DEXPOLICYEVO: EVOLUTION OF DEXTEROUS GRASPING POLICIES THROUGH FAILURE-DRIVEN SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review



Figure 1: We propose **DexPolicyEvo**, a system that enables dexterous hand reaching-and-grasping policies to evolve autonomously toward open-world object grasping capabilities through failure-driven self-improvement.

ABSTRACT

Enabling dexterous grasping policies to continually improve toward open-vocabulary object grasping remains an open challenge. Achieving this efficiently requires understanding a policy's current capabilities and weaknesses to tailor its training accordingly. With this goal in mind, we introduce **DexPolicyEvo**, a system that enables dexterous grasping policies to evolve autonomously through failure-driven self-improvement. Its key idea is to connect failure diagnosis with targeted training-data construction and geometric curriculum design through measurable and editable object properties. Six collaborating agents implement this improvement loop: the Diagnostic Designer and Policy Evaluator construct tests across everyday scenarios and assess policy capabilities; the Failure Analyst and Repair Designer identify weaknesses and devise targeted training data and curricula; and the Policy Trainer and Policy Verifier update the policy and validate its improvement. As the policy improves, the agents revise its training plan to address emerging weaknesses and further expand its grasping capabilities. Experiments demonstrate that DexPolicyEvo continually improves policy generalization while preserving existing grasping capabilities.

1 INTRODUCTION

Grasping is a fundamental capability for robots interacting with the diverse objects found in everyday environments. Anthropomorphic dexterous hands offer greater flexibility than simple grippers and support more complex manipulation, but their high degrees of freedom make training reaching-and-grasping policies more challenging (Yu & Wang, 2022). Large-scale datasets and advances in policy learning have substantially advanced the development of generalizable dexterous grasping policies (Wang et al., 2023; Xu et al., 2023; Wan et al., 2023; Wang et al., 2025; Huang et al., 2025b). However, enabling these policies to continually improve their capabilities and generalization

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

toward open-world object grasping remains an open challenge. Everyday environments expose robots to new object categories and geometries beyond those anticipated during initial training. Although large-scale training improves grasping performance, naive expansion of training datasets can be inefficient when it overlooks the current policy's learning needs. Training effort may be disproportionately allocated to objects the policy already handles well, while its specific weaknesses remain insufficiently addressed. Moreover, how much a policy benefits from training on a given object depends on its current capabilities: overly challenging objects can hinder learning progress, whereas overly easy ones offer limited additional gains. These challenges call for a self-improving system that uses feedback from policy execution to guide continued learning, enabling the policy to handle an increasingly diverse range of objects.

Recent studies of self-improving LLM agents demonstrate the potential of such feedback-driven improvement, achieving gains in task performance through repeated evaluation and adaptation (Zhang et al., 2025; Acikgoz et al., 2025; Weng et al., 2026). These findings suggest that execution feedback could similarly guide continued policy improvement in embodied learning. Recent robotic systems have demonstrated improvements through autonomous experience collection and feedback-guided learning, including in dexterous manipulation (Zhou et al., 2024; Zhu et al., 2025b;a). Agentic frameworks further automate parts of this process by using execution results to guide subsequent training and experimentation (Xiao et al., 2026b; You et al., 2026). A central challenge for sustained self-improvement in dexterous grasping is to turn execution failures into actionable training decisions: which object properties to target, how to construct training objects around them, and how to make difficult targets learnable as the policy evolves.

We introduce **DexPolicyEvo**, a failure-driven self-improving system in which six agents collaborate to evolve dexterous grasping policies. Its key idea is to use measurable and editable object properties to connect failure diagnosis with targeted training-data construction and geometric curriculum design. To advance toward open-world grasping, the system constructs diagnostic data across diverse everyday scenarios. In each improvement loop, agents use the object categories associated with a selected scenario and available diagnostic history to construct a new diagnostic set, then evaluate the current policy on it. By contrasting low- and high-performing objects and their execution outcomes, they identify object properties associated with grasp failures. These findings guide the construction of new training objects, with curricula introduced when direct training makes insufficient progress. The agents then update the policy using the resulting data and check for gains on training objects, transfer to diagnostic objects, and retention of existing capabilities. An accepted policy becomes the starting point for the next loop, while diagnostic findings and training outcomes inform subsequent data construction. Shared memory and replay support continued improvement and capability retention.

We integrate two dexterous grasping policies into DexPolicyEvo and let them evolve autonomously. After each loop, we evaluate the updated policies on held-out diagnostic objects, a fixed unseen-object challenge set, and their original benchmarks. The results demonstrate continued improvement in both policies without compromising their existing capabilities.

Our contributions are summarized as below:

- We propose **DexPolicyEvo**, an agentic system that enables dexterous grasping policies to evolve autonomously through coordinated diagnosis, analysis, repair, and policy updates.
- We propose a failure-driven self-learning mechanism that connects failure diagnosis with targeted training-data construction and geometric curriculum design, guiding both what the policy learns next and how it learns to grasp challenging objects.
- Extensive experiments show that DexPolicyEvo improves generalization across loops, preserves original benchmark performance, and outperforms data-expansion baselines under matched training-asset budgets.

2 Related Work

2.1 Dexterous Policy Learning

Large-scale parallel simulation has enabled reinforcement learning policies to scale to diverse objects and dexterous grasping settings (Makoviychuk et al., 2021; Huang et al., 2025a; Lum et al., 2024; She et al., 2022; Yuan et al., 2025; Kedia et al., 2026). For universal object grasping, most approaches

combine large-scale grasp proposals with goal-conditioned reinforcement learning (Wu et al., 2023; Zhang et al., 2024; Liu et al., 2026), and employ object curricula and teacher–student distillation to obtain vision-based policies that generalize to unseen objects (Xu et al., 2023). UniDexGrasp++ (Wan et al., 2023) further introduces geometry-aware curricula and iterative generalist–specialist learning to alleviate optimization conflicts across large object collections. ResDex (Huang et al., 2025b) improves training efficiency using base policies with weak cross-object generalization, a mixture of experts, and residual multi-task reinforcement learning, whereas UniGraspTransformer (Wang et al., 2025) trains object-specific reinforcement learning policies to produce large-scale successful trajectories and distills them offline into a unified Transformer policy. These methods primarily improve policy learning over predefined object distributions without an automatic mechanism for repairing insufficient coverage. In contrast, DexPolicyEvo continuously expands a trained policy's generalization boundary through closed-loop diagnosis and targeted repair.

2.2 Self-Improving Robot Learning

Recent work has moved beyond fixed demonstration datasets toward autonomous interaction loops for policy improvement. A pretrained policy is deployed in real or imagined environments, and the resulting trajectories are processed and incorporated into subsequent policy updates (Bousmalis et al., 2023; Zhou et al., 2024; Liang et al., 2026; Zhu et al., 2025a; Ghasemipour et al., 2025; Xiao et al., 2026a; Guo et al., 2026; Yang et al., 2026a). Although differing in how this experience is generated and utilized, these methods share the goal of expanding the trajectory distribution available for policy improvement. More closely related to our work, failure-directed methods make testing and data selection dependent on the current policy. They search for vulnerable contexts, probe failure states, or select existing trajectories according to their predicted utility for downstream improvement (Parashar et al., 2025; Sagar et al., 2026; Xiao et al., 2026a; Dass et al., 2026). Unlike broad experience expansion, these approaches explicitly use policy outcomes to determine where to test or which data to reuse. However, their search space is typically defined over contexts, visited states, environmental variations, or trajectories already available. Recent agentic systems further allow foundation-model agents to operate the improvement loop itself, automating self-resetting data collection, revising policy-training procedures, or repairing executable skills (Li et al., 2026a; Xiao et al., 2026b; Lu et al., 2026). The persistent component being improved is therefore the policy data, training procedure, control program, or skill memory. DexPolicyEvo brings this feedback-driven agentic paradigm for simulation-based dexterous grasping. Unlike systems that mainly expand or reuse trajectories, it actively diagnoses input-conditioned failures and converts them into measurable and editable specifications for constructing targeted training objects.

2.3 Generative Simulation for Robot Learning

Generative simulation provides scalable alternatives to manually constructed robot-learning environments. Procedural systems generate diverse interactive worlds through explicit scene and asset rules (Deitke et al., 2022b; Raistrick et al., 2024), while controllable methods condition scene construction on language, task specifications, or physical constraints (Yang et al., 2024b;a; Lee et al., 2025). Recent agentic systems further integrate planning, asset retrieval, scene assembly, and physical validation into iterative generation processes (Hu et al., 2024; Yang et al., 2026b; Xia et al., 2026). For robot learning, related systems jointly generate assets, environments, tasks, and training supervision (Katara et al., 2024; Wang et al., 2024; Hua et al., 2024), while object-level methods produce interactive, simulation-compatible assets (Dai et al., 2024; Le et al., 2025; Li et al., 2026b). FATE further uses feedback to repair scene configurations and policy specifications, improving the executability of generated tasks (Wei et al., 2026). These advances provide the generative infrastructure needed to instantiate DexPolicyEvo's diagnostic and repair assets.

3 Methodology

3.1 System Overview

Given a dexterous grasping policy π_0, DexPolicyEvo coordinates six agents to improve it through iterative diagnosis, analyze, repair, and update as shown in Figure 2. The **Diagnostic Designer** constructs diagnostic objects, and the **Policy Evaluator** collects execution evidence. The **Failure**

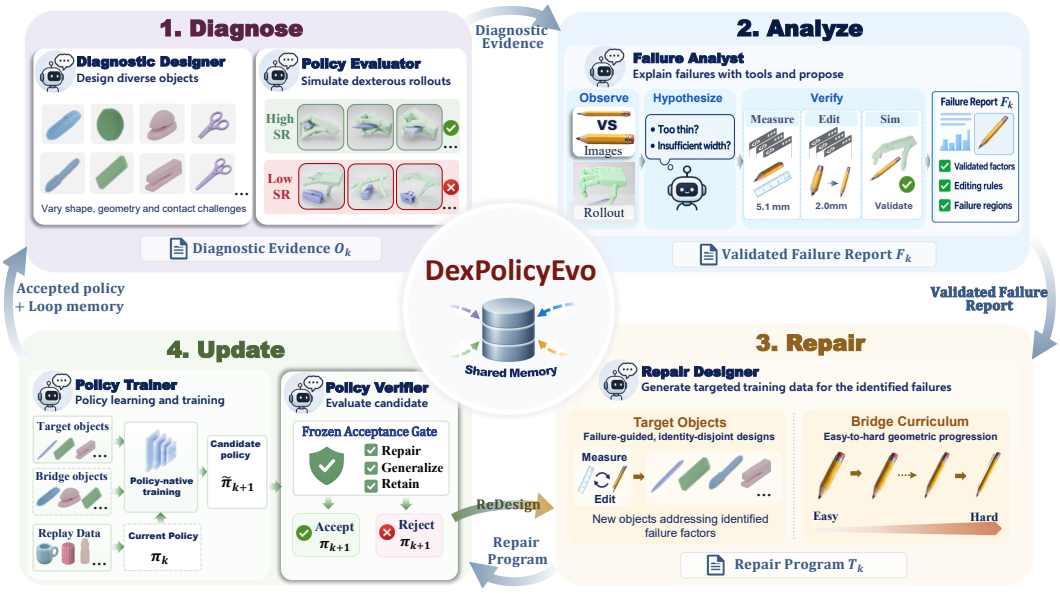

Figure 2: **DexPolicyEvo transforms failures into targeted policy improvements.** Collaborating agents **Diagnose** policy weaknesses, **Analyze** actionable failure factors, **Repair** through targeted data generation, and **Update** the policy, with verification closing the loop for continued self-improvement.

Analyst identifies measurable and editable failure factors, which the **Repair Designer** uses to construct training objects and curricula. The **Policy Trainer** adapts the policy through its native training procedure, and the **Policy Verifier** assesses candidate updates. Shared memory carries diagnoses, training outcomes, and verification results across loops.

Let $\mathcal{Z}$ denote the space of object configurations, where each $z \in \mathcal{Z}$ specifies an object instance, scale, initial stable pose, and simulator parameters. We denote the policy and shared memory at loop k by π_k and $\mathcal{H}_k$, respectively.

3.2 Active Diagnostic Construction

To construct diagnostic data across diverse everyday contexts, we first prompt the Diagnostic Designer to randomly generate a set of scenario categories. For each scenario, the agent identifies associated object categories and specifies corresponding object shapes to generate.

Within these scenario contexts, the Diagnostic Designer uses the task schema, prior diagnostic coverage, and unresolved regions in memory $\mathcal{H}_k$ to propose a diagnostic plan $\mathcal{P}_k$, informed by previously constructed diagnostic sets $\mathcal{D}^{\text{diag}}_{<k} = \bigcup_{j<k} \mathcal{D}^{\text{diag}}_j$. Each element of $\mathcal{P}_k$ specifies the scenario, an associated object family, editable-factor ranges, scene constraints, and a sampling budget. An asset toolchain generates the corresponding 3D object shapes from these specifications and checks physical validity and simulator compatibility, yielding the loop-k diagnostic set $\mathcal{D}^{\text{diag}}_k$. The configurations generated from each proposal form a diagnostic group indexed by c, sharing the proposed object family and scenario constraints.

The Policy Evaluator then freezes object membership, initial states, rollout seeds, and success criteria, and evaluates π_k on $\mathcal{D}^{\text{diag}}_k$. This produces the diagnostic evidence $\mathcal{O}_k$, which contains per-seed success labels, stage-aligned trajectory summaries, and key failure states for each object, connecting object properties to performance for subsequent analysis.

3.3 Failure Analysis

The Failure Analyst identifies measurable and editable factors that connect execution failures to targeted training-data construction.

For each diagnostic group c, the Failure Analyst first contrasts low- and high-performing object sets, denoted by $\mathcal{D}_{k,c}^{\text{low}}$ and $\mathcal{D}_{k,c}^{\text{high}}$, according to empirical success rates. When reliable within-group controls are unavailable, better-performing groups provide reference comparisons. Following an *evidence–hypothesis–validation* procedure, the analyst compares rendered objects, trajectory summaries, and key failure states to propose falsifiable hypotheses about the causes of failure, such as an excessively thin cross-section or an insufficient graspable region.

For each hypothesis h, the analyst retrieves or synthesizes a deterministic scalar evaluator $\phi_h : \mathcal{Z} \to \mathbb{R}$ to measure the proposed factor. A factor is retained only when its measured values sufficiently separate low- and high-performing cases. For each retained factor, a decision stump identifies a failure interval $\mathcal{R}_{c,h}^{\text{fail}}$ associated with a performance gap between instances inside and outside the interval, and the factor is paired with a semantics-preserving editor $e_h(z, \delta)$. The editor returns a configuration edited toward the requested factor value δ, in the same units as $\phi_h(z)$; achieved values are remeasured after editing. The resulting failure-analysis report is

$$\mathcal{F}_k = \{(c, h, \phi_h, e_h, \mathcal{R}_{c,h}^{\text{fail}}, \rho_{c,h})\},$$

where $\rho_{c,h}$ records supporting evidence. Each group may retain multiple factors. Statistical checks and tool validation are detailed in the appendix.

3.4 Training Data Construction for Policy Repair

Using $\mathcal{F}_k$, the Repair Designer constructs *Target objects* to address diagnosed weaknesses and, when needed, easier intermediate variants called *Bridge objects*. *Target objects* remain identity-disjoint from the diagnostic objects, which are reserved for evaluation.

For each diagnostic group c, let q denote the asset toolchain's conditional generator and H_c the number of retained factors, indexed locally by $h = 1, \ldots, H_c$. Let $z^{\text{tar}} \in \mathcal{Z}$ denote a candidate *Target object* configuration. Its generation and feature-based admission are defined by

$$\begin{aligned} z^{\text{tar}} &\sim q\Big(z \mid c, \mathcal{D}_{k,c}^{\text{low}}, \{\phi_h, \mathcal{R}_{c,h}^{\text{fail}}\}_{h=1}^{H_c}\Big), \\ A(z^{\text{tar}}) &= \mathbb{I}\big[\exists h \in \{1, \ldots, H_c\}, \quad \phi_h(z^{\text{tar}}) \in \mathcal{R}_{c,h}^{\text{fail}}\big]. \end{aligned} \tag{1}$$

Here, $A : \mathcal{Z} \to \{0, 1\}$ is the feature-based admission indicator for the current diagnostic group, and $\mathbb{I}[\cdot]$ equals 1 when its condition holds and 0 otherwise. Thus, $A(z^{\text{tar}}) = 1$ means that the candidate falls into the diagnosed failure region for at least one retained factor. Candidates are edited when needed and remeasured, and they must also pass physical-validity, simulator-compatibility, and identity-disjointness checks. Admitted candidates form the loop-k Target set $\mathcal{D}_k^{\text{tar}}$.

DexPolicyEvo first attempts direct training on each original *Target object* under a preset budget. If a *Target object* remains too difficult for the current policy, the Repair Designer creates an easier variant $z^{(0)}$ whose feature values lie in the high-performing reference range, and then constructs a Bridge curriculum $z^{(0)} \to z^{(1)} \to \cdots \to z^{(L)} \to z^{\text{tar}}$ through controlled edits. Here, $z^{(\ell)}$, $\ell = 0, \ldots, L$, denotes a *Bridge object* configuration, with L indexing the final Bridge stage before the original *Target object* z^{tar}. Along the curriculum, feature values progressively return to those of the original *Target object* while preserving object identity and task semantics. Each completed stage warm-starts the next, and curriculum success requires eventual mastery of the unedited *Target object*.

3.5 Policy Update

The Policy Trainer fine-tunes the current policy on *Target objects*, available *Bridge objects*, and Replay data from earlier accepted loops. If verification fails, training observations guide additional *Bridge objects* and further training.

The Policy Verifier compares the candidate with the incumbent on the Target set, diagnostic set, and a held-out validation set to assess repair, transfer, and retention. Acceptance requires improved Target performance, no diagnostic degradation, and validation performance within a small tolerance of the incumbent. An accepted candidate replaces the incumbent, and the loop's findings and successful training data update shared memory and Replay, respectively; otherwise, the incumbent is retained.

4 EXPERIMENTS

4.1 EXPERIMENTAL GOALS AND SETUP

We validate the effectiveness of DexPolicyEvo through the following four questions: (1) Can its closed-loop repair process continually improve performance on newly discovered failure regions while retaining existing dexterous grasping capabilities? (2) Does failure-guided repair-data construction outperform conventional data expansion strategies? (3) When Target objects are difficult to learn directly, can the Bridge curriculum further facilitate policy repair? (4) Can the discovered failure factors be validated through measurable features and controlled geometric edits?

4.1.1 BASE POLICIES AND TASK

We evaluate DexPolicyEvo on UniGraspTransformer (Wang et al., 2025) and ResDex (Huang et al., 2025b), which are two simulation-based reinforcement-learning dexterous grasping policies with distinct learning and policy-integration paradigms. Both policies are initialized from checkpoints pretrained on the DexGraspNet (Wang et al., 2023) object distribution. In its original training pipeline, UniGraspTransformer trains object-specific expert policies to collect successful trajectories and distills them into a unified policy. In contrast, ResDex uses residual reinforcement learning for multi-object policy optimization. DexPolicyEvo updates each current policy through its native training interface using the generated repair data and Replay buffer. An accepted checkpoint initializes the next evolution loop.

All experiments are conducted in Isaac Gym with a Shadow Hand on a state-based single-object grasping task. DexPolicyEvo does not modify the policy observation space, action space, network architecture, or native training objective. Instead, a fixed policy adapter converts newly constructed object assets into the native training and evaluation inputs required by each policy.

4.1.2 EVALUATION SUITES AND METRICS

We use object-level success rate as the primary metric. For each object, the success rate is the fraction of successful rollouts, and the performance of an evaluation set is the macro average over its objects. We organize the evaluation into three functional groups: failure-region repair, capability expansion, and original-capability retention.

Failure-region repair. At loop k, $\Delta\mathrm{Target}$ and $\Delta\mathrm{Diag.}$ measure the success-rate changes from the incumbent policy π_{k-1} to the candidate policy $\widetilde{\pi}_k$ on the Target and Diagnostic set, respectively. $\Delta\mathrm{Target}$ measures adaptation on the current repair targets, while $\Delta\mathrm{Diag.}$ evaluates whether the improvement generalizes to identity-disjoint diagnostic objects not used for training. Because these sets are constructed for the corresponding loop and repair strategy, the two metrics characterize within-loop repair rather than a globally shared test distribution.

Capability expansion. For a multi-loop run, $\mathrm{Cum.\text{-}Frontier}$ is the candidate policy's macro-averaged success rate over all frozen Diagnostic-Test sets constructed through all loops. It measures the accumulated coverage of failure regions discovered during evolution. For cross-method comparison, we use $\mathrm{Shared\ Frontier}$, which aggregates the frozen Target sets constructed across all compared methods. Every final policy is evaluated on this common object test set. More importantly, we additionally evaluate generalization on a frozen Challenge set. It contains the 500 lowest-success objects under the initial policy among the first 1,500 uniformly preprocessed Objaverse (Deitke et al., 2022a) objects. This set provides a fixed external probe of policy performance on some Out-of-Distribution difficult objects.

Original-capability retention. Following the benchmark definitions of the base methods, we use their predefined $\mathrm{Seen\text{-}Cat.}$ and $\mathrm{Unseen\text{-}Cat.}$ splits on DexGraspNet. We report absolute success rates on these splits to evaluate whether failure-region repair preserves the capabilities represented by the original policy benchmarks.

4.2 MULTI-LOOP POLICY IMPROVEMENT

We evaluate DexPolicyEvo on two policies, UniGraspTransformer (Wang et al., 2025) and ResDex (Huang et al., 2025b), enabling each to autonomously self-improve over three loops. After each

Table 1: Three-loop evaluation of DexPolicyEvo on two dexterous grasping policies. ΔTarget and ΔDiag. measure improvement over the policy at the beginning of each loop, while Cum.-Frontier evaluates all Diagnostic-Test sets accumulated through the three loops. Challenge is a frozen difficult-object set, and Seen-Cat. and Unseen-Cat. evaluate retention on the original benchmark distribution. Results are success rates (%).

Policy	Stage	Failure-region repair		Capability expansion		Original-capability retention	
		ΔTarget ↑	ΔDiag. ↑	Cum.-Frontier ↑	Challenge ↑	Seen-Cat. ↑	Unseen-Cat. ↑
UniGrasp Transformer (Wang et al., 2025)	Base	–	–	40.43	14.53 ± 0.13	76.50 ± 0.78	79.49 ± 0.53
	Loop 1	+46.00	+14.20	49.60	31.51 ± 0.14	76.59 ± 0.44	78.70 ± 0.12
	Loop 2	+35.40	+11.69	58.31	40.25 ± 0.06	77.22 ± 0.20	79.63 ± 0.47
	Loop 3	+40.05	+4.42	**59.89**	$\mathbf{42.28 \pm 0.09}$	$\mathbf{78.61 \pm 0.36}$	$\mathbf{81.24 \pm 0.41}$
ResDex (Huang et al., 2025b)	Base	–	–	3.44	0.570 ± 0.01	59.17 ± 0.41	62.24 ± 0.41
	Loop 1	+84.54	+78.98	60.46	17.72 ± 0.17	83.20 ± 0.18	$\mathbf{84.85 \pm 0.24}$
	Loop 2	+12.81	+7.69	66.29	47.55 ± 0.17	79.98 ± 0.25	80.08 ± 0.48
	Loop 3	+7.23	+6.26	**69.54**	$\mathbf{51.07 \pm 0.19}$	$\mathbf{83.27 \pm 0.17}$	84.49 ± 0.55

loop, we evaluate failure-region repair, capability expansion, and retention of original capabilities. Table 1 summarizes the results.

The joint improvements on Target and Diagnostic support the effectiveness of converting diagnosed failures into repair data. Target gains establish that the constructed objects support policy improvement, while gains on identity-disjoint Diagnostic objects show that this improvement transfers to the failure regions that motivated repair. Because Diagnostic objects are excluded from training, this transfer provides evidence that the repair data capture weaknesses shared across object identities. Both policies exhibit this pattern in every loop, including after earlier repairs have updated the policy, supporting the closed-loop design's ability to identify and address further weaknesses as the policy evolves.

The before-and-after examples in Figure 3 and Figure 4 illustrate these repairs at the level of grasp execution. On the displayed objects, the evolved policies form grasps around shapes that remain unsecured in the base-policy examples, including cylindrical, rounded, and flat or hollow objects. These cases complement the aggregate results by showing recovered grasping behaviors across varied geometries for both policy backbones.

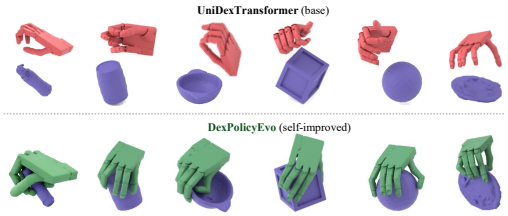

Figure 3: Qualitative results of UniGraspTransformer (Wang et al., 2025) before and after improvement with DexPolicyEvo.

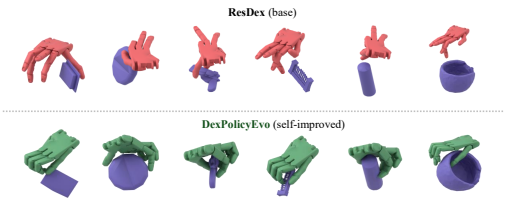

Figure 4: Qualitative results of ResDex (Huang et al., 2025b) before and after improvement with DexPolicyEvo.

The repair gains are accompanied by increasing success on the accumulated diagnostic frontier and, more decisively, on the fixed Challenge set. Challenge evaluates the same initially difficult objects after every loop. Its consistent improvement across successive updates therefore provides evidence of capability expansion on a common external evaluation set. This indicates that the benefits of failure-driven evolution extend beyond the loop-specific repair and diagnostic objects.

Both final policies also exceed their initial performance on Seen-Cat. and Unseen-Cat. Together with the Challenge improvements, these results show that the observed capability expansion preserves overall performance on the original benchmarks. The same combination of transferable repair, continued expansion, and final benchmark retention holds for both expert distillation and residual reinforcement learning, supporting DexPolicyEvo's applicability across distinct policy training pipelines through its fixed policy adapters.

Table 2: Comparison of repair-data construction strategies and curriculum variants at Loop 2 on UniGraspTransformer (Wang et al., 2025). ΔTarget and ΔDiag. report improvement from the common accepted Loop 1 policy to the policy updated at Loop 2. Shared Frontier evaluates all final policies on a common frozen set aggregating Target identities discovered across methods. Results are reported as success rates (%).

Method	Failure-region repair		Capability expansion		Original-capability retention	
	ΔTarget ↑	ΔDiag. ↑	Shared Frontier ↑	Challenge ↑	Seen-Cat. ↑	Unseen-Cat. ↑
Base	–	–	33.52 ± 0.23	31.51 ± 0.14	76.59 ± 0.44	78.70 ± 0.12
Repair-data construction						
Class-Matched Random	+12.22	+6.20	41.32 ± 0.24	34.42 ± 0.40	**79.93** ± 0.26	81.87 ± 0.70
Failure-Neighborhood	+12.65	+5.39	39.11 ± 0.33	34.05 ± 0.09	72.65 ± 0.25	75.78 ± 0.57
Bridge curriculum						
w/o Bridge	+7.79	+3.46	39.47 ± 0.10	32.68 ± 0.16	79.08 ± 0.25	**83.03** ± 0.40
DexPolicyEvo*	+29.18	+10.39	50.00 ± 0.16	36.67 ± 0.36	77.41 ± 0.91	81.79 ± 0.12
DexPolicyEvo	**+35.40**	**+11.69**	**54.13** ± 0.28	**40.25** ± 0.06	77.22 ± 0.20	79.63 ± 0.47

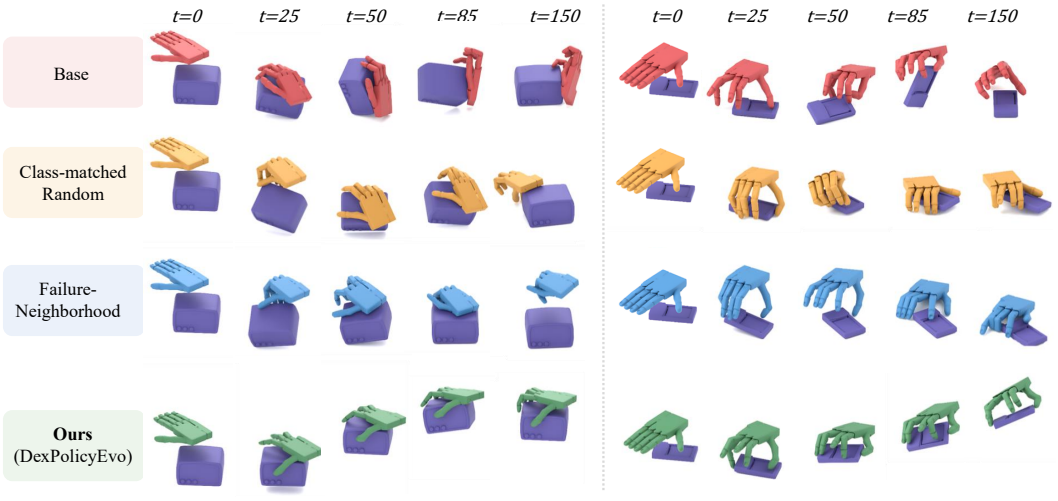

Figure 5: **Qualitative results of reaching and grasping sequences.** We compare the results of the base policy, two data-construction ablation variants, and DexPolicyEvo on two objects. All methods start from identical initial states, and columns show matched simulation steps.

4.3 Ablation Study on Repair-Data Construction

We evaluate three ablation settings: Class-Matched Random samples objects from the affected categories; Failure-Neighborhood generates objects semantically and morphologically similar to low-success examples; and DexPolicyEvo* reduces the number of original Targets so that its total number of Target and Bridge assets matches the other two settings. The first two settings omit measured failure constraints and use the same number of original Targets as the full method. All variants and the full method share the same accepted Loop 1 UniGraspTransformer (Wang et al., 2025) checkpoint and Diagnostic objects.

As shown in Table 2, DexPolicyEvo achieves stronger repair and better generalization on Shared Frontier and Challenge than Class-Matched Random and Failure-Neighborhood. DexPolicyEvo* retains these advantages under a matched training-asset budget, supporting the effectiveness of failure-guided repair beyond simply adding more data. These results highlight the value of constructing repair data around measured policy weaknesses over relying on category matching or semantic and morphological similarity alone.

Figure 5 further compares grasp execution from identical initial states at matched simulation steps. On both illustrated objects, the policy updated by DexPolicyEvo progresses from approach to an enclosing grasp and retains the objects through the final shown frames. The comparison sequences show object reorientation without maintaining an enclosing grasp or loss of contact. These examples

illustrate improvements in grasp acquisition and retention, complementing the quantitative advantage of failure-guided repair over generic data expansion.

4.4 Ablation Study on the Bridge Curriculum

To assess the contribution of Bridge after factor-guided Target construction, we compare DexPolicyEvo with a w/o Bridge variant. Both start from the same accepted Loop 1 policy and use the same Target objects satisfying the identified LOW-region constraints. The w/o Bridge variant trains directly on these Targets, whereas DexPolicyEvo introduces a curriculum when direct training makes insufficient progress. Starting from a variant $z^{(0)}$ in the high-performing reference range, controlled edits produce measured intermediate variants that progressively approach the unedited Target. Mastery of the unedited Target remains the final objective.

With the same Target objects, Bridge substantially improves repair and generalization over direct training (Table 2). The experts also master more original Targets, supporting the curriculum's role in making difficult repair cases learnable. The accompanying gains on Diagnostic, Shared Frontier, and Challenge show that these benefits transfer beyond the training objects and strengthen the resulting unified policy.

4.5 Failure Factor Validation

To examine whether failure analysis provides actionable information for policy repair, we test whether the discovered factors can be measured, edited, and translated into training objects satisfying specified geometric conditions.

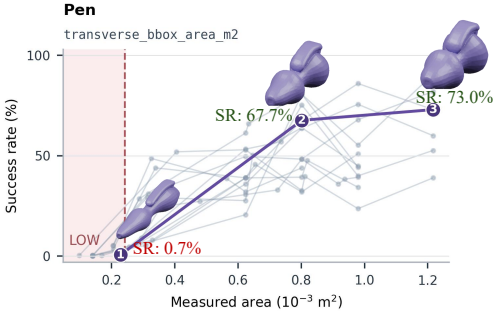

Figure 6: Controlled Pen factor edits and their success rates under a frozen policy.

Taking Pen as an example, failure analysis identifies the *transverse bounding-box area* as a measurable and editable geometric factor associated with grasp performance. It is the product of the two shorter dimensions of the object's bounding box, measured after upright alignment at the scale used by the policy. The diagnosed LOW region contains objects whose measured area does not exceed $0.242 \times 10^{-3}\,\mathrm{m}^2$. For visualization, Figure 6 shows three remeasured versions of one source identity. Their measured areas are $(0.228, 0.802, 1.217) \times 10^{-3}\,\mathrm{m}^2$, and success under the same frozen incoming Loop 2 policy is $(0.7, 67.7, 73.0)\%$. Across all Pen target identities, identity-macro success is 1.4% inside the LOW region versus 46.7% outside it ($\Delta_{\mathrm{LOW}} = 45.3$ percentage points; 95% identity-bootstrap CI $[40.3, 50.9]$); all 12 identity-level contrasts have the predicted direction. The example visualizes a measurable and editable geometric factor, while the all-identity comparison provides intervention-supported evidence of its behavioral relevance. Neither result identifies a unique physical cause of grasp failure

5 Conclusion

We presented DexPolicyEvo, a system for the evolution of dexterous grasping policies through failure-driven self-improvement. Its central idea is to connect failure diagnosis with training-data construction through measurable and editable object properties. These properties guide the generation of new repair targets and, when direct learning is difficult, geometric curricula that return progressively to the original targets. Native policy updates, replay, verification, and shared memory complete an iterative improvement loop. Experiments with UniGraspTransformer (Wang et al., 2025) and ResDex (Huang et al., 2025b) show that three loops improve performance on newly discovered failure regions and a fixed difficult-object challenge set while preserving or improving original benchmark performance. Matched-budget comparisons and curriculum ablations further support the effectiveness of targeted data construction and progressive training on difficult objects. These results suggest that execution feedback can guide both what a policy learns next and how it learns it. Our evaluation is limited to state-based grasping in simulation; extending the approach to visual observations, physical robots, and longer-horizon manipulation remains future work.

AI USE STATEMENT

In this work, we used generative AI as a component of DexPolicyEvo to propose diagnostic object configurations, formulate and test failure hypotheses, generate and validate measurement and editing tools, and design repair objects and Bridge curricula. We also used image and 3D generation models to create candidate synthetic object assets. These proposals, tools, and assets were subject to the measurement, validity, simulator, and evaluation procedures described in the paper. Separately, generative AI tools assisted the authors with literature discovery, research ideation and execution, drafting parts of the manuscript, and language editing. All AI-assisted outputs were reviewed and verified by the authors. The authors take responsibility for the final content, claims, code, and artifacts presented in this work.

REPRODUCIBILITY STATEMENT

We provide the algorithm, dataset construction and isolation protocols, training configurations, evaluation seeds, acceptance criteria, and additional experimental details in the appendices and supplementary materials.

REFERENCES

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. Self-improving LLM agents at test-time. *arXiv preprint arXiv:2510.07841*, 2025. URL https://arxiv.org/abs/2510.07841.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauzá, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. *arXiv preprint arXiv:2410.07408*, 2024.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. Datamil: Selecting data for robot imitation learning with datamodels. In *International Conference on Learning Representations*, volume 2026, pp. 1311–1336, 2026.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13142–13153, 2022a.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied ai using procedural generation. *Advances in neural information processing systems*, 35:5982–5994, 2022b.

Seyed Kamyar Seyed Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, and Igor Mordatch. Self-improving embodied foundation models. *arXiv preprint arXiv:2509.15155*, 2025.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *ICML*, pp. 19252–19282, 2024.

Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. Gensim2: Scaling robot data generation with multi-modal and reasoning llms. *arXiv preprint arXiv:2410.03645*, 2024.

Dongchi Huang, Tianle Zhang, Yihang Li, Ling Zhao, Jiayi Li, Zhirui Fang, Chunhe Xia, and Xiaodong He. Dexterous grasping with real-world robotic reinforcement learning. *arXiv preprint arXiv:2503.04014*, 2025a.

Ziye Huang, Haoqi Yuan, Yuhui Fu, and Zongqing Lu. Efficient residual learning with mixture-of-experts for universal dexterous grasping. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=BUj9VSCoET.

Pushkal Katara, Zhou Xian, and Katerina Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6672–6679. IEEE, 2024.

K. Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C. Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation. *ArXiv*, abs/2602.16863, 2026.

Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model. In *International Conference on Learning Representations*, volume 2025, pp. 17578–17602, 2025.

Sangmin Lee, Sungyong Park, and Heewon Kim. Dynscene: Scalable generation of dynamic robotic manipulation scenes for embodied ai. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12166–12175. IEEE, 2025.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. Roboclaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026a.

Zhe Li, Xiang Bai, Jieyu Zhang, Zhuangzhe Wu, Che Xu, Ying Li, Chengkai Hou, and Shanghang Zhang. Urdf-anything: Constructing articulated objects with 3d multimodal language model. *Advances in Neural Information Processing Systems*, 38:94974–95002, 2026b.

William Liang, Sam Wang, Hung-Ju Wang, Osbert Bastani, Yecheng Jason Ma, and Dinesh Jayaraman. Autonomous functional play with correspondence-driven trajectory warping. In *International Conference on Learning Representations*, volume 2026, pp. 111546–111571, 2026.

Qingtao Liu, Zhengnan Sun, Yu Cui, Haoming Li, Gaofeng Li, Lin Shao, Jiming Chen, and Qi Ye. Dexrepnet++: Learning dexterous robotic manipulation with geometric and spatial hand-object representations. *IEEE Transactions on Robotics*, 2026.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic/skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D. Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. *ArXiv*, abs/2407.02274, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, N. Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *ArXiv*, abs/2108.10470, 2021.

Anjali Parashar, Joseph Zhang, Yingke Li, and Chuchu Fan. Cost-aware discovery of contextual failures using bayesian active learning. In *9th Annual Conference on Robot Learning*, 2025.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, et al. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21783–21794. IEEE, 2024.

Som Sagar, Jiafei Duan, Sreevishakh Vasudevan, Yifan Zhou, Heni Ben Amor, Dieter Fox, and Ransalu Senanayake. Robomd: Uncovering robot vulnerabilities through semantic potential fields. In *International Conference on Learning Representations*, volume 2026, pp. 110599–110624, 2026.

Qijin She, Ruizhen Hu, Juzhan Xu, Min Liu, Kai Xu, and Hui Huang. Learning high-dof reaching-and-grasping via dynamic representation of gripper-object interaction. *ACM Transactions on Graphics (SIGGRAPH 2022)*, 41(4), 2022.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. UniDexGrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. DexGraspNet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2023.

Wenbo Wang, Fangyun Wei, Lei Zhou, Xi Chen, Lin Luo, Xiaohan Yi, Yizhong Zhang, Yaobo Liang, Chang Xu, Yan Lu, Jiaolong Yang, and Baining Guo. Unigrasptransformer: Simplified policy distillation for scalable dexterous robotic grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12199–12208, June 2025.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 51936–51983, 2024.

Bingchuan Wei, Bingqi Huang, Jingheng Ma, Zeyu Zhang, and Sen Cui. FATE: Closed-loop feasibility-aware task generation with active repair for physically grounded robotic curricula. *arXiv preprint arXiv:2603.01505*, 2026. URL https://arxiv.org/abs/2603.01505.

Zhaotian Weng, Antonis Antoniades, Deepak Nathani, Zhen Zhang, Xiao Pu, and Xin Eric Wang. Group-evolving agents: Open-ended self-improvement via experience sharing. *arXiv preprint arXiv:2602.04837*, 2026. URL https://arxiv.org/abs/2602.04837.

Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. In *Conference on Robot Learning*, pp. 618–629. PMLR, 2023.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, et al. Sage: Scalable agentic 3d scene generation for embodied ai. *arXiv preprint arXiv:2602.10116*, 2026.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, volume 2026, pp. 13209–13236, 2026a.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. Enpire: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026b.

Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, Tengyu Liu, Li Yi, and He Wang. UniDexGrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026a.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16262–16272. IEEE, 2024a.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. *Advances in neural information processing systems*, 38:140319–140351, 2026b.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16277–16287. IEEE, 2024b.

Haoxiang You, Zeyu Shen, Yilang Liu, Zhicheng Zheng, Lihan Zha, Kashu Yamazaki, Mingtong Zhang, Suning Huang, Jiankai Sun, Qianzhong Chen, Lucy He, Kaiyuan Liu, Haoran Chang, Katerina Fragkiadaki, Dhruv Shah, Mac Schwager, Peter Henderson, Ian Abraham, and Canwen Xu. EmbodiedSWE: Coding agents for long horizon dexterous robotics. *arXiv preprint arXiv:2609.27308*, 2026. URL https://arxiv.org/abs/2609.27308.

Chunmiao Yu and Peng Wang. Dexterous manipulation for multi-fingered robotic hands with reinforcement learning: A review. *Frontiers in Neurorobotics*, 16:861825, 2022. doi: 10.3389/fnbot.2022.861825.

Haoqi Yuan, Bohan Zhou, Yuhui Fu, and Zongqing Lu. Cross-embodiment dexterous grasping with reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hui Zhang, Sammy Christen, Zicong Fan, Otmar Hilliges, and Jie Song. Graspxl: Generating grasping motions for diverse objects at scale. In *European Conference on Computer Vision*, pp. 386–403. Springer, 2024.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025. URL https://arxiv.org/abs/2505.22954.

Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

Kefei Zhu, Fengshuo Bai, YuanHao Xiang, Yishuai Cai, Xinglin Chen, Ruochong Li, Xingtao Wang, Hao Dong, Yaodong Yang, Xiaopeng Fan, and Yuanpei Chen. DexFlyWheel: A scalable and self-improving data generation framework for dexterous manipulation. *Advances in Neural Information Processing Systems*, 38:4236–4265, 2025a. URL https://papers.neurips.cc/paper_files/paper/2025/hash/066c5542795287822f4f076cf330e5a2-Abstract-Conference.html.

Yufei Zhu, Yiming Zhong, Zemin Yang, Peishan Cong, Jingyi Yu, Xinge Zhu, and Yuexin Ma. EvolvingGrasp: Evolutionary grasp generation via efficient preference alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11665–11674, 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zhu_EvolvingGrasp_Evolutionary_Grasp_Generation_via_Efficient_Preference_Alignment_ICCV_2025_paper.html.

APPENDIX CONTENTS

A Detailed Algorithm **15**

A.1 Closed-loop execution 15

A.2 Agent artifacts and tool contracts 15

B Policy and Task Adapters **16**

B.1 Shared task and interface 16

B.2 UniGraspTransformer 16

B.3 ResDex 17

C Diagnostic and Target Construction **17**

C.1 Diagnostic proposals and admission 17

C.2 Measured failure conditions 18

C.3 Measurement and Editing Tools 19

C.4 Target Generation and Bridge Curriculum 20

D Additional Experimental Details **20**

D.1 UniGraspTransformer training configuration 20

D.2 ResDex training configuration 20

D.3 Evaluation sets and statistical conventions 21

E Additional Results **21**

E.1 Full multi-loop results 21

E.2 Bridge outcomes 22

E.3 Repair-data and Bridge comparison 22

E.4 Measured factors and curriculum paths 23

F Failure Cases and Limitations **24**

A Detailed Algorithm

A.1 Closed-loop execution

Algorithm 1 specifies the execution order omitted from the main-text overview. In particular, it identifies the point at which each evaluation manifest is frozen, the condition that triggers Bridge, and the different state changes following Gate acceptance and rejection.

Algorithm 1: DexPolicyEvo closed-loop evolution

Input: Checkpoint π_0, memory $\mathcal{H}_0$, policy adapter $\mathcal{A}_\pi$, asset tools $\mathcal{U}_\pi^{\text{asset}}$, pre-existing evaluation manifests, and prespecified loop budgets B_k and Target-design attempt limits R_k.
Output: Final policy π_K and memory $\mathcal{H}_K$.

for $k = 0, \dots, K-1$ **do**
  Fix the evaluation protocol, seed, criteria, budgets, pre-existing evaluation suites, and Gate;
  $\mathcal{P}_k \leftarrow \textsc{Diagnose}(\pi_k, \mathcal{H}_k, B_k)$;
  $\mathcal{C}_k^{\text{diag}} \leftarrow G_\pi(\mathcal{P}_k; \mathcal{U}_\pi^{\text{asset}})$;
  $\mathcal{D}_k^{\text{diag}} \leftarrow \textsc{ValidateAndAdmit}(\mathcal{C}_k^{\text{diag}})$;
  Freeze admitted Diagnostic identities and scales;
  $\mathcal{O}_k \leftarrow \textsc{Evaluate}(\pi_k, \mathcal{D}_k^{\text{diag}})$;
  $\mathcal{F}_k \leftarrow \textsc{Analyze}(\mathcal{D}_k^{\text{diag}}, \mathcal{O}_k)$;
  $\mathcal{E}_k \leftarrow \emptyset$;
  accepted $\leftarrow$ false;
  $j \leftarrow 0$;
  while $j < R_k$ ***and*** $\neg$accepted **do**
    $j \leftarrow j + 1$;
    $\mathcal{C}_{k,j}^{\text{tar}} \leftarrow \textsc{GenerateMeasureEdit}(\mathcal{F}_k, \mathcal{E}_k)$;
    $\mathcal{D}_{k,j}^{\text{tar}} \leftarrow \textsc{ValidateAndAdmit}(\mathcal{C}_{k,j}^{\text{tar}})$;
    Freeze admitted original-Target identities and scales for attempt j;
    $\mathcal{T}_{k,j} \leftarrow (\mathcal{D}_{k,j}^{\text{tar}}, \emptyset, \mathcal{D}^{\text{ret}})$;
    foreach $z \in \mathcal{D}_{k,j}^{\text{tar}}$ **do**
      Attempt native training on original Target z under $\mathcal{T}_{k,j}$;
      if *the fixed progress criterion is unmet* **then**
        Construct measured $\mathcal{D}_{k,j}^{\text{bridge}}(z)$;
        Train through valid Bridge levels, returning to the original z;
        Add valid levels to $\mathcal{T}_{k,j}$;
    $\widetilde{\pi}_{k,j} \leftarrow \mathcal{A}_\pi(\pi_k, \mathcal{T}_{k,j})$;
    $\mathcal{V}_{k,j} \leftarrow \textsc{Verify}(\pi_k, \widetilde{\pi}_{k,j})$;
    Record $(\mathcal{T}_{k,j}, \mathcal{V}_{k,j})$ in the attempt log;
    if $\mathcal{V}_{k,j}$ *passes the frozen Gate* **then**
      $\pi_{k+1} \leftarrow \widetilde{\pi}_{k,j}$;
      accepted $\leftarrow$ true;
      Update future Replay eligibility;
    else
      $\mathcal{E}_k \leftarrow \mathcal{E}_k \cup \{\text{permitted repair feedback from attempt } j\}$;
  if $\neg$accepted **then**
    $\pi_{k+1} \leftarrow \pi_k$;
  $\mathcal{H}_{k+1} \leftarrow \mathcal{H}_k \cup \{(\mathcal{P}_k, \mathcal{F}_k, \text{all Target attempts, Gate decisions, and final status})\}$;
return $\pi_K, \mathcal{H}_K$;

The failure rule and evaluation membership are immutable after the indicated freeze points; rejected candidates remain in the evidence log. Experimental Δ_{Target} and $\Delta_{\text{Diag.}}$ compare the incoming and candidate checkpoints within the displayed loop.

A.2 Agent artifacts and tool contracts

The executable Failure Report stores $(c, h, \phi_h, e_h, \mathcal{R}_{c,h}^{\text{fail}}, \rho_{c,h})$, including measured feature units, the LOW direction and boundary, evidence IDs and tool versions. The Repair Program $\mathcal{T}_k$ specifies admitted object codes, lineage, constraints, per-object scale, edit/measurement traces, native training

Table 3: Typed contracts of the six fixed roles. Tool failure creates an auditable invalid or retried item, not an unrecorded replacement.

Role	Input	Output	Tools	Failure path
Diagnostic Designer	π_k, $\mathcal{H}_k$, policy profile and budget	$\mathcal{P}_k$, admitted $\mathcal{D}_k^{\mathrm{diag}}$ manifest	generation, preprocessing, asset checks	reject invalid candidates before suite freeze
Policy Evaluator	checkpoint and frozen suite	$\mathcal{O}_k$, object-level success and keyframes	native simulator/evaluator	retry unchanged manifest or record an evaluation failure
Failure Analyst	$\mathcal{O}_k$, meshes, contrasts and tool library	$\mathcal{F}_k$ with factors, ϕ_h, e_h, $\mathcal{R}_{c,h}^{\mathrm{fail}}$, $\rho_{c,h}$	Measure, statistical checks, tool registration	retain counterevidence and unsupported hypotheses
Repair Designer	$\mathcal{F}_k$, category and budget	$\mathcal{T}_k$, Target/Bridge manifests and edit trace	u_{gen}, u_{check}, e_h, simulator validation	retain rejected/invalid attempts; stop on bounded retries
Policy Trainer	π_k, $\mathcal{T}_k$, $\mathcal{A}_\pi$, budget	native training records and $\widetilde{\pi}_{k+1}$	expert or residual trainer, trajectory collection, unified update	preserve π_k if training fails
Policy Verifier	incoming/candidate checkpoints and frozen suites	$\mathcal{V}_k$, Gate decision and $\mathcal{H}_{k+1}$	native evaluator, metric aggregation	reject if a required test fails or is missing

schedule, optional Bridge levels, Replay membership and stop conditions. This program is an explicit input to the Policy Trainer; it is not merely prose accompanying the asset files. Newly generated evaluators or editors are registered only after interface and black-box checks.

Table 3 summarizes the role-specific inputs, outputs, tool interfaces, and failure paths.

Agent decision scope and fixed controls. We further clarify the autonomy boundaries of DexPolicyEvo by specifying the decisions delegated to the agents and the controls fixed in advance. The agents determine the semantic and constructive content of the self-improvement process: the Diagnostic Designer proposes scenarios, object families, shape specifications, and diagnostic coverage; the Failure Analyst formulates candidate failure hypotheses and selects or synthesizes measurable factors and associated editors; and the Repair Designer constructs Target objects and, when needed, Bridge variants that progressively approach difficult Targets.

In contrast, the task definition, asset-generation interface, geometric and simulator-validity checks, policy-native training interface, training and generation budgets, evaluation protocol, success criteria, and candidate acceptance Gate are fixed in advance. Diagnostic assets are generated and validated before their identities, scales, checkpoint, and rollout protocol are frozen for evaluation. Generated Target objects must satisfy the corresponding remeasured geometric constraints and validity checks. The Policy Trainer therefore executes the Repair Program through the fixed native learning procedure of the underlying policy, while the Policy Verifier applies a prespecified quantitative Gate rather than making an open-ended acceptance decision.

B Policy and Task Adapters

B.1 Shared task and interface

Both adapters address single-object grasping with a Shadow Hand in Isaac Gym. The Repair Program $\mathcal{T}_k$ provides the identities, scales, optional Bridge ordering and Replay eligibility; the adapter maps these inputs into the policy's native object representation and training procedure. It leaves the policy observation, action space, architecture and native reward schedule unchanged. An object enters training or evaluation only after its geometry, collision model, initial pose and policy-required auxiliary inputs refer to the same identity and scale. The incoming and candidate policies are then evaluated on the same frozen object and rollout manifests. The two adapters below follow the same object-interface, update and curriculum contract.

B.2 UniGraspTransformer

Object interface. Each admitted object is converted into a simulator asset and a complete-object point cloud. The latter is encoded as part of the state-based grasp policy's observation; Target and Bridge variants retain their own measured geometry and scale rather than sharing a nominal category representation.

Native update. The update first trains object-specific experts with the policy's native PPO procedure. Experts that master an original Target provide successful trajectories for the subsequent unified-policy update. That update combines original-Target trajectories with eligible Replay trajectories and starts from the incoming unified policy. Thus the candidate submitted to the Gate is the updated unified policy, not a collection of object experts.

Bridge and output. If direct expert training does not meet the fixed progress criterion, the same expert is trained through easier, measured Bridge variants and then retested on the original Target. A Bridge variant can improve the expert's training reachability, but does not count as mastery of the original Target or become a Target trajectory by itself. The adapter records the expert attempts, Bridge transitions, original-Target trajectory provenance and resulting unified-policy checkpoint.

B.3 ResDex

Object interface. Each admitted object is converted into a scale-matched collision asset, point-cloud feature and grasp prior. The prior specifies a reference hand configuration and object supporting pose used by the native state-based task. For an edited Target or Bridge variant, these components must be regenerated or validated against the edited geometry at its actual scale; a mesh that loads without a consistent prior is not a valid ResDex input.

Native update. ResDex keeps its base grasp experts frozen and trains a residual actor–critic with PPO. Given state s, its action has the native form

$$a(s) = \sum_{j=1}^{J} \alpha_j(s) a_j(s) + \delta a(s), \qquad \boldsymbol{\alpha}(s) = \mathrm{softmax}(\mathbf{w}(s)), \tag{2}$$

where a_j is a frozen expert action and the residual policy predicts both the mixture weights and action correction. Its native training uses a grasp-proposal-guided stage followed by a stage emphasizing task completion. The candidate is the updated residual policy together with the unchanged base experts needed to execute it; no object-specific expert training or trajectory distillation is inserted by DexPolicyEvo.

Bridge and output. The same Repair Program is realized by the sequence of objects presented to residual-policy training. When direct Target learning is insufficient, validated Bridge variants precede the original Target; eligible Replay objects remain available for retention. Stage transitions continue the same residual policy while keeping the base experts fixed. The adapter records the object schedule, policy continuation and complete checkpoint dependencies. This is an adapter requirement; evidence that a particular run executed every transition belongs to its run-level provenance, not to the general interface definition.

Table 4 collects the concrete fields that differ between the two policy implementations.

C Diagnostic and Target Construction

C.1 Diagnostic proposals and admission

Diagnostic construction precedes failure analysis. This subsection specifies three reproducible decisions: what configurations are proposed, how they become policy-compatible assets, and which assets are admitted to the frozen evaluation set. It does not use the resulting grasp outcomes to select Diagnostic objects or enforce a failure interval that has yet to be found.

Proposal and coverage. In each loop, the Diagnostic Designer focuses on a specific object-use scenario spanning approximately 10–15 related semantic categories. It then allocates a category-conditioned proposal budget and records the intended coverage of object configurations. Given the limited reliability of direct text-to-3D generation for fine-grained object morphology, we use a text-to-image-to-3D pipeline. Semantic proposals are first materialized as reference images, batched into a 4×4 sheet when the image-generation interface requires it. Proposal identities and their planned scales are tracked before policy evaluation; a failed proposal is not replaced on the basis of its eventual grasp success.

Table 4: Policy-specific implementation fields. Entries marked as checkout defaults are not asserted as resolved settings of the reported ResDex runs.

Field	UniGraspTransformer	ResDex
Object-side inputs	Simulator mesh/collision asset and complete-object point cloud at assigned scale.	Collision asset, scale-matched point-cloud feature and grasp-prior sidecar.
Observation/action	400-dimensional state plus a 128-dimensional encoding of 1,024 object points; 24 hand actions.	`full_state` state-based task; frozen base-expert actions plus learned residual and mixture weights.
Native update	Object-expert RL, successful original-Target trajectories, then unified-policy update from the incoming policy.	PPO update of the residual actor–critic while base experts remain frozen; no trajectory distillation.
Bridge continuation	A passed expert checkpoint warm-starts the next geometric level and original Target.	The same residual-policy parameters continue across levels; base experts remain fixed.
Training environments	1,000 parallel environments for native expert training.	1,000 parallel environments for native expert training.
Runtime parameters	200-step episodes; assigned scale tokens $\{0.06, 0.08, 0.10, 0.12, 0.15\}$.	200-step episodes; assigned scale tokens $\{0.06, 0.08, 0.10, 0.12, 0.15\}$.
Candidate output	Original-Target training evidence and candidate unified-policy checkpoint.	Candidate residual checkpoint together with its frozen-expert dependencies.

Asset realization. Reference images are reconstructed into 3D meshes using TRELLIS. The reconstruction then passes through mesh repair, stable-pose estimation, collision preparation and point-cloud generation. The adapter in Appendix B produces the additional representation required by each policy. The resulting record binds the source proposal, mesh version, scale, pose and policy-native asset, so that a measured or evaluated object can be traced back to the geometry actually loaded by the simulator.

Admission and freeze. An object enters $\mathcal{D}_k^{\mathrm{diag}}$ only if its geometry and collision asset are valid and the policy-native representation loads in the simulator at the assigned scale and pose. Construction and loading failures retain their failure status but contribute no grasp rollouts. The admitted identity and scale manifest, incoming checkpoint and rollout protocol are fixed before the Policy Evaluator produces $\mathcal{O}_k$; subsequent outcomes cannot change membership in that evaluated set. Run-specific category and identity counts are reported with the corresponding manifests in Appendix E.

C.2 Measured failure conditions

Performance contrasts. Rollouts are aggregated by source-object identity within diagnostic group c. For groups with at least six identities, the bottom and top thirds of object success rates define $\mathcal{D}_{k,c}^{\mathrm{low}}$ and $\mathcal{D}_{k,c}^{\mathrm{high}}$; a within-group contrast requires a mean gap of at least 20 percentage points. Otherwise, better-performing groups may provide category-level references, without implying a matched contrast.

Feature selection and LOW interval. Each hypothesis is implemented as a deterministic scalar measurement $\phi_h : \mathcal{Z} \to \mathbb{R}$. The analyst checks normalized mean shift, standardized effect size, range overlap, and pairwise ordering consistency between LOW/HIGH measurements, then fits a one-dimensional boundary to measured Diagnostic values. The lower-performing side defines the LOW interval $\mathcal{R}_{c,h}^{\mathrm{fail}}$. The evidence record $\rho_{c,h}$ retains the feature definition, units, boundary, support statistics and counterexamples; multiple supported factors may be retained.

Editability and Target use. An editor $e_h(z, \delta)$ is bound when it can change ϕ_h while preserving a valid object of the same category. Edited assets are remeasured at their assigned scale. A primary factor may guide editing, but Target geometry follows the main-text OR rule over supported LOW intervals:

$$A_{\mathrm{geom}}(z) = \mathbb{I}\big[\exists h : \phi_h(z) \in \mathcal{R}_{c,h}^{\mathrm{fail}}\big] . \tag{3}$$

Category and physical validity are checked separately; an editor's requested value cannot substitute for the remeasured $\phi_h(z)$.

C.3 Measurement and Editing Tools

The unified reference implementation pairs each geometric measurement with an editor and a post-edit validity check. Registration makes a tool available; evidence from Diagnostic outcomes determines whether its factor is used in a particular loop.

Measurement convention. Asset preprocessing tests candidate upright axes in the simulator and rotates the selected axis onto canonical z before exporting the policy mesh. Measurements use this aligned mesh at its assigned policy scale. When normalization is requested, the mesh is centered at its bounding-box center, divided by its longest extent and then scaled to the policy size; a mesh already in physical units is not normalized again. Each measurement records the mesh version and scale so that a LOW threshold is applied in the same units in which it was defined.

Registered tools. Table 5 lists the single-object mesh Measure–Edit pairs in the unified reference registry. The descriptions specify what each function measures and which coordinates its paired editor changes; they do not imply that every requested value is reachable on every mesh, or that every registered factor was selected in every loop. These descriptors measure shape, not a unique physical cause of grasp failure. For ranked bounding-box extents, $d_1 \leq d_2 \leq d_3$; the reported upright height is measured after simulator-guided alignment at policy scale.

Table 5: Single-object mesh Measure–Edit tools in the unified reference implementation.

Measure function	Measured quantity	Paired editor / action
`upright_z_max`	Upright height $H_z = z_{\max} - z_{\min}$ (m).	`set_upright_z_max` *Scale canonical z about the mesh center.*
`transverse_bbox_area_m2`	Transverse bbox area $d_1 d_2$ (m^2).	`set_transverse_bbox_area_m2` *Scale both shorter axes; preserve d_3 and their ratio.*
`cross_section_area_ratio`	Relative cross-section $d_1 d_2 / d_3^2$.	`set_cross_section_area_ratio` *Jointly scale the two shorter axes.*
`min_over_max`	Shortest-to-longest extent d_1/d_3.	`set_min_over_max` *Adjust non-longest extents toward the requested ratio.*
`mid_over_max`	Middle-to-longest extent d_2/d_3.	`set_mid_over_max` *Adjust the middle-ranked extent toward the requested ratio.*
`min_over_mid`	Shortest-to-middle extent d_1/d_2.	`set_min_over_mid` *Scale the shortest-ranked axis.*
`canonical_xy_aspect_ratio`	Longer/shorter canonical XY span.	`edit_canonical_xy_aspect_ratio` *Scale the shorter XY axis.*
`canonical_xy_span_over_z`	Longer XY span divided by Z span.	`edit_canonical_xy_span_over_z` *Jointly scale X and Y, preserving Z.*
`central_band_support_ratio`	Maximum transverse span in the central half of the dominant axis, divided by its full span.	`edit_central_band_support_ratio` *Scale transverse coordinates in that band.*
`lateral_asymmetry_ratio`	Largest normalized left–right span imbalance in XY.	`edit_lateral_asymmetry_ratio` *Extend vertices on the more prominent side.*
`longitudinal_balance_offset`	Dominant-axis vertex-mean offset from bbox center, divided by its span.	`edit_longitudinal_balance_offset` *Extend vertices on the offset side.*
`outer_edge_thickness_ratio`	Z span of the outer radial quartile, divided by maximum XY span.	`edit_outer_edge_thickness_ratio` *Scale local outer-edge Z coordinates.*
`planform_radial_variation`	XY vertex-radius standard deviation divided by its mean.	`edit_planform_radial_variation` *Scale the larger-span XY axis.*
`rim_basin_depth_ratio`	Outer–inner 90th-percentile top-height gap, normalized by maximum XY span.	`edit_rim_basin_depth_ratio` *Shift upper outer-rim vertices along Z.*

Edit and verification contract. An editor receives a mesh, requested feature value and seed, and returns a new mesh with an edit record. Its output is always remeasured: a requested value is not treated as an achieved value. Generated evaluators and editors undergo interface checks and black-box tests before use. Editor tests check deterministic execution, non-degenerate output and the expected direction of feature change. Target admission additionally requires the measured LOW condition, category semantics, valid collision geometry and successful simulator load. An invalid or ineffective edit remains in the attempt log rather than counting as a satisfying Target.

C.4 Target Generation and Bridge Curriculum

Target generation. For each original Target slot, the Repair Designer specifies a category, failure semantic cue, assigned primary factor and LOW interval from $\mathcal{F}_k$. A proposed source object passes through image-to-3D reconstruction, mesh and collision preparation, canonical upright alignment, scale assignment and simulator validation. The assigned Measure function is evaluated on the policy-scale mesh. If its value misses the designated interval, the paired editor is applied under the slot's attempt budget and the resulting mesh is measured again. An unsuccessful edit may lead to another candidate for the same slot; neither the requested edit value nor the semantic prompt is used as evidence that a numerical constraint was met. Construction retries are bounded, and unsuccessful attempts are discarded when the slot's attempt budget is exhausted.

Target admission. A candidate enters $\mathcal{D}_k^{\mathrm{tar}}$ only if its remeasured geometry satisfies A_{geom} (Appendix C.2) and it passes category, mesh/collision and simulator validation at the assigned scale. The Target record retains the assigned factor, scale, pre/post-edit measurements and validation outcomes. Its difficulty under the frozen incoming policy is reported separately from geometric admission.

Bridge curriculum. Each admitted Target first undergoes native training within a fixed budget. If it remains difficult, the Repair Designer uses the retained factor–editor pairs to derive easier variants from that Target, selecting a starting level by its *measured* feature value and training viability. Further edited levels approach the original Target in feature space while keeping its identity, category and task configuration fixed. The checkpoint from a passed level initializes the next; progression requires the same object-level success criterion at each stage. A chain is successful only when training returns to and masters the original Target.

D Additional Experimental Details

D.1 UniGraspTransformer training configuration

Expert training. UGT first trains an object-level expert on each original Target using its native PPO implementation. A direct attempt has a 5,000-iteration cap. The expert is evaluated on its Target; a success rate below 20% triggers the Bridge curriculum.

Bridge curriculum. For a Target that fails direct training, the curriculum proposes at most five Bridge levels, with up to four design attempts per level. Each trained level has a cumulative budget of 2,000 expert-training iterations. An evaluation after 1,000 iterations provides an early-stop check: if success remains zero, training at that level stops; otherwise it continues for at most another 1,000 iterations. A level advances only after meeting the same 20% object-level threshold, and the curriculum ultimately returns to the original Target.

Replay and unified-policy distillation. Successful Target-expert trajectories are combined with eligible historical Replay trajectories for the unified-policy update. Replay is drawn from a base-distribution bank and, in later loops, from previously accepted loops; its prescribed sampling share is 25% of the update batches. Distillation is capped at 20 epochs. Checkpoints at epochs 5, 10, and 20 are evaluated on the frozen Target, Diagnostic, and Validation sets; the checkpoint used for the Gate is selected by the prespecified repair-and-retention criterion.

D.2 ResDex training configuration

Residual-policy updates. All reported ResDex updates use the native state-based residual PPO backend. Throughout the evolution process, the four base experts remain frozen, and PPO updates only the residual actor–critic, including the mixture-weight head and the residual-action head. Accordingly, each candidate checkpoint is evaluated together with the same frozen base experts used during its training. We use an episode horizon of 200 steps. Object translations are fixed, while the yaw angle is reset from a continuous uniform distribution. All evolution runs reported in this work use the `no-static-prior` initialization path, preventing object-specific grasp priors from providing an additional recovery signal.

Training mixture and curriculum. At each residual-policy update, the admitted Target assets and the Replay pool receive equal sampling probability mass,Target:Replay = 1:1.The Replay pool is held fixed within each update and is expanded across evolution loops with Target data accumulated from previous loops. Given a fixed manifest and random seed, the object sampler is deterministic. When a Target requires a Bridge, the same residual-policy parameters are trained continuously through a validated sequence of geometric difficulty levels before transitioning to the original Target. Across these levels, only the designated geometric attribute used to construct the Bridge is varied; the frozen experts, reward function, simulator backend, object scale, and reset protocol remain unchanged. Bridge construction and use follow the rules below:

1. Within-category analysis is the sole primary path for constructing a Bridge.
2. Each Bridge curriculum varies exactly one explicit and interpretable geometric attribute.
3. If no successful comparison example exists within the same category, additional samples are first generated from the within-category neighborhood rather than directly relying on cross-category inference.
4. Before formal RL training, each generated Bridge undergoes a small-scale validation to verify that task difficulty increases approximately monotonically as the selected attribute moves from an easy anchor toward the Target.
5. If this validation fails, the Bridge is re-analyzed and reconstructed, and the corresponding Target does not proceed to formal RL training.
6. Bridge samples are strictly additive training data: they do not replace or reduce the prescribed numbers of Target or Replay samples. Thus, the base Target–Replay sampling budget is preserved when Bridge training is introduced.

PPO budget and candidate selection. Each candidate is trained for 5,000 PPO updates in 1,000 parallel environments, collecting eight simulator steps per update (40 million environment transitions per candidate). The PPO configuration uses clipping 0.2, five optimization epochs, four minibatches, $\gamma = 0.96$, $\lambda = 0.95$, an adaptive learning-rate schedule with target KL 0.016, and zero entropy coefficient. For each loop, the three learning rates $\{1, 3, 5\} \times 10^{-4}$ are predeclared; candidate selection is then made by the frozen Gate using the prescribed Target and retention evaluations, not by training return. Evaluation uses the same frozen object/scale manifests as the incoming checkpoint and is separate from candidate selection rollouts.

D.3 Evaluation sets and statistical conventions

Δ_{Target} and $\Delta_{\text{Diag.}}$ are percentage-point differences between the candidate and its own incoming checkpoint on the same method's current-loop Target and Diagnostic suites. Cum.-Frontier is the object-macro success over frozen Diagnostic suites accumulated across all loops. Shared Frontier is a common frozen cross-method pool; the Loop 2 comparison combines three method-specific target pools before deduplication and has 468 listed objects in the archived manifest. The challenge is the frozen set of 500 low-success objects selected by the initial policy from the first 1,500 uniformly preprocessed Objaverse objects. Seen-Cat. (141 objects) and Unseen-Cat. (100 objects) follow the base DexGraspNet benchmark partitions. Seen-Cat., Unseen-Cat., and Challenge are used only for reporting; none enters checkpoint selection or the candidate-acceptance Gate.

For the three evaluation seeds $s \in \{0, 1, 2\}$, rollouts are first averaged within object and seed; set success is then macro-averaged across objects. The main tables' "$\pm$" denotes the sample standard deviation of three seed-level set means.

E Additional Results

E.1 Full multi-loop results

The main table reports each loop's repair gain, frontier coverage, and fixed-benchmark performance. This section complements it with the absolute success rates underlying repair and object-level Challenge examples.

Diagnostic coverage. The three UGT loops cover distinct everyday-object scenes. Loop 1 examines 15 categories of drinkware, tableware, and tea utensils; Loop 2 examines 13 categories of desk and stationery objects; and Loop 3 examines 12 categories of small household and personal objects. ResDex uses its own policy-specific Diagnostic objects, not the UGT failure regions.

Absolute repair rates. Table 6 reports the incoming and reported-policy success rates on each loop's Target and Diagnostic objects. The adjacent Table 7 shows performance on previously diagnosed regions.

Table 6: Target/Diagnostic SR (%; before → after).

Policy	Stage	Target	Diag.
UGT	Loop 1	$40.36 \rightarrow 86.36$	$44.92 \rightarrow 59.12$
	Loop 2	$34.62 \rightarrow 70.02$	$46.86 \rightarrow 58.55$
	Loop 3	$38.83 \rightarrow 78.88$	$54.88 \rightarrow 59.30$
ResDex	Loop 1	$4.58 \rightarrow 89.12$	$5.85 \rightarrow 84.83$
	Loop 2	$30.67 \rightarrow 43.48$	$55.93 \rightarrow 63.62$
	Loop 3	$27.28 \rightarrow 34.51$	$47.99 \rightarrow 54.25$

Table 7: Diagnostic retention across previously diagnosed regions (%).

Policy	Stage	Diag. 1	Diag. 2	Diag. 3
UGT	Initial	44.92	40.46	35.91
	Loop 1	59.12	46.86	42.82
	Loop 2	61.50	58.55	54.88
	Loop 3	66.21	54.16	59.30
ResDex	Initial	5.85	2.21	2.27
	Loop 1	84.83	55.93	40.63
	Loop 2	87.26	63.62	47.99
	Loop 3	88.41	65.95	54.25

For both policies, the reported Target and Diagnostic rates increase within every loop. The retention table additionally shows how later checkpoints perform on earlier Diagnostic suites, rather than reporting only the current loop's gain.

Challenge grasp examples. We select three Challenge objects for each adjacent UGT and ResDex policy comparison (Initial–Loop 1, Loop 1–Loop 2, and Loop 2–Loop 3) and visualize their grasp outcomes in Figure 11 and Figure 12. For the same object and initial configuration, the incoming policy fails to lift the object, whereas the updated policy succeeds. These examples show how successive loops recover difficult grasps; the reported object-level success rates summarize 100 rollouts per policy rather than the single frames shown.

E.2 Bridge outcomes

Figure 7 summarizes Target-level training outcomes across the three evolution loops. For UGT, direct training succeeds on 54.4%, 30.1%, and 36.8% of Targets in Loops 1–3, while Bridge recovers an additional 40.6%, 48.7%, and 39.6%, respectively. These results show that Bridge consistently recovers a substantial fraction of Targets that are not mastered by direct training alone.

Figure 7: Target-level training outcomes (%). Direct and Bridge denote verified success before and after curriculum recovery, respectively.

Policy	Stage	n	Direct	Bridge	Remaining
UGT	Loop 1	180	54.4	40.6	5.0
	Loop 2	156	30.1	48.7	21.2
	Loop 3	144	36.8	39.6	23.6

E.3 Repair-data and Bridge comparison

Training-step allocation. For Class-Matched Random and Failure-Neighborhood, each Target object is allocated a training-step budget equal to the average cumulative training steps per original Target in our reduced-asset variant, DexPolicyEvo*, which uses approximately 75 original Targets in this experiment. The cumulative count includes direct training and the subsequent Bridge curriculum stages.

Figure 8 cross-evaluates the Loop 2 UGT policies on three method-specific Target pools and their common Shared Frontier. Because Class-Matched Random, Failure-Neighborhood, and DexPolicyEvo construct different Target identities, performance on a method's own Target pool alone does

not provide a common basis for comparison. We therefore report all cross-pool results together with the Shared Frontier.

DexPolicyEvo achieves the highest Shared Frontier success rate of 54.13%, compared with 41.32% for Class-Matched Random and 39.11% for Failure-Neighborhood. It also obtains the highest success on the Random pool (51.85%) and the factor-guided pool (70.22%). Failure-Neighborhood performs best on its own pool (41.62% versus 40.33% for DexPolicyEvo), showing that the advantage is not uniform across every method-specific Target set.

The Bridge ablation further isolates the effect of curriculum recovery. On the factor-guided Target pool, removing Bridge reduces success from 70.22% to 43.03%, while the Shared Frontier score decreases from 54.13% to 39.47%. Under the reduced-data setting, DexPolicyEvo* reaches 62.25% on the factor-guided pool and 50.00% on the Shared Frontier. Together, these results indicate that the observed gains depend on both failure-factor-guided repair-data construction and the ability to recover difficult Targets through Bridge, rather than on performance on the method's own Target pool alone.

E.4 Measured factors and curriculum paths

Figure 9 shows four traceable UGT identities from Loops 1–3 for which the Easy Anchor, Bridge and original Target meshes, measured factors and native-training outcomes are all archived. The rows show how the edited geometry approaches the original Target; the numbers beneath each mesh are the measured factor and the success rate of the corresponding stage-specific expert. The red annotation gives the original Target's direct-training rate. These are illustrative complete paths, not a post-hoc estimate of curriculum yield; the full Target-slot denominators are reported in Figure 7.

Figure 8: Cross-evaluation of Loop 2 UGT policies on three method-specific Target pools and their shared union. Results are object-macro success rates (%) averaged over seeds 0–2.

Policy	Random	Neighborhood	Ours	Shared Frontier
Base	37.30	28.79	34.46	33.52
Class-Matched Random	49.80	36.73	37.43	41.32
Failure-Neighborhood	40.22	41.62	35.48	39.11
w/o Bridge	41.85	33.53	43.03	39.47
DexPolicyEvo*	49.27	38.48	62.25	50.00
DexPolicyEvo	51.85	40.33	70.22	54.13

Factors shown in Figure 10. For ranked bounding-box extents $d_1 \le d_2 \le d_3$, Pen uses $\texttt{transverse_bbox_area_m2} = d_1 d_2$ (m^2), with LOW defined by a measured value at most $0.242 \times 10^{-3}\,\mathrm{m}^2$. Scissors uses $\texttt{cross_section_area_ratio} = d_1 d_2 / d_3^2$ (unitless), with LOW at most 0.0763. These are two distinct geometric factors, not a shared physical scale. Under the frozen incoming policy, all 12 Pen and all five Scissors source identities have lower mean success inside their predicted LOW region than outside it. Their identity-macro LOW/non-LOW rates are 1.4/46.7% and 15.6/34.6%, respectively. The curves support within-identity behavioral relevance of the measured factors; they do not establish a universal causal failure mechanism. For each highlighted identity, the three rendered levels are the minimum, median and maximum measured values. The Pen example is an illustrative monotonic path (0.7%, 67.7%, 73.0%); it is not the basis for the all-identity contrast. The separate mesh gallery above illustrates native Bridge construction and does not use frozen-policy success rates.

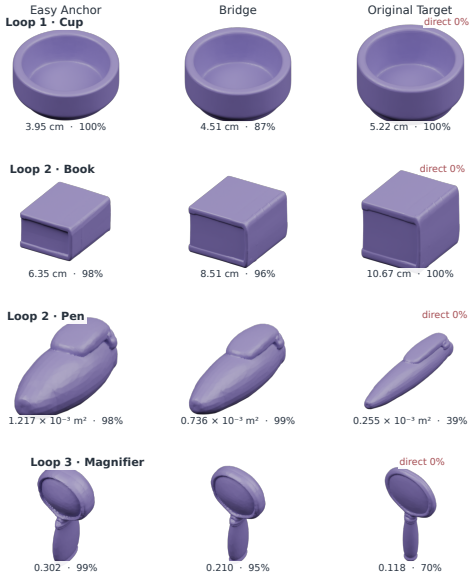

Figure 9: Four UGT curriculum paths. Each row shows an Easy Anchor, a Bridge, and the original Target.

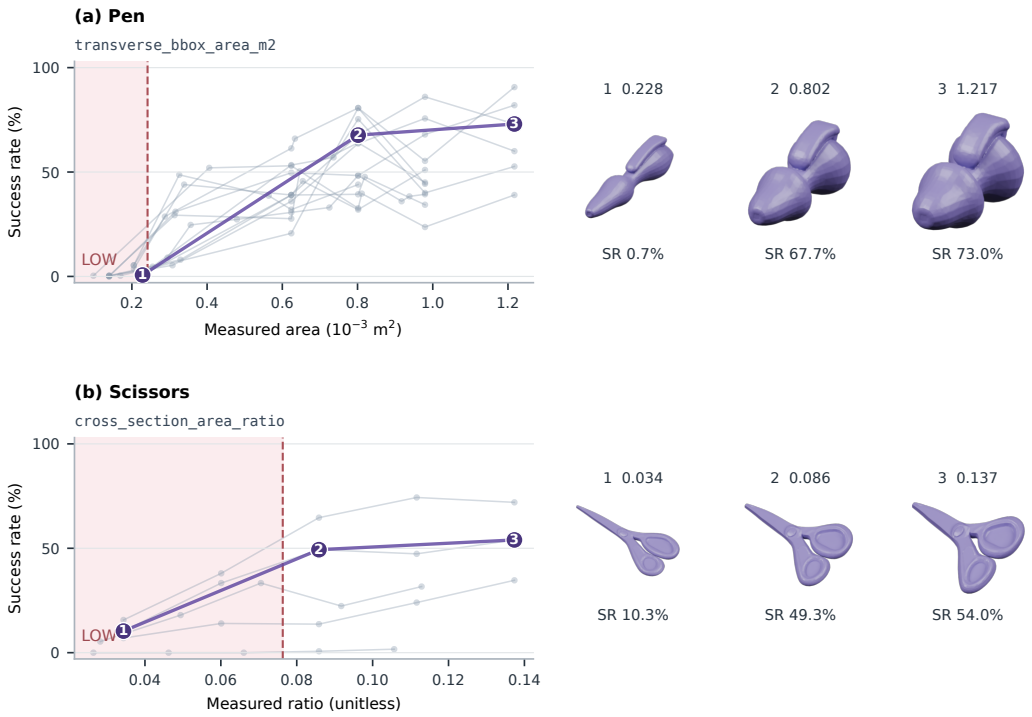

Figure 10: Measured factors, frozen-policy success and the corresponding geometries. The numbered purple points show the minimum, median and maximum measured levels of one traceable source identity per category; the same numbers identify Blender renders of the policy-scale meshes. Grey paths show all measured identities. LOW regions are shaded, and success rates are asset-level means over three evaluation seeds under the same frozen incoming UGT policy.

F Failure Cases and Limitations

Coverage of actionable factors. Our current framework primarily uses measurable and editable geometric factors to guide repair. Material properties, contact dynamics, and interactions among multiple factors remain less explored. Extending the measurement and editing tool library could support a broader range of failure modes.

Repair under finite budgets. Although Bridge facilitates learning on difficult Targets, some objects remain unresolved within the allocated training budget. More adaptive curriculum planning and budget allocation could further expand repair coverage.

Scope of evaluation. We evaluate DexPolicyEvo on two policies with distinct native training pipelines, focusing on simulated single-object dexterous grasping. Extending the framework to physical robots, different hand embodiments, and more complex manipulation tasks remains an important direction for future work.

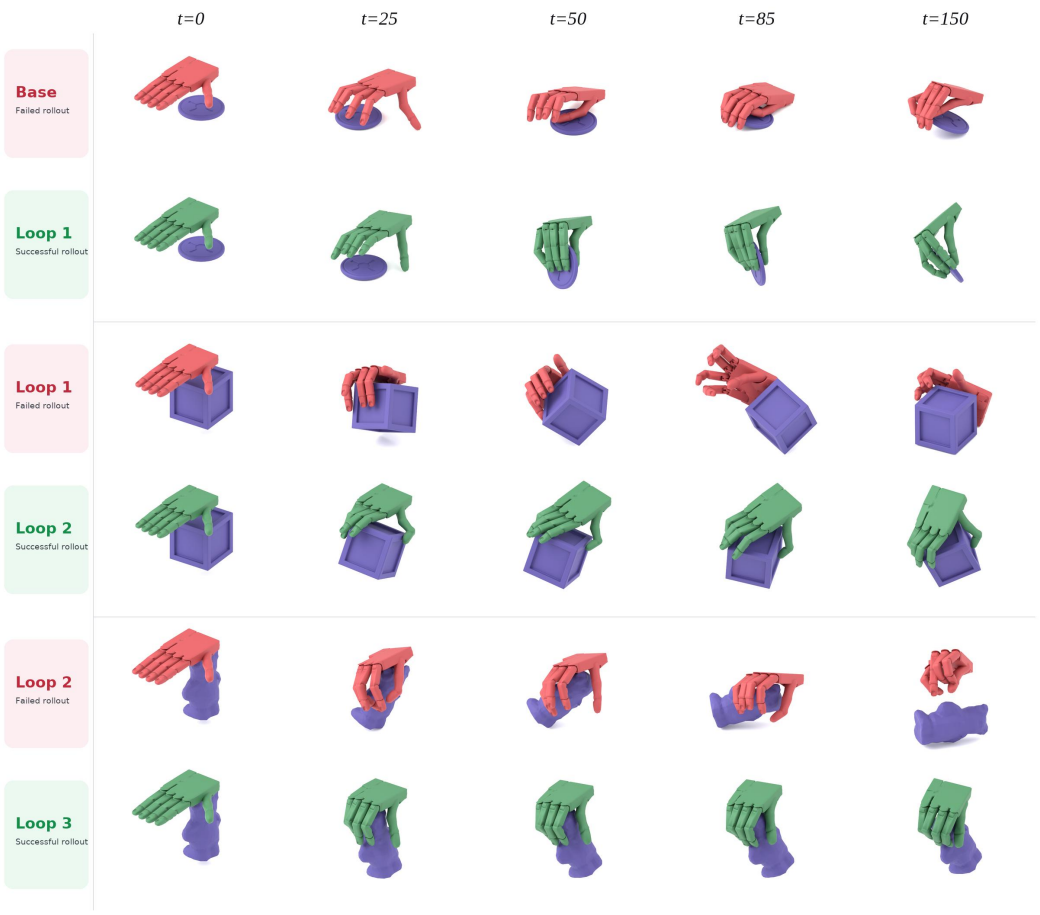

Figure 11: Representative Challenge grasp sequences across three repair loops. Each pair of rows compares adjacent UGT policies on the same object and initial simulator state. Columns show simulator steps $t \in \{0, 25, 50, 85, 150\}$; red and green hands denote the preceding (failed) and updated (successful) rollouts, respectively.

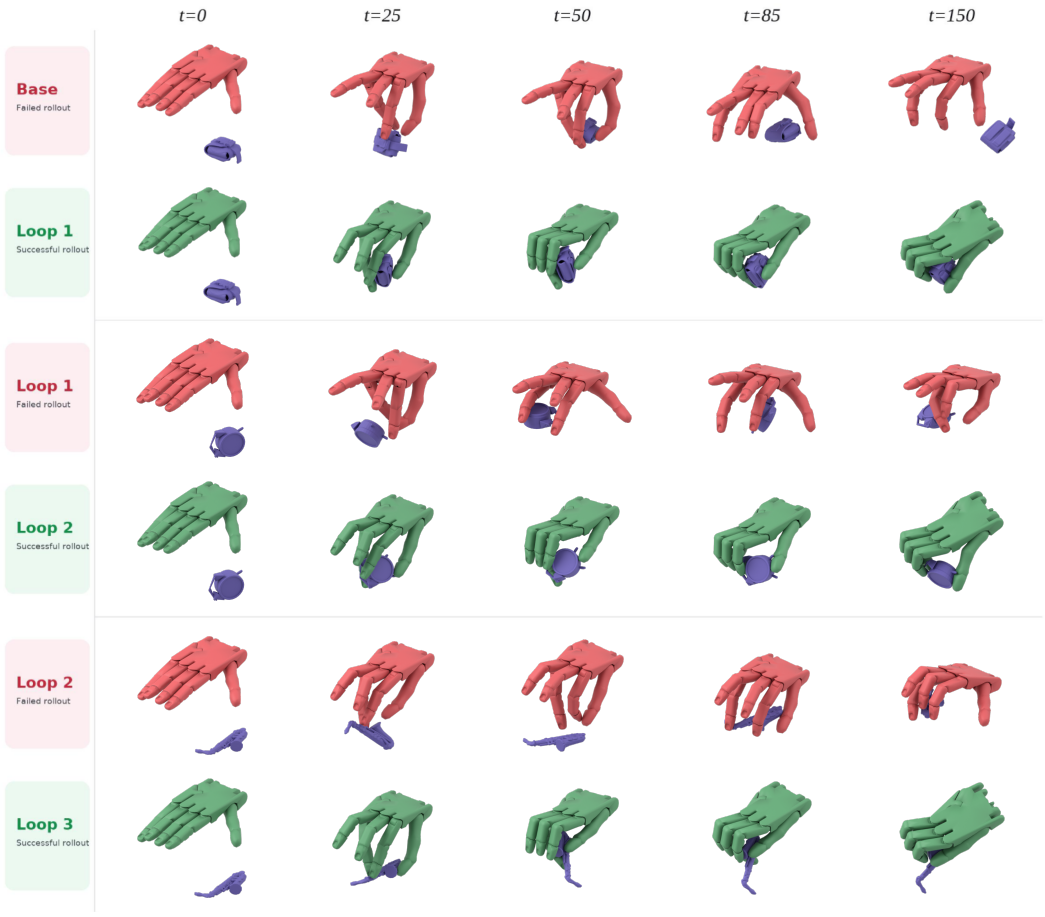

Figure 12: Representative Challenge grasp sequences across three repair loops. Each pair of rows compares adjacent ResDex policies on the same object and initial simulator state. Columns show simulator steps $t \in \{0, 25, 50, 85, 150\}$; red and green hands denote the preceding (failed) and updated (successful) rollouts, respectively.