UNDERSTANDING ROBOT CO-DESIGN: EXPLOITING THE ASYMMETRY BETWEEN THE DESIGNER AND CONTROLLER

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot co-design jointly optimizes morphology and control against the same final objective, but does that mean the two optimizers should behave alike? To answer this, we create a test environment and explore it through the lens of three new metrics: spread, exploration, and generalization. By varying these properties independently, we find that effective co-design requires opposite behavior from each optimizer: the controller requires broad generalization to new designs and suffers from wide sampling, whereas the designer barely benefits from generalization but must sample the design space broadly to find good designs. We show that the source of this asymmetry is due to the sequential structure of co-design in which the body is chosen before the controller acts, and that it can be explained through the Expected Value of Perfect Information. Our test environment also shows that broader morphology search can discover better designs but may simultaneously worsen the selection when the controller cannot evaluate unfamiliar bodies reliably. We observe optimization profiles in existing co-design methods, and combine our findings to construct SCOPER which is tailored to exploit the optimizer asymmetry by maintaining a broad morphological search space, and using representation learning to increase controller generalization. In our experiments, SCOPER demonstrates the value of these principles, achieving 50% higher task performance than prior co-design methods while halving runtime.

1 INTRODUCTION

Recent advances in robot learning have produced increasingly capable control policies, but a robot's capabilities depend on both its controller and its body. Co-design optimizes both: a designer proposes morphologies, while a controller learns to use them. Although both optimize the same final objective, they play different roles: the designer selects a morphology first, and the controller then acts conditioned on that choice. This sequential "designer $\rightarrow$ controller" structure raises a simple question: should the two optimizers behave in the same way?

The effects of this coupling in co-design are hard to interpret as a morphology's value is determined by the controller that may still be learning. A design may score poorly because it is unsuited to the task, or because the controller has not adapted to it. Final return cannot distinguish these cases. To deconstruct this nuance, we derive three metrics from the common structures in modern co-design methods, which apply to both optimizers: spread, how broadly an optimizer samples; exploration, how often an optimizer samples an uncertain point; and generalization, how well it handles unfamiliar samples. In a controlled test environment we vary these properties independently and document a clear asymmetry. Our data shows that the designer's performance rises significantly with spread but minimally in generalization, while the controller's performance does the inverse. Since the designer commits before the controller acts, uncertainty in the design is resolved in time for the controller to respond to it, whereas uncertainty in the action is resolved after all choices have already been made. The difference we see in reward originates from the expected value of perfect information, which we detail in Eq. 6 (Raiffa & Schlaifer, 1968; Birge & Louveaux, 2011).

We see the same behavior in robotic locomotion tasks. Evaluating NGE (Wang et al., 2018b), an Evolutionary Algorithm, we show that its generator achieves a broad population, but its controllers

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.

over-fit and prohibit proper design-space search. BodyGen (Lu et al., 2025), a set of PPO-trained transformers, generalizes across bodies but collapses onto a single design early in training. We build Structured Co-design optimization with Physical Embodiment Representations (SCOPER) to satisfy each criterion: its designer maintains broad samples, while its controller acts near-greedily and shares a single transformer with the designer, learning representations that carry across bodies. SCOPER reaches $\sim$50% higher return than the strongest prior method in under half the wall-clock time, and our ablations confirm that each component moves the metrics in the intended direction. See our website https://iclr-submission-2027.github.io/ for videos of our algorithm and an interactive demo of the test environment.

2 Related Work

Morphology–control co-design and cross-morphology transfer. Co-design has progressed from evolutionary optimization of morphology and control (Sims, 1994; Cheney et al., 2018), to methods that combine morphology search with reinforcement-learned control (Schaff et al., 2019; Luck et al., 2020; Schaff & Walter, 2022). However, learning control from scratch for every candidate is expensive. In parallel, graph-structured, modular, and Transformer-based policies have shown that control knowledge can transfer across diverse robot morphologies (Wang et al., 2018a; Pathak et al., 2019; Huang et al., 2020; Kurin et al., 2020; Gupta et al., 2021a; Xiong et al., 2023). Co-design methods have therefore increasingly exploited this transfer through inherited or shared controllers and representations (Schaff et al., 2019; Wang et al., 2018b; Schaff & Walter, 2022; Lu et al., 2025; Wang et al., 2023), while related approaches use pre-trained cross-embodiment models (Strgar & Kriegman, 2025; Fay et al., 2025) or value and dynamics models to guide morphology search (Bohlinger & Peters, 2026; Ha et al., 2026). Consequently, candidate morphologies are increasingly evaluated using knowledge acquired on other bodies, making the reliability of cross-morphology transfer directly relevant to morphology selection.

Controller-dependent morphology search. Prior work provides evidence that the controller used during evaluation can alter morphology selection. Evolutionary studies report a morphological Baldwin effect, favoring bodies that support faster learning (Gupta et al., 2021b), while co-optimization can undervalue morphologies with greater eventual performance (Mertan & Cheney, 2025). Premature fitness evaluation can similarly underestimate morphological potential and bias selection toward faster-learning designs (Song et al., 2026), while jointly training a universal controller on a morphology population can cause diversity collapse, as search converges toward similar bodies that are easier for the shared controller to operate (Strgar & Kriegman, 2025). Recent work has begun to address this interaction more explicitly, incorporating controller adaptation into morphology updates (Dai et al., 2026) and analyzing low-dimensional directions of variation that can be exploited to make joint morphology–control search more efficient (Vaish & Brock, 2026). Together, these studies show that morphology selection can reflect the capabilities of the controller used for evaluation, rather than morphology potential alone. We isolate how morphology-search breadth interacts with controller generalization, and how limited generalization can cause promising morphologies to be lost during selection.

3 Studying Joint-optimization

Typical co-design algorithms sample from a designer and then a controller, with the joint goal of maximizing task return. However, comparisons based only on final return conflate a poor morphology with a controller that has not yet adapted to it. Separating these requires two things: a vocabulary for what each optimizer does, rather than how well the pair performs, and a setting in which that behavior can be set rather than merely observed. This section provides both. We first formalize co-design as a bi-level problem (Sec. 3.1), then identify three properties shared by the optimizers of existing methods, spread, exploration and generalization (Sec. 3.2), and construct a test environment in which each can be controlled independently on either side (Sec. 3.3). Varying them there reveals a consistent asymmetry between designer and controller, which we identify as a result of the Expected Value of Perfect Information (Sec. 3.4).

3.1 PROBLEM SETTING

Co-design selects a design $d \in \mathcal{D}$ together with a controller $\pi_C(a \mid s, d)$ to maximize the fitness $F(\pi_C, d) = \mathbb{E}\big[\sum_t r_t\big]$ (Lu et al., 2025). The two choices do not play symmetric roles: the designer π_D commits to a design before the controller acts on it. Co-design is therefore a bi-level problem (Pujara & Sinha, 2025), in which an upper-level decision, the design, is evaluated only through the lower-level response it induces:

$$\max_{d \in \mathcal{D}} \; F(\pi_d^*, d) \quad \text{s.t.} \quad \pi_d^* \in \arg\max_{\pi} F(\pi, d). \tag{1}$$

Because both levels share the same objective, the upper level seeks to maximize the *optimal value function* $\varphi(d) = \max_\pi F(\pi, d)$ (Pujara & Sinha, 2025): the best fitness achievable on d by any controller, which we call its *potential*.

In practice, the lower level is rarely solved for every candidate design. Joint co-design methods instead evaluate d using the controller at the current training iteration τ, so the designer sees not φ but the *current fitness*

$$F_C(d) \equiv F(\pi_{C,\tau}, d) \;\le\; \varphi(d), \tag{2}$$

a landscape that shifts whenever the controller improves. The designer therefore searches using a moving approximation to morphology potential. Because rollouts are expensive, many designers go one step further and observe F_C through a learned critic $V_D(d) \approx F_C(d)$, analogous to the meta-models used to approximate φ in evolutionary bi-level optimization (Pujara & Sinha, 2025). The failure modes of Sec. 1 now separate cleanly. A design may score poorly because its potential $\varphi(d)$ is low, because the current controller fails to realize its potential, $\varphi(d) - F_C(d) > 0$, or because the designer's critic misjudges its current fitness $V_D(d) \neq F_C(d)$. Final return alone cannot distinguish them.

3.2 ANATOMY OF A CO-DESIGN OPTIMIZER

To compare otherwise different co-design methods, we describe the behavior of their generator and controller with three common metrics:

- **Spread**, ς: how broadly an optimizer samples its search space.
- **Exploration**, ϵ: how much value its own critic believes it gives up by sampling rather than committing to its best choice.
- **Generalization**, ρ: how well it copes with bodies far from those the designer typically produces through value prediction for the designer and control performance for the controller.

Subscripts D and C denote the designer and controller. These metrics are defined analogously across the two optimizers, but their operations depend on what each optimizer predicts or controls.

For measurement we assume each optimizer has a well-defined committed choice: the design $\bar{d}$ it would hand over, and the action $\bar{a}$ its policy would take deterministically. Let $\Delta(d, d')$ count the differences between two bodies, and let P_k be a distribution of bodies at mean distance k from $\bar{d}$. Then

$$\varsigma_D = \mathbb{E}_{d,d' \sim \pi_D}\big[\Delta(d, d')\big], \qquad \varsigma_C = \mathbb{E}_{s,d}\,\mathbb{E}_{a,a' \sim \pi_C(\cdot \mid s,d)}\big[\|a - a'\|\big], \tag{3}$$

$$\epsilon_D = \frac{V_D(\bar{d}) - \mathbb{E}_{\pi_D} V_D}{\mathrm{sd}_{\pi_D} V_D}, \qquad \epsilon_C = \frac{\mathbb{E}_{\bar{\pi}_C} V_C - \mathbb{E}_{\pi_C} V_C}{\mathrm{sd}_{\pi_C} V_C}, \tag{4}$$

$$\rho_D(k) = 1 - \frac{\mathrm{RMS}_{d \sim P_k}\big[V_D(d) - F_C(d)\big]}{F_C(\bar{d})}, \qquad \rho_C(k) = \frac{\mathbb{E}_{d \sim P_k} F_C(d)}{F_C(\bar{d})}. \tag{5}$$

Spread is the mean distance between two samples. Exploration is the regret of a typical sample relative to the committed choice, measured by the optimizer's own critic and standardized by the spread of its predicted values. Generalization is the fraction of performance retained k body differences from the committed design: for the designer, the accuracy of its critic; for the controller, the fraction of its return. Both equal 1 at $k = 0$, and 1 at every k means full generalization. For methods that train an independent controller for every body, cross-morphology controller generalization is not required, it becomes critical when control knowledge is shared across candidate designs. Controller spread is averaged over the states s it visits and the bodies d it acts on. Since its critic $V_C(s, d)$

scores states rather than actions, its exploration compares the states reached by acting on the committed action $\bar{a}$, $s \sim \bar{\pi}_C$, with those reached by sampling, $s \sim \pi_C$, from the same start. Appendix D maps these properties to representative co-design methods. Additional measurement details for real systems are given in Table 5.

3.3 A CO-DESIGN TEST ENVIRONMENT

To conduct our experiments in a clean, controlled setting, we created a simple test environment. This environment consists of a generic reward landscape, randomized by seed.

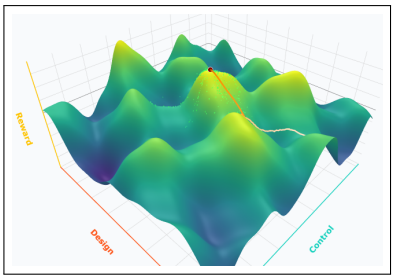


Figure 1: **The test environment**

The environment preserves the following four features of co-design and discards nearly everything else. *(i)* The coupling is emergent. Nothing in the environment links the two optimizers; the designer sees only F_C, exactly as in equation 2. *(ii)* One controller serves every design, so controller generalization is a scarce resource rather than a free one, as in methods with a shared controller. *(iii)* The landscape is multi-scale and fully interacting: broad basins decide which local hills are tall, and where a hill sits along one coordinate depends on every other coordinate, of both design and action. *(iv)* Its dimensionality N scales freely, so each effect can be followed as the problem grows.

Designs and actions are both points in $\mathbb{R}^N$, and a single-step fitness $F^{\text{toy}} : \mathbb{R}^{2N} \to [-1, 1]$ scores each pair, so that $F_C(d) = \mathbb{E}_{a\sim\pi_C(\cdot|d)} F^{\text{toy}}(d, a)$. The landscape is randomly oriented over the joint space, so design and action coordinates are statistically alike and the two optimizers differ only in their role (App. C).

The designer and controller both use the same toy optimizer architecture. Each is a fixed-variance Gaussian evolution strategy: at iteration τ it samples a population of λ candidates around its mean m_τ, optionally improves each by a short climb uphill along its own coordinates, keeps the best μ as elites, and moves m_τ toward their rank-weighted mean by a fraction $\alpha \in (0, 1]$, the recombination rate (Hansen & Ostermeier, 2001). The climb stands in for the model's knowledge about the local space, and is roughly equivalent to a critic or a planner. Each property then has one metric. **Spread** ς^{toy} is the sampling standard deviation. **Generalization** ρ^{toy} is the climb radius: the climb's strength decays with distance from what the optimizer has recently seen, the designer's from its own mean, the controller's from the designs it was last trained on. **Exploration** ϵ^{toy} is the probability that a sample is left unclimbed. An unclimbed sample is one the optimizer's own model knows it could improve, so leaving it raw gives up believed value, which is exactly what the regret of Eq. (4) measures.

The designer proposes a batch of designs, and the controller plays one action on each for κ of its own updates before the designer proposes again; a design's fitness is its mean over that window. At the default $\kappa = 1$, both optimizers rank the same evaluated pairs, but the designer is unconditional while the controller is conditioned on design. Every comparison is made at a fixed evaluation budget B.

3.4 TEST ENVIRONMENT EXPERIMENTAL RESULTS

To measure the effect of these metrics, we run a sweep for $N \in \{1, 2, 4, 8, 16\}$ over random landscapes. Runs are scored by the best design fitness reached, normalized within each landscape (App. C). The results reveal a consistent asymmetry: broader designer search improves discovery, whereas controller performance depends strongly on acting reliably across the designs that search produces. This effect increases with the number of dimensions.

Spread. The designer and controller spreads have opposing effects (Fig. 2, left). Increasing designer spread improves fitness by giving the algorithm a means to find greater optima. Conversely, increasing controller spread reduces fitness. We attribute this asymmetry to EVPI, detailed below (Eq. (6)).

Exploration. Increasing exploration, i.e. sampling more randomly, has little effect on either the controller or designer (Fig. 2, center).

Generalization. Although both the controller and designer benefit from increased generalization, the controller benefits more than 3x as much as at N=16 (Fig. 2, right). A controller whose generalization range does not match the designers' spread fails to utilize those bodies well, and effectively wastes the samples

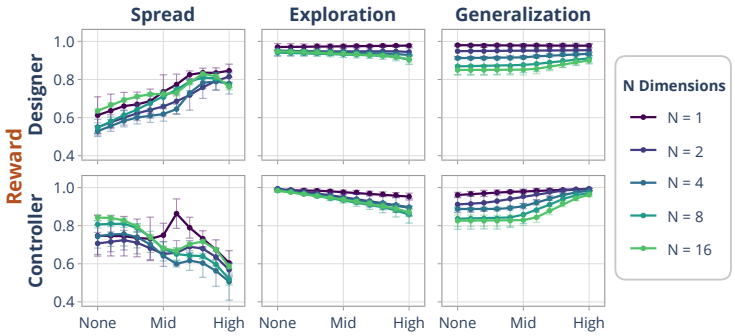


Figure 2: **Spread, exploration and generalization, on each side.** normalized best design fitness against each marginalized metric.

Commitment Gap. Broad designer search, however, is only useful if the promising designs it discovers survive selection. We measure this using the commitment gap: $\Delta\varphi = \max_{d\in\mathcal{H}} \varphi(d) - \varphi(\bar{d})$, as the difference in potential between the best design encountered during search, $\mathcal{H}$, and the design finally committed to, $\bar{d}$. Widening designer spread raises the potential of the best design found, but also widens this gap (Fig. 3a). Increasing controller generalization narrows it (Fig. 3b). Thus broad search can discover a high-potential design without retaining it: if the current controller performs poorly on that design, its observed fitness understates its potential and it can be passed over.

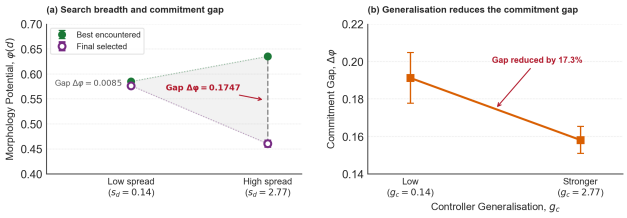


Figure 3: **Broad search finds more, but can commit to less of it.** *(a)* Widening the designer's spread ς^{toy} raises the potential φ of the best design encountered, but the committed design lags further behind, widening the commitment gap $\Delta\varphi$. *(b)* Widening the controller's generalization ρ^{toy} narrows the gap, so more of what broad search finds is kept. Error bars are ± 1 SEM over 64 paired random initialisations of one fixed $N{=}1$ landscape.

Value of perfect information. The common source to both commitment failure and the broader asymmetry is the order of the optimization. The designer samples first, and the controller outputs an action conditioned on that design. Therefore, uncertainty in the design is resolved before the controller acts, but uncertainty in the controller's action is resolved only after the final decision. If $a^*(d)$ is the best response to design d, and $(\bar{d}, \bar{a})$ are the committed pair:

$$\begin{aligned} C_C &= F(\bar{d},\bar{a}) - \mathbb{E}_{a\sim\pi_C}\big[F(\bar{d},a)\big], \\ C_D &= F(\bar{d},\bar{a}) - \mathbb{E}_{d\sim\pi_D}\big[F(d,a^*(d))\big] = C_D^{\text{fixed}} - \text{EVPI}(\pi_D), \\ C_D^{\text{fixed}} &= F(\bar{d},\bar{a}) - \max_a \mathbb{E}_d[F(d,a)] \\ \text{EVPI}(\pi_D) &= \mathbb{E}_d[\max_a F(d,a)] - \max_a \mathbb{E}_d[F(d,a)] \geq 0 \end{aligned} \tag{6}$$

where C_D^{fixed} is what design spread would cost if the action could not respond, and $\text{EVPI}(\pi_D)$ is the expected value of perfect information (Birge & Louveaux, 2011; Raiffa & Schlaifer, 1968): the gain from deciding after uncertainty is resolved rather than before. We can see that design spread is discounted by the EVPI, whereas action spread is paid in full. A random design followed by its best action loses little; the best design followed by a random action loses a lot.

Cross-side coupling These equations formalize the coupling between spread and generalization. The more an optimizer can generalize over the spread, the more EVPI is recovered; this is why stronger controller generalization narrows the commitment gap, and rewards high designer spread. Due to the optimization being sequential from designer to controller, the controller's spread acts as noise on the fitness of the chosen design, which the designer's generalization cannot recover.

Joint sweeps over all six properties and sampling ratios preserve the asymmetry (Fig. 6), and the best fixed-budget ratio is a fresh design for every evaluation, $\kappa = 1$ (Sec. F.0.1). Broader morphology search can improve discovery, but only as far as the controller generalizes across it, so that promising designs stay competitive during selection.

4 Co-Design on Robots

The test environment shows what co-design should look like if each property could be set by hand. Real methods offer no such control. Their spread, exploration, and generalization emerge together from architecture and parameterization, and none is tuned in isolation.

In this section, we verify if the metrics are still meaningful in real co-design algorithms, and if the test environment's findings hold, by measuring each in a modular locomotion robotics environment.

4.1 Setup and measurement

Environment and methods. We evaluate our metrics on the Ant task (Brockman et al., 2016), a locomotion environment in which the body is assembled from limb modules (Fig. 5). We measure on two disparate co-design algorithms: NGE (Wang et al., 2018b), an Evolutionary Algorithm, and BodyGen (Lu et al., 2025), a set of PPO (Schulman et al., 2017) trained transformers.

In Figure 4 each of these methods were retuned for this task with the same effort, and run in Vsim parallelized as far as its algorithm permits, and trained on 3 seeds. The methods train at such different speeds that no single sample budget is fair to all of them, so we plot return against both environment samples and wall-clock time, and each property against the fraction of a run completed.

Measuring on robots. The metrics of Sec. 3.2 are defined for any co-design problem, but each must be interpreted for a new algorithm. For starters, now that we are comparing robots, we need a metric of distance between two bodies. We use a structural edit distance Δ, the number of modules that must change to turn one body into another. A method's committed design $\bar{d}$ is the body it would most strongly recommend at that point in training: the most likely design for a design policy, or the fittest population member for NGE. For generalization, the test bodies should be ones the method might plausibly propose, so rather than mutating $\bar{d}$ by hand, we widen each designer's own distribution in steps, from its committed design out to uniform, choosing each step so that its bodies lie k modules from $\bar{d}$ on average. To define controller spread, we measure how much the controller's commanded actions vary between rollouts of the same body, averaged over bodies. For controller exploration, we value each sampled action by the states it leads to, using the controller's state-value critic. Table 5 lists what each metric is measured from, per method.

4.2 Existing methods through the metrics

Figure 4 reveals the contrasting optimization behavior in NGE and Bodygen through our metrics beyond final return. Return measures the final coupled performance, but cannot distinguish broad morphology search from early commitment followed by control fine-tuning.

NGE maintains a broad designer spread throughout training, but its controllers transfer poorly across morphology changes. It improves bodies over generations through mutation and passing on a parent controller to its children. As such, newly sampled bodies are initially evaluated by controllers adapted to structurally nearby ancestors. The poor transfer makes early evaluations of mutated children particularly sensitive to controller mismatch. Changing a single module of NGE's committed body cuts its return to 43%, compared with 76% for BodyGen, and by the end of training its commanded actions vary over 3 times as much between rollouts of the same body. Thus, a child's observed fitness can reflect controller mismatch as well as morphology quality. With no design critic, designer exploration and generalization are undefined for NGE.

Section 3.4 found that a controller must generalize across the designs it is given to make the most of them. BodyGen shows a contrasting profile. Its shared controller retains substantially more return on nearby morphologies, but its designer rapidly loses spread. It starts out sampling about as broadly as NGE's, but loses most of its spread within the first fifth of training, settling on a single body by around half-way and fine-tuning it for the remaining period. This mirrors the feedback loop described in Sec. 1: a designer that learns from the current controller is drawn toward the bodies that controller already plays well, and the controller in turn specializes on what it is shown.BodyGen trains for under a quarter of SCOPER's sample budget (Table 2); read at that same point (Fig. 4, diamonds), SCOPER's designer spread is already 8.8 modules against BodyGen's eventual 0.02, and its controller already retains essentially all of its return on nearby designs, against BodyGen's final 0.38.

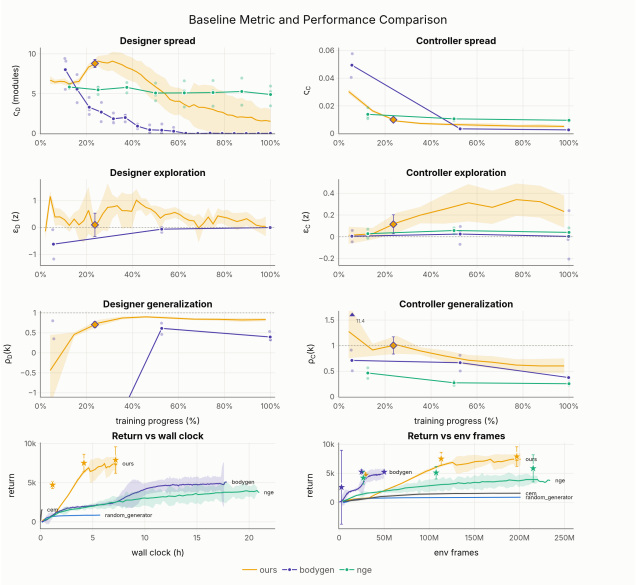


Figure 4: **Baseline Comparison** Rows 1-3 show spread, exploration and generalization of the designer (left) and controller (right), for NGE, BodyGen and SCOPER (Sec. 5). NGE has no designer exploration and generalization, because it has no designer critic. Stars mark specialized return (Sec. F.0.1) at the start, middle and end of each run. Diamonds mark SCOPER's own value read at BodyGen's finishing budget rather than at SCOPER's own end, the one point in rows 1-3 compared on absolute samples instead of each arm's own training progress. Shaded regions are 95% CIs.

Together, these profiles align with the trade-off exposed by the controlled study: NGE maintains search breadth but has weaker cross-morphology evaluation, whereas BodyGen has stronger controller retention but narrows its morphology search early. Neither combines both behaviors.

5 SCOPER

We now ask whether the test environment metrics can guide the design of a method as well as diagnose one. We built SCOPER, to target the asymmetry of Sec. 3.4: its designer samples broadly and converges slowly, while its controller acts near-greedily and shares a representation with the designer, so that it generalizes across the bodies the designer proposes.

Three choices target this profile. First, a designer entropy bonus roughly eighty times larger than the controller encourages broad morphology search while the controller stays sharp. Second, a single transformer shared by designer and controller, trained with forward-dynamics and forward-kinematics auxiliary losses, supports cross-morphology generalization. Third, each parallel environment samples its own body, giving one fresh design per evaluation ($\kappa = 1$), as favored by the controlled study (Fig. 7). If the metrics capture the desired behavior, SCOPER should exhibit this profile and its ablations should reveal which components produce it.

5.1 Algorithm

SCOPER takes inspiration from BodyGen (Lu et al., 2025), reading a body as a set of module tokens processed by a transformer, and training both the designer and controller via PPO (Schulman et al., 2017). The difference is what is shared. While BodyGen uses separate transformers for the designer and controller, SCOPER uses a single trunk for both (Fig. 5). When outputting new morphologies, the trunk reads a partly built body and proposes its next module recursively, until the body is complete. When controlling a body, it reads the body's current state from the set of morphological tokens, which contain local observations, and the CLS token, which contains global observations. It outputs an action for each actuated module, and predicts the auxiliary head's outputs

as well. Both the designer and controller have an actor and a critic head. As they share a backbone, every update to either optimizer therefore changes how the other will act, which Sec. 5.2 tests. Appendix H details the tokenizer and heads.

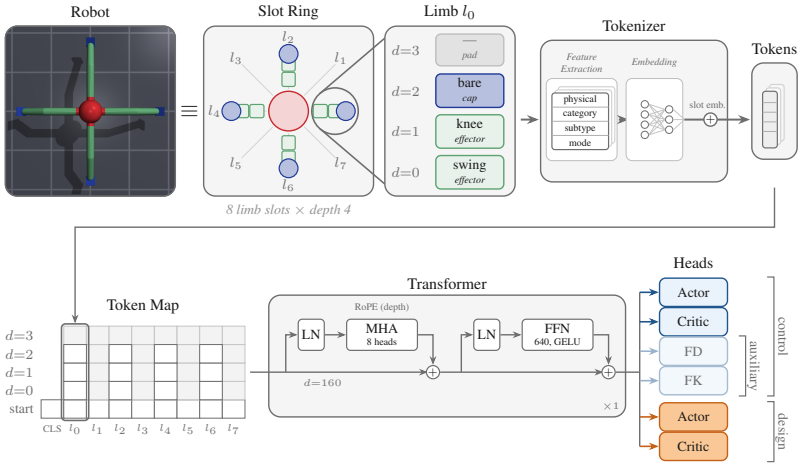


Figure 5: **SCOPER Architecture.** The ordered components of SCOPER. Details in App. H.

Training proceeds in windows of one episode. At the start of each window, the designer samples a separate body for each of the thousands of parallel environments, so that every episode is played on a fresh design. Where BodyGen evaluates a dozen bodies in parallel, we sample thousands. To achieve this, we implement our method in Vlearn (Vsim Technology Ltd, 2026), a highly-parallel GPU reinforcement learning framework powered by the Vsim simulator that supports distinct morphologies across parallel environments. Within the window, the controller trains on every body using PPO (Schulman et al., 2017), with forward-dynamics and forward-kinematics predictions as auxiliary losses. At its end, the designer is updated by policy gradient on the returns its bodies earned, with its critic as a baseline. The entropy bonuses enter here: large in the designer's update, small in the controller's. Appendix I gives the full training loop and pseudocode.

5.2 Ablations

Before testing representation sharing, we verify that the transformer's cross-limb attention is used: masking communication between limbs slows fixed-body learning and reduces return (App. G.1).

Entropy asymmetry. We test whether the two entropy coefficients move only the cells assigned to them in Table 7, sweeping each in isolation from $0.1\times$ to $10\times$ its tuned value with the other held fixed (Fig. 12 and Table 7). Changing each entropy term does move its own designated cells: the generator's takes designer spread from 0.1 to 6.4 modules and designer exploration from 0.08 to 0.76 z; the controller's takes controller generalization from 0.64 to 0.30. But the separation is not clean. Raising the generator's entropy also raises controller generalization, from 0.49 at $0.1\times$ to 0.93 at $10\times$ – a broader body distribution for the designer to sample evidently gives the controller more to generalize over, independent of the controller's own entropy. It moves controller exploration by a comparable amount to the controller's own parameter, too. The reverse leak, from the controller's entropy onto the designer's cells, is smaller throughout. The entropy asymmetry we tune is evidence for the metrics' predicted direction, not proof that the two levers act independently.

Representation Learning. We ablate for representation learning ability in two stages. In **split** the designer and controller have separate networks, and controller knowledge is distilled into the designer through an auxiliary head. In **decoupled**, distillation is also removed. Both substantially reduce co-design performance (Table 1), by roughly one third for split and more than half for decoupled. Importantly, both variants retain substantially broader designer spread than SCOPER, showing that search breadth alone is insufficient for co-design, rather broad morphology search must be paired with a controller that generalizes across the resulting designs. **Split** also reduces controller performance on the bodies its designer samples. In the **decoupled** variant, the controller can achieve higher return on more distant bodies than on the committed morphology (Table 1), indicating misalignment between the morphology selected by the designer and those the controller

can exploit. Overall, sharing the representation improves coordination between morphology search and control rather than simply increasing search breadth.

Physical Auxiliary Objectives. Forward-dynamics and forward-kinematics objectives induce morphological representation learning. Removing these auxiliaries consistently reduces online co-design return after the transition to RL training (Fig. 10).

Table 1: Best value $\pm$ 95% CI, 8 seeds. Controller spread in units of 10^{-3}.

				Designer	Controller		
Ablation	Variant	Return	specialized	Spread	Return (retention)	Spread	Exploration
—	SCOPER	7441 ± 1026	7892 ± 1684	0.5 ± 0.6	6846 ± 998 (0.98 ± 0.02)	5.2 ± 0.9	0.23 ± 0.15
Representation	split	4672 ± 797	6574 ± 1277	4.3 ± 1.2	4878 ± 878 (0.90 ± 0.15)	7.1 ± 1.4	0.03 ± 0.05
	decoupled	2918 ± 1864	4482 ± 3125	4.2 ± 2.7	2960 ± 1876 (0.97 ± 0.23)	3.8 ± 1.0	0.01 ± 0.16
FD/FK	off	5968 ± 2021	6814 ± 1752	0.6 ± 0.5	5484 ± 976 (0.97 ± 0.04)	6.0 ± 4.5	0.40 ± 0.05
Clones	no control clone	7058 ± 906	7312 ± 1894	1.2 ± 1.0	6404 ± 537 (0.98 ± 0.01)	4.9 ± 0.9	0.31 ± 0.13
	control clone, KL only	6806 ± 1720	6111 ± 2767	0.7 ± 0.9	5955 ± 1517 (1.00 ± 0.02)	6.8 ± 1.7	0.27 ± 0.17
	control clone, value only	5626 ± 2337	6119 ± 2403	2.1 ± 2.3	4984 ± 1889 (1.04 ± 0.22)	5.8 ± 1.5	0.42 ± 0.19
	generator clone added	6510 ± 1396	7837 ± 1052	0.9 ± 0.8	6763 ± 1472 (0.98 ± 0.14)	4.0 ± 1.5	0.21 ± 0.23

5.3 Comparisons

Figure 4 places SCOPER alongside NGE and BodyGen for direct comparison of reward, designer spread, and controller generalization. Its designer converges gradually – half-way through training its designer spread is still about 5.9 modules, where BodyGen's has already collapsed to 0.5, and its controller generalizes better than either baseline. SCOPER's reach is also the widest: return stays above 80% of the committed body's return out to 3.4 modules at the final checkpoint, against 0.9 for BodyGen and 0.5 for NGE. Its designer's critic, meanwhile, predicts returns substantially more accurately than BodyGen's.

SCOPER reaches the highest final return of the methods we compare (Table 2). It uses more environment samples than BodyGen, but reaches that return in much less wall-clock time. This is because SCOPER is designed to be highly parallelized, which not only increases throughput, but also lowers κ to 1 (Sec. 5). BodyGen, by comparison, evaluates each design over roughly 30 rollouts across its evolutionary lifetime ($\kappa \approx 30$). Specialized return (stars in Fig. 4; Sec. F.0.1) ablates the differences between a generalized policy and a committed one. See App. G.2 for more details.

Table 2: Comparison of co-design algorithm performance and runtime on the Modular Ant task. Specialized return is evaluated for the final committed body. Mean $\pm$ 95% CI over eight seeds, or three for NGE and BodyGen.

Method	Return	Specialized	Samples	Wall-clock (h)
Random design sampling	861 ± 21	–	201M	5.7
CEM	1560 ± 118	–	205M	0.2
NGE	3974 ± 506	5850 ± 2341	225M	20.9
BodyGen	4971 ± 862	5225 ± 261	50M	17.6
SCOPER	7441 ± 1026	7892 ± 1684	201M	7.3

6 Conclusion

In this paper we investigate a fundamental quirk of co-design optimization, which is that the body's and the controller's optimizers are asymmetric in their preferred behaviors. Our experiments in both theoretical and applied environments demonstrate that spread is useful to the designer and not the controller, while generalization has a stronger benefit to the controller than the designer. Based on these results, we present a simple pair of principles to guide the development of future co-design algorithms: the designer should maintain sampling breadth, while the controller must generalize well and sample greedily.

Additionally, we demonstrate that the co-design algorithms NGE (Wang et al., 2018b) and BodyGen (Lu et al., 2025) are both limited by the underperformance of one of their optimizers. To test our hypothesis, we developed SCOPER, which uses representation learning and a shared-network design to improve controller generalization, and high entropy on the designer heads to enforce designer spread. Our ablations corroborate both the efficacy of these additions for producing the desired behaviors, and for increasing performance.

AI USE STATEMENT

Generative AI tools were utilized during both the research and manuscript preparation phases of this work in accordance with the ICLR 2027 AI Policy. Specifically, AI programming agents and environments (including Claude Code, Google Antigravity, and OpenAI Codex) were employed to assist in software implementation, as well as with paper revision. The human authors manually reviewed all code and text produced, and assume full accountability for the accuracy and integrity of the entire submission. All reported numbers were computed from saved experimental outputs. AI was used to implement our own ideas, not to decide research directions or autonomously perform research.

REFERENCES

John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2 edition, 2011. doi: 10.1007/978-1-4614-0237-4.

Nico Bohlinger and Jan Peters. Shape Your Body: Value Gradients for Multi-Embodiment Robot Design. 2026.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Nick Cheney, Josh Bongard, Vytas SunSpiral, and Hod Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15 (143):20170937, June 2018. ISSN 1742-5689. doi: 10.1098/rsif.2017.0937. URL https://doi.org/10.1098/rsif.2017.0937.

Yanning Dai, Yuhui Wang, Dylan Ashley, and Jürgen Schmidhuber. Efficient Morphology-Control Co-Design via Stackelberg Proximal Policy Optimization. *International Conference on Learning Representations*, 2026:144280–144316, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e9750610639c3e7a849cff746bf60dbd-Abstract-Conference.html.

Kehlani Fay, Darin Anthony Djapri, Anya Zorin, James Clinton, Ali El Lahib, Hao Su, Michael T. Tolley, Sha Yi, and Xiaolong Wang. House of Dextra: Cross-embodied Co-design for Dexterous Hands. *arXiv preprint*, December 2025. doi: 10.48550/arXiv.2512.03743. URL http://arxiv.org/abs/2512.03743. arXiv:2512.03743 [cs] version: 1.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. MetaMorph: Learning Universal Controllers with Transformers. October 2021a. URL https://openreview.net/forum?id=Opmqtk_GvYL.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 12(1):5721, October 2021b. ISSN 2041-1723. doi: 10.1038/s41467-021-25874-z. URL https://www.nature.com/articles/s41467-021-25874-z.

Huy Ha, C. Karen Liu, and Shuran Song. Transformer Transformer: A Unified Model for Motion-Conditioned Robot Co-design, July 2026. URL http://arxiv.org/abs/2607.25798. arXiv:2607.25798 [cs.RO].

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 4455–4464. PMLR, November 2020. URL https://proceedings.mlr.press/v119/huang20d.html.

V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson. My Body is a Cage: the Role of Morphology in Graph-Based Incompatible Control, 2020. URL https://openreview.net/forum?id=N3zUDGN5lO. Conference Name: ICLR 2021 - 9th International Conference on Learning Representations Meeting Name: ICLR 2021 - 9th International Conference on Learning Representations Volume: 9.

Haofei Lu, Zhe Wu, Junliang Xing, Jianshu Li, Ruoyu Li, Zhe Li, and Yuanchun Shi. BodyGen: Advancing Towards Efficient Embodiment Co-Design. In *The Thirteenth International Conference on Learning Representations (ICLR)*, January 2025. URL https://openreview.net/forum?id=cTR17xl89h.

Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient Co-Adaptation of Morphology and Behaviour with Deep Reinforcement Learning. In *Proceedings of the Conference on Robot Learning*, pp. 854–869. PMLR, May 2020. URL https://proceedings.mlr.press/v100/luck20a.html.

Alican Mertan and Nick Cheney. Evolutionary Brain-Body Co-Optimization Consistently Fails to Select for Morphological Potential. MIT Press, October 2025. doi: 10.1162/ISAL.a.891. URL https://dx.doi.org/10.1162/ISAL.a.891.

Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei Efros. Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/c26820b8a4c1b3c2aa868d6d57e14a79-Abstract.html.

Dhaval Pujara and Ankur Sinha. A Review of Bilevel Optimization: Methods, Emerging Applications, and Recent Advancements, November 2025. URL https://arxiv.org/abs/2511.03448. arXiv:2511.03448 [math.OC].

Howard Raiffa and Robert Schlaifer. *Applied Statistical Decision Theory*. M.I.T. Press, Cambridge, Mass., student edition, April 1968.

Charles Schaff and Matthew R. Walter. N-LIMB: Neural Limb Optimization for Efficient Morphological Design, September 2022. URL http://arxiv.org/abs/2207.11773. arXiv:2207.11773 [cs.RO].

Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R. Walter. Jointly Learning to Construct and Control Agents using Deep Reinforcement Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9798–9805. arXiv, May 2019. doi: 10.1109/ICRA.2019.8793537. URL https://ieeexplore.ieee.org/abstract/document/8793537.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pp. 15–22, New York, NY, USA, July 1994. Association for Computing Machinery. ISBN 978-0-89791-667-7. doi: 10.1145/192161.192167. URL https://dl.acm.org/doi/10.1145/192161.192167.

Junru Song, Yang Yang, Yaqing Xu, Ying Wen, Wei Peng, Guozhen Li, Wei'en Zhou, and Wen Yao. Shaping the Evolutionary Dynamics of Robot Morphology via Adaptive Control Learning, August 2026. URL http://arxiv.org/abs/2608.23100. arXiv:2608.23100 [cs.RO].

Luke Strgar and Sam Kriegman. Accelerated co-design of robots through morphological pretraining. *arXiv preprint*, February 2025. doi: 10.48550/arXiv.2502.10862. URL http://arxiv.org/abs/2502.10862. arXiv:2502.10862 [cs].

Apoorv Vaish and Oliver Brock. Identifying and Exploiting Structure in Robot Co-Design, August 2026. URL http://arxiv.org/abs/2604.11768. arXiv:2604.11768 [cs.RO].

Vsim Technology Ltd. Vsim: Next-gen physics simulation platform, 2026. URL https://v-sim.co.uk/. Accessed: 2026-09-25.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. NerveNet: Learning Structured Policy with Graph Neural Networks. February 2018a. URL https://openreview.net/forum?id=S1sqHMZCb.

Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural Graph Evolution: Towards Efficient Automatic Robot Design. September 2018b. URL https://openreview.net/forum?id=BkgWHnR5tm.

Yuxing Wang, Shuang Wu, Tiantian Zhang, Yongzhe Chang, Haobo Fu, Qiang Fu, and Xueqian Wang. PreCo: Enhancing Generalization in Co-Design of Modular Soft Robots via Brain-Body Pre-Training. In *Proceedings of The 7th Conference on Robot Learning*, pp. 478–498. PMLR, December 2023. URL https://proceedings.mlr.press/v229/wang23b.html.

Zheng Xiong, Jacob Beck, and Shimon Whiteson. Universal Morphology Control via Contextual Modulation. June 2023. URL https://openreview.net/forum?id=PzlO3SSk3A.

Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris M. Kitani. Transform2Act: Learning a Transform-and-Control Policy for Efficient Agent Design. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=UcDUxjPYWSr.

Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. RoboGrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics*, 39(6):1–16, December 2020. ISSN 0730-0301, 1557-7368. doi: 10.1145/3414685.3417831. URL https://dl.acm.org/doi/10.1145/3414685.3417831.

Appendices

A Notation **14**

B Limitations **14**

C The Test Environment **14**

D Mapping existing methods onto the co-design anatomy **15**

E Metric Main Effects **16**

F Additional Experimental Details **16**

G Additional Results **17**

H Architecture Details **19**

I SCOPER Details **20**

A NOTATION

Table 3: Notation.

Symbol	Meaning
Problem setting	
$d \in \mathcal{D}$	design; specialized to a graph G in Sec. 5
π_D, θ_D	designer and its parameters
$\pi_C(a \mid s, d)$, θ_C	controller and its parameters
s_t, a_t, r_t	state, action and reward at timestep t
$F(\pi_C, d)$	fitness, $\mathbb{E}\big[\sum_t r_t\big]$
π_d^*	optimal controller for design d
$\varphi(d)$	potential (optimal value function), $\max_\pi F(\pi, d)$
τ	optimizer iteration; dropped when implied
$F_C(d)$	current fitness, $F(\pi_{C,\tau}, d) \le \varphi(d)$
$V_D(d)$	designer critic, $\approx F_C(d)$
Metrics (subscripts D, C denote designer and controller)	
$\bar{d}$, $\bar{a}$	committed design and action (the policy mean)
$\Delta(d, d')$	body distance: number of body differences
k, P_k	distance level; bodies at mean distance k from $\bar{d}$
ς	spread: mean pairwise Δ between samples
ϵ	exploration: z-scored regret under the optimizer's own critic
$\rho(k)$	generalization at distance k; 1 is full generalization
$\mathcal{H}$	designs encountered during search
$\Delta\varphi$	commitment gap, $\max_{d\in\mathcal{H}} \varphi(d) - \varphi(\bar{d})$
ϵ_{clip}	PPO clipping parameter
Test environment	
N	dimensionality, per side
$F^{\text{toy}}(d, a)$	test-environment fitness, $\mathbb{R}^{2N} \to [-1, 1]$
Q	random rotation of the joint space
m_τ	optimizer mean at iteration τ
i	sample index within a population
λ, μ	population size; number of elites
α	recombination rate
$\varsigma^{\text{toy}}, \epsilon^{\text{toy}}, \rho^{\text{toy}}$	parameters: sampling std, P(unclimbed), climb radius
κ	sampling ratio: controller updates per design batch
B	evaluation budget
SNR_D, SNR_C	fitness signal-to-noise ratio of each side
η^2	share of variance explained

B LIMITATIONS

Our metrics necessarily simplify co-design behavior: structural distance is only a proxy for meaningful morphological change, and not every metric cleanly applies to every algorithm. Specialist evaluation considers only the committed morphology, while our robot experiments are limited to Ant and a single modular grammar. Finally, the controlled test environment abstracts away richer sequential design and control decisions, so the observed asymmetry should be tested across broader tasks and morphology spaces.

C THE TEST ENVIRONMENT

The landscape is a sum of sine waves, each with a randomly drawn frequency, phase and amplitude, evaluated in a randomly rotated frame. One draw is fixed by (N, seed).

A design and an action are both points in $\mathbb{R}^N$, and the landscape scores the joint point $x = (d, a) \in \mathbb{R}^{2N}$. Before anything is evaluated, x is rotated by a Haar-uniform $Q \in O(2N)$, drawn as the QR factor of a Gaussian matrix with signs fixed so that R has a positive diagonal:

$$z = Qx. \tag{7}$$

The rotation is what couples the two optimizers. No coordinate of z belongs to the designer or the controller alone, so every design coordinate is mixed with every action coordinate, even at $N = 1$.

The rotated coordinates are read in pairs: dimension $i \in \{1, \dots, N\}$ takes z_i together with $z_{i'}$, where $i' = N + i$. Each pair is scored by four products of two sines, written below with $u_j^t = f_j^t z_j + \phi_j^t$ for frequency f_j^t and phase ϕ_j^t at coordinate $j \in \{1, \dots, 2N\}$. One product is broad — the *basin* β_i, whose half-period exceeds any sampling cloud we use — and three are finer: a coarse sinusoid, a fine one, and a high-exponent *spike* that is near zero except in small regions. The latter three form the *texture* T_i:

$$\beta_i = \sin u_i^b \sin u_{i'}^b, \qquad T_i = A_i^m \sin u_i^m \sin u_{i'}^m + A_i^w \sin u_i^w \sin u_{i'}^w + A_i^s |\sin u_i^s|^{n_i} |\sin u_{i'}^s|^{n_{i'}}. \quad (8)$$

The basin enters twice, as a term and as an envelope, so hills on a crest are $(1 + \gamma)$ times taller than those in a valley ($\gamma = 0.75$). One term is exempt from the rotation, on the raw coordinates, and is the only non-interacting structure:

$$P_i = A_i^b \beta_i + (1 + \gamma \beta_i) T_i, \qquad S_i = A_i^p \sin(f_i^p d_i + \phi_i^p) + A_{i'}^p \sin(f_{i'}^p a_i + \phi_{i'}^p), \quad (9)$$

$$F^{\text{toy}}(d, a) = \frac{1}{N} \sum_{i=1}^{N} \big(P_i + S_i\big) + c. \quad (10)$$

Every frequency, exponent and amplitude is drawn as base $\cdot\, e^{\varepsilon}$, $\varepsilon \sim \mathcal{N}(0, s^2)$, phases uniform on $[0, 2\pi)$; design and action share one base per constant, so neither side is structurally coarser. Bases (s): $f^b = 2\pi/24\,(0.15)$ capped at $2\pi/16$, $f^m = 3.5\,(0.30)$, $f^w/f^m = 2.8\,(0.15)$, $f^s = 1.29\,(0.50)$, $n = 20\,(0.35)$ rounded and ≥ 2, $f^p = 1.67\,(0.50)$; amplitudes $A^b, A^m, A^w, A^s, A^p = 3.0, 2.04, 0.55, 1.07, 2.0$, all (0.15). Amplitudes are scaled and c chosen so the range implied by the draw's own amplitudes is exactly $[-1, 1]$ at every N; realized extrema are not controlled and grow rarer with N, so runs are normalized within their landscape. Runs start in $[-2, 2]$.

Climb. Four gradient-ascent steps along one side's coordinates, the other fixed, of size $4/L$ with L an analytic Lipschitz bound on ∇F^{toy}. Generalization attenuates each sample's step by $w = \exp(-\|p - \bar{p}\|^2 / 2\rho^2)$, its squared joint distance from the optimizer's center (per dimension, so ρ means the same at every N); exploration gates it, leaving a sample raw with probability ϵ.

D MAPPING EXISTING METHODS ONTO THE CO-DESIGN ANATOMY

Table 4: How existing co-design methods realize spread, exploration and generalization in each optimizer. An empty cell ("none") marks a mechanism the method does not use. *Measured on robots in Sec. 4; the remaining rows are read from each method's design, to show how the metrics apply, and are not run. This assumes NGE is run without GM-UC, as per their defaults.

	Designer			Controller		
Method	Spread	Exploration	generalization	Spread	Exploration	generalization
Per-body ES (Sims, 1994)	mutation, crossover	none	none	mutation, crossover	none	not needed (per body)
NGE* (Wang et al., 2018b)	mutation	none	none	Gaussian σ	Gaussian σ	per body, inherited from parent
RoboGrammar (Zhao et al., 2020)	ϵ random rules	ϵ-greedy	GNN heuristic	MPPI samples	none (executes mean)	not needed (per body)
Transform2Act (Yuan et al., 2022)	stochastic policy	sampling	shared GNN value	Gaussian σ	Gaussian σ	shared GNN + JSMLP
BodyGen* (Lu et al., 2025)	stochastic policy	sampling	design-stage value	Gaussian σ	Gaussian σ	shared MoSAT + TopoPE
SCOPER * (ours, Sec. 5)	stochastic policy	sampling	GenCrit, shared trunk	Gaussian σ	Gaussian σ	shared transformer trunk

The mapping in Table 4 highlights that different co-design algorithms can be described in terms of the same three properties, while differing in which of them they explicitly realize. An empty cell does not indicate a deficiency in the method but that the corresponding mechanism is absent or unnecessary. NGE, for instance, has neither designer exploration nor designer generalization in the form we run: it never predicts a design's value, and so never estimates a design's potential before trying it. Likewise, methods that train a controller per body obtain controller generalization for free, since no controller acts on a body it was not trained for, at the cost of a full inner optimization per candidate; every method that shares a controller must instead earn it through its architecture, which is where most recent work has concentrated.

This view also makes visible an asymmetry that is usually implicit in algorithm design: existing methods employ different mechanisms for the designer and controller, but do not generally formulate how the desired behavior of the two optimizers should differ. Section 3.4 studies that question directly in a setting where spread, exploration, and generalization can be varied independently.

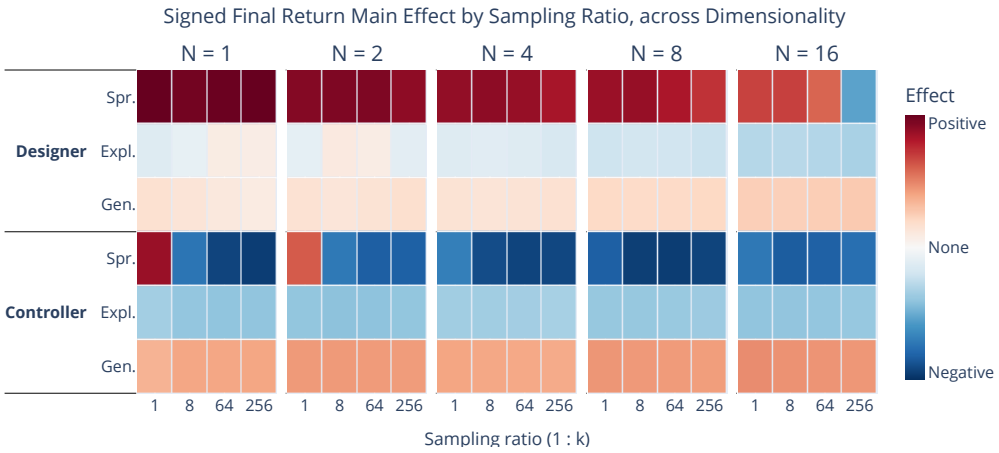


Figure 6: **Comparing Metric Effects.** This shows the main effect of each metric on normalized best design fitness, marginalizing over all other metrics, at four sampling ratios κ and five dimensions N.

E Metric Main Effects

Sweeping all six metrics jointly confirms the asymmetry (Fig. 6). The designer's and controller's spread contribute the largest effect, increased by N. Controller exploration and generalization follow in effect size, while the designer's remain close to zero throughout. At large N, longer adaptation windows weaken the benefit of designer spread, reversing it only at $N = 16$ and $\kappa = 256$.

At high values of N and κ, we see the designer's spread preference invert. This is a natural consequence of needing to be more greedy with our design choices when only a small number of design updates are available.

F Additional Experimental Details

F.0.1 Metric Measurements

Table 5: What each metric is measured from, per method.

	NGE	BodyGen	SCOPER
Committed design $\bar{d}$	fittest member	most likely design	most likely design
Designer critic V_D	(not measurable)	attribute critic	designer critic
Designer spread cohort	population	replay buffer	designer samples
Controller critic	state value	state value	state value

Sampling ratio. At a fixed evaluation budget, design fitness falls as the sampling ratio κ grows, at every N, whether the other six metrics are averaged over or tuned separately for each ratio (Fig. 7). The drop is steepest at large N.

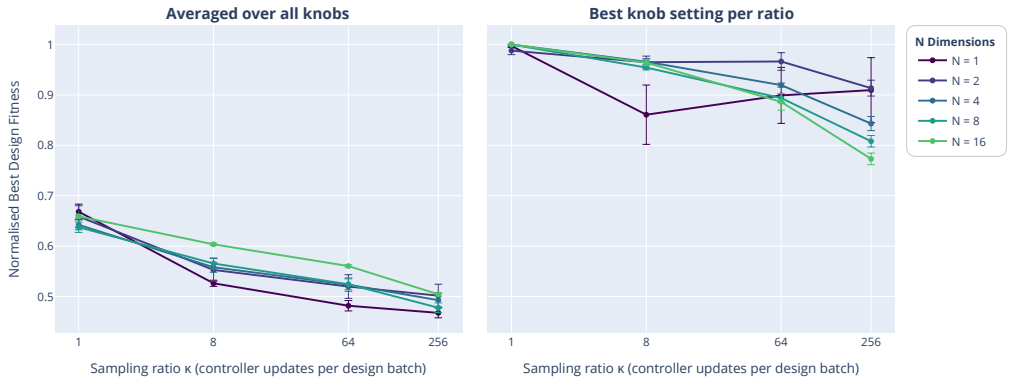


Figure 7: **A fresh design for every evaluation is best.** normalized best design fitness against the sampling ratio κ at a fixed evaluation budget. *Left:* averaged over the other six metrics. *Right:* the best metric setting at each ratio. Error bars are ± 1 SEM over four landscapes.

Specialized return. Process metrics show how a method searches, not whether it found a better body: BodyGen's early concentration, for instance, could be premature collapse or the efficient discovery of a strong design. specialized return separates the body from the controller that co-designed it. At the beginning, middle and end of each run, we take the method's committed design $\bar{d}$, fine-tune its controller on $\bar{d}$ alone for the same 32.8M environment samples (250 PPO epochs) for every method, and report the mean return of the resulting greedy policy over 32 episodes on each of 256 environments, taken at the fine-tuning checkpoint with the highest training return. This estimates the potential $\varphi(\bar{d})$ of Sec. 3.1, for the body the method commits to rather than the best body in the library. For SCOPER, $\bar{d}$ is the modal greedy design, since the greedy generator's output varies with the order in which limbs are grown.

G Additional Results

G.1 Ablations

Cross-limb attention. We test whether the controller benefits from communication between limb tokens. The full SCOPER controller uses unrestricted attention over morphology-aware control tokens. We compare it with two masked variants: `self_cls`, where each limb token attends only to itself and the global torso token, and `self`, where each token attends only to itself. These runs do not resample morphology and therefore isolate control learning from the co-design process.

Restricting attention reduces both learning speed and asymptotic return. Full attention reaches the reference threshold, 90% of the `self_cls` asymptote, in roughly one third of the training epochs required by `self_cls`, while the fully independent `self` variant converges to a much lower return. Cross-limb communication therefore contributes materially to control learning on the morphology-aware representation.

G.2 Comparison Metrics

Ablation curves. Figure 8, 9, 10 and 11 show the training curves behind Table 1.

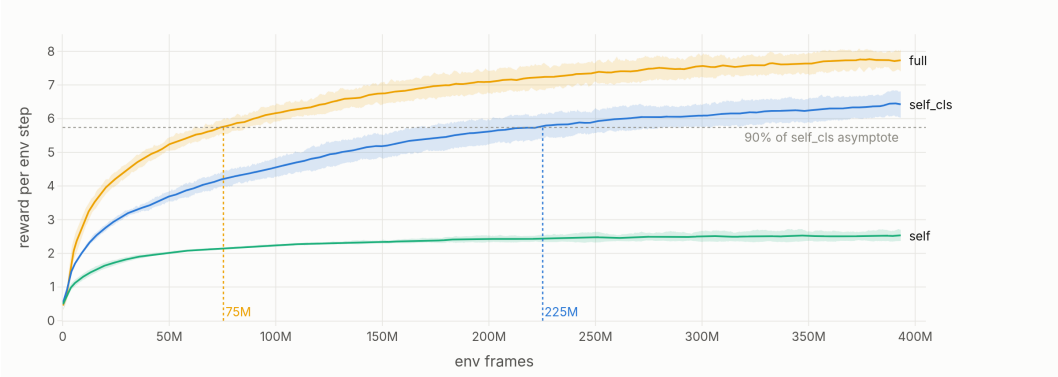


Figure 8: Ablation of cross-limb attention.

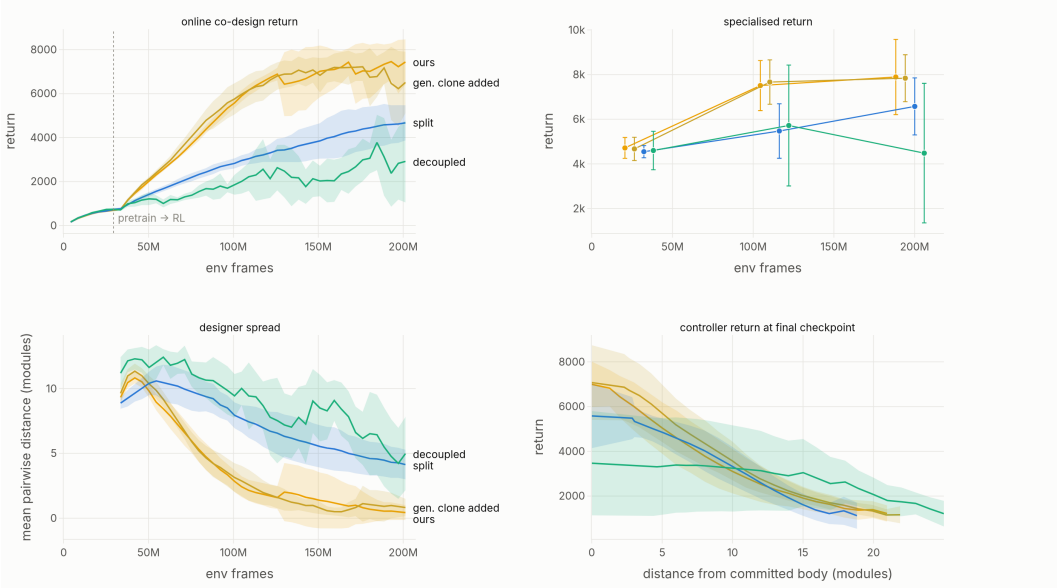


Figure 9: Ablation of the shared backbone.

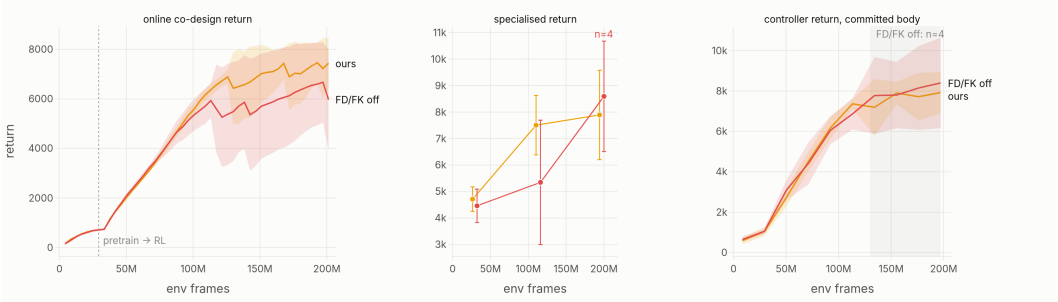


Figure 10: Ablation of the FD/FK auxiliary objectives.

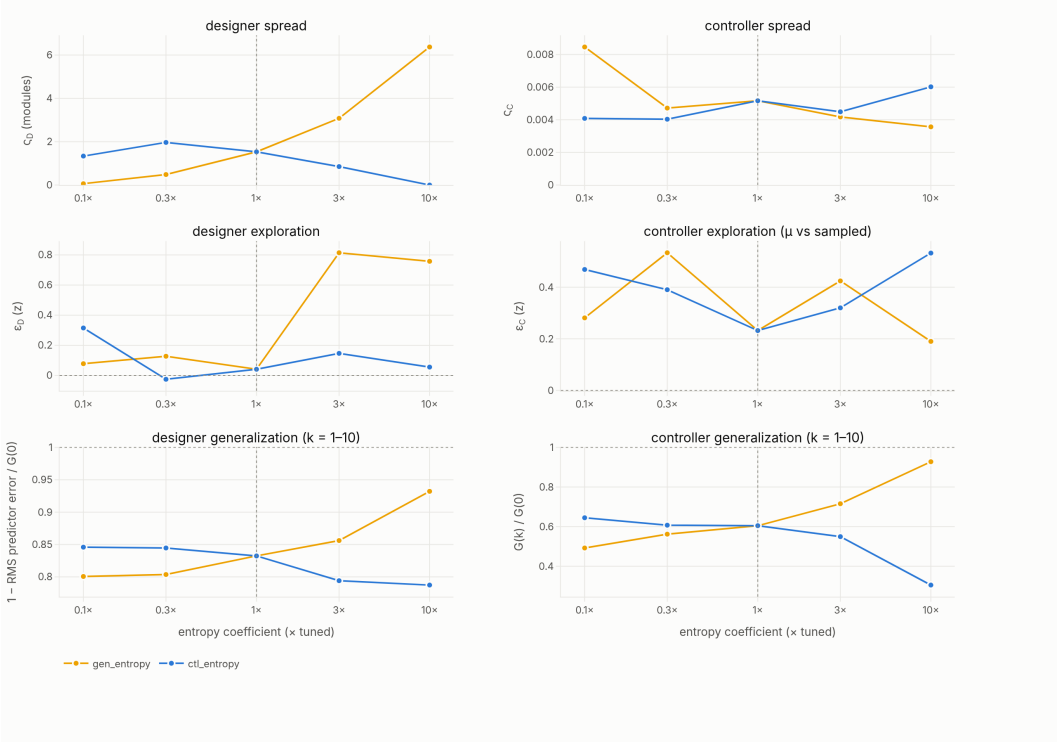


Figure 12: Sweeping each entropy coefficient in isolation, 0.1–10× its tuned value, against the six search-configuration cells (Sec. G.2).

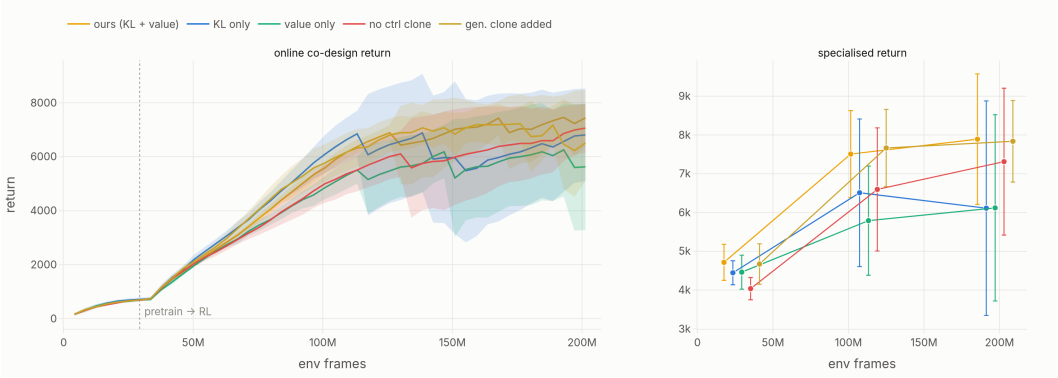


Figure 11: Ablation of controller preservation (control clone).

H Architecture Details

Tokenizer. The tokenizer runs in two stages (Fig. 5). Every module, an unoccupied one included, is first turned into the same four-block feature vector: its physical state; a category one-hot (effector, cap or pad); a subtype one-hot over the module library (`swing`, `knee`, `bare`, ...); and a three-valued mode field. A single linear map, shared across modules, then embeds each vector, and an additive slot embedding marks its position. Neither stage mixes one module into another; that is the trunk's job. An unoccupied slot is neutralized by its content rather than by an attention mask, so the design path needs no mask.

Table 7: Entropy coefficient sweep, 0.1–10× the SCOPER default value

		Designer			Controller		
Lever	× tuned	ς_D	ε_D	$\rho_D(k)$	ς_C ($\times 10^{-3}$)	ε_C	$\rho_C(k)$
—	1×	1.5 ± 1.5	0.04 ± 0.37	0.83 ± 0.04	5.2 ± 0.9	0.23 ± 0.15	0.60 ± 0.15
Generator	0.1×	0.1	0.08	0.80	8.5	0.28	0.49
	0.3×	0.5	0.13	0.80	4.7	0.53	0.56
	3×	3.1	0.81	0.86	4.2	0.42	0.72
	10×	6.4	0.76	0.93	3.6	0.19	0.93
Controller	0.1×	1.3	0.31	0.85	4.1	0.47	0.64
	0.3×	2.0	−0.03	0.84	4.0	0.39	0.61
	3×	0.9	0.15	0.79	4.5	0.32	0.55
	10×	0.0	0.06	0.79	6.0	0.53	0.30

Algorithm 1 SCOPER training loop.

Require: shared trunk with heads ContAct, ContCrit, GenAct, GenCrit; teacher sampler q; warmup length W; E environments

for window $w = 0, 1, 2, \ldots$ **do**
  for each environment $e = 1, \ldots, E$ **do** ▷ one body per environment
    $d_e \sim \pi_D$ with probability $\min(w/W, 1)$, else $d_e \sim q$ ▷ storing V_D at every prefix
  end for
  Rebuild the simulator with bodies $d_{1:E}$
  repeat ▷ control phase
    Collect a rollout on every body
    Update trunk, ContAct and ContCrit by PPO, with FD and FK auxiliary losses
  until one episode of steps has elapsed
  $R_e \leftarrow$ mean episode return on d_e
  Snapshot ContAct and ContCrit
  for 16 epochs **do** ▷ designer phase
    if $w < W$ **then**
      Fit GenAct to teacher bodies by NLL
    else
      Update GenAct by PPO, $A_t = V_D(p_{t+1}) - V_D(p_t)$, with entropy bonus
    end if
    Fit GenCrit to R_e at every prefix of d_e and on live rollout states
    Control clone: KL and MSE hold ContAct and ContCrit near their snapshot
  end for
end for

Token map. The tokens form a grid with one column per limb slot and one row per depth, with the CLS token in column 0 of the start row. A single limb is therefore a column, strided by 8 in the sequence, rather than a contiguous run. Depth enters inside attention, as a rotary embedding on the queries and keys.

Modes and heads. The trunk carries four heads: the controller's actor and critic (ContAct, ContCrit) and the designer's actor and critic (GenAct, GenCrit). In design mode the physical fields are zero and the CLS token is a learned parameter. GenCrit is fit in both modes: on partly designed bodies during generation, and on live rollout states during control, both against the same episode return.

I SCOPER Details

Algorithm 1 and Fig. 13 give the full loop.

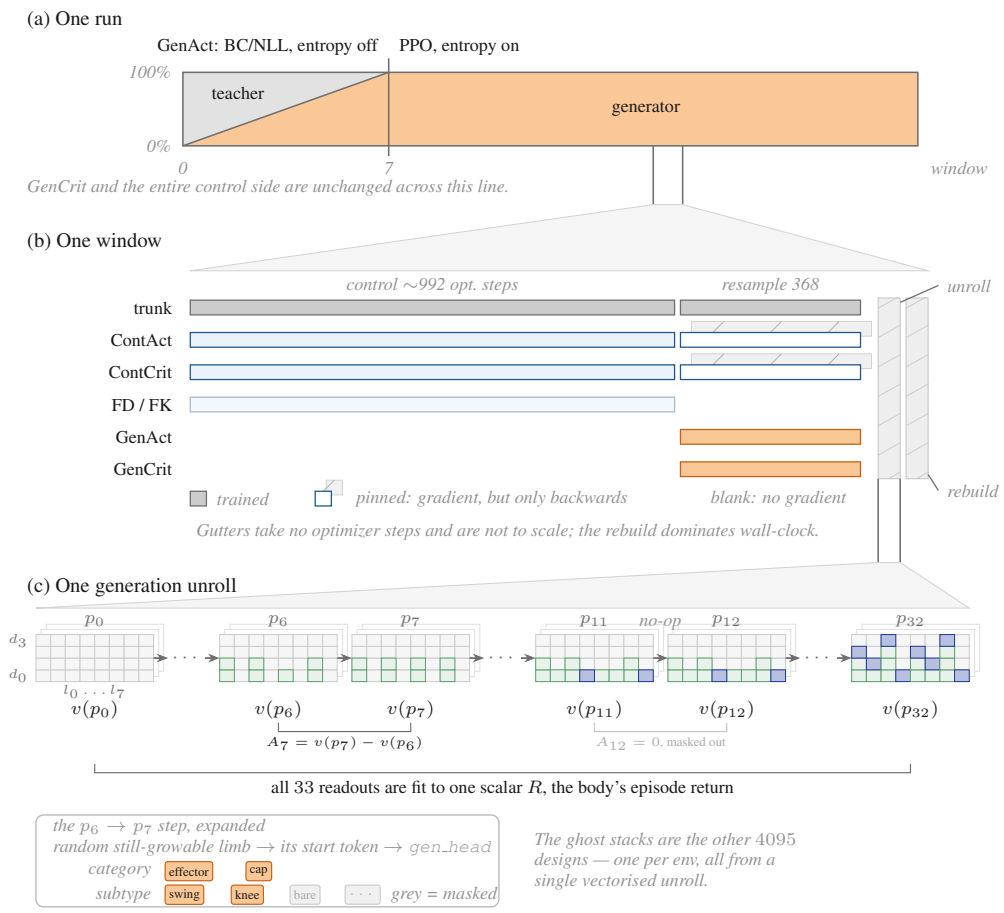


Figure 13: **Training loop.** (a) Over a run, designer bodies replace the teacher's. (b) Each window trains the controller, then the designer, then rebuilds the simulator with new bodies. (c) The designer grows a body limb by limb; its critic scores every prefix against the body's return.

Control and generator clones. Every designer update moves the shared trunk, and with it the controller, as a side effect. The control clone counteracts this: during each designer update, a KL term holds the controller's actor near its pre-update policy, and a squared-error term holds its critic near its pre-update values, with coefficients 0.119 and 0.399. The generator clone is its mirror image: it holds the designer's actor and critic near their outputs while the controller's updates move the trunk, anchored once per window on the bodies being played. SCOPER keeps the control clone only because its configuration was tuned with it on; the ablation finds no consistent effect of removing it (Table 1). The generator clone is not part of SCOPER.

Representation ablation arms. Split gives designer and controller disjoint trunks and token embeddings, each with its own optimizer, over the same morphology-conditioned inputs. The designer's trunk learns the controller's function by distillation: copies of the controller's actor and critic, built on the designer's trunk, are fit to the controller's outputs over a reservoir of transitions. Beyond this, the two share only the observation normalizer and forward-kinematics statistics. Decoupled removes the distillation and every other channel from controller to designer, leaving each body's return as the only shared signal. Its designer still reads morphology tokens and rest-pose observations through its own encoder, and is trained to predict forward kinematics from them; no controller action, value, hidden state or representation reaches it. Distillation coefficients are left untuned at 1.0; all other coefficients are those tuned for SCOPER.