When Search Becomes Memory: Accelerating Robot Design Discovery with Self-Evolving Skills

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly used as proposal generators for evolutionary robot design, yet most loops remain memoryless: simulator results shape the next population but are not preserved as reusable design knowledge. We present AUTO-ROBOTIST, a self-evolving LLM agent that distills morphology-search traces into an explicit natural-language skill library. Each skill stores a structural archetype, evidence-grounded positive and negative rules, and the evaluated designs that support them, making design memory inspectable rather than implicit in a population. During search, the agent retrieves skills to condition LLM edits of elite bodies while retaining a Genetic Algorithm (GA) mutation path for exploration; after evaluation, it updates the library through ADD, DIAGNOSE, and MERGE. Across seven EVOGYM tasks spanning locomotion, traversal, and object interaction, AUTO-ROBOTIST improves cold-start 5×5 search and transfers learned skills to 10×10 design spaces, where reference-conditioned transfer outperforms GA on every task. These results suggest that LLM agents can convert expensive physical evaluations into reusable, auditable design principles. Our code will be released upon acceptance.

1 Introduction

Robot-design search is usually judged by its final artifact: the best body found before the evaluation budget is exhausted. Yet every evaluated morphology is also an experiment about support, contact, actuation, stability, and failure. Classical morphology search uses expensive evaluations to discover robot bodies and controllers (Sims, 2023; Cheney et al., 2014, 2018; Mouret and Clune, 2015; Pugh et al., 2016), but the resulting experience is typically compressed into a final population or a single elite. A human designer would not discard that evidence: after enough trials, isolated blueprints become principles, such as which frames stabilize load-bearing limbs, which symmetries suppress rotation, and which plausible structures repeatedly collapse into poor gaits.

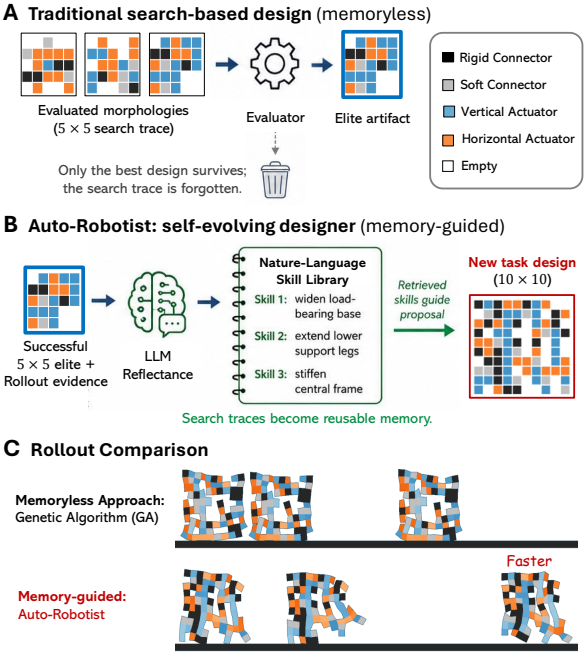

Figure 1: **Search traces as reusable design knowledge. (A)** Traditional search-based design is memoryless: many 5×5 morphologies are evaluated, but only the elite artifact is retained, leaving the evidence behind success and failure unused. **(B)** AUTO-ROBOTIST reflects over successful elites and rollout evidence to build a simulator-grounded natural-language skill library. Retrieved skills guide proposals for a new 10×10 design task, while each candidate remains validated by simulation. **(C)** In rollout comparison, memory-guided search reaches effective locomotion with fewer evaluations than a genetic algorithm baseline. The central distinction is that AUTO-ROBOTIST transfers *design knowledge*, not merely robot bodies: search becomes an accumulating, auditable source of reusable skills.

This paper asks whether a language agent can make the same transition: *from search experience to transferable robot-design skills*. As illustrated

in Figure 1, the key distinction is not whether an LLM can propose another morphology, but whether it can write down what costly evaluations reveal and reuse that knowledge when the design space changes. We study this question in voxel-based soft-robot morphology design in EvoGym (Bhatia et al., 2021), where an $n \times n$ robot is a grid of modular components and each fitness query requires Proximal Policy Optimization (PPO) (Schulman et al., 2017) controller training. A 10×10 robot already contains 100 component choices, yielding an unconstrained space of $5^{100} > 10^{69}$ bodies for 5 modules. In such spaces, forgetting why a walker failed or why a carrier succeeded discards exactly the evidence that should make future search easier.

Prior work improves robot design through evolutionary algorithms, quality diversity, learned encodings, Bayesian or reinforcement-learning policies, and morphology-aware architectures (Ha, 2019; Luck et al., 2020; Gupta et al., 2021; Yuan et al., 2021; Lu et al., 2025; Li et al., 2025). Recent LLM systems propose candidates, design rewards, or reflect on optimization outcomes (Lehman et al., 2023; Yang et al., 2024; Ma et al., 2024b; Song et al., 2025; Chen et al., 2026; Ringel et al., 2025; Fang et al., 2025), while NLP agents increasingly study long-term memory and reusable skills (Wang et al., 2023; Tan et al., 2025; Kang et al., 2025; Salama et al., 2025; Xu et al., 2026). Together, these lines of work raise a sharper question: *what should a language agent remember after costly physical evaluations?*

A useful design memory cannot be a raw log, a copied elite body, or unconstrained reflection. It must be **abstract** enough to transfer across resolutions and tasks, **grounded** in evaluated bodies and fitness changes, and **editable** as new evidence arrives. We therefore treat memory as a revisable design theory rather than a passive transcript.

We introduce Auto-Robotist, a self-evolving language agent with a persistent library of robot-design skills. Each skill separates an archetype, such as `portal-frame`, from positive and negative structural rules and the observations that support them. During search, the agent retrieves relevant skills, asks the LLM to edit elite parents toward those skills, validates the proposed bodies, and evaluates them in simulation. The library is then updated through Add, Diagnose, and Merge: new archetypes are created from unassigned evidence, observations are distilled into rules and failure modes, and redundant skills are consolidated. The LLM weights remain fixed; the agent's explicit domain memory evolves.

This framing changes the evaluation standard. A proposal generator should improve one run; a design agent should leave behind knowledge that improves the next. We evaluate Auto-Robotist in cold-start 5×5 search, where skills must emerge online, and in $5 \times 5 \rightarrow 10 \times 10$ transfer, where the learned library initializes a larger design problem. Across seven EvoGym tasks, Auto-Robotist matches or exceeds GA in cold-start search and, under reference-conditioned transfer, improves over GA on all seven 10×10 tasks, especially when contact and load-bearing structure are central.

Our contributions are threefold:

- We reframe LLM-based robot design as converting costly search experience into transferable design skills, rather than only generating high-fitness bodies.
- We introduce an evidence-grounded, self-evolving skill library that separates archetypes, rules, and observations, and couples skill-conditioned LLM proposals with simulator feedback.
- We evaluate on seven EvoGym tasks and analyze the learned memory, showing competitive cold-start search, consistent $5 \times 5 \rightarrow 10 \times 10$ transfer gains, and evidence that skills encode reusable structural relations.

2 Related Work

Morphology and embodiment co-design. Automated body-controller co-design has a long history, from evolving virtual creatures (Sims, 2023) to voxel-based soft robot evolution (Cheney et al., 2014, 2018) and quality-diversity search (Mouret and Clune, 2015; Pugh et al., 2016). EvoGym (Bhatia et al., 2021) made this setting a controlled benchmark by pairing discrete voxel bodies with reinforcement-learning controllers. Later work improved search with learned representations, Bayesian or RL-based design policies, and embodied pretraining (Ha, 2019; Luck et al., 2020; Gupta et al., 2021; Yuan et al., 2021). Recent systems further push this direction: coarse-to-fine and morphology-aware representations improve sample efficiency; BodyGen uses topology-aware self-attention and temporal credit assignment for efficient embodiment co-design (Lu et al.,

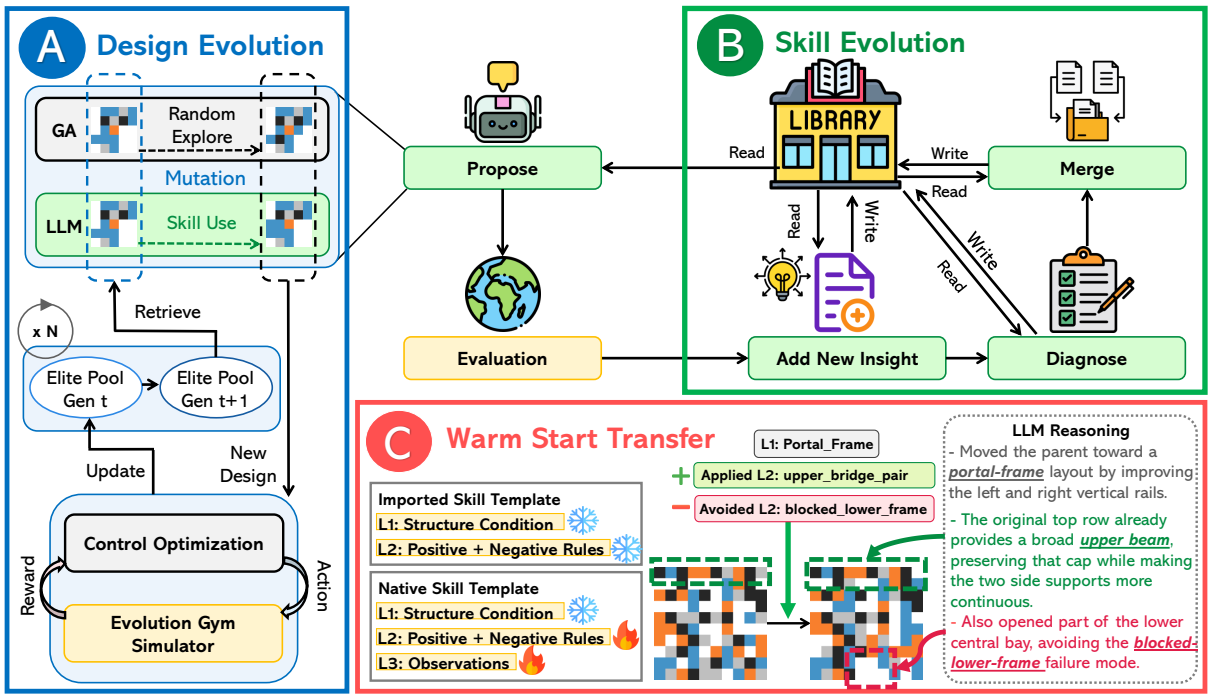

Figure 2: **Overview of AUTO-ROBOTIST. (A)** During design evolution, AUTO-ROBOTIST augments genetic mutation with skill-conditioned LLM proposals. Retrieved skills guide edits to elite parents, valid candidates are evaluated in EVOGYM with PPO-trained controllers, and the resulting fitness updates the elite pool. **(B)** During skill evolution, evaluation evidence is written back into a persistent library through ADD, DIAGNOSE, and MERGE. This converts search traces into revisable design knowledge while keeping the LLM parameters fixed. **(C)** During warm-start transfer, learned skills are imported into a new design space. Each skill separates a structural archetype (L_1), positive and negative design rules (L_2), and supporting observations (L_3), enabling a 5×5 `portal_frame` principle to guide a 10×10 proposal by preserving useful relations, extending support, and avoiding known failures.

2025); and freeform endoskeletal robot generation expands the design space beyond fully soft or fully rigid bodies by jointly optimizing external tissues and internal skeletons (Li et al., 2025). These methods make the optimizer or representation stronger. Our question is complementary: after a design run finishes, what reusable knowledge should remain?

LLMs for optimization, robotics, and robot design. LLMs can propose candidate programs (Lehman et al., 2023), optimize language objectives (Yang et al., 2024), synthesize rewards (Ma et al., 2024b), and refine sim-to-real reward functions (Ma et al., 2024a). For robot morphology, LASeR uses LLM-aided evolutionary search and diversity reflection (Song et al., 2025); other systems use foundation models to select, initialize, or co-optimize morphology from language and preferences (Chen et al., 2026; Ringel et al., 2025; Fang et al., 2025). These works show that LLMs are useful proposal operators. AUTO-ROBOTIST places the abstraction not only in the current prompt or candidate, but in a persistent skill library revised across generations and reused across runs.

Agent memory, reflection, and reusable skills. LLM agents increasingly use memory to extend beyond a single context window. Voyager stores executable skills for open-ended exploration (Wang et al., 2023); reflective-memory systems revise long-term memories based on future evidence (Tan et al., 2025); and recent memory architectures organize agent memories dynamically for retrieval and update (Kang et al., 2025; Salama et al., 2025; Xu et al., 2026). Our setting imposes a stricter grounding requirement: a robot-design memory is useful only if it changes future physical-search behavior and remains tied to evaluated bodies. This is why AUTO-ROBOTIST stores not only a natural-language rule but also its supporting observations and parent-relative gains.

3 Method

3.1 Problem Formulation

We study morphology search for modular soft robots in EVOGYM (Bhatia et al., 2021). The goal is to design a body that performs well after controller training on tasks such as walking, climbing,

balancing, carrying, or pushing.

Design space and validity. At scale n, a robot body is an $n \times n$ matrix $d \in \mathcal{V}^{n\times n}$ whose entries are one of five voxel types: *empty*, *rigid connector*, *soft connector*, *horizontal actuator*, or *vertical actuator*. Empty cells contain no material; connectors provide passive structure; actuators expand and contract along their axes. A feasible body $d \in \mathcal{D}_n \subset \mathcal{V}^{n\times n}$ must have a single connected non-empty component and at least one actuator.

Optimization objective. For task $\mathcal{T}$, each feasible body is paired with a PPO controller trained under a fixed protocol. The simulator then returns a scalar fitness $f(d, \mathcal{T})$, and morphology search solves

$$d^* = \arg\max_{d\in\mathcal{D}_n} f(d, \mathcal{T}), \tag{1}$$

under a budget B of body evaluations. One evaluation proposes a body, trains its controller, and measures task fitness.

Search trace as evidence. The combinatorial space is large, but the central inefficiency is epistemic: each expensive evaluation teaches something about stability, contact, actuation, symmetry, or failure, yet standard search keeps this knowledge only implicitly in the population. AUTO-ROBOTIST instead maintains a mutable skill library $\mathcal{S}_t$; the LLM parameters, PPO protocol, component alphabet, and validity constraints remain fixed.

3.2 From Search Experience to Skills

At generation t, AUTO-ROBOTIST maintains an evaluated population $\mathcal{P}_t$ and a skill library $\mathcal{S}_t$. Algorithm 1 alternates between *using* memory and *updating* memory. First, the agent retrieves task- and scale-relevant skills, samples one by evidence-weighted utility, and asks the LLM to edit elite parents toward the retrieved rule; a GA mutation path runs in parallel for coverage and invalid-output fallback. Then, evaluated children become observations, are routed to existing skills or an unassigned evidence pool, and are distilled into revised rules. The loop therefore compresses individual simulations into reusable design knowledge.

Lines 3–5 of Algorithm 1 show how the library steers exploration; Lines 8–9 show how simulator evidence rewrites the library. This separation prevents the LLM from becoming an oracle: its proposals must pass validity checks, its abstractions are trusted only after evaluation, and its memory is tied to observed fitness changes.

Algorithm 1 AUTO-ROBOTIST: converting search traces into transferable skills

1: Initialize evaluated population $\mathcal{P}_0$ and skill library $\mathcal{S}_0$.
2: **for** generation $t = 0, \ldots, T-1$ **do**
3: Retrieve task- and scale-relevant skills from $\mathcal{S}_t$.
4: Propose LLM children by editing elite parents toward retrieved skills.
5: Propose GA children by random mutation for coverage.
6: Reject invalid children; repair or replace them when possible.
7: Evaluate valid children with PPO to obtain fitness and parent-relative gain.
8: Route each observation to an existing skill or to the unassigned evidence pool.
9: Update $\mathcal{S}_{t+1}$ with ADD, DIAGNOSE, and MERGE.
10: **end for**
11: **return** best evaluated body and final skill library.

3.3 Skill Representation

A skill $s \in \mathcal{S}$ is a three-level record:

- L_1 is the **archetype**: a compact structural concept with an applicability condition, e.g., `portal-frame` for load-bearing arches around actuators.
- L_2 contains **rules**: positive rules that tend to improve fitness and negative rules that describe failure modes. Each rule stores a structural description, supporting evidence, and running mean parent-relative gain.
- L_3 contains **observations**: evaluated child and parent bodies, task, scale, fitness, gain, validity status, and proposal attribution.

This structure gives memory stability and accountability: L_1 is retrievable, L_2 specifies how to instantiate or avoid the concept, and L_3 preserves the audit trail. Rules use morphology-level relations rather than absolute coordinates; "support a central actuator column with a rigid side frame" can transfer, while "set cell $(4, 2)$ to rigid" cannot.

3.4 Skill-Conditioned Proposal

Let $g_{s,i}$ be the parent-relative fitness gain of the i-th observation attributed to skill s, and let n_s be the number of such observations. We score each skill by a smoothed usefulness estimate:

$$w_s = \frac{1 + \sum_{i=1}^{n_s} \mathrm{clip}(g_{s,i}/\delta_{\max}, 0, 1)}{2 + n_s}. \tag{2}$$

After normalizing these scores over retrieved skills, the agent samples a skill and prompts the LLM with the task, scale, parent body, archetype, relevant rules, and a strict output schema. Positive

evidence increases reuse probability; repeated non-improvement lets better-supported skills dominate. The child must satisfy legal-voxel, connectivity, and actuator-presence checks before evaluation. Local, unambiguous invalidities are repaired; otherwise, the slot is filled by GA mutation.

3.5 Library Maintenance

After each evaluated batch, the agent updates the library with three operations.

ADD. Observations that do not match existing skills enter an unassigned evidence pool. Once per generation, the LLM receives this evidence and the current archetypes, and may create a new L_1 only for a recurring structure not already represented. This prevents one-skill-per-anecdote growth.

DIAGNOSE. For each skill receiving new evidence, the agent updates its rules. If a child targeted a rule, its gain updates that rule's running mean:

$$\bar{g}_\ell \leftarrow \bar{g}_\ell + \frac{g_i - \bar{g}_\ell}{m_\ell}, \tag{3}$$

where m_ℓ is the updated support count. Rule-free observations are passed to the LLM, which decides whether they support, contradict, create, or leave undecided a rule. Negative rules are first-class entries because failure modes often transfer as reliably as successes.

MERGE. LLM-generated archetypes can be redundant. The agent clusters skills by L_1 identity and task family, merging only those that describe the same structural principle. Rules and observations from absorbed skills are preserved, consolidating evidence without erasing the audit trail.

3.6 Cross-Scale Transfer

To transfer from source scale to target scale, the target run imports the source skill library. Imported skills retain L_1 and L_2 but drop raw source observations, so they act as priors rather than target-scale evidence. During the first target generations, retrieval emphasizes imported skills; later observations enter the unassigned pool and can form native target-scale skills. The agent thus moves from prior-guided exploration to target-grounded refinement.

No voxel upsampling is required. A 5×5 rule such as "use a rigid side frame to stabilize vertical actuators" can guide 10×10 search because it names relations among functional parts. This is the core advantage of representing search experience as design skills rather than raw examples.

4 Experiments

A design-memory agent should pass three tests: improve the run that creates the memory, reuse that memory after the design space changes, and transfer physical relations rather than copy an elite body. We therefore ask: (i) can skills emerge from an empty library during 5×5 search; (ii) do 5×5 skills help in the larger 10×10 space; and (iii) are gains better explained by relational rules than by direct resizing or visual imitation? We answer these questions with fitness curves, endpoint summaries, rollouts, and library inspection.

4.1 Experimental Setup

Benchmark. We evaluate on EVOGYM (Bhatia et al., 2021), a 2D soft-robot benchmark where performance depends on support, contact, actuation, and task geometry. We use seven tasks spanning locomotion, traversal, balance, and object interaction: Walker, BridgeWalker, Balancer, Carrier, Climber, Jumper, and Pusher. Each task is tested in cold-start 5×5 search and $5\times5 \rightarrow 10\times10$ transfer.

Budget and controller training. At 5×5, Walker and BridgeWalker use 250 morphology evaluations; Balancer, Carrier, and Climber use 500; and Jumper and Pusher use 750. At 10×10, all tasks use 1,000 evaluations. Each body is evaluated by training a PPO controller (Schulman et al., 2017) for 5.12×10^5 environment steps, following EVOGYM (Bhatia et al., 2021). Full hyperparameters are in Appendix B.

Baselines and variants. The main baseline is the standard EVOGYM genetic algorithm (GA), run with the same budget, population size 25, and survivor-rate schedule. For transfer, *with reference* imports the 5×5 library and shows the source elite in the prompt; *skill-only* imports the library but removes the source-body exemplar, isolating rule transfer from visual imitation.

Metrics and reporting. Let $f_A(e)$ and $f_G(e)$ denote the best fitness found by AUTO-ROBOTIST and GA within the first e morphology evaluations, under total budget B. We report best-fitness-so-far curves and three scalar summaries. *Maximal Fitness* is the final value $f_A(B)$. *Convergence Speedup (S)* compares how quickly AUTO-ROBOTIST reaches the GA endpoint with how quickly GA reaches its own endpoint,

$$S = \frac{\min\{e : f_G(e) \geq f_G(B)\}}{\min\{e : f_A(e) \geq f_G(B)\}}, \tag{4}$$

Table 1: 5×5 **cold-start skill discovery.** Δ = Auto-Robotist − GA. S is convergence speedup relative to GA, and L is lead fraction over the budget.

Task	GA	AUTO-ROBOTIST	Δ	S	L
Walker	9.54	**9.56**	+0.02	1.30	0.72
BridgeWalker	3.65	**3.66**	+0.01	1.61	0.50
Balancer	0.13	**0.14**	+0.01	1.04	0.10
Carrier	6.40	**7.14**	+0.74	1.67	0.85
Climber	**0.57**	**0.57**	+0.00	1.05	0.86
Jumper	5.58	**6.52**	+0.94	1.64	0.94
Pusher	8.45	**9.87**	+1.42	1.95	0.91

with $S > 1$ indicating that AUTO-ROBOTIST reaches the GA endpoint faster than GA does. *Lead Fraction (L)* is the proportion of the search budget for which AUTO-ROBOTIST is strictly ahead,

$$L = \left|\{e : f_A(e) > f_G(e)\}\right|/B. \tag{5}$$

All curves plot cumulative morphology evaluations, so methods that consume the budget at different rates are compared on equal footing. We treat the results as descriptive comparisons under a fixed evaluation protocol.

4.2 Implementation Details

For AUTO-ROBOTIST, `gpt-5.5` is used for PROPOSE, ADD, DIAGNOSE, and MERGE, with temperature 1.0. Each generation evaluates 25 designs: 15 skill-conditioned Path A proposals and 10 GA-style Path B proposals. Thus, the agent biases search toward evidence-supported hypotheses without removing mutation-based exploration. Candidate bodies are checked for legal voxel types, connectivity, and actuator presence before simulation; invalid or unrecoverable LLM outputs are replaced by GA mutations.

4.3 Cold-Start Discovery: Can Skills Emerge from Search?

Cold-start search tests whether useful memory can be built during the same run that uses it. As shown in Table 1 and the top row of Figure 4, AUTO-ROBOTIST matches or exceeds GA on all seven tasks, with 1.47× average convergence speedup and a 70% average lead fraction. The strongest endpoint gains appear on Carrier, Jumper, and Pusher, where success depends on persistent contact, load paths, and coordinated impulse rather than merely adding actuation. Climber ties GA at the endpoint but leads for 86% of the budget. Thus, the library primarily improves the search trajectory, especially early exploration, while GA remains a strong optimizer for local search and exploration.

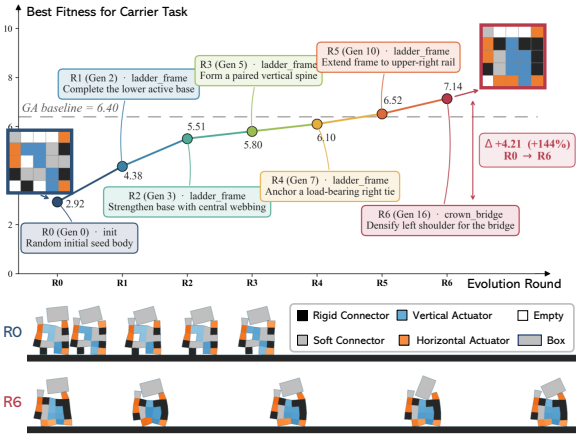

Figure 3: **Design evolution trace of AUTO-ROBOTIST on Carrier.** Each point marks the best body after an evidence-guided library update, annotated with the skill or rule that motivated the next design edit. AUTO-ROBOTIST transforms a weak random seed into a load-bearing carrier by completing the active base, adding central webbing, forming a vertical spine, and densifying bridge support. The bottom rollouts show that the final body better preserves payload support over time.

Design evolution trace. Figure 3 shows how memory is formed during a single Carrier run. Starting from a random seed body, the agent repeatedly writes evaluation evidence into named structural hypotheses, such as completing a lower active base, strengthening central webbing, forming a paired vertical spine, and densifying a left shoulder for bridge-like support. The final body improves from 2.92 to 7.14, surpassing the GA endpoint of 6.40. This trace makes the cold-start result concrete: the agent is not only sampling new bodies, but converting evaluated edits into reusable rules about payload contact and load-bearing structure.

4.4 Cross-Scale Transfer: Do Skills Survive a Larger Design Space?

The bottom row of Figure 4 and Table 2 show that 5×5 skills remain useful at 10×10. Reference-conditioned AUTO-ROBOTIST beats GA on all seven tasks, improves endpoint fitness by +1.55 on average, and maintains a lead fraction of at least 90% on six tasks. The largest gain is Carrier (+5.71), where success requires moving while preserving payload support; Walker and BridgeWalker also benefit, indicating that frames, legs, and compliant contact pads survive scale change. Pusher is the only speedup exception ($S = 0.97$): AUTO-ROBOTIST finishes better but crosses the GA endpoint slightly later, suggesting a quality-over-earliness tradeoff rather than failed transfer.

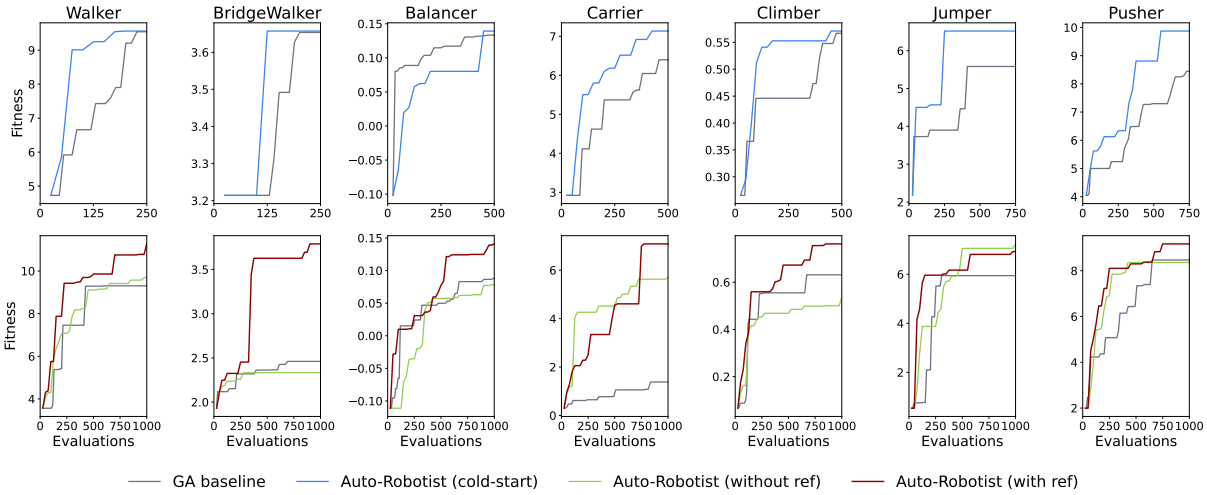

Figure 4: **Best fitness vs. morphology evaluations. Top**: 5×5 cold-start search, where AUTO-ROBOTIST learns from empty memory. **Bottom**: 5×5 → 10×10 transfer, where the learned library is reused in a larger space. The green curve removes the source-body exemplar, isolating skill transfer from visual imitation.

Table 2: 10×10 **cross-scale transfer.** AUTO-ROBOTIST vs the Genetic Algorithm (GA) baseline at 10×10. *Upsampling* columns evaluate the 5×5 elite after upsampling to 10×10 via 2×2 voxel tiling; *Full search* columns use a 1,000-eval 10×10 budget. *w/ ref* and *w/o ref* indicate whether a 5×5 reference design is included in the proposal prompt. Δ = AUTO-ROBOTIST w/ ref − GA; S and L are computed for AUTO-ROBOTIST *w/ ref* against the GA.

Task	*Upsampling* 5×5 *to* 10×10		*Full search* at 10×10			*w/ ref vs GA*		
	GA	AUTO-ROBOTIST	GA	AUTO-ROBOTIST w/o ref	AUTO-ROBOTIST w/ ref	Δ	S	L
Walker	4.67	4.20	9.31	9.72	**11.27**	+1.96	2.98	0.95
BridgeWalker	1.24	0.42	2.46	2.33	**3.79**	+1.33	1.97	0.93
Balancer	-1.44	-0.14	0.09	0.08	**0.14**	+0.05	1.81	0.72
Carrier	3.41	0.16	1.38	5.69	**7.09**	+5.71	8.36	0.95
Climber	0.34	0.29	0.63	0.54	**0.76**	+0.13	1.49	0.92
Jumper	1.55	2.71	5.94	**7.24**	6.93	+0.99	2.05	0.93
Pusher	5.85	0.76	8.47	8.36	**9.17**	+0.70	0.97	0.90

Rollout evidence. Figure 5 explains the aggregate gains behaviorally. GA elites often contain enough material and actuation to move, but lose the task-critical relation: contact breaks, support bends, or energy is spent on internal deformation. AUTO-ROBOTIST more often preserves the functional motif–a stable walking frame, a supported carrier body, or a pusher that keeps object contact. The learned memory therefore changes which physical invariants search protects, not only the final fitness score.

Why not just upsample the source robot morphology? Table 2 and Figure 6 give a negative control. Deterministic 2×2 tiling preserves source pixels but often destroys source function, especially for Carrier and Pusher, where scaling changes contact, leverage, and load distribution. Full 10×10 search is necessary; AUTO-ROBOTIST helps because it transfers morphological relations–support around actuators, compliant contact backed by structure, balanced limb placement–that target-scale search can re-instantiate.

Skill memory versus design copying. Figure 6 also separates rule transfer from exemplar copying. Skill-only transfer removes the source-body exemplar but retains the library; it still beats GA on Walker, Carrier, and Jumper and remains competitive on Pusher. Thus, the library stores functional information beyond the appearance of a 5×5 elite. BridgeWalker, Balancer, and Climber benefit most when the reference body is also supplied, indicating that some abstract rules need a geometric anchor. The best transfer combines both: skills specify *what relation should hold*, while the reference suggests *where to begin instantiating it*.

4.5 Anatomy of a Learned Skill Library

Figure 7 inspects the Walker 5×5 library. The memory stabilizes at five L_1 archetypes–lower-bridge, portal-frame, twin-spine-frame, side-braced-spine, and tapered-shell–with 12 positive and 17 negative L_2 rules. Sampling weights remain distributed

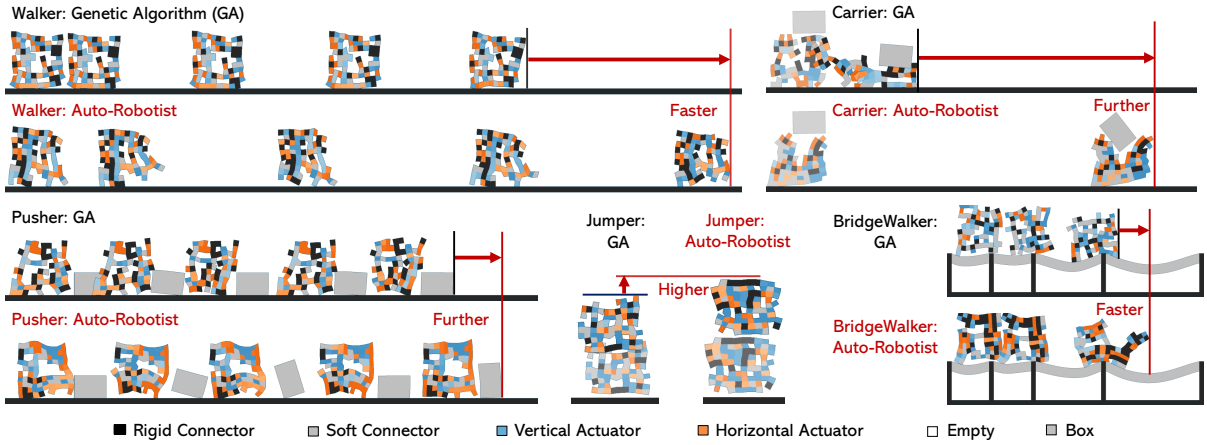

Figure 5: **Representative best** 10×10 **transfer rollouts.** AUTO-ROBOTIST more often preserves support and contact through time, yielding faster movement, higher jumps, or farther object displacement than GA.

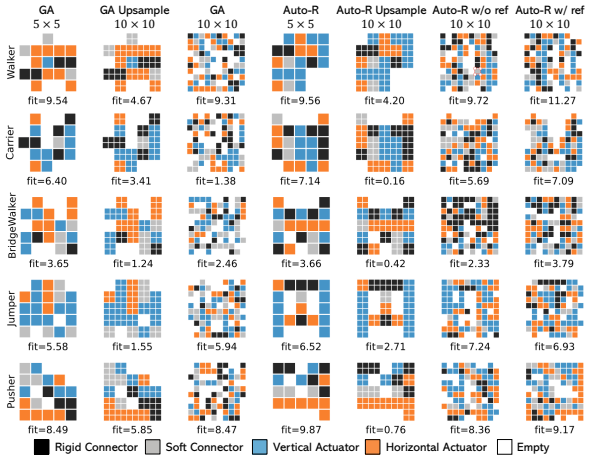

Figure 6: **Elite robot designs across** 5×5 **and** 10×10**.** Direct upsampling preserves geometry but often loses function; skill-guided search reconfigures source motifs into viable target-scale morphologies.

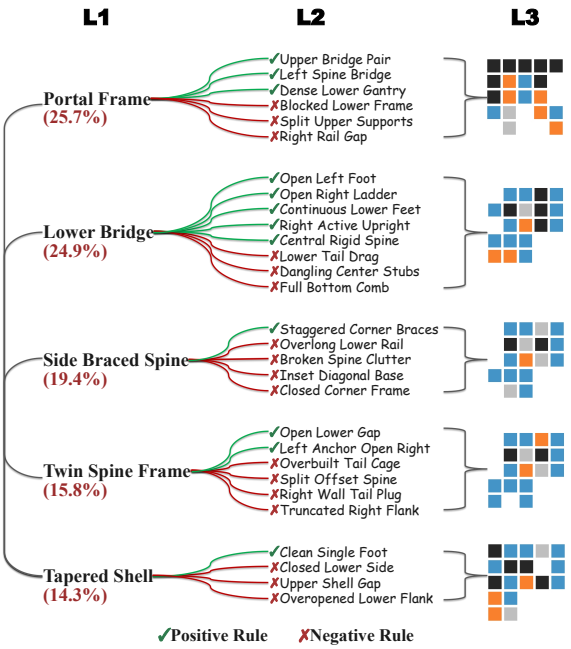

Figure 7: **A learned Walker skill library.** Evaluations are distilled into L_1 archetypes, L_2 positive/negative rules, and L_3 supporting observations, exposing both reusable structures and grounded failure modes.

across all five skills, suggesting that MERGE reduces redundancy without collapsing the library into one template.

The library's value is abstraction with evidence. Portal-frame and lower-bridge skills encode support paths that can be re-instantiated at another scale; negative rules preserve recurring traps such as brittle bottom connections, overlong rails, and closed-shell gaps. Failure evidence becomes a constraint on future proposals. Transfer is therefore not coordinate copying, but skill-guided re-search using compact hypotheses about load paths, contact surfaces, and actuator placement.

5 Conclusion

We introduced AUTO-ROBOTIST, a self-evolving language agent that turns robot morphology search from disposable trial-and-error into transferable design knowledge. Rather than retaining only the best body, the agent distills evaluated designs into an explicit skill library of archetypes, evidence-grounded rules, and supporting observations. Across EVO-GYM tasks, these skills emerge during cold-start 5×5 search and improve 10×10 transfer, especially when contact, support, and load-bearing structure matter. More broadly, AUTO-ROBOTIST suggests a path for improving LLM agents in expensive scientific and engineering domains: not only generating better proposals, but learning what to remember.

6 Limitations

Our experiments study 2D voxel morphology search in EvoGym under fixed-budget PPO evaluation. This controlled setting lets us isolate whether search traces can be distilled into transferable design skills, but it does not by itself establish performance for 3D robots, hardware deployment, or environments with unmodeled dynamics. The current skill library is also grounded in simulator feedback from a fixed component alphabet, controller-training protocol, and validity checker; transferring the approach to other morphologies or physical platforms will require new safety constraints, uncertainty-aware skill scoring, and sim-to-real validation. The main practical risk is over-interpreting simulation-derived skills as hardware-ready designs. We therefore view AutoRobotist as a research framework for auditable design-memory construction, not an autonomous hardware-deployment pipeline. Future work should extend the framework to 3D and hardware-facing morphologies, open-source LLM backbones, explicit rule-conflict resolution, calibrated uncertainty estimates, and stress tests for unsafe or brittle designs. These limitations do not change the central thesis: expensive physical evaluations should leave grounded abstractions that future agents can reuse, inspect, and audit.

References

Ferran Alet, Tomás Lozano-Pérez, and Leslie P Kaelbling. 2018. Modular meta-learning. In *Conference on robot learning*, pages 856–868. PMLR.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. 2021. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, 34:2201–2214.

Changhe Chen, Xiaohao Xu, Xiangdong Wang, Jiaqi Wang, and Xiaonan Huang. 2026. Natural selection via foundation models for soft robot evolution. In *2026 IEEE 9th International Conference on Soft Robotics (RoboSoft)*, pages 422–429.

Nick Cheney, Josh Bongard, Vytas SunSpiral, and Hod Lipson. 2018. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143).

Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. 2014. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061.

Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. 2023. Dreamcoder: growing generalizable, interpretable knowledge with wake–sleep bayesian program learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 381(2251).

Jiawei Fang, Yuxuan Sun, Chengtian Ma, Qiuyu Lu, and Lining Yao. 2025. Robomore: Llm-based robot co-design via joint optimization of morphology and reward. *arXiv preprint arXiv:2506.00276*.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. 2021. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721.

David Ha. 2019. Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25972–25981.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O Stanley. 2023. Evolution through large models. In *Handbook of evolutionary machine learning*, pages 331–366. Springer.

Muhan Li, Lingji Kong, and Sam Kriegman. 2025. Generating freeform endoskeletal robots. In *International Conference on Learning Representations*, volume 2025, pages 41952–41976.

Haofei Lu, Zhe Wu, Junliang Xing, Jianshu Li, Ruoyu Li, Zhe Li, and Yuanchun Shi. 2025. Bodygen: Advancing towards efficient embodiment co-design. *arXiv preprint arXiv:2503.00533*.

Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. 2020. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR.

Jason Ma, William Liang, Hung-Ju Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. 2024a. Dreureka: Language model guided sim-to-real transfer. RSS.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, and 1 others. 2024b. Eureka: Human-level reward design via coding large language models. In *International conference on learning Representations*, volume 2024, pages 26516–26560.

Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. 2020. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40.

Ryan P Ringel, Zachary S Charlick, Jiaxun Liu, Boxi Xia, and Boyuan Chen. 2025. Text2robot: Evolutionary robot design from text descriptions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5789–5797. IEEE.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. 2025. Meminsight: Autonomous memory augmentation for llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33124–33140.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Karl Sims. 2023. Evolving virtual creatures. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 699–706.

Junru Song, Yang Yang, Huan Xiao, Wei Peng, Wen Yao, and Feifei Wang. 2025. Laser: Towards diversified and generalizable robot design with large language models. In *The Thirteenth International Conference on Learning Representations*.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, and 1 others. 2025. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pages 12028–12068.

Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris Kitani. 2021. Transform2act: Learning a transform-and-control policy for efficient agent design. *arXiv preprint arXiv:2110.03659*.

A Prompt Templates

This appendix lists the prompt templates used by AUTO-ROBOTIST. Curly-brace fields such as {task_desc} are runtime substitutions; all other text is sent to the LLM as is. Sections follow the loop steps of §3.

A.1 Shared Blocks

Voxel legend.

```
Voxel type integers (robot body):
  0 = EMPTY   (no robot voxel)
  1 = RIGID   (structural, cannot actuate)
  2 = SOFT    (deformable, passive)
  3 = H_ACT   (horizontal actuator --
  expands/contracts horizontally)
  4 = V_ACT   (vertical actuator --
  expands/contracts vertically)
  5 = FIXED   (environment geometry only; do
  not use in robot bodies)
```

Body requirements.

```
- body is a {grid_size} x {grid_size} integer
grid (rows top-to-bottom, columns
left-to-right)
- robot-body entries must be 0, 1, 2, 3, or 4
- the robot MUST be fully connected (all
non-empty voxels reachable via 4-connectivity,
no isolated groups)
- MUST contain at least one actuator (3=H_ACT
or 4=V_ACT)
- do not use 5=FIXED in robot bodies
```

A.2 Propose

Cold-start initialization.

```
You are an expert soft-robot morphology
designer for the EvoGym simulation platform.

Task: {task_desc}

Voxel grid legend:
{voxel_legend}

Generate exactly {n_designs} diverse robot
designs as a JSON object with this schema:
{
  "designs": [
    {
      "body": <{grid_size}x{grid_size} integer
      matrix>,
      "reasoning": "brief explanation of
      design choices"
    }
  ]
}

Requirements:
<BODY_REQUIREMENTS>
- Explore diverse structures -- vary leg count,
body shape, actuator placement, symmetry
- Every design must be structurally different
from the others
```

Skill-conditioned mutation.

```
You are an expert soft-robot morphology
designer for the EvoGym simulation platform.

Task: {task_desc}

{transfer_context_block}
Voxel grid legend:
{voxel_legend}

Here is the parent design to mutate
(fitness={parent_fitness:.3f}):
{parent_body}

Skill assignments for this proposal batch:
{skill_assignments_block}

{static_reference_block}
Previous mutation history on this exact
parent:
{history_block}

Your task:
Propose exactly {n_designs} new mutations of
this parent, one for each assigned slot above,
using a two-part process:

1. Direction from the assigned skill:
   Use the skill's L1 condition as the target
   structural archetype for that slot.
   If the skill has no L2 rules yet, still
   move the parent toward the L1 condition.

2. Tactics from L2 rules and exact-parent
history:
   Use L2 positive rules as helpful
   sub-patterns when they fit this parent.
   Avoid L2 negative rules when relevant.
   Use exact-parent history to avoid repeats
   and avoid edits that already failed on this
   parent.

Each history entry gives you raw evidence
only:
- the exact child_fitness achieved on this
parent
- the voxel_diff from parent to child
- the full child_body after that mutation

Use this evidence directly to judge which
edits seem promising, harmful, or ambiguous.

Hard constraints:
- You must produce new child bodies distinct
from all history entries listed above.
- For each slot, use the specific assigned
skill for that slot only.
- For each slot, also output
`intended_leaf_id`: the L2 leaf you are trying
to instantiate.
  IMPORTANT: this MUST be the leaf_id (e.g.
  "pos_0", "neg_1") shown in the skill's L2
  rules block, NOT the claim string. Look for
  "leaf_id=..." in the rules listing.
  If the assigned skill has no leaves yet,
  output null.
- If an assigned skill is not a natural fit for
this parent, propose the smallest edit that
still moves in that skill direction without
destroying the parent's core structure.
```

```
- Do not refuse to propose. An imperfect
mutation is still useful learning material.

Generate exactly {n_designs} mutated
variations of this parent as a JSON object
with this schema:
{
  "designs": [
    {
      "slot_index": 0,
      "body": <{grid_size}x{grid_size} integer
      matrix>,
      "reasoning": "what you changed from the
      parent and why",
      "based_on_skill": "skill_id string or
      null",
      "intended_leaf_id": "leaf_id string or
      null"
    }
  ]
}

Requirements:
<BODY_REQUIREMENTS>
- Each design should modify {mutation_range}
voxels from the parent -- keep what works,
change what could improve
- L1 chooses the mutation direction; L2 rules
and exact-parent history choose the concrete
edits
- History can veto repeated or clearly harmful
edits, but it should not replace the assigned
L1 direction
- Do not repeat any exact child body already
present in the history block
- Return exactly one design per slot_index
listed above
- Every mutation must be structurally
different from the parent and from the other
mutations
```

A.3 Attribute

```
You are classifying robot designs by their
main structural archetype.

Skill library (each skill is described by its
L1 archetype plus top positive sub-patterns):
{skills_block}

Designs to classify:
{designs_block}

For each design, determine which skill it
structurally matches.

Use L1 condition as the PRIMARY criterion:
does the design exhibit the main
load-bearing arrangement described by L1? Use
top_positive_leaves as supplementary
evidence to recognize concrete instances of
successful sub-patterns within
that archetype.

If the design does not clearly match any
skill's L1 archetype, return null.

Constraints:
- Use structural matching only -- do not
consider fitness or task performance.
- A design may match at most one skill (return
the best match).
- Prefer null over a weak match. Skills that
don't fit will not gain useful evidence.
- L1 dominates over L2: if a design clearly
matches one skill's L1 but somewhat
  resembles another skill's positive leaves,
  attribute by L1.

Return JSON:
{
  "assignments": [
    {"local_index": 0, "skill_id": "skill_id"
    or null, "reason": "short structural
    reason"}
  ]
}
```

A.4 Add

```
You are a robot design analyst. Your job is to
add useful L1 structural skill identities for
robot morphology search.

High-fitness designs (top 6 from this
generation, body grids + fitness):
{high_designs}

Low-fitness designs (bottom 6 from this
generation, body grids + fitness):
{low_designs}

Existing skills for this task (each with L1
condition + top 2 positive L2 leaves):
{existing_skills_block}

Look for at most ONE new L1 structural
archetype in this generation.
Start from the high-fitness designs. If
several share a simple main structure,
you may add it as a new skill. Use the
low-fitness designs only as a light contrast
signal: it is enough if the structure is
absent, broken, weaker, or less coherent
there.
Be willing to add a new L1 when the high
designs show a reusable structure that is not
an obvious duplicate of an existing L1
condition. Return no_add only when there is no
clear nameable structure in the high designs,
or the best candidate is almost the same
as an existing L1 condition.

L1 should describe the robot's main connected
load-bearing arrangement. Use simple
structure words such as frame, rail, column,
bridge, arch, tripod, fork, wedge, tail,
crawler, shell, or beam. Avoid both
performance goals and exact voxel-level
details.
The Add step creates only the L1 identity; L2
and L3 must stay empty at birth.

Return JSON:
{
  "decision": {
    "action": "add" or "no_add",
    "inspired_obs_ids": [<int>, ...],
    "skill": {
      "skill_id": "short_name (max 3 words,
      snake_case, no version suffix)",
```

```
      "task_family": ["{task_name}"],
     "condition": "same text as l1.condition",
      "l1": {
        "structure": "one coarse structure
        word",
        "condition": "10-25 words describing
        one concrete structural archetype"
      },
      "l2": {
        "positive": [],
        "negative": [],
        "next_leaf_id_counter": 0
      },
      "l3": {
        "observations": [],
        "next_obs_id_counter": 0
      }
    } or null,
    "reasoning": {
      "supporting_high_labels": [<int>, ...],
      "contrast_signal": "short explanation of
      the high-vs-low structural difference",
      "nearest_existing_skill": "skill_id
      string or null",
      "duplicate_risk": "none" or "low" or
      "high",
      "why_add_or_no_add": "short explanation"
    }
  }
}

Rules:
- This is a lightweight discovery step, not a
strict filter.
- Add when high-fitness designs reveal a
simple reusable L1 structure and
duplicate_risk is not high.
- Reject only generation-level summaries,
performance goals, local patches,
voxel-coordinate descriptions, and
near-duplicate L1 conditions.
- skill_id must be max 3 words in snake_case,
with no version suffix.
- If action is "add", inspired_obs_ids must
list the observations that best exemplify this
pattern; if no_add, set skill to null and
inspired_obs_ids to [].
- For any added skill, l2.positive,
l2.negative, and l3.observations MUST be empty
lists.
- reasoning is audit-only. It will not be
shown to later stages, so be explicit and
honest.
- Candidates have gain > 0 (computed as
child_fitness - parent_fitness). Only consider
these for new L1 archetypes.
- When evaluating duplicate_risk, compare
against both L1 conditions AND
top_positive_leaves of existing skills.
  Two skills with similar archetype but
  different positive sub-patterns may still be
  distinct.
- inspired_obs_ids must list the obs_ids from
the input that best exemplify this pattern
(use the obs_id values shown).
{low_skill_hint}
```

A.5 Diagnose

```
You are diagnosing L2 leaves for a robot
design skill.

Skill L1 condition: {l1_condition}

Existing L2 leaves (with current statistics):
{leaves_json}

Unassigned observations (need leaf decisions):
{unassigned_json}

Context observations (already assigned this
generation; for distribution awareness only):
{context_json}

Generation statistics:
gen_mean={gen_mean:.3f}, p25={gen_p25:.3f}
(context only, not primary criterion).

{cold_start_note}

For each unassigned observation, decide its
leaf assignment. Optionally propose new
standalone leaves
that capture cross-cutting patterns visible
across multiple obs.

Decision rules:
- Primary criterion: gain (= child_fitness -
parent_fitness). gen_mean / p25 are weak
context.
- Positive leaf creation (lenient): obs.gain >
0 AND reusable structural sub-pattern.
- Negative leaf creation (strict): obs.gain <<
0 (significantly negative) AND obvious failure
structure.
- Prefer "no_leaf" when there is no clear
sub-pattern; the observation will be
reconsidered in future generations
  (up to a hard cap of 3 attempts).
- standalone_new_leaves: only when >= 2 obs
share the same not-yet-captured sub-pattern.
- description_update applies only to
"match_existing" decisions.

Refinement granularity rules (apply to all
claim/description text including standalone
leaves):
- ALLOWED: relative regions ("lower-right
corner", "upper half", "leftmost column"),
  shape language ("U-shape opening upward",
  "tapered top"),
  counts and proportions ("4 H_ACT", "30%
  RIGID", "at least 2 anchors"),
  structural relations ("anchor connects to
  lower rail").
- FORBIDDEN: exact voxel coordinates ("voxel
at (5,4)"), numeric row/column indices
  ("row 0", "column 4"), full-body matrix
  templates.
- Prefer "approximately N" / "at least N" over
rigid "exactly N".

Return JSON:
{
  "leaf_assignments": [
    {"obs_id": <int>, "decision":
    "match_existing", "leaf_id": "<id>",
```

```
      "description_update": {"mode":
      "overwrite" | "append" | null, "text":
      "..." or null}},
    {"obs_id": <int>, "decision": "new_leaf",
    "polarity": "positive" | "negative",
      "claim": "snake_case_1_to_4_words",
      "description": "structural sentence"},
    {"obs_id": <int>, "decision": "no_leaf"}
  ],
  "standalone_new_leaves": [
    {"polarity": "positive" | "negative",
    "claim": "...", "description": "...",
      "supporting_obs_ids": [<int>, <int>,
      ...]}
  ]
}
```

A.6 Merge

```
You are checking a robot design skill library
for obvious duplicate L1 skill identities.

Skills in the library (L1 identity only):
{skills_full_content}

Return merge clusters only for obvious
duplicate or near-duplicate L1 skill
identities.
The main comparison target is L1 condition:
merge skills only when they describe the
same main structural archetype in different
words.

L1 structure is only a weak hint. Two skills
may both use "frame", "rail", or "column"
and still be different if their L1 conditions
describe different connected layouts.

Return JSON:
{
  "clusters": [
    {"group_label": "short_name (max 3 words)",
    "skill_ids": ["id1", "id2"], "reason":
    "why these are the same mechanism"}
  ]
}

Rules:
- group_label must be max 3 words,
descriptive, snake_case
- Return only multi-skill clusters that should
be merged; return [] if no obvious duplicates
exist
- Do not create single-skill groups
- Do not merge skills just because they share
the same performance goal, same task, or same
broad structure word
- If there is any meaningful doubt, keep the
skills separate
```

A.7 Transfer-Context Blocks

Same-grid transfer.

```
=== Transfer Context ===
The skill library below comes from a prior run
on the same task
({current_env}, {source_grid} grid, source
experiment "{source_exp}").
Treat the L1/L2 rules as validated patterns
from previous experimentation.
Note: avg_gain values reflect the prior run's
parent fitness distribution;
treat them as relative ranking signals, not
absolute predictions.
```

Cross-grid transfer.

```
=== Transfer Context ===
Current task: {current_env} ({current_grid}
voxel grid).
The skill library below comes from a prior run
on a related task with a
{source_grid} voxel grid (source experiment
"{source_exp}").

The skills' L1/L2 rules describe abstract
structural principles (e.g.
"vertical rails joined by horizontal
crossbeam") that should generalize
across grid sizes. The current {current_grid}
grid affords richer / more
redundant structures than {source_grid}.
```

Elite-design addendum (same-grid).

```
Reference designs (top-fitness exemplars from
the source run) are also
provided below. Use them as concrete examples
of what the L1/L2 rules
look like in practice.
```

Elite-design addendum (cross-grid).

```
Reference designs (top-fitness exemplars) are
also provided below as
concrete {source_grid} examples -- extract
structural patterns from them,
do NOT copy voxel-level arrangements directly.
```

B Experimental Details and Hyperparameters

B.1 Tasks and Environment IDs

Table 3: **EvoGym tasks and environment IDs.** We evaluate seven tasks at both 5×5 and 10×10 design scales.

Task	Objective / family	5×5 env.	10×10 env.
Walker	Flat-ground locomotion	`Walker-v0`	`Walker-v0-10x10`
BridgeWalker	Bridge traversal	`BridgeWalker-v0`	`BridgeWalker-v0-10x10`
Balancer	Balance on a beam	`Balancer-v0`	`Balancer-v0-10x10`
Carrier	Object carrying	`Carrier-v0`	`Carrier-v0-10x10`
Climber	Vertical traversal	`Climber-v0`	`Climber-v0-10x10`
Jumper	Coordinated jumping	`Jumper-v0`	`Jumper-v0-10x10`
Pusher	Object pushing	`Pusher-v0`	`Pusher-v0-10x10`

B.2 Agent, Search, and Controller Hyperparameters

Table 4: **Agent, search, and PPO controller hyperparameters.** Values are shared across tasks unless otherwise noted.

Hyperparameter	5×5	10×10
AUTO-ROBOTIST agent		
Designs per generation	25	25
Path A skill-conditioned slots	15	15
Path B GA slots	10	10
Elite Pool size k	5	5
LLM mutation range (voxels)	1–3	1–10
LLM backbone	`gpt-5.5`	
LLM temperature	1.0	
Skill-weight saturation cap $\delta_{\max}$	2.0	
Unassigned-pool re-attribution threshold	30	
Prior-only generations K	–	5
Static elite-inject pool size	–	5
Reference design from 5×5 elite	–	yes in w/ref variant
Evaluation budget		
Walker / BridgeWalker budget	250 evals	1,000 evals
Balancer / Carrier / Climber budget	500 evals	1,000 evals
Jumper / Pusher budget	750 evals	1,000 evals
PPO controller training		
Controller architecture	Multilayer perceptron	
Total timesteps per morphology	512,000	
Parallel envs n_{envs}	4	
Rollout steps n_{steps}	128	
Update epochs	4	
Batch size	128	
Learning rate (initial)	2.5×10^{-4}	
Learning rate schedule	linear decay to 0	
Clip range	0.1	
Discount γ	0.99	
GAE λ	0.95	
Entropy coefficient	0.01	
Value-loss coefficient	0.5	
Max gradient norm	0.5	
Log / eval interval (updates)	50	

B.3 Validity Checks

Every proposed body is checked before simulation. A valid body must contain only legal robot-body voxel values, include at least one actuator, and form a connected component after empty cells are ignored. The environment-reserved `FIXED` voxel type is not allowed inside the robot body. Invalid LLM outputs are repaired only when the edit is local and unambiguous; otherwise, the proposal slot is replaced by GA mutation. These checks ensure that all reported comparisons spend their simulation budget on feasible morphology evaluations rather than on malformed bodies.

C Additional Results and Qualitative Analysis

C.1 Representative Skill Types

The learned Walker skill library is visualized in the main paper (Figure 7). Here we summarize additional skill types that recur across tasks and help explain why the library transfers better than raw upsampling.

Load-bearing frame. A frame skill describes rigid or semi-rigid support around actuators so that actuation produces task-relevant motion rather than body collapse. This skill is useful for Walker, BridgeWalker, and Carrier because each task rewards controlled force transmission through the body.

Compliant contact pad. A contact skill places soft material near the interaction surface while preserving stiffer internal support. It is useful in Carrier and Pusher because the body must maintain contact with an object during motion rather than merely moving itself.

Actuator anchor. An actuator-anchor skill pairs active voxels with neighboring rigid or soft support. The rule prevents isolated actuation from producing local deformation, rotation, or internal cancellation without useful displacement.

C.2 Component Ablations on Pusher

To probe which mechanisms of §3 carry the gain, we disable one component of the library pipeline at a time and measure best fitness on Pusher under the same 5×5 cold-start protocol as Table 1. (A1) **No Diagnose** skips the DIAGNOSE call, so L_2 rules never form or update. (A2) **No Merge** skips the per-generation MERGE pass, letting L_1 duplicates accumulate. (A3) **Pure LLM** drops GA mutation so all 25 slots come from skill-conditioned LLM mutation. (A4) **No L_2+L_3** keeps the pipeline intact but strips L_2 rules and L_3 observations from the PROPOSE prompt.

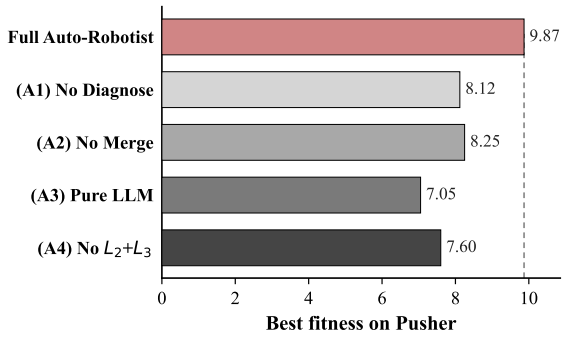

Figure 8: Best fitness on Pusher per ablation condition.

From Figure 8, (A3) at -28.6% indicates that GA mutation remains a useful source of structural exploration the skill-conditioned proposer alone fails to recover. (A4) at -23.0% indicates that L_2 rules sharpen the abstract L_1 archetypes into ac-

tionable positive/negative guidance, and L_3 observations ground both L_1 and L_2 in concrete voxel instances the LLM can interpret. (A1) at -17.8% and (A2) at -16.4% further confirm that library maintenance is load-bearing on this long-horizon manipulation task. All four mechanisms thus contribute materially to the gain on Pusher.

D Extended Discussion

D.1 What Makes a Design Skill Transferable?

The experiments suggest that transferable design skills sit between two extremes. A raw elite body is too concrete: it may work at one grid size but fail when naively resized. A vague reflection such as "be stable" is too abstract: it does not constrain a proposal enough to improve search. Useful skills name structural relations that remain meaningful under scale changes, such as a support frame around actuators, a compliant contact surface backed by rigid structure, or an actuator band anchored by passive material. These relations are the right level of abstraction because they can be instantiated differently while preserving function.

D.2 Why Evidence Grounding Matters

Natural language is expressive enough to describe physical design principles, but it can also produce plausible explanations unsupported by evidence. AUTO-ROBOTIST addresses this risk by treating a skill as an evidence-linked record rather than a free-form reflection. L_3 observations preserve the evaluated bodies, tasks, scales, and gains; L_2 rules summarize repeated positive or negative patterns; and the sampling weight in Equation 2 gives stronger influence to rules with consistent support. This design makes the library auditable: a user can inspect which evaluations caused a rule to matter, and future updates can revise or merge rules when new evidence contradicts them.

D.3 Boundary Conditions for Transfer

Transfer is strongest when source-scale skills describe relations that remain valid at the target scale. Carrier is the clearest example: load-bearing and contact-stabilizing motifs learned at 5×5 remain useful at 10×10. Some tasks additionally require geometric anchoring. BridgeWalker, Balancer, and Climber without a reference design illustrate this case: the library alone captures useful structural priors, but the reference-conditioned variant supplies topology and environment-specific anchoring that abstract rules do not fully determine.

These cases highlight two design pressures for future memory systems. First, useful rules may interfere when they prescribe incompatible edits, such as placing a rigid spine where another rule prefers a central actuator band. Second, a library must avoid crystallizing a concept before enough evidence distinguishes a general principle from a lucky design. AUTO-ROBOTIST addresses these pressures through evidence tracking and merging; richer uncertainty-aware retrieval and explicit rule-conflict resolution are natural next steps.

E Responsible NLP, Artifacts, and Reproducibility

This appendix summarizes the responsible-research considerations for AUTO-ROBOTIST and provides ARR checklist-ready responses. The work studies simulated robot morphology search, not human-subject data or user-facing deployment. The main responsibility issues are therefore artifact use, reproducibility, simulator-to-real limitations, and auditability of LLM-assisted design decisions.

E.1 Scope and Artifacts

Scope. AUTO-ROBOTIST is a research system for studying whether language agents can convert expensive morphology-search evaluations into reusable design knowledge. The method operates in EVOGYM and uses simulated robot morphologies, controller-training results, fitness values, LLM proposal traces, and learned skill-library records. It does not collect, annotate, infer, or release human-subject data. No personally identifying information, demographic attributes, private text, or user-generated natural-language corpus is used.

Scientific artifacts. The main scientific artifacts used in this work are EVOGYM environments, PPO controller training, the genetic-algorithm baseline, LLM prompts, generated voxel morphologies, simulation logs, and learned skill libraries. The paper cites EVOGYM and PPO in the main text, documents prompt templates in Appendix A, and reports task descriptions, validity constraints, evaluation budgets, metrics, and hyperparameters in Appendix B. The artifacts created by this work are intended for research and reproducibility: they should be used to study morphology search, language-agent memory, and simulator-grounded design rules, not as deployment-ready hardware designs.

Licenses and intended use. EvoGym is a public research benchmark for evolving soft robots. Our use is consistent with this intended purpose: we compare morphology-search algorithms in simulation and do not deploy generated designs as physical robots. Any released code, prompt files, robot-body grids, logs, or skill-library records will preserve attribution to the original benchmark and third-party packages. Released artifacts should include explicit license information and should be treated as research artifacts rather than safety-certified robot designs.

Computational experiments and budget. All experiments are computational. We report compute in platform-independent units because wall-clock time depends on parallelization and hardware. Each morphology evaluation trains a PPO controller for 5.12×10^5 environment steps. At the 5×5 scale, Walker and BridgeWalker use 250 morphology evaluations; Balancer, Carrier, and Climber use 500; and Jumper and Pusher use 750. At the 10×10 scale, all tasks use 1,000 morphology evaluations. Auto-Robotist evaluates 25 designs per generation, with 15 skill-conditioned LLM-proposal slots and 10 GA-mutation slots. The LLM is used as a fixed prompting component for Propose, Add, Diagnose, and Merge; its parameters are not trained or fine-tuned.

Statistical reporting. The paper reports best-fitness-so-far curves, endpoint fitness, convergence speedup, and lead fraction under a fixed evaluation protocol. These quantities are descriptive comparisons of search behavior under equal morphology-evaluation budgets. Unless otherwise stated, the reported values should not be interpreted as formal statistical-significance claims. Future work should add multi-seed confidence intervals and wall-clock/hardware-normalized cost analysis.

Risks and safety. The main risks are not privacy or fairness risks, because the work uses no personal or demographic data. The relevant responsibility risks concern robotics and automation. A language agent may discover morphologies that exploit simulator artifacts, overfit to benchmark-specific physics, or fail under real-world manufacturing tolerances. We therefore frame Auto-Robotist as a simulator-facing research prototype. Any physical deployment should require human engineering review, stress testing, uncertainty-aware validation, and safety constraints outside the training simulator.

Auditability. The proposed skill library is designed to make LLM-assisted design more auditable. Each skill stores an archetype, positive and negative rules, and supporting observations. Thus, a generated proposal can be traced to the retrieved skill, the rule that motivated the edit, and the simulation evidence behind that rule. This evidence link is important because natural language alone can produce plausible but unsupported explanations.

AI-assistant use. This work uses LLMs as part of the proposed method: the agent prompts a fixed LLM for proposal, skill addition, diagnosis, and merging. These prompts are documented in Appendix A. Separately, AI assistants were used for drafting, editing, formatting, and project-organization support. They are not authors, did not determine the scientific claims, and do not replace author responsibility. The authors remain responsible for the correctness of experiments, citations, figures, tables, claims, and final submission content.

E.2 ARR Responsible NLP Checklist Responses

A1. Limitations. Yes. The main paper includes a dedicated Limitations section. The claims are restricted to simulated 2D voxel morphology search in EvoGym under fixed evaluation budgets. Appendix E.1 further clarifies scope, artifact use, safety risks, and reproducibility boundaries.

A2. Potential risks. Yes. The main risks are simulator exploitation, brittle morphology heuristics, overfitting to benchmark-specific physics, and unsafe direct transfer from simulation to physical robots. These risks are discussed in Appendix E.1. Because the work uses no personal or demographic data, direct privacy and fairness risks are limited; the central responsibility issue is safe and auditable use of generated designs.

B. Use or creation of scientific artifacts. Yes. The study uses EvoGym environments, PPO controller training, GA baselines, LLM prompts, generated robot morphologies, simulated-evaluation logs, and learned skill-library records. It also creates experiment outputs such as curves, tables, prompt templates,

rollout visualizations, and skill-library summaries.

B1. Citation of artifact creators. Yes. The main paper cites EvoGym, PPO, and the relevant robot co-design, evolutionary-search, LLM-optimization, robotics, and agent-memory systems. Appendix A documents prompt templates, and Appendix B documents the simulation protocol and hyperparameters.

B2. License or terms for artifact use and distribution. Yes. Appendix E.1 discusses artifact use, attribution, and intended use. EvoGym and third-party packages are used as research artifacts for simulated morphology-search benchmarking. Any released code, prompt files, robot-body grids, logs, or skill-library records will include explicit license information and preserve attribution to EvoGym and other third-party dependencies.

B3. Consistency with intended use. Yes. EvoGym is intended as a benchmark for developing and comparing soft-robot co-design algorithms. Our use is consistent with that purpose: we evaluate morphology-search methods in simulation and do not deploy generated designs as physical robots. The artifacts created by this work are intended for research and reproducibility, not for direct commercial, physical, or safety-critical deployment.

B4. Personally identifying information or offensive content. Not applicable. The work does not use natural-language corpora, user-generated text, human-subject data, private data, or demographic information. The generated artifacts are voxel grids, numerical rewards, controller logs, prompt traces, and structured skill records. These artifacts do not identify individuals and do not contain offensive textual content by construction.

B5. Documentation of artifacts. Yes. The main text and Appendix B document task names, task families, grid sizes, morphology encoding, validity constraints, evaluation budgets, PPO settings, baselines, variants, and metrics. Appendix A documents prompt templates. Appendix C provides additional qualitative and quantitative results, including representative skill-library analyses.

B6. Statistics for data or artifacts. Yes, where applicable. The work does not use train/dev/test splits from a language dataset. Instead, it reports the relevant statistics for the simulator setting: seven tasks, two morphology scales, task-dependent 5×5 source budgets, a $1{,}000$-evaluation 10×10 target budget, 5.12×10^5 PPO environment steps per morphology, endpoint fitness, convergence speedup, and lead fraction.

C. Computational experiments. Yes. The experiments run fixed-budget morphology search in EvoGym with PPO controller training for each valid robot body.

C1. Model size, computational budget, and infrastructure. Partially. The paper reports platform-independent computational budgets: morphology-evaluation counts and PPO environment steps per morphology. The LLM is used as a fixed prompting component and is not trained or fine-tuned. If the exact LLM is a hosted model whose parameter count is not publicly disclosed, the parameter count cannot be reliably reported. The final code release should pin the model name/version, software environment, and hardware configuration.

C2. Experimental setup and hyperparameters. Yes. Section 4.1 defines the benchmark, tasks, budgets, baselines, variants, and metrics. Appendix B reports PPO hyperparameters, GA settings, LLM-proposal settings, and validity checks. We use a fixed protocol rather than task-specific hyperparameter tuning; the reported settings are shared across tasks, grid sizes, and search methods unless otherwise stated.

C3. Descriptive statistics. Yes. The paper reports best-fitness-so-far curves and scalar summaries: final maximal fitness, convergence speedup, and lead fraction. It explicitly treats results as descriptive comparisons under a fixed protocol and does not present them as formal statistical-significance claims. Where a result is based on one fixed run, that should remain clear in the text, tables, and captions.

C4. Package parameters. Yes. The paper cites EvoGym and PPO, reports the EvoGym tasks and controller settings, and specifies validity checks for robot bodies. If released code

modifies third-party packages or wrappers, the release should document those modifications and exact software versions.

D. Human subjects or annotators. No. The work uses no human annotators, crowdworkers, or human-subject experiments.

D1. Participant instructions. Not applicable. There were no human participants or annotators. Prompt templates for the LLM agent are provided in Appendix A for reproducibility, but these are not human-subject instructions.

D2. Recruitment and payment. Not applicable. No participants or annotators were recruited or paid.

D3. Consent. Not applicable. The study does not collect or curate human data.

D4. Ethics review board approval. Not applicable. The study does not involve human-subject data collection. If a future extension includes human design ratings, user studies, or human-subject annotations, the authors should seek appropriate ethics review before data collection.

E. AI assistants in research, coding, or writing. Yes. LLMs are used as part of the proposed method for proposal generation and skill-library maintenance. In addition, AI assistants were used for drafting, editing, formatting, and project-organization support. They are not authors, do not determine scientific claims, and do not replace author responsibility.

E1. Information about AI-assistant use. Yes. Appendix E.1 provides this disclosure. The authors remain responsible for the paper's content, experimental validity, citation correctness, figures, tables, and final submission.

E.3 Social Impact

AUTO-ROBOTIST is intended as a research system for studying how language agents can preserve useful knowledge from expensive scientific evaluations. Its potential benefits are methodological: making morphology search more sample-efficient, making design priors inspectable through evidence-linked skills, and reducing repeated simulation effort by reusing what earlier searches discovered.

The main risks arise if such agents are moved from simulation to physical design without safeguards. Automatically generated morphologies can exploit simulator artifacts, fail under real-world tolerances, or encode brittle design heuristics that appear reliable only under a narrow benchmark. We therefore frame the system as a simulator-facing research prototype, not as a direct hardware-deployment tool. Any physical use should require human engineering review, safety constraints, stress testing, uncertainty-aware validation, and validation outside the training simulator.

Because the work uses no personal or demographic data, direct privacy and fairness risks are limited. The more relevant responsibility concern is auditability. The proposed library stores not only skill names but also rules, supporting observations, and failure evidence, so users can inspect why a design prior was retrieved rather than treating the LLM proposal as an opaque recommendation.

F Additional Related Work

Curriculum, transfer, and abstraction in design search. A second relevant thread studies how agents acquire reusable structure across tasks. Curriculum learning and automatic curriculum learning shape the order of experience to improve generalization (Bengio et al., 2009; Portelas et al., 2020), while environment-design methods seek training distributions that induce robust transfer (Dennis et al., 2020). Modular meta-learning and program-library induction show that reusable components can make future learning more efficient (Alet et al., 2018; Ellis et al., 2023). AUTO-ROBOTIST shares this motivation but operates in a different setting: it does not pretrain a model or learn a latent module. Instead, it writes simulator evidence into natural-language design rules that can be inspected, edited, and reused by an LLM agent.