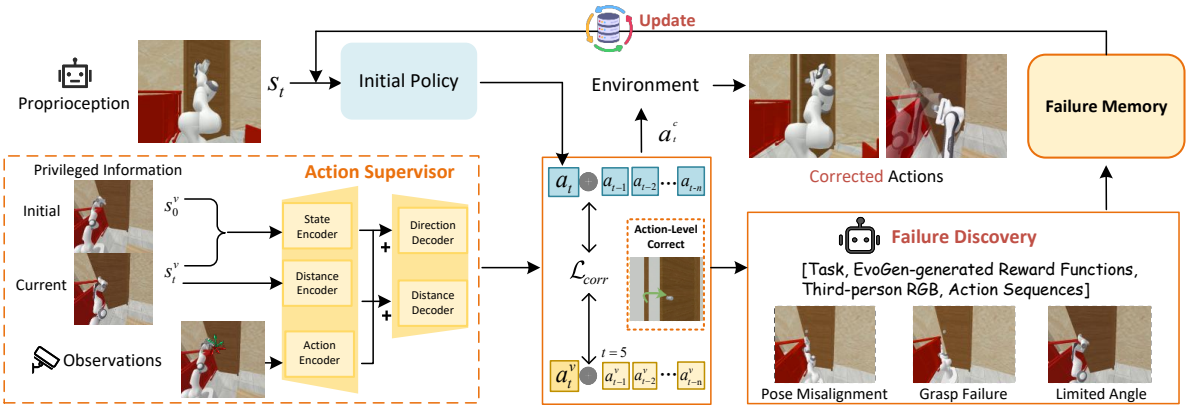

Fig. 3. **Details of TTO Module**. A vision-based policy serves as a pretrained supervisor, which takes RGB-D as input and generates an end-effector action to refine the action of a control-based policy.

area for the robot to ensure feasible execution. The spatial occupancy rate is computed as

$$D_{occ} = \frac{\sum_{i=1}^{N} \text{Vol}(Object_i)}{\text{Vol}(E)}, \tag{3}$$

where $Object_i$ denotes the i_{th} object and $\text{Vol}(E)$ represents the volume of the current environment.

After each evaluation round, EvoGen receives a feedback signal from TTO, denoted as $feedback \in \{-1, 0, +1\}$. This signal enables closed-loop refinement by guiding adjustments in scene complexity. -1 indicates reducing complexity by removing obstacles or decreasing the number of interactive objects. 0 indicates that the current complexity is maintained to reinforce robustness. $+1$ encourages introducing new obstacles to increase the occupancy rate.

Failure-Aware Randomization. We leverage the structured failure descriptions stored in the failure memory $\mathcal{M}$ from TTO, and prompt an LLM to automatically determine how to randomize scenes based on the observed failure types. This guides targeted perturbations to existing objects in the scene, covering texture, shape and spatial layout. For these perturbations, we design an adaptive selection mechanism based on a curated library of textures and shapes, which enables the policy to learn spatial layouts, geometric relationships, and diverse visual appearances. For spatial layout, we apply random offsets to the original position $\mathbf{p}_i + \Delta\mathbf{p}$, and angular noise to the original orientation $\theta_i + \Delta\theta$. The perturbation magnitudes are sampled from Gaussian distributions: $\Delta\mathbf{p} \sim \mathcal{N}(0, \sigma_p^2 I), \quad \Delta\theta \sim \mathcal{N}(0, \sigma_\theta^2)$. This mechanism constructs failure-adaptive scenes and modulates environmental difficulty according to policy capabilities. σ_p and σ_θ are adaptively tuned, allowing the system to dynamically adjust perturbation levels. This ensures that each iteration systematically challenges the policy on its previously identified weaknesses.

C. TTO: Test-Time Optimization

Action Supervisor. To support test-time correction, we leverage an action supervisor that provides privileged visual supervision unavailable to the deployed policy (see Fig.3). Specifically, we introduce a vision-based policy with privileged information to generate candidate action sequences for manipulating general articulated objects. We integrate DINO [36] to perform object detection, target workspace localization, and grasp point refinement. Additionally, GraspNet-1B [37] predicts feasible approach vectors, distances, in-plane rotations along the approach axis, and gripper width. The action supervisor π^{v} processes the third-person RGB-D observations from both the initial state s_0^v and the current state s_t^v, to generate an action suggestion $a_t^v = \pi^{\text{v}}(s_0^v, s_t^v)$ for the end-effector. In parallel, the initial policy π trained via PPO [38], operates on proprioception state S_t to predict action a_t. This module enhances failure discovery by providing additional context on action-level mismatches.

Failure Discovery. We introduce failure discovery module to analyze discrepancies between the goal and the observed execution outcomes. It interprets failure types in a human-like manner, relying solely on action-level visual supervision. After execution, it collects the end-effector trajectories $\tau = a_0, a_1, \ldots, a_t$, the supervisor trajectory τ^v, the final state image, and reward signals R_t, and then employs GPT-4o to summarize and classify the failures. All failures are classified into predefined failure types: *Pose Misalignment*: significant deviation between the predicted and actual end-effector pose at the target keyframe; *Reach Failure*: the robot fails to establish physical contact with the target object; *Limited Angle*: task failure due to the end-effector approaching the target with an improper orientation.

Finetune via Test-Time Optimization. TTO performs online reward adjustment and policy fine-tuning, guided by failure discovery. When a failure case $f \in \mathcal{F}$ is detected, the failure memory module stores its type, a descriptive summary, and suggestions for environment randomization and reward shaping, as follows:

$$\mathcal{M}_i = [\text{Type}, \text{Description}, \text{Suggestion}]\,. \tag{4}$$

These records are used to prompt the LLM to generate a failure-aware reward function $\mathcal{R}_{\text{opt}}$, which integrates semantic

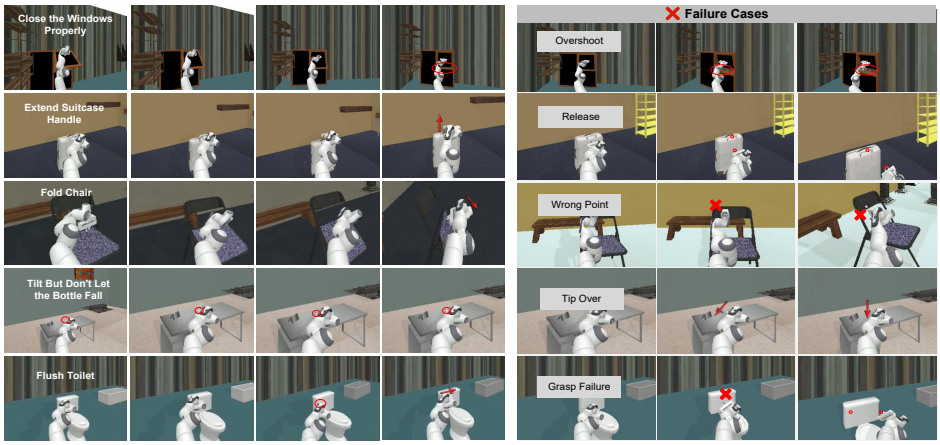

Fig. 4. **Main results of CorrectManip in various tasks.** Visual performances across sample tasks (left) and representative failure cases (right).

feedback from the observed failures. We then re-evaluate the advantage estimates for these trajectories and perform a limited number of additional PPO updates. Additionally, we integrate privileged action-level supervision from the action supervisor to improve sample efficiency and robustness. We introduce $\mathcal{L}_{\text{corr}}$ as an additional loss in the actor loss:

$$\mathcal{L}_{\text{corr}} = \frac{1}{T}\sum_{t=1}^{T}\Big(\alpha\, D_{\text{KL}}\Big(\pi(a_t \mid s_t) \,\|\, \pi^v(a_t^v \mid (s_0^v, s_t^v))\Big) + \beta\, \|a_t - a_t^v\|^2\Big), \tag{5}$$

where D_{KL} measures the divergence between the policy and supervisor distributions, and the L2 term enforces closeness of their outputs in the action space. The coefficients α and β balance these two objectives. This design ensures sample-efficient online fine-tuning by adapting the reward structure to observed failures and aligning the policy's behavior with privileged supervision.

IV. Experiments

In this section, we evaluate CorrectManip across simulated and real-world environments, answering four key questions: 1) Does our closed-loop framework improve open-world generalization over baseline methods? 2) What is the individual contribution of the components? 3) How effective are EvoGen and TTO in enhancing generalization? 4) How does CorrectManip perform in the real world compared to the baselines?

A. Experimental Setup

Baseline Methods. We compare CorrectManip with RoboGen [15] and GenSim2 [16], two state-of-the-art generative frameworks designed to create unlimited data for skill learning. For the EvoGen module, we conduct comparisons with other data generation frameworks. For the TTO module, we compare against AHA [22] and RACER [23], both are leading methods for failure discovery.

Train & Evaluation Setup. Each RL training loop collects rollouts from 8 parallel workers, with batches of 12,800 transitions and 50 epochs per update. We evaluate every 20,000 timesteps, saving the best policy based on the mean return. Environments for each task are automatically constructed by EvoGen using assets from PartNet Mobility [39] and Objaverse [40]. We train policies on a mobile manipulator and humanoid robots in PyBullet and Isaac Sim, respectively. The action supervisor is based on UMPNet [41], pretrained on the PartNet Mobility, covering diverse object categories and joint types. Replay buffer size is 6,400, and CEM sampling is used with 64 action candidates per step, and fine-tuned with EvoGen-generated scenarios.

Fig. 5. **Main results of CorrectManip in randomized environments.** Example task (*Open the Door*) demonstrating robust generalization across diverse spatial layouts, object shapes, and textures.

B. Main Results

Results in Various Tasks. We evaluate CorrectManip across six representative tasks. As shown in Table I, CorrectManip achieves an average success rate of 85.98%, significantly outperforming RoboGen [15] (25.16%) and GenSim2 [16] (40.76%). Notably, CorrectManip maintains superior performance even in challenging multi-stage tasks (TASK 4 and 5). Fig. 4 illustrates CorrectManip's robustness and failure recovery by showing both successful executions and its response to representative failure cases.

TABLE I
COMPARISON OF SUCCESS RATE BETWEEN CORRECTMANIP AND THE BASELINE METHODS ON SIX TASKS.

	TASK 1 *Open the Door*	**TASK 2** *Close the Windows Properly*	**TASK 3** *Extend Suitacase Handle*	**TASK 4** *Fold the Chair*	**TASK 5** *Tilt But don't Let the Bottle Fall*	**TASK 6** *Flush the Toilet*	**Average**
RoboGen [15]	7.16	24.43	31.25	25.19	34.71	28.24	25.16
GenSim2 [16]	29.70	45.54	56.43	43.56	20.79	48.51	40.76
CorrectManip	**84.14**	**86.78**	**87.52**	**81.25**	**91.64**	**84.60**	**85.98**

Results in Various Scenes. In randomized scenes (Fig. 5), CorrectManip generalizes across diverse layouts, shapes, and textures, while baselines suffer significant drops (Table II), even when equipped with the same perception modules. These results confirm that CorrectManip's joint perception-action learning and closed-loop adaptation are key to robust generalization.

TABLE II
COMPARISON OF SUCCESS RATE BETWEEN CORRECTMANIP AND THE BASELINE METHODS ON EXAMPLE TASKS IN RANDOMIZED ENVIRONMENTS. * DENOTES WE ADD A SET OF COMPARISON EXPERIMENTS USING THE SAME PERCEPTION MODULE.

Method	**TASK 1**			**TASK 2**		
	Shape	Texture	Spatial	Shape	Texture	Spatial
RoboGen	-	14.28	-	-	12.5	-
RoboGen*	9.52	21.34	11.06	4.88	9.37	7.23
GenSim2*	22.58	34.37	25.81	21.95	36.38	30.77
CorrectManip	**72.45**	**75.51**	**80.61**	**78.57**	**87.13**	**86.73**

TABLE III
ABLATION STUDY OF TTO UNDER DIFFERENT CONFIGURATIONS ON THREE SCENE TYPES (SPACIOUS, MODERATE, COMPACT). "ACTION SUPERVISOR W/O FT" DENOTES A PRE-TRAINED SUPERVISOR APPLIED DIRECTLY AT TEST TIME WITHOUT FINE-TUNING.

Module		**Scene Types**		
Action Supervisor	**Failure Discovery**	***Spacious***	***Moderate***	***Compact***
✗	✗	33.67	32.61	28.51
✗	✓	63.42	60.18	57.20
✓	✗	74.65	70.83	64.28
✓ (w/o FT)	✗	71.28	58.14	50.85
✓ (w/o FT)	✓	73.46	67.34	61.89
✓	✓	**78.13**	**74.73**	**68.61**

C. Ablation Studies

Ablation on EvoGen. We compare EvoGen with popular simulation frameworks in Table IV. EvoGen achieves the lowest values in both scene and task diversity. Here, lower scores indicate higher consistency and less redundancy, which highlights EvoGen's capability to create adaptive scenarios that support robust policy learning within a closed-loop framework.

Ablation on TTO. Table III shows the ablation results of the TTO module under varying scene complexities. The baseline without any adaptation achieves low success rates. Enabling failure discovery module improves performance by recovering from failures. Adding the action supervisor further improves performance by providing action-level correction. However, using the action supervisor without fine-tuning yields lower performance compared to the fine-tuned variant, particularly in complex scenes, yet still surpasses configurations without the action supervisor. Full TTO configuration consistently achieves the highest success rates, demonstrating that both components are complementary and fine-tuning is essential for robust adaptation.

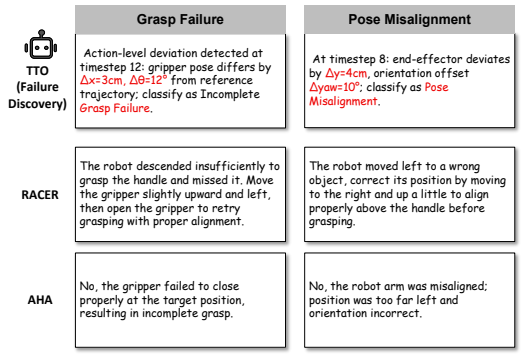

Fig. 6. Comparison of the Failure Discovery module with RACER [23] and AHA [22] in explaining two representative failure types: Grasp Failure and Pose Misalignment.

Comparison of Failure Discovery Module. Our failure discovery module identifies action-level failures, enabling precise classification of failure types, as shown in Fig. 6. In contrast, RACER [23] focuses on high-level description without quantifying the mismatch, while AHA [22] reports coarse explanations without actionable metrics. This comparison illustrates that our module offers more precise and actionable guidance, supporting robust policy correction in diverse failure types.

Closed-loop iterative improvement analysis. To evaluate CorrectManip under closed-loop iteration, we quantify its effectiveness across multiple iterations. Fig. 7(a) and (b) illustrate the diversity of scenes generated by EvoGen during the iteration. Fig. 7(c) presents the spatial distribution heatmap of generated objects, showing that scene coverage becomes more balanced as iteration proceeds. Fig. 7(d) presents the average success rates over iterations for different configurations. The complete CorrectManip achieves a consistent improvement, outperforming all ablations. Notably, CorrectManip (full) shows continuous improvement in all iterations. Without TTO, performance improves initially but soon plateaus. The CorrectManip (baseline) lacks closed-loop optimization and merely accumulates data, leading to limited improvement. These results validate that closed-loop optimization is the key to sustained generalization improvement.

TABLE IV

COMPARISON WITH THE LATEST SIMULATION FRAMEWORKS FOR ROBOTIC MANIPULATION. SCENE DIVERSITY (VIT) MEASURES THE VISUAL VARIATION BETWEEN SCENES AND TASK DIVERSITY (SELF-BLEU) QUANTIFIES THE SEMANTIC VARIATION BETWEEN TASK DESCRIPTIONS.

Simulation Framework	Simulation Platform	Scene Type	Autonomous Generation	Visual Perception	Closed-loop Adaptation	Scene Diversity ↓	Task Diversity ↓
RoboGen [15]	Pybullet / Genesis	**Unlimited**	✓	✗	✗	0.19	0.28
GenSim2 [16]	SAPIEN	**Unlimited**	✓	✓	✗	0.65	0.32
MetaWorld [4]	MuJoCo	1	✗	✗	✗	0.52	0.33
iGibson 2.0 [12]	Pybullet	15 homes, 108 rooms	✓	✓	✗	0.68	0.41
RoboCasa [5]	MuJoCo	12	✓	✓	✗	0.72	0.38
Maniskill2	SAPIEN	1	✗	✓	✗	0.33	0.67
RLBench [6]	V-REP + PyRep	1	✓	✓	✗	0.38	0.32
Behavior-1K [8]	Omniverse	8	✗	✗	✗	0.39	0.30
EvoGen (CorrectManip)	Pybullet / Isaac Sim	**Unlimited**	✓	✓	✓	**0.14**	**0.26**

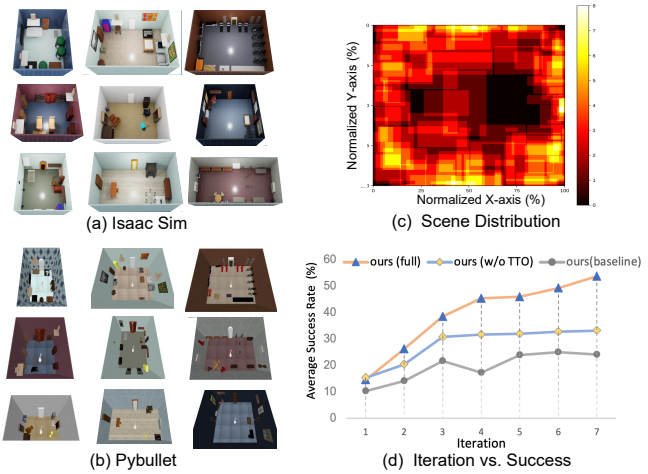

Fig. 7. **Iterative evaluation of CorrectManip.** (a) Example scenes generated in Isaac Sim and (b) PyBullet. (c) Normalized spatial distribution of generated objects, showing that scene coverage becomes more balanced as iterations progress. (d) Performance improvements across iterations, reporting the average success rate (%) for three settings: ours (**full**) with both TTO and EvoGen, ours **w/o TTO**, and ours (**baseline**) without closed-loop iteration.

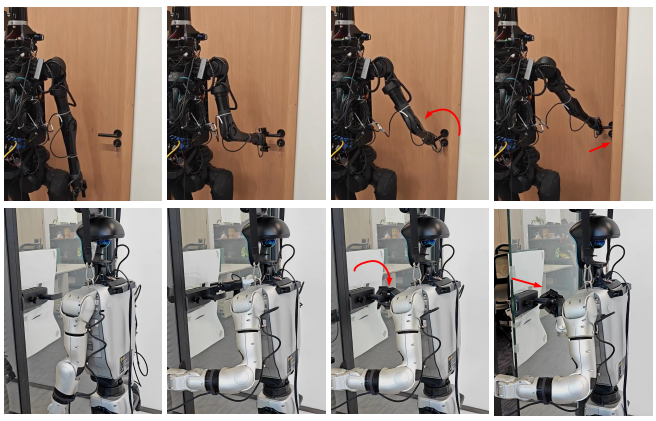

Fig. 8. Deployment of the reinforcement learning policy on physical humanoid robots Unitree H1 and G1 for door-opening tasks.

D. Real-world Experiments

Sim-to-real Transfer. We deploy our framework on two humanoid robots—a Unitree G1 with dexterous hands (Unitree Dex3-1) and an H1 with 2-finger grippers (Robotiq 2F-85)—to validate sim-to-real transfer and cross-platform capability. Considering the strong coupling between path planning and gait control in humanoid robots, as well as their high-dimensional, nonlinear, and hybrid dynamics, navigation tasks become significantly complex [42]. Therefore, we only deploy our policy for a manipulation task. The policy directly processes observations: visual input from RGB-D cameras (Intel Realsense D435i for G1, Orbbec 336L for H1) and real-time joint states from encoders. The policy runs on an NVIDIA Jetson AGX Orin NX module at 100 Hz. TTO provides privileged, action-level corrections and records failures (no parameter updates on the robot). During simulation iterations, these failures drive reward shaping and limited TTO updates (online finetuning) to produce the next policy. Fig. 8 illustrates the robot successfully grasping, rotating a handle, and pushing/pulling a door, demonstrating our framework's cross-platform transfer capabilities.

TABLE V

ITERATIVE PERFORMANCE ON THE REAL-WORLD DOOR-OPENING TASK. *DENOTES WE ADD A SET OF COMPARISON EXPERIMENTS USING THE SAME PERCEPTION MODULE AS CORRECTMANIP.

Method	Iter 3	Iter 5	Iter 7
RoboGen*	0/10	1/10	2/10
GenSim2*	0/10	2/10	3/10
CorrectManip	1/10	3/10	5/10

Experimental Performance Evaluation. Quantitative results across iterations are shown in Table V. The results clearly show that CorrectManip's success rate steadily increases with more iterations, robustly demonstrating the effectiveness of our framework. In each iteration, failures encountered in the real world guide the EvoGen to generate more targeted training scenarios, progressively addressing the policy's shortcomings. By iteration 7, CorrectManip achieves 5/10 successes, outperforming baselines RoboGen (2/10) and GenSim2 (3/10). This validates the closed-loop adaptation's role in enhancing sim-to-real transferability.

V. CONCLUSION

We introduced CorrectManip, a closed-loop framework that continuously enhances policy generalization. CorrectManip integrates TTO, which identifies failure modes during

execution and provides fine-grained corrective signals, and EvoGen, which adaptively generates training data conditioned on the discovered failures. Together, they establish a robust paradigm for autonomous and continual skill refinement. Extensive experiments across diverse tasks demonstrate that CorrectManip achieves an average success rate of 85.98%, significantly outperforming baseline methods by 45.22%. Preliminary deployment on Unitree humanoid robots provides further evidence of its sim-to-real transfer capability.

Limitations. Our real-world evaluation has so far been limited to Unitree H1 and G1 and a small set of tasks, leaving broader validation across diverse robotic platforms and long-horizon tasks as an open challenge. Future work will focus on expanding adaptability to a wider range of robotic platforms and more complex manipulation tasks, moving toward scalable and robust open-world autonomy.

VI. Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (NSFC) under grant No.62403389, the Provincial Natural Science Foundation of Zhejiang under grant No. QKWL25F0301, and the Zhejiang Key Laboratory of Low-Carbon Intelligent Synthetic Biology (2024ZY01025).

References

[1] A. Hurst *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[2] G. Team *et al.*, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint arXiv:2403.05530*, 2024.

[3] Z. Jiang *et al.*, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 923–16 930.

[4] T. Yu *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.

[5] S. Nasiriany *et al.*, "RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots," Jun. 2024.

[6] S. James *et al.*, "RLBench: The Robot Learning Benchmark & Learning Environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.

[7] J. Gu *et al.*, "Maniskill2: A unified benchmark for generalizable manipulation skills," *arXiv preprint arXiv:2302.04659*, 2023.

[8] C. Li *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 80–93.

[9] J. Huang *et al.*, "An Embodied Generalist Agent in 3D World," May 2024, arXiv:2311.12871.

[10] D. Ghosh *et al.*, "Octo: An Open-Source Generalist Robot Policy," in *Robotics: Science and Systems XX*. Robotics: Science and Systems Foundation, Jul. 2024.

[11] X. Puig *et al.*, "Habitat 3.0: A co-habitat for humans, avatars and robots," 2023.

[12] C. Li *et al.*, "iGibson 2.0: Object-Centric Simulation for Robot Learning of Everyday Household Tasks," Nov. 2021, arXiv:2108.03272.

[13] A. Mandlekar *et al.*, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *7th Annual Conference on Robot Learning*, 2023.

[14] Z. Xue *et al.*, "Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning," *arXiv preprint arXiv:2502.16932*, 2025.

[15] Y. Wang *et al.*, "Robogen: Towards unleashing infinite data for automated robot learning via generative simulation," *arXiv preprint arXiv:2311.01455*, 2023.

[16] P. Katara *et al.*, "Gensim2: Scaling robot data generation with multi-modal and reasoning llms," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6672–6679.

[17] H. Bharadhwaj *et al.*, "Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4788–4795.

[18] T. Chen *et al.*, "Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation," *arXiv preprint arXiv:2506.18088*, 2025.

[19] W. Wang and G. D. Hager, "Domain adaptation of visual policies with a single demonstration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 17 208–17 215.

[20] P. Yin *et al.*, "Rapidly adapting policies to the real-world via simulation-guided fine-tuning," in *The Thirteenth International Conference on Learning Representations*, 2025.

[21] Y. Chen *et al.*, "Conrft: A reinforced fine-tuning method for vla models via consistency policy," *arXiv preprint arXiv:2502.05450*, 2025.

[22] J. Duan *et al.*, "AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[23] Y. Dai, J. Lee, N. Fazeli, and J. Chai, "Racer: Rich language-guided failure recovery policies for imitation learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[24] K. Black *et al.*, "π_0: A Vision-Language-Action Flow Model for General Robot Control," Nov. 2024, arXiv:2410.24164.

[25] A. O'Neill *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," Jun. 2024, arXiv:2310.08864.

[26] M. J. Kim *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," Sep. 2024, arXiv:2406.09246.

[27] J. Wen *et al.*, "TinyVLA: Towards Fast, Data-Efficient Vision-Language-Action Models for Robotic Manipulation," Nov. 2024, arXiv:2409.12514.

[28] Y. Ma *et al.*, "A Survey on Vision-Language-Action Models for Embodied AI," Nov. 2024, arXiv:2405.14093.

[29] C. Finn and S. Levine, "Deep Visual Foresight for Planning Robot Motion," Mar. 2017, arXiv:1610.00696.

[30] F. Ebert *et al.*, "Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning," Oct. 2018, arXiv:1810.03043.

[31] W. Huang *et al.*, "Inner Monologue: Embodied Reasoning through Planning with Language Models," Jul. 2022, arXiv:2207.05608.

[32] A. Ajay *et al.*, "Compositional Foundation Models for Hierarchical Planning," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023, pp. 22 304–22 325.

[33] Q. Bu *et al.*, "Closed-Loop Visuomotor Control with Generative Expectation for Robotic Manipulation," Sep. 2024, arXiv:2409.09016.

[34] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[35] N. Reimers, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[36] S. Liu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[37] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.

[38] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[39] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 097–11 107.

[40] M. Deitke *et al.*, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 142–13 153.

[41] Z. Xu *et al.*, "Universal manipulation policy network for articulated objects," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2447–2454, 2022.

[42] C. Peng *et al.*, "Real-time safe bipedal robot navigation using linear discrete control barrier functions," 2024.