EVOLVING DEXTEROUS ROBOTS FROM SCRATCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Little is known about how to manually design agents capable of dexterous manipulation. Some design principles have been inferred from close examination of how animals manipulate objects, but these structures and behaviors have so far resisted biomimicry and may not be optimal for artificial machines. Here we evolve freeform robots to pick up, hold, rotate, and use diverse objects. Unlike other approaches to optimizing robot hands, we do not presuppose the presence, articulation, or geometry of any part of the body. Although familiar prehensile forms such as tails, beaks, paws and claws may emerge spontaneously under certain conditions—and while such conditions could be of interest to evolutionary biologists—de novo manipulator design can also reveal whole new solutions, overlooked or unknown structures which may be better suited for the task at hand. We use contrastive learning to create a highly searchable genetic embedding of design space, an autoregressive developmental model to decode designs, evolutionary strategies to find good designs, and reinforcement learning to train each evolved design. Winning designs were automatically converted into a manufacturable blueprint, printed, assembled and tested in the real world in a zero-shot manner. The results represent the state-of-the-art in evolutionary robotics in terms of performance, diversity and complexity. Project page: evodexbots.github.io.

1 INTRODUCTION

A diversity of prehensile structures have evolved in animals to support effective object manipulation (Sugasawa et al., 2021). While the specific solutions found in nature (e.g. an elephant's trunk, an insect's mandibles, a primate's hand) might not be suitable for artificial machines, which are made of very different components and coordinated by very different control systems, it may still be beneficial to mimic the evolutionary mechanisms responsible for the emergence, complexification, and diversification of nature's designs (Nolfi & Floreano, 2002). That is, it can be useful and interesting to design robots automatically with minimal assumptions. However, co-optimizing a robot's physical layout and control policy is a notoriously challenging problem (Cheney et al., 2018; Strgar et al., 2024; Strgar & Kriegman, 2026).

Several approaches for automatic robot design have been reported in the literature. The majority focused on locomotor performance and completely ignored object manipulation (Lipson & Pollack, 2000; Hiller & Lipson, 2012; Brodbeck et al., 2015; Kriegman et al., 2020; Matthews et al., 2023; Yu et al., 2026). However, some manipulator structures have also been optimized. Usually the size and shape of given links are deformed within a manually determined kinematic structure, such as a serial chain (Xu et al., 2021) or parallel jaw gripper (Ha et al., 2020; Xu et al., 2025). Similarly, Liu et al. (2018) optimized the 2D topology of compliant jaws, and Bai et al. (2026) optimized the relative position and orientation of three soft fingers as well as the size, spacing, and tendon routing of four component blocks along each finger. However, these underactuated grippers could only grasp and release, and radically different anatomies (e.g. more or less fingers) could not emerge.

Fay et al. (2026) optimized the kinematic structure of a manipulator using stacks of cubic modules to form inward-bending fingers of variable position and length. The optimized design, when built, exhibited faster in-hand object rotation than an anthropomorphic baseline (Shaw et al., 2023). However, the shape and articulation of each finger was largely presupposed, and their position was locked in plane about a 2D palm. As a result, a limited diversity of designs could arise, and differences between unoptimized and optimized manipulators could be difficult to discern. Mirzaee et al. (2026) likewise optimized the distribution and length of modular rectangular fingers as well as the

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.

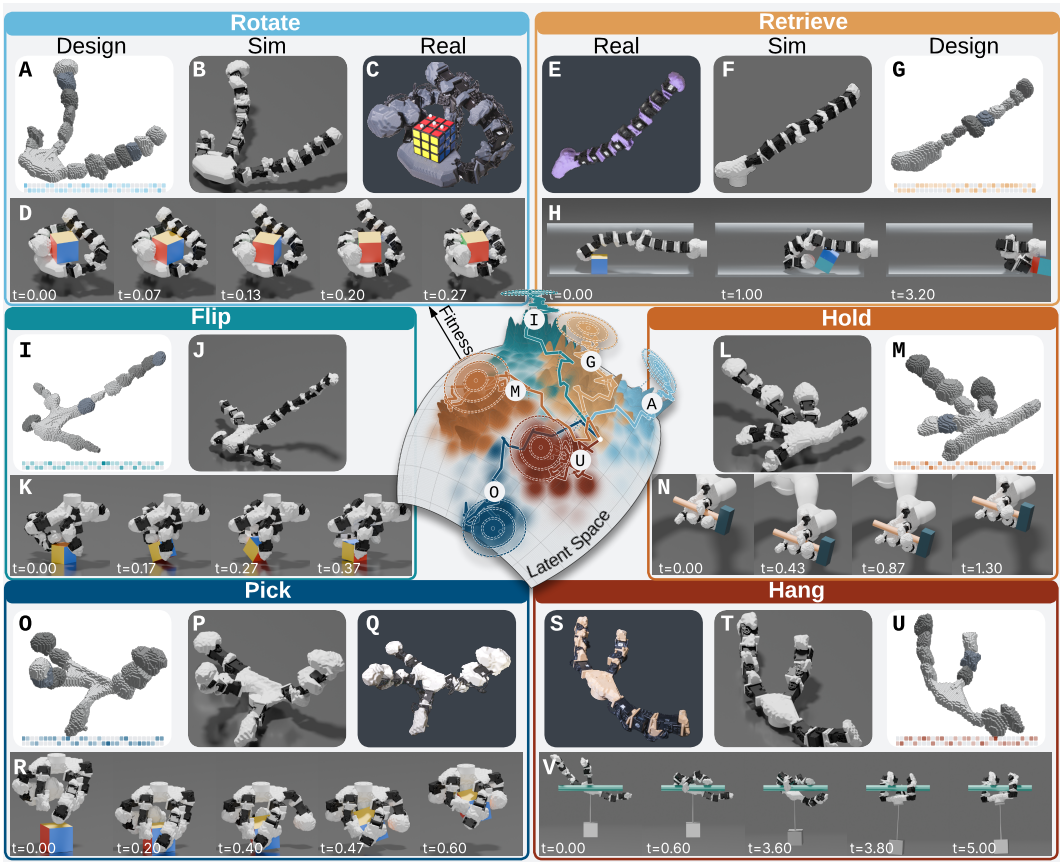

Figure 1: **Evolving freeform manipulators.** Under different selection pressures and environmental conditions, unique morphologies emerged to rotate (**A-D**), retrieve (**E-H**), flip (**I-K**), hold (**L-N**), pick up (**O-R**), and hang from (**S-V**) objects. Evolution operates in a latent space (multicolored surface, center inset) of genotypes, which are visualized as colored barcodes below each decoded design (A,G,I,M,O,U). The resulting robot was trained in simulation (D,H,K,N,R,V), and in some cases built and tested in the real world (C,E,Q,S).

3D surface curvature of the palm and the relative bend angle of each finger, which improved grasp stability. However, their anthropomorphic design template added more assumptions than it relaxed.

Here we explore the open-ended evolution of novel dexterous robots with freeform geometry and articulation (Fig. 1). For fair comparison, we use the same actuators as the most relevant prior work (Fay et al., 2026; Mirzaee et al., 2026). Unlike prior work however, we do not assume fingers about a palm and do not constrain the position and orientation of actuators or the geometry of the segments between them. After jointly co-optimizing the robot's controller and complex morphology from scratch in simulation, we automatically convert the design into a manufacturable blueprint (Guo et al., 2026), which we directly follow to print, assemble, and test the robot in the real world.

To support the evolution of freeform manipulators, we encode the enormous, combinatorial space of possible morphologies—morphospace (Raup, 1966)—into a compressed latent genome (Mitchell & Cheney, 2025). The related work mentioned above (Fay et al., 2026; Mirzaee et al., 2026), by contrast, operated within manually devised parameterizations of design space, as did other investigations of alternative hand designs (Gilday et al., 2025). In locomotion studies it is becoming increasingly popular to use a variational autoencoder (VAE; Kingma & Welling (2014)) to construct a latent embedding of design space (Hu et al., 2023; Li et al., 2025; Yu et al., 2026; Wang et al., 2026; Guo et al., 2026). Similarly, Wei et al. (2026) used a VAE to operate kinematically distinct anthropomorphic hands with a unified control policy. However, VAEs can unintentionally obstruct evolution by mapping nearly identical designs (phenotypes) to very distant latent codes (genotypes). Here, we use contrastive learning (Chen et al., 2020; Wang & Isola, 2020; Kirchoff et al., 2024)

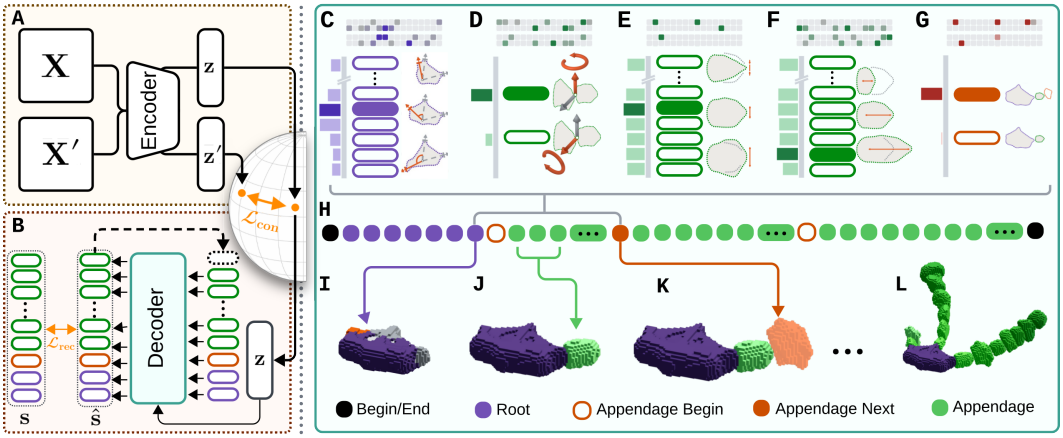

Figure 2: **Latent genome and developmental decoder.** Morphologies were embedded on a latent manifold (**A**) and autoregressively decoded back to morphospace (**B**) through a developmental program (**C-G**) that translates design tokens (**H**) into articulated bodies (**I-L**). Dimensions of an evolved genotype (barcodes in C-G) most strongly correlated with each design token are highlighted in the corresponding color at each growth step. The decoder outputs a probability distribution (vertical histogram in C) for each token that determines how new structures are layered on top of earlier structures, and their 3D geometry (E,F) and articulation (D,G).

to project morphospace onto a genetic hypersphere, which yields a phenotypically coherent—and more easily searchable—latent manifold.

Another limitation of prior genetic embeddings has been the decoder, the genotype-to-phenotype map (Pigliucci, 2010; Mitchell & Cheney, 2025). This mapping was either too constrained, e.g. limiting phenotypes to stick figures (Hu et al., 2023; Yu et al., 2026), or too unconstrained, i.e. with the capacity to produce nonsensical anatomies, such as those with floating, unconnected body parts (Li et al., 2025; Guo et al., 2026). Here we introduce an autoregressive developmental model that iteratively grows one body part after another, ensuring validity at each step. It does so by translating a sequence of design tokens (specified by the latent genotype) into morphological features, with different tokens determining how, when and where to layer new structures on top of older ones, and the shape and articulation of each part (Fig. 2).

TL;DR: In short, here we show for the first time de novo manipulator design. Key to our approach is a well-structured latent space for freeform morphological evolution and a pipeline to instantly CADify decoded designs, obviating the need for standardized body modules. The results are more diverse, complex, and dexterous than other evolved robots reported in the literature to date.

2 Methods

In this section we describe how designs were indexed (Sect. 2.1), encoded and decoded (Sect. 2.2), translated into to a realizable form (Sect. 2.3), trained (Sect. 2.4), evolved (Sect. 2.5), manufactured and tested in reality (Sect. 2.6).

2.1 Morphospace

Designs were parameterized by a root (Fig. 2I) and up to five appendages (kinematic chains) that may grow outward from any point along the edge of the root (Fig. 2L). Each design contained up to 25 joints. Its root was parameterized by 39 variables (Fig. 9) and each segments along an appendage was parameterized by 12 variables (Fig. 10). A compiler converts these variables into a body plan, voxelized on a 128-by-128-by-128 grid with two channels. The first channel indicates which voxels belong to the root; non-root voxels were set to -1. The second channel indicates topological distance from the root using relative coordinates, where voxels within proximate and distal segments were indexed as 0 and 1, respectively. Each intermediate segment, k, along an appendage with K segments was indexed as $k/(K-1)$. Non-appendage voxels in the second were set to -1. Finally, a kinematic graph stores the relative location and hinge axis of each joint.

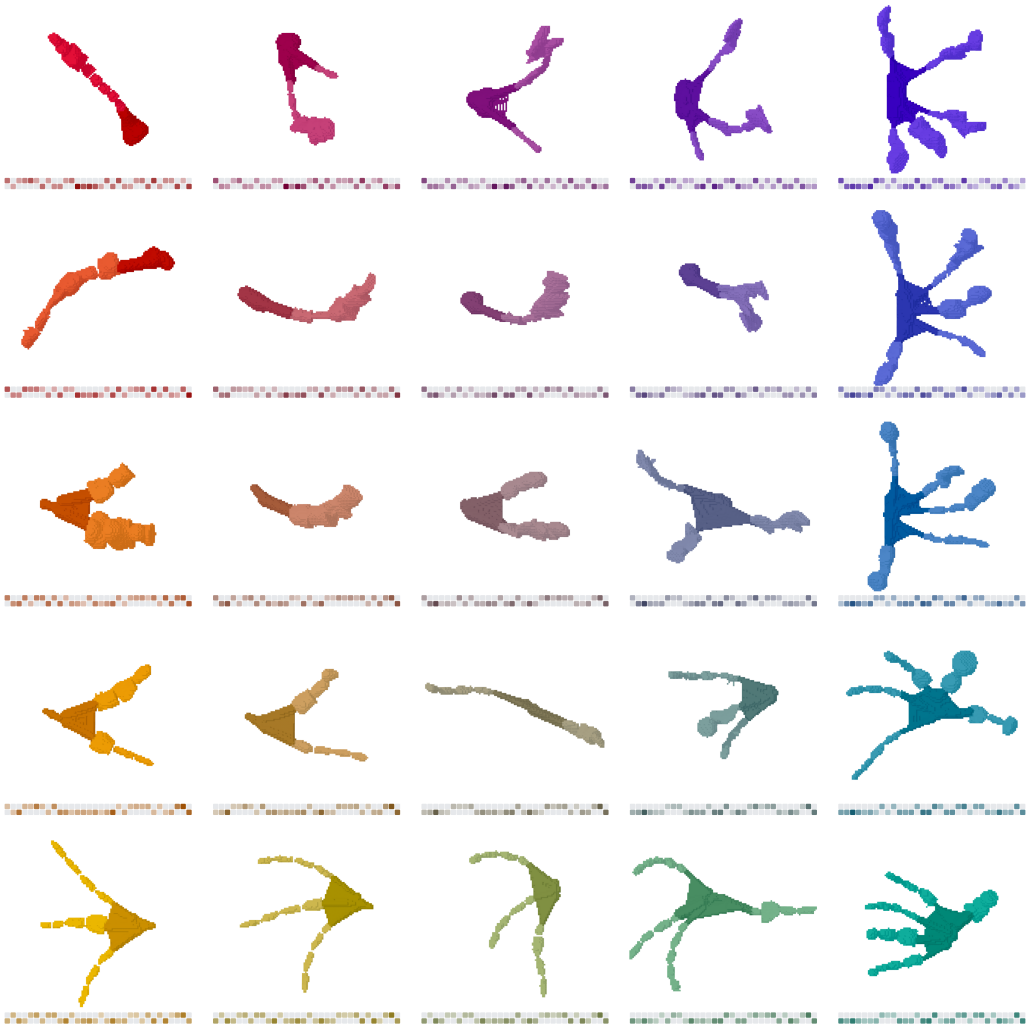

Figure 3: **Genetic interpolation between four evolved manipulators.** Spherical linear interpolation (slerp) between a retriever (top left), a rotator (top right), a picker (bottom right), and a hanger (bottom left). The latent genotype, $z_i \in \mathbb{S}^{31}$, below each decoded design is visualized as a double barcode. The upper and lower rows of the barcode show the positive and negative component of each latent dimension, respectively, with darker colors indicating greater magnitude.

2.2 Latent Space (The Genome)

We used contrastive learning (Chen et al., 2020) to create a latent genome in which similar morphologies have similar genes and dissimilar morphologies have genes that are farther away in the latent space. The voxel grid was encoded by a 3D convolutional network (Maturana & Scherer, 2015) and the kinematic tree was encoded by a graph network (Ying et al., 2018). The two inputs streams were combined into a single latent vector z and normalized onto the surface of a hypersphere (Wang & Isola, 2020). More information about the encoder can be found in Appx. B.2.

The decoder maps a latent genotype to a sequence of design tokens (Fig. 2H) that specify the morphology. Continuous morphological parameters were quantized into multiple tokens. An autoregressive transformer (Vaswani et al., 2017) conditioned on the latent genotype vector predicts the sequence one token at a time (Fig. 2B). During body generation, a grammar masks erroneous tokens such that every body is decoded from a complete and valid set of morphological parameters. More information about the decoder can be found in Appx. B.3.

Two million example body plans were generated and each was converted into a ground-truth token sequence, s. (See Appx. B.1 for details.) This synthetic data was used to optimize the encoder and

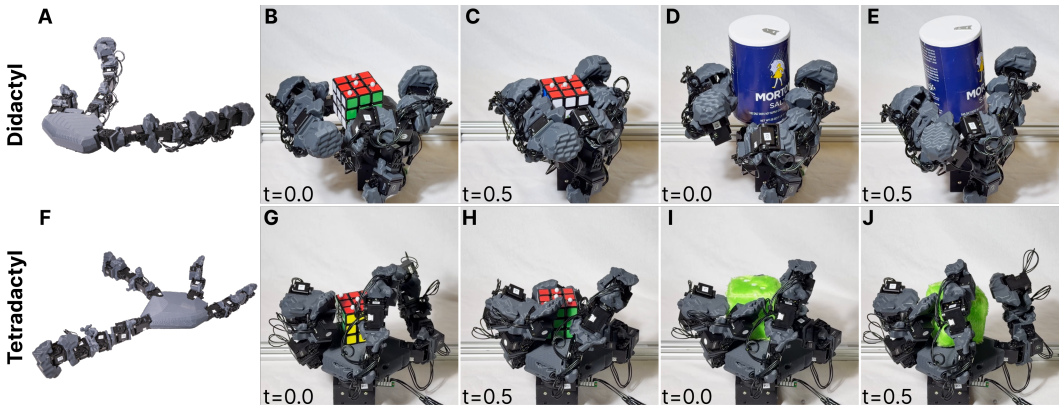

Figure 4: **Two evolved rotators.** In our main experiment, robots were evolved to rotate diverse objects. The best was essentially didactyl (**A-E**). It possessed two functional appendages and a third, very short vestigial appendage that did not contribute to object rotation. Another highly fit evolved design which we built was tetradactyl (**F-J**); all four of its appendages were utilized.

decoder simultaneously with the loss function $\mathcal{L} = \mathcal{L}_{\text{con}} + \lambda \mathcal{L}_{\text{rec}}$. The contrastive term, $\mathcal{L}_{\text{con}}$, uses NT-Xent loss (Chen et al., 2020) to update the encoder. For each example design, z, a morphologically similar counterpart, z', was generated through small perturbations and transformations to the original body plan, rescaling, reorienting, repositioning, or slightly deforming it on the voxel grid or removing distal segments. These "positive pairs" (z and z') are brought closer together and farther away from other examples ("negative pairs"). The reconstruction term, $\mathcal{L}_{\text{rec}}$, uses next-token cross-entropy to optimize the decoder. See Appx. B.4 for more details.

2.3 Translating Designs into Robots

After a design was decoded from latent space to voxel space, it was automatically converted into a manufacturable form (Guo et al., 2026). A printable mesh was extracted (Lorensen & Cline, 1987) for each body part and servos were placed at every joint. If all of the motors did not fit into the body, the design was rejected. Body parts were carved out slightly to accept the hardware, with cavities for each motor and its mounting bracket, and channels for screws and cables. To prevent self-collision, each joint was swept through its range of motion and any material of either segment that would collide was removed. This sweep stopped before a segment lost too much of its volume or was broken apart into disconnected pieces, and this safe range became the joint limit. Finally, the motors and their brackets were merged into the body model together with their masses. The resulting, assembly-ready robot was exported as a URDF file for simulation and as set of meshes for printing.

2.4 Learning to Manipulate Diverse Objects

Robot were trained in Isaac Lab (Mittal et al., 2025). Before policy training, a neural pose was found in simulation by optimizing joint angles such that enough segments get close to the object without self-collisions. If and when a valid neutral pose could not be found, the design was discarded. A control policy was then trained from scratch for each design with proximal policy optimization (PPO; Schulman et al. (2017)) across thousands of parallel environments. The policy observes joint angles, previous commands, and the state of the object, and outputs a target position for every joint. After training, the optimized policy was replayed without exploration noise for a fixed window, and fitness was taken to be average reward achieved across environments. See Appx. C for details.

2.5 Evolving Novel Robots

Following Li et al. (2025), robots were designed by covariance matrix adaptation evolutionary strategies (CMA-ES; Hansen & Ostermeier (2001)). Briefly, genotypes were sampled from a normal distribution in the latent space. During evolution, the distribution is pulled toward sampled latents with the highest fitness, stretched along the most promising dimensions of covariation, and com-

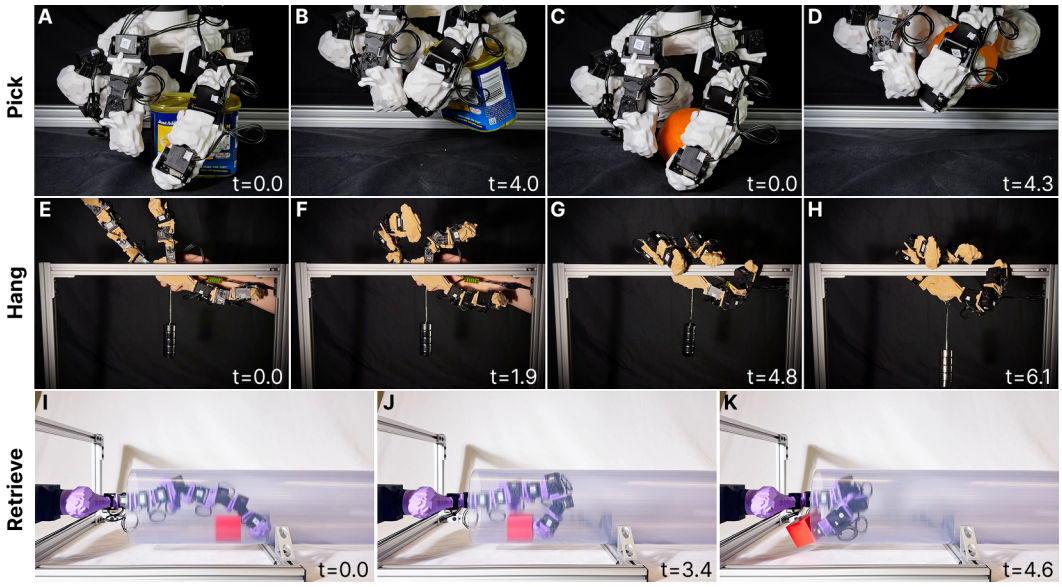

Figure 5: **An evolved picker, hanger, and retriever.** We evolved, trained, and realized robots that pick objects off a surface (**A-D**), hang from a horizontal bar with a weight suspended below them (**E-H**), and retrieve objects from within a narrow pipe (**I-K**).

pressed along others. Depending on how far the distribution mean is moving across the latent space, the step size is either increased or decreased, dynamically. For more details see Hansen (2016). Hyperparameters can be found in Table 14.

2.6 Reality Check

Winning designs were fabricated directly from their automatically generated blueprints. Body parts were 3D-printed using PLA (polylactic acid) filament and the Dynamixel servos were inserted and attached by screws into heat inserts. (A complete bill of materials can be found in Table. 17.) During evolution the policy observes privileged information, i.e. the state of the object. The policy of the winning design was distilled into a proprioceptive controller that observes only the joint positions and previous commands, and was finetuned under domain randomization (Table 15). The finetuned policy was deployed zero-shot on the physical robot at the same control rate as used during training in simulation (30 Hz). See Appx. C for details.

3 Results

We being this section by examining the latent genome (Sect. 3.1). Next we consider six tasks: rotating (Sect. 3.2), retrieving (Sect. 3.5), flipping (Sect. 3.6), and picking (Sect. 3.3) objects, holding a tool (hammer; Sect. 3.7), and hanging from the environment (a bar; Sect. 3.4). We evolved specialists for each task and generalists capable of multiple tasks (Sect. 3.8). Finally, we conduct an ablation study to determine how different genetic encodings may help or hinder evolution (Sect. 3.9).

Although it is difficult to decompose animal behavior into discrete "tasks", the capacity to retrieve, pick, and hold objects (e.g. food) is clearly important for survival. Likewise, gripping and hanging rather than falling from objects (e.g. a tree branch) can be a matter of life or death. More sophisticated behaviors require reorienting (rotating, flipping) the object. It can be equally important—and very interesting—for robots to evolve anatomies that support these vital behaviors.

3.1 Genomics

In order to evaluate the smoothness and coherence of the latent genome, we generated a test set of 2,000 unique body plans that were not used for training and sequenced their latent genotypes. For each design we created an augmented copy and measured how often the copy and original were

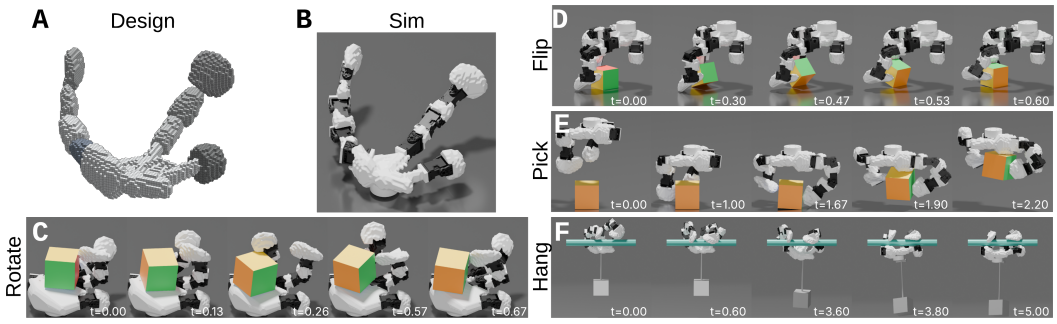

Figure 6: **An evolved multitasker.** We also evolved a generalist to carry out several tasks. The winning design (**A,B**) could rotate (**C**), flip (**D**), pick up (**E**), and hang (**F**) from objects in simulation.

nearest genetic neighbors. We refer to this as "recall" (in Table 11) and found that indeed all original-modified pairs were closer together than to any other design, a recall of 100%. We also measured, on a separate test set of 4,500 designs spanning 17 topologies, how often the kinematic structure of each design matched the most common structure among its 10 nearest genetic neighbors, a statistic we term "topology 10-NN accuracy", and found it to be 92%. Next we created amputated copies of the test designs with up to four distal segments removed and measured the correlation (Spearman's) between how many segments were amputated and the genetic distance from the original (edit-distance correlation; $\rho = 0.97$). Finally, we encoded and decoded designs and measured how often the decoded design had the expected number of appendages (appendage accuracy; 100%) and joints (joint accuracy; 96%), as well as how often the decoded design stayed within the voxel grid with nonoverlapping appendages (validity; 82%). See Table 11 for complete analysis.

3.2 Rotate

We first considered "in hand" rotation, a standard manipulation benchmark (Handa et al., 2023; Yin et al., 2023; Shaw et al., 2023; Fay et al., 2026). The robot was mounted to the environment with its root facing upward to accept an object. The objective is to vertically rotate the object placed on top of the root, i.e. in its "palm" (Fig. 1A-D). Fitness was the net rotation rate of the object over a 30 second evaluation period. Five independent evolutionary trials were conducted and compared against an anthropomorphic baseline (Shaw et al., 2023) as well as Fay et al. (2026), who optimized manipulators for the same task and which, until now, represented the state-of-the-art (Fig. 7). More details about the task environment and reward function used for policy training can be found in Appx. C.1.

We realized two evolved designs (Fig. 4). The policy learned during evolution was distilled so as to no longer rely on privileged (object state) information and finetuned under domain randomization (Table 15). After finetuning, the evolved didactyl robot rotated objects at 3.66 rad/s in simulation, compared to 2.09 rad/s on the anthropomorphic hand (Shaw et al., 2023) and 3.3 rad/s reported by Fay et al. (2026) using privileged information. Deployed zero-shot in the real world, the didactyl robot rotated a diverse set of previously unseen objects (weighing from 16 to 687 g) at 1.39 rad/s on average, almost twice as fast as the best design reported by Fay et al. (2026),which achieved 0.74 rad/s on average. The evolved tetradactyl robot rotated the five cubes at 1.78 rad/s on average, but could not rotate the other test objects. After the same training process, the anthropomorphic hand (Shaw et al., 2023) failed to rotate any of the previously unseen test objects. Even with a specialized policy trained to rotate a cube, the hand was no better than the evolved designs at rotating cubes. See Table 3 for details.

3.3 Pick

Robots were mounted above a tabletop with their root facing downward and evolved to pick up objects (Fig. 1O-R). Highly fit robots pick the object and keep it from falling in the under external perturbations (see Appx. C.2 for more details). We conducted a single evolutionary trial; the winning design picked up and held onto objects 99% of the time. We realized this design (Figs. 1Q and 5A-D), and deployed its policy zero-shot after the same distillation and finetuning procedure described

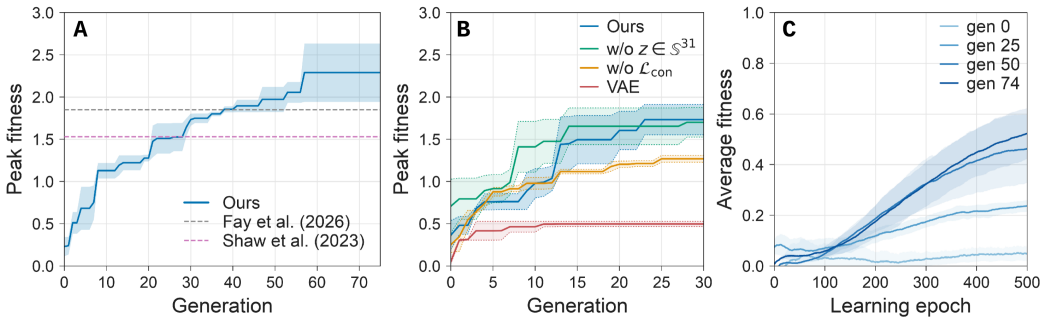

Figure 7: **The evolution of rotators** (fitness measured during evolution; results after finetuning are in Sect. 3.2). Our approach consistently yields novel robots that rotate objects faster than those produced by related prior work, where our full model was trained with the 2 million sample dataset (**A**) and the proposed genetic representation increased evolvability beyond what could be achieved with a VAE, both trained in the smaller 200,000 sample dataset (**B**). In the full setting, morphologies evolved that made learning much easier (**C**). Shaded regions indicate standard error of the mean. Here results are reported before policy distillation and finetuning.

above. Across 24 previously unseen objects, it succeeded in eight out of ten attempts on average, and picked up every object at least once except the heaviest, a 687 g salt container (Table 6).

3.4 HANG

Robots were positioned near a horizontal bar and evolved to wrap around it such that, when released, the body hangs from the bar with a 500 g weight suspended below it (Fig. 1S-V). Fitness was based on how long the robot stayed on the bar, discounted by the motor torque expended during an evaluation period of 20 sec (see Appx. C.3 for details). The winning design held on for the entire evaluation period in 99.9% of training episodes, using half the torque of the best designs found earlier in evolution. We realized this design (Fig. 1S); deployed zero-shot with a finetuned policy, it gripped and hung from a bar under a 500 g load (Fig. 5E-H).

3.5 RETRIEVE

Robots were mounted at the opening of a narrow horizontal pipe containing a small cube (Fig. 1E-H). Robots were evolved to reach into the pipe and pull the cube out (see Appx. C.4 for details). The winning design, which was successful in every training episode, was realized, finetuned, and deployed zero-shot as described above. It was equally successful in reality (Fig. 5I-K).

3.6 FLIP

Robots were mounted above a table and evolved to flip a cube over, again and again, in a randomly chosen direction, as quickly as possible. Each flip moves the cube one edge length along the table, so the mounted robot must also pull the cube back within reach to keep flipping it. Fitness was the number of full rotations of the cube over a 60 sec evaluation period (see Appx. C.5 for details). The winning design (Fig. 1I-K) achieved 7.75 full rotations in 60 sec.

3.7 HOLD

The manipulator was mounted on the end of an arm and evolved to hold a hammer by its handle as arm swings the hammer up and down (Fig. 1N). Fitness was how little the hammer moved relative to its mounted root during arm swinging and under external perturbations that pulled the hammer in random directions (see Appx. C.6 for details). The winning design (Fig. 1L) dropped the hammer in just 3% of episodes. It was the only winning design with a pentadactyl morphology that evolved across the six task environments we tested.

3.8 MULTITASKING

Finally, we evolved manipulators to perform five of the six tasks described above (all but retrieve). Fitness was based on performance across all five tasks (see Appx. C.7 for details). Early in evolution, the best design in the population had mastered object rotation and flipping and hanging from the bar. Later, the best design evolved the ability to pick and hold objects but as it did it got worse at rotating them. The final, winning design (Fig. 6) flipped and hung as well as the specialists evolved for those tasks alone, and lifted almost every object but held on under the pulling force for only a quarter of the evaluation window; it rotated objects slowly and dropped the hammer.

3.9 ABLATIONS

We compared our latent embedding against three alternatives (Table 11). The first two were otherwise equivalent but lacked the spherical projection (w/o $z \in \mathbb{S}^{31}$) or contrastive loss (w/o $\mathcal{L}_{\text{con}}$). The third alternative was a VAE (Hu et al., 2023; Li et al., 2025; Mitchell & Cheney, 2025; Yu et al., 2026). All were trained (ours was retrained) with a smaller set of 200,000 example body plans and share the same encoder and decoder architecture. Three evolutionary trials were conducted for the task of object rotation (Fig. 7B). Without contrastive learning, winning designs were significantly worse. The VAE suffered posterior collapse, rarely reconstructing the correct number of appendages and joints, and so evolution quickly fell into a local optimum. See Appx. D for complete analysis.

4 DISCUSSION

In this paper, we evolved morphologically complex robots to flexibly handle diverse objects across a wide array of tasks (Fig. 1). The resulting robots were shown to be more dexterous than those generated by otherwise equivalent state-of-the-art baselines (Fig. 7) which operate within a more restrictive design space (Fay et al., 2026) or assume an anthropomoric layout (Shaw et al., 2023). We used contrastive learning to map the combinatorial space of possible designs onto a spherical latent genome in which similar body plans were genetically closer than dissimilar bodies (Fig. 22) and an autoregressive decoder to retain finegrain morphological features and ensure validity (Fig. 2). This genetic representation significantly increased the evolvability of robots beyond the best known alternative (Fig. 7), which relied on a simple but effective VAE framework of reconstruction and regularization (Mitchell & Cheney, 2025). Although it sounds plausible in retrospect that contrastive learning would yield a latent space that aids morphological evolution, this result was quite surprising; distinct designs could have been inadvertently clustered on disjoint genetic islands, obstructing the evolution of novel forms. On the contrary, we found contrastive loss to be essential; without it, manipulators were significantly less evolvable (Fig. 7). Kirchoff et al. (2024) used a similar approach to design drugs based on their molecular graphs but they did not test if the sphere projection mattered. Anecdotally, this projection slightly increased evolvability on average, though this comparison was not statistically significant and the dis/advantages for evolving robots remain unclear.

There were, of course, several important limitations of these experiments. The design space was limited to a one material (PLA), one mode of actuation (in-joint servos), and a single sensor modality (proprioception); evolution and learning were conducted entirely in simulation without feedback from the real environment; the desired behaviors were cognitively simple and could be executed without prospection or planning; and the robots themselves were sessile, they could only handle nearby objects—everything else was out of reach. Future work will study the evolution of more thoroughly sensorized, multi-material agents that move themselves from place to place, interacting with objects along the way and learning from this real world experience.

It is also important to note the necessity of simulation; without it, the evolution remains prohibitively expensive for all but the simplest robots (Brodbeck et al., 2015). An interesting open question is therefore whether and how relevant morphology-conditioned dynamics that are mismodeled by or missing from the simulator may be learned on the fly with each deployed robot (O'Connell et al., 2022; Matthews & Kriegman, 2026).

Despite these and many other limitations, the dexterous robots in this paper were not limited by human preconceptions about what a manipulator should look like; rather, they were free (more or less) to evolve whatever structural and architectural features they needed to solve the task at hand.

REFERENCES

Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Proceedings of The Conference on Robot Learning (CoRL)*, pp. 3453–3467, 2023.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Xueqian Bai, Nicklas Hansen, Adabhav Singh, Michael T Tolley, Yan Duan, Pieter Abbeel, Xiaolong Wang, and Sha Yi. Learning to design soft hands using reward models. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2026.

Luzius Brodbeck, Simon Hauser, and Fumiya Iida. Morphological evolution of physical robots through model-free phenotype development. *PloS ONE*, 10(6):e0128444, 2015.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1597–1607, 2020.

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, 2018.

Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S Srinivasa, Michael Erdmann, Matthew T Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 1578–1585, 2014.

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Kieran Gilday, Chapa Sirithunge, Fumiya Iida, and Josie Hughes. Embodied manipulation with past and future morphologies through an open parametric hand design. *Science Robotics*, 10(102): eads6437, 2025.

Zihan Guo, Muhan Li, Shuzhe Zhang, and Sam Kriegman. Creating manufacturable blueprints for coarse-grained virtual robots. *arXiv preprint arXiv:2603.13582*, 2026.

Huy Ha, Shubham Agrawal, and Shuran Song. Fit2Form: 3D generative model for robot gripper form design. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 5977–5984, 2023.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 770–778, 2016.

John Healy and Leland McInnes. Uniform manifold approximation and projection. *Nature Reviews Methods Primers*, 4(1):82, 2024.

Jonathan Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2012.

Jiaheng Hu, Julian Whitman, and Howie Choset. GLSO: Grammar-guided latent space optimization for sample-efficient robot design automation. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 1321–1331, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Kathryn E Kirchoff, Travis Maxfield, Alexander Tropsha, and Shawn M Gomez. Salsa: Semantically-aware latent space autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13211–13219, 2024.

Sam Kriegman, Douglas Blackiston, Michael Levin, and Josh Bongard. A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences*, 117(4): 1853–1859, 2020.

Muhan Li, Lingji Kong, and Sam Kriegman. Generating freeform endoskeletal robots. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974, 2000.

Chih-Hsing Liu, Ta-Lun Chen, Chen-Hua Chiu, Mao-Cheng Hsu, Yang Chen, Tzu-Yang Pai, Wei-Geng Peng, and Yen-Pin Chiang. Optimal design of a soft robotic gripper for grasping unknown objects. *Soft Robotics*, 5(4):452–465, 2018.

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

David Matthews and Sam Kriegman. Zero-shot robot design with residual physics. In *Proceedings of the Conference on Artificial Life (ALife)*, 2026.

David Matthews, Andrew Spielberg, Daniela Rus, Sam Kriegman, and Josh Bongard. Efficient automatic design of robots. *Proceedings of the National Academy of Sciences*, 120(41):e2305180120, 2023.

Daniel Maturana and Sebastian Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015.

Mohammad Amin Mirzaee, Harsh Gupta, and Wenzhen Yuan. Function-based parametric co-design optimization of dexterous hands. *arXiv preprint arXiv:2604.27557*, 2026.

Kevin J Mitchell and Nick Cheney. The genomic code: The genome instantiates a generative model of the organism. *Trends in Genetics*, 41:462–479, 2025.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Stefano Nolfi and Dario Floreano. Synthesis of autonomous robots through evolution. *Trends in Cognitive Sciences*, 6(1):31–37, 2002.

Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022.

Massimo Pigliucci. Genotype–phenotype mapping and the end of the 'genes as blueprint' metaphor. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1540):557–566, 2010.

David M Raup. Geometric Analysis of Shell Coiling: General Problems. *Journal of Paleontology*, pp. 1178–1190, 1966.

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

Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. LEAP Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning. *Robotics: Science and Systems (RSS)*, 2023.

Luke Strgar and Sam Kriegman. Accelerated co-design of robots through morphological pretraining. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Luke Strgar, David Matthews, Tyler Hummer, and Sam Kriegman. Evolution and learning in differentiable robots. In *Robotics: Science and Systems (RSS)*, 2024.

Shoko Sugasawa, Barbara Webb, and Susan D Healy. Object manipulation without hands. *Proceedings of the Royal Society B: Biological Sciences*, 288(1947):20203184, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 3868–3879, 2023.

Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. *arXiv preprint arXiv:2210.02697*, 2022.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 9929–9939, 2020.

Yibin Wang, Muhan Li, Zihan Guo, and Sam Kriegman. ECo-MoE: Embodiment-Conditioned Mixture of Experts Increases the Evolvability of Robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Zhenyu Wei, Yunchao Yao, and Mingyu Ding. One hand to rule them all: Canonical representations for unified dexterous manipulation. *arXiv preprint arXiv:2602.16712*, 2026.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An end-to-end differentiable framework for contact-aware robot design. In *Robotics: Science and Systems (RSS)*, 2021.

Xiaomeng Xu, Huy Ha, and Shuran Song. Dynamics-guided diffusion model for sensor-less robot manipulator design. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 4446–4462, 2025.

Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without seeing: Towards in-hand dexterity through touch. *Robotics: Science and Systems (RSS)*, 2023.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Chen Yu, David Matthews, Jingxian Wang, Jing Gu, Douglas Blackiston, Michael Rubenstein, and Sam Kriegman. Agile legged locomotion in reconfigurable modular robots. *Proceedings of the National Academy of Sciences*, 123(10):e2519129123, 2026.

Haoqi Yuan, Bohan Zhou, Yuhui Fu, and Zongqing Lu. Cross-embodiment dexterous grasping with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 81413–81434, 2025.

Wenxuan Zhou and David Held. Learning to grasp the ungraspable with emergent extrinsic dexterity. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 150–160, 2023.

APPENDIX

A Design space **14**
A.1 The root . . . 14
A.2 The appendage . . . 17

B Latent Genome Details **19**
B.1 Synthetic data generation . . . 19
B.2 Encoder . . . 19
B.3 Decoder . . . 19
B.4 Training . . . 19
B.5 Evolution . . . 21

C Task Details **21**
C.1 Rotate . . . 21
C.2 Pick . . . 23
C.3 Hang . . . 24
C.4 Retrieve . . . 24
C.5 Flip . . . 25
C.6 Hold . . . 25
C.7 Multitask . . . 26
C.8 Real-World Deployment . . . 27

D Ablation analysis **33**
D.1 Validity . . . 33
D.2 Smoothness . . . 34

E Hyperparameters and BOM **35**

A DESIGN SPACE

A design is made of one root and up to five appendages (Fig. 8). The root is the central body. It has three components: the hub, a flat elliptical disc at its centre; five spokes that radiate from the hub, each either built or not; and the webs, which close the gap between neighbouring built spokes. Its 39 parameters (Table 1) set the size and shape of the hub and the direction, length, tilt and thickness of each spoke, so that the root can range from a thin flat frame to a thick cupped body, with its appendages starting at any angle around it (Appx. A.1).

One appendage grows from the tip of each built spoke. An appendage is a chain of segments that never branches, each segment joined to the one before it by a joint. Each segment, with its joint, has 12 parameters, which set how its joint turns and where it points, and the length, cross-section and surface of the segment itself (Appx. A.2). A design has at most 25 segments, and so at most 25 joints.

A design is turned into a voxelized body in three steps. First, the parameters place a skeleton: the hub, each spoke from its foot to its tip, and each chain of segments, every segment starting where the one before it ends. Second, every part of the skeleton is given a solid shape, and the shapes that make up one part are blended smoothly into each other. Third, the body is voxelized on a 128^3 grid, with every voxel labelled by the part it belongs to.

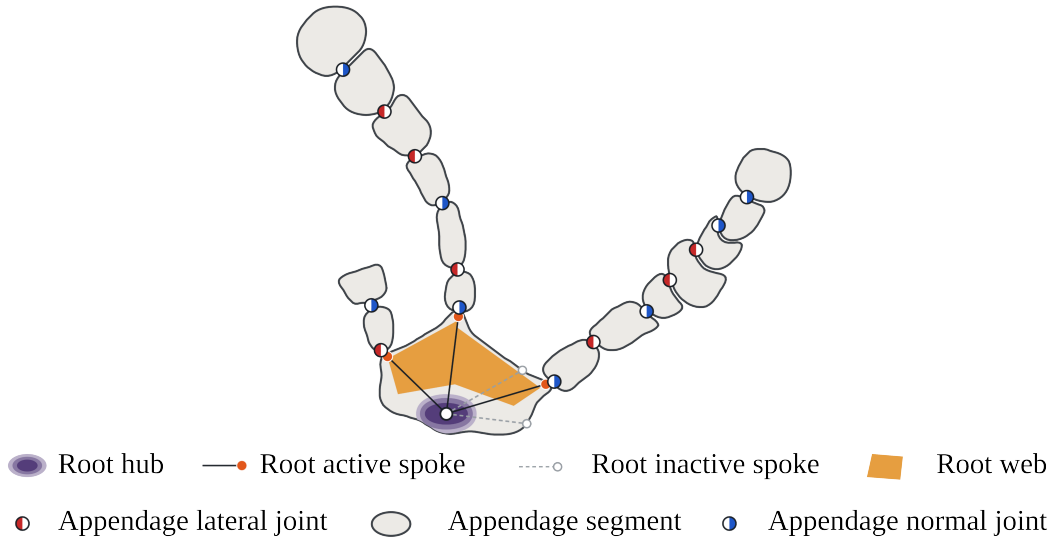

Figure 8: **One design.** The root is the central body: the hub at its centre and five spokes around it. Three spokes are active and so built, each ending at the tip from which an appendage grows; two are inactive and not built. A web fills the space between neighbouring built spokes. Each appendage is a chain of segments, and each joint between two segments is marked by the hinge axis it takes, lateral or normal. Segments are drawn slightly apart so that every joint shows. The parameters of the root and of an appendage are shown one by one in Figs. 9 and 10.

A.1 THE ROOT

The root has three parts: the hub at its centre, the five spokes around it, and the webs between neighbouring built spokes. The letter after each name is its panel in Fig. 9.

The hub has four parameters.

Root hub aspect	(A)	stretches the hub from a disc into an oval: round at 1, and three times as long as wide at 3.
Root hub ring fraction	(B)	sets where the feet of the spokes sit on the hub, from halfway out (0.5) to the rim (1.0).

Root hub radius (C) sets the size of the hub, and with it how far apart the feet of the spokes are.

Root hub infill (D) sets how much material the hub carries, from a thin frame between the feet of the spokes to a thick solid body.

Each of the five spokes, with its web, has seven parameters.

Root spoke azimuth (E) sets where around the hub the spoke comes out, and so where its appendage starts. Spokes with similar azimuths sit side by side; spokes half a turn apart face each other across the hub.

Root spoke activation (F) decides whether the spoke is built: it is built if its activation is greater than 0, and such a spoke is called active; otherwise it is inactive and not built. If no activation is greater than 0, the spoke with the largest activation is built, so a design always has at least one appendage.

Root spoke elevation (G) tilts the spoke out of the plane of the hub, so that the appendage starts above or below the hub. Tilting all spokes the same way cups the root.

Root spoke radius (H) sets how thick the spoke is. A spoke is thickest at its foot and narrows toward its tip.

Root spoke length (I) sets how far from the centre of the hub the tip of the spoke lies, and so how far out the appendage starts. The length of a spoke, from foot to tip, is kept between 3.5 and 16 cm.

Root web radius (J) sets how thick the web is. A thin web leaves the spokes distinct; a thick web merges them into one plate.

Root web reach (K) sets how far along the spoke the web extends toward the tip. The web between two spokes goes only as far as the smaller of their two values.

A spoke that is not built still places its foot on the hub, so its azimuth and elevation, and where its foot sits, still shape the hub, while its length, radius and web reach have no effect.

Table 1: Design-space parameters. The root is described by 39 values and each segment of an appendage, with its joint, by 12. Rows follow the panels of Figs. 9 and 10. Bins is the number of bins used to write the parameter as a token (Appx. B.3).

Parameter	Range	Bins	Parameter	Range	Bins
Root hub			*Appendage joint*		
root hub aspect	0.8 to 3.0	12	appendage joint yaw bend	-0.24 to 0.24 rad	32
root hub ring fraction	0.5 to 1.0	12	appendage joint hinge axis	lateral or normal	2
root hub radius	0.02 to 0.05 m	12	appendage joint pitch bend	-0.26 to 0.26 rad	32
root hub infill	1.0 to 4.0	8	appendage joint length	0.03 to 0.06 m	32
Root spoke and web			*Appendage segment*		
root spoke azimuth	$-\pi$ to π rad	128	appendage segment asymmetry	0.0 to 2.0	8
root spoke activation	built if > 0	2	appendage segment gain	0.55 to 1.20	8
root spoke elevation	-1 to 1	64	appendage segment lobe	0.40 to 1.10	8
root spoke radius	0.01 to 0.03 m	8	appendage segment phase	$-\pi$ to π rad	16
root spoke length	0.06 to 0.10 m	32	appendage segment width	0.01 to 0.04 m	12
root web radius	0.6 to 1.2	8	appendage segment depth	0.01 to 0.04 m	24
root web reach	0.7 to 1.0	8	appendage segment taper	0.10 to 1.5	12
			appendage segment twist	-1.20 to 1.20 rad	8

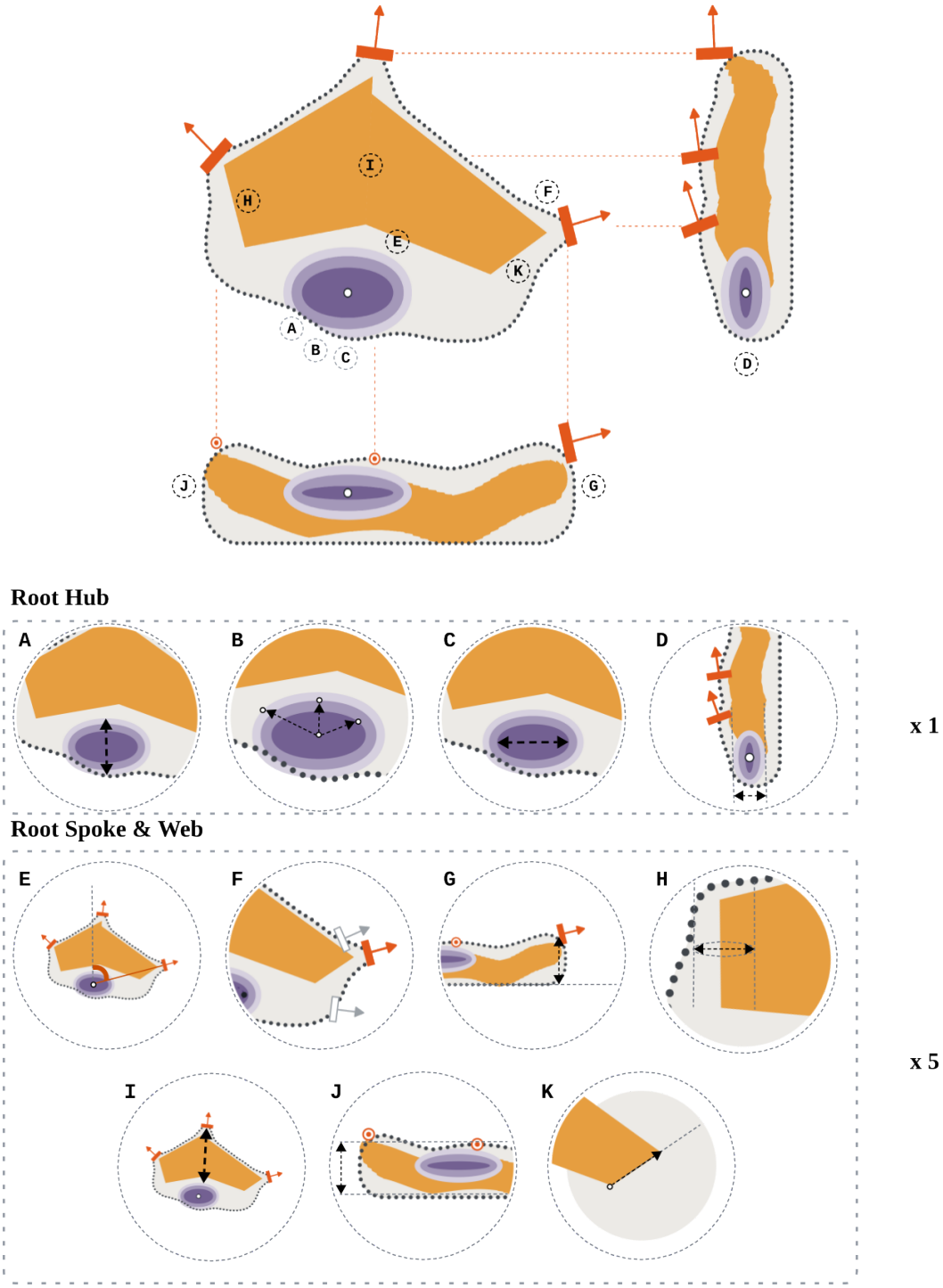

Figure 9: **The root.** Top: one root from the design space, seen from above (left), from the end (right) and from the side (bottom). The purple ellipse is the hub. An orange bar with an arrow marks the tip of each spoke that is built. **A–K**: one parameter each, cut and enlarged from the views above; the small circle on the views marks where each was cut. Root hub: A **Root hub aspect**, B **Root hub ring fraction**, C **Root hub radius**, D **Root hub infill**. Root spoke and web: E **Root spoke azimuth**, F **Root spoke activation**, G **Root spoke elevation**, H **Root spoke radius**, I **Root spoke length**, J **Root web radius**, K **Root web reach**.

A.2 THE APPENDAGE

The appendage has two parts: its segments, the rigid pieces of the chain, and its joints, one between each segment and the one before it. The letter after each name is its panel in Fig. 10.

Each joint has four parameters.

Appendage joint yaw bend	(A)	turns the segment toward its side, relative to the segment before it, when the joint is at rest.
Appendage joint hinge axis	(B)	sets which way the axis of the joint runs: across the width of the previous segment (lateral), so that the segment folds toward its face, or across its depth (normal), so that the segment swings toward its side. Each joint chooses separately, so an appendage can fold at one joint and swing at the next.
Appendage joint pitch bend	(C)	turns the segment toward its face in the same way. Together with yaw bend it gives an appendage a built-in curve before any joint moves.
Appendage joint length	(D)	sets how long the segment is, and so how far this joint lies from the next. Long segments reach far with few joints; short segments put more joints into the same reach.

Each segment has eight parameters.

Appendage segment asymmetry	(E)	makes the segment lopsided: the body leans to one side near the joint and to the other side further along. At 0 the segment is symmetric.
Appendage segment gain	(F)	sets how strong the bulges and dents on the surface of the segment are.
Appendage segment lobe	(G)	adds one bulge about two thirds of the way along the segment and sets its size, which gives the segment a pad near its far end.
Appendage segment phase	(H)	sets where along the segment the bulges and dents fall.
Appendage segment width	(I)	sets the size of the cross-section from side to side.
Appendage segment depth	(J)	sets the size of the cross-section from face to face. Equal width and depth give a round segment; unequal values give a flattened one.
Appendage segment taper	(K)	sets how the cross-section changes along the segment: below 1 the segment narrows toward its far end, above 1 it widens.
Appendage segment twist	(L)	rotates the cross-section about the long axis of the segment, from none at the joint to the full angle at the far end.

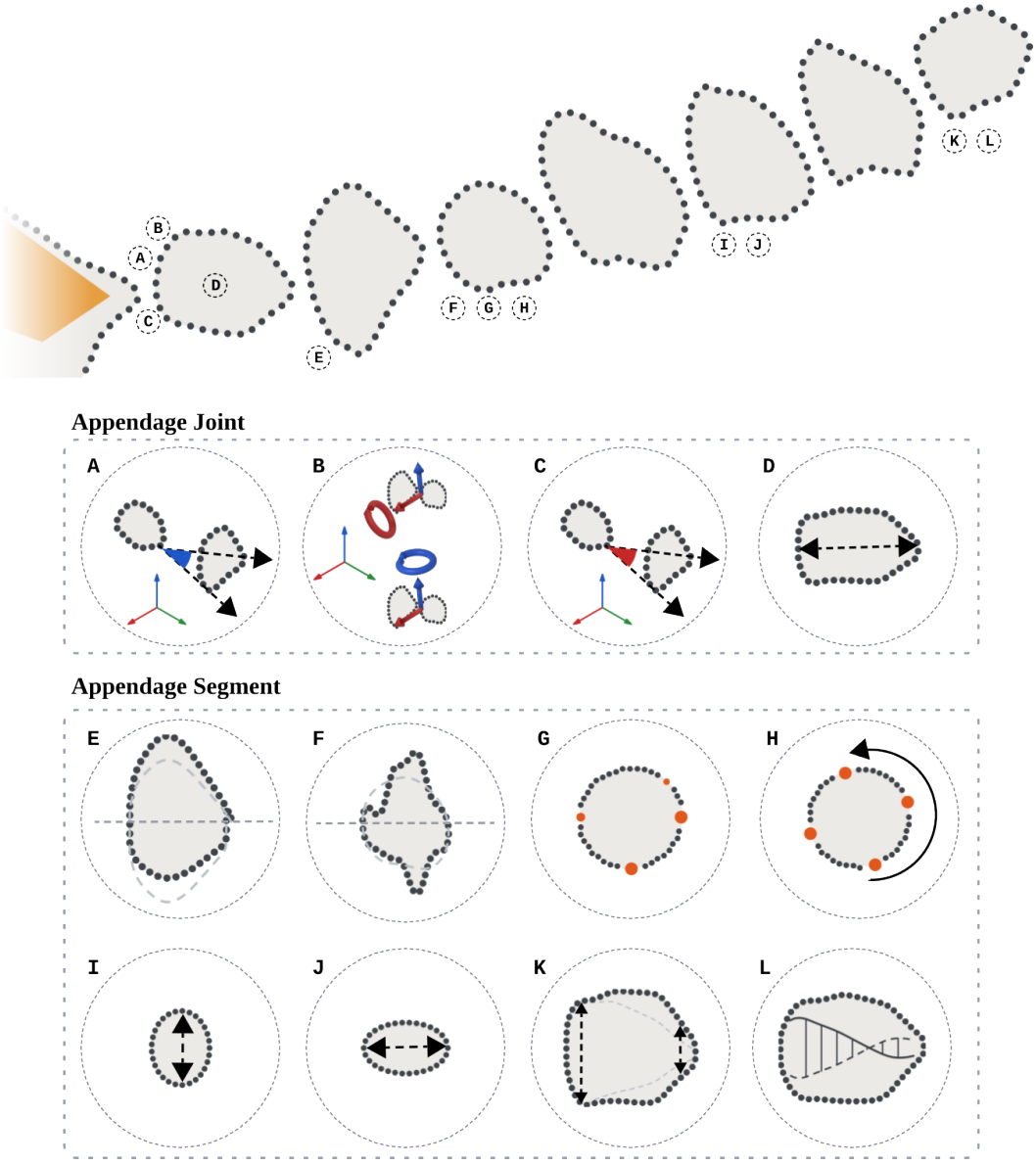

Figure 10: **The appendage.** Top: one appendage of the same root, growing from its spoke tip at the left; each segment is drawn as a chain of beads around its outline. **A–L**: one parameter each, cut and enlarged from the appendage above; the small circle marks where each was cut. Appendage joint: A **Appendage joint yaw bend**, B **Appendage joint hinge axis**, C **Appendage joint pitch bend**, D **Appendage joint length**. Appendage segment: E **Appendage segment asymmetry**, F **Appendage segment gain**, G **Appendage segment lobe**, H **Appendage segment phase**, I **Appendage segment width**, J **Appendage segment depth**, K **Appendage segment taper**, L **Appendage segment twist**.

B LATENT GENOME DETAILS

B.1 SYNTHETIC DATA GENERATION

We generated 2 million designs from the design space described in Appx. A. Each design was created in three steps. First, we chose how many appendages it has, from one to five. Second, we chose how many joints it has in total, up to 25, and divided them at random among the appendages, so that every appendage has at least one joint. Third, we drew every design parameter of the root and of each segment uniformly at random from its range (Table 1).

Designs with more appendages have more possible topologies, because there are more ways to divide the segments among the appendages: with one appendage there are 25 possible topologies (one to 25 segments), and with five appendages there are 1,125. To make sure that the latent space sees every topology about equally often, we generated more designs with more appendages, in the ratio $1:6:18:34:45$ for one to five appendages. This gave 19,240, 115,440, 346,320, 654,000 and 865,000 designs, respectively.

B.2 ENCODER

The encoder has two branches (Table 12). The first branch reads the two-channel voxel grid with a 3D residual network (He et al., 2016). The network shrinks the grid in five steps, from 128 voxels per side to 4, while the number of channels grows from 32 to 512. Average pooling then yields a vector of 512 numbers. The second branch reads the kinematic graph. Each node is the root or a joint, and stores the location and hinge axis of that joint. Three rounds of message passing share information between each joint and its neighbors along the appendage, in both directions. The mean and the maximum over all nodes are then combined into a vector of 128 numbers. The two vectors are joined, passed through two linear layers, and projected to a vector of 32 numbers. This vector is rescaled to unit length, so the latent genome z lies on the unit sphere $S^{31} \subset \mathbb{R}^{32}$.

B.3 DECODER

A design is written as a sequence of tokens (Fig. 2H) : first the 39 variables of the root, then the 12 variables of each segment, one appendage after another. A control token before each segment says whether the segment starts a new appendage or extends the current one, and a final token ends the sequence.

Each continuous variable is cut into equal bins over its range and written as the token of its bin. We conducted a sensitivity analysis (Fig. 11) on the parameters of the design in order to decide how many bins each one needs. Across 1,000 designs, each variable in turn was changed by a random amount (10% of its range on average), and we measured how much of the voxelized body changed. Variables that changed the body more were given more bins, from 8 to 128 (Table 1). For example, the variable that sets where an appendage starts around the edge of the root changed the body the most (86% of it) and was given 128 bins; the variables that set the length and the bend of a segment changed 18–22% of the body and were given 32 bins; and the variables that set the fine surface detail of a segment changed less than 6% and were given 8 bins.

The decoder is a transformer (Vaswani et al., 2017) that predicts the next token from the tokens before it (Table 13). A linear layer turns the latent genome z into eight vectors, and every layer of the transformer attends to them.

B.4 TRAINING

Positive pairs. Each design in a batch is paired with a changed copy of itself, a process known as data augmentation (Fig. 12). The copy is rotated about the vertical axis of the voxel grid in steps of $30°$, rescaled by a factor between 0.8 and 1.2 along each axis, sheared by up to 0.1, and moved to a random position at which it still fits inside the grid. With a probability of 0.3, every voxel of the copy is also shifted by up to one voxel. The same rotation, rescaling, shear and shift are applied to the joints of the kinematic graph. With a probability of 0.1, the terminal segment of one appendage is removed from the original design.

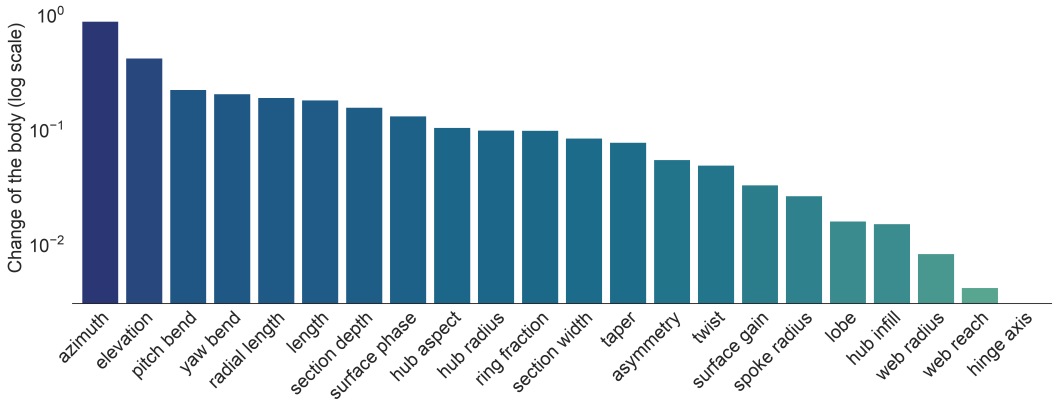

Figure 11: **Sensitivity analysis of the design variables.** We measured how much each variable of the design changes the body. Across 1,000 designs, each variable in turn was changed by a random amount (10% of its range on average) and the body was voxelized before and after the change. The change of the body is one minus the intersection over union (IoU).

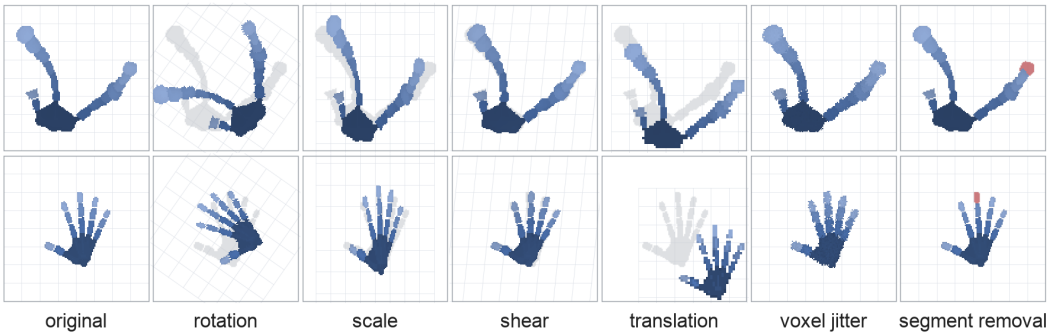

Figure 12: **Data augmentation.** Two designs and the operations used to create their positive pairs. The original design is shown in gray behind each augmented copy.

Loss. The encoder and decoder are trained together with a loss that has two terms,

$$\mathcal{L} = \mathcal{L}_{\text{con}} + \lambda \mathcal{L}_{\text{rec}}, \tag{1}$$

a contrastive term $\mathcal{L}_{\text{con}}$ and a reconstruction term $\mathcal{L}_{\text{rec}}$, balanced by the weight λ.

For the contrastive term, a batch of N designs and their N copies is encoded into $2N$ latent genomes, $z_1, \ldots, z_{2N}$. For a design i and its copy j, the NT-Xent loss (Chen et al., 2020) is

$$\ell_{i,j} = -\log \frac{\exp(z_i^\top z_j / \tau)}{\sum_{k \neq i} \exp(z_i^\top z_k / \tau)}, \tag{2}$$

where τ is a temperature. Because every z has unit length, $z_i^\top z_j$ is the cosine similarity of the two latent genomes. $\mathcal{L}_{\text{con}}$ is the average of $\ell_{i,j}$ and $\ell_{j,i}$ over the N pairs.

For the reconstruction term, the decoder reads the latent genome of the original design and, at each position t of its token sequence $s_1, \ldots, s_T$, predicts a distribution $\hat{s}_t$ over the next token given the true tokens before it. The reconstruction term is

$$\mathcal{L}_{\text{rec}} = -\frac{1}{T} \sum_{t=1}^{T} \log \hat{s}_t[s_t], \tag{3}$$

averaged over the batch, where $\hat{s}_t[s_t]$ is the probability given to the true token s_t. The reconstruction term updates the decoder and also the encoder, so that designs that are close in latent space can still be told apart by the decoder.

Implementation details. We set the temperature to $\tau = 0.5$ and the weight of the reconstruction term to $\lambda = 1$. The model was trained for 50 epochs (31,200 steps) with AdamW and a weight decay of 10^{-6}. The learning rate rose linearly to 10^{-3} over the first five epochs and then followed a cosine decay to zero. Gradients were clipped to a norm of 10, and training used bfloat16 precision. Training ran on eight NVIDIA B300 GPUs with a batch of $N = 400$ designs per GPU, and took 16 hours.

B.5 Evolution

Our genes lie on the unit sphere $\mathbb{S}^{31}$, but CMA-ES samples from a normal distribution, which lives in $\mathbb{R}^{32}$. We therefore let CMA-ES search in $\mathbb{R}^{32}$ and project every sample $\mathbf{x}$ onto the sphere before decoding it,

$$\mathbf{z} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \in \mathbb{S}^{31}. \tag{4}$$

The search starts from a random point on the sphere. Since only the direction of a sample matters, the length of the distribution's mean carries no information and is free to drift. The two ablations without the spherical projection have no sphere to search on, so their samples are decoded as they are, the search starts at the mean of the training genes, and the initial step size is scaled by their standard deviation.

C Task Details

All tasks were simulated in Isaac Lab (Mittal et al., 2025). For every design and every task, a control policy was trained from scratch with PPO (Schulman et al., 2017) across thousands of parallel environments, using a recurrent policy and value function (Table 16). To speed up training, we chose to use privileged observations in simulation. For deployment on the physical robot, the policy of the winning design was first distilled into a proprioceptive policy and then finetuned under domain randomization (Table. 15). Below, we describe the setup of each task and the reward it uses and the symbols in the reward tables are defined in Table 2.

Table 2: Symbols used in the reward tables.

Symbol	Meaning
$\mathbf{p}$, $\mathbf{p}_0$	position of the object, and its start position
$\mathbf{p}^*$	point under the root at which the object is held
r	distance of the object from its start position along the table, $\lvert\mathbf{p}_{xy} - \mathbf{p}_{0,xy}\rvert$
h	height of the object above its start position
θ	angle between the orientation of the object and the goal orientation
ω	rotation rate of the object about the task axis
ϕ	angle between the actual and the intended rolling axis of the object
v	linear speed of the object
g, $g^\star$	progress of the object toward the opening of the pipe, from 0 to 1, and its largest value so far
p	proximity of the nearest link to the object
s	slip: how far the object has moved relative to the root since release
d_i	distance from the tip of appendage i to the object
f_i	contact force between the tip of appendage i and the object, clipped to $[0, 1]$ N
f_{self}	contact force between appendages
$\mathbf{a}_t$	action at step t
$\mathbf{q}_t$, $\mathbf{q}_0$	joint position targets at step t, and the initial pose
$\boldsymbol{\tau}$	torque of every joint divided by the torque limit of its motor
Δ	change of a quantity over one step

C.1 Rotate

We trained on 16 objects from Yin et al. (2023), with an equal number of environments per object (Fig. 13). The design is mounted with its root facing upward, and the object is placed on the root.

The policy observes the joint positions, the joint position targets, and the position of the object relative to its start position. A goal orientation starts at the orientation of the object and advances by 22.5° about the vertical axis every time it is reached, so a policy that keeps reaching goals rotates the object continuously. The reward (Table 4) follows the goal-based scheme of Handa et al. (2023) and Fay et al. (2026). It rewards progress toward the goal and reaching it, and penalizes the object drifting from where it started. An episode ends early when the object falls off the root, tips over, or drifts too far. Fitness is the net rotation rate of the object over a 30 second window, in which nothing ends early.

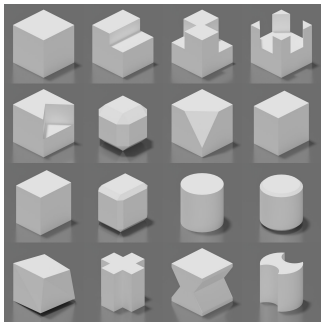

Figure 13: The 16 training objects.

The policy trained during evolution observes the position of the object, which the physical design cannot sense. We therefore distilled it into a proprioceptive policy, one that observes only the joint positions and the joint position targets. The evolution policy acts as a frozen teacher, and the proprioceptive policy is trained to output the action the teacher would take from the same state. The proprioceptive policy was then fine-tuned further with PPO under the domain randomization listed in Table 15. Following Andrychowicz et al. (2020), the randomization starts at zero and is raised, step by step, each time the policy has adapted to the current level.

Table 3: Real-world in-hand rotation performance across diverse unseen objects. We report the mean rotation speed (ω, rad/s) and stable rotation time (T, s), with sample standard deviations shown as subscripts. Rotation speeds are evaluated in two trials. Stable rotation time is evaluated in three 30 s trials. For each object, both metrics are bolded for the design with the highest mean rotation speed, determined before rounding. An em dash indicates failure to sustain in-hand rotation.

	Wooden Cube (134.5 g)		**Rubber Cube** (26.8 g)		**Rubik's Cube 3x** (92.9 g)		**Rubik's Cube 4x** (132.3 g)		**Fabric Cube** (16.1 g)	
Design	ω	T	ω	T	ω	T	ω	T	ω	T
LEAP (Multi-obj)	—		—		—		—		—	
LEAP (Cube-only)	$1.7_{\pm 0.1}$	$20.0_{\pm 7.0}$	$0.2_{\pm 0.0}$	$30.0_{\pm 0.0}$	$0.3_{\pm 0.0}$	$30.0_{\pm 0.0}$	$0.3_{\pm 0.0}$	$30.0_{\pm 0.0}$	—	
Ours (Tetradactyl)	$\mathbf{1.8_{\pm 0.1}}$	$\mathbf{30.0_{\pm 0.0}}$	$1.9_{\pm 0.0}$	$30.0_{\pm 0.0}$	$1.7_{\pm 0.1}$	$30.0_{\pm 0.0}$	$1.8_{\pm 0.3}$	$21.3_{\pm 7.0}$	$\mathbf{1.7_{\pm 0.1}}$	$\mathbf{30.0_{\pm 0.0}}$
Ours (Didactyl)	$1.5_{\pm 0.3}$	$30.0_{\pm 0.0}$	$\mathbf{2.4_{\pm 0.5}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{2.0_{\pm 0.0}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{1.9_{\pm 0.2}}$	$\mathbf{30.0_{\pm 0.0}}$	$1.4_{\pm 0.2}$	$30.0_{\pm 0.0}$

	Tomato Soup Can (353.2 g)		**Moon Ball** (30.9 g)		**Apple** (364.6 g)		**Orange** (212.6 g)		**Avocado** (223.8 g)	
Design	ω	T	ω	T	ω	T	ω	T	ω	T
LEAP (Multi-obj)	—		—		—		$0.2_{\pm 0.0}$	$30.0_{\pm 0.0}$	—	
LEAP (Cube-only)	—		—		$0.3_{\pm 0.0}$	$30.0_{\pm 0.0}$	$0.6_{\pm 0.0}$	$30.0_{\pm 0.0}$	—	
Ours (Tetradactyl)	—		—		—		$0.9_{\pm 0.1}$	$30.0_{\pm 0.0}$	—	
Ours (Didactyl)	$\mathbf{1.0_{\pm 0.3}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{1.3_{\pm 0.1}}$	$\mathbf{26.8_{\pm 2.9}}$	$\mathbf{1.3_{\pm 0.1}}$	$\mathbf{24.7_{\pm 1.5}}$	$\mathbf{1.8_{\pm 0.1}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{1.4_{\pm 0.1}}$	$\mathbf{21.7_{\pm 4.9}}$

	Disk Toy (77.3 g)		**Hand Soap** (315.6 g)		**Salt Container** (687.3 g)		**Jam Jar** (577.7 g)		**Energy Stick** (68.7 g)	
Design	ω	T	ω	T	ω	T	ω	T	ω	T
LEAP (Multi-obj)	—		—		—		—		—	
LEAP (Cube-only)	—		—		$0.2_{\pm 0.0}$	$30.0_{\pm 0.0}$	—		—	
Ours (Tetradactyl)	—		—		—		—		—	
Ours (Didactyl)	$\mathbf{1.0_{\pm 0.4}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{0.8_{\pm 0.0}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{0.9_{\pm 0.1}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{0.6_{\pm 0.1}}$	$\mathbf{30.0_{\pm 0.0}}$	$\mathbf{1.6_{\pm 0.1}}$	$\mathbf{26.2_{\pm 2.7}}$

We then deployed the policy zero-shot on the physical design and tested 15 objects that were not used in training (Table 3). A design that could not keep an object rotating is marked with a dash. The didactyl design rotated every object, at 1.39 rad/s on average, and sustained rotation for the full 30 seconds on 11 of them. The tetradactyl design rotated the five cubes and the orange but no other object. On the cubes it was as fast as the didactyl design, at 1.78 rad/s on average. The LEAP Hand trained on all 16 objects failed to rotate any of the test objects, so we trained a second policy for it

Table 4: Reward terms of the rotate task.

Term	Equation	Weight
rotation toward goal	$1/(\lvert\theta\rvert + 0.1)$	1
goal reached	$\lvert\theta\rvert \leq 0.2$	250
steady rotation	$0.25 < \omega_z < 1.5$	1
object displacement	$\lvert\mathbf{p} - \mathbf{p}_0\rvert$	-10
object drift	$\max(\lvert\mathbf{p} - \mathbf{p}_0\rvert - 0.03,\, 0)$	-0.1
action magnitude	$\lvert\mathbf{a}_t\rvert^2$	$-2\text{e}{-4}$
pose deviation	$\lvert\mathbf{q}_t - \mathbf{q}_0\rvert^2$	-0.1
drop	episode ends	-10

on a single cube. That policy rotated the cubes and three other objects, at 0.51 rad/s on average, a third of the speed of the didactyl design.

C.2 PICK

We trained on the same 16 objects as in the rotate task. The design is mounted above a table with its root facing downward, and the object is placed on the table beneath it. The object counts as held when it is lifted off the table and kept under the root. The reward (Table 5) follows the usual shape for learning to grasp (Wan et al., 2023; Yuan et al., 2025). It pays for approaching and touching the object, for lifting it up to the root, and for holding it still there, and penalizes the object drifting sideways or moving fast. An episode ends early when a held object falls back to the table or escapes from under the root. Fitness is how long the object stays held over a 60 second window in which a force pulls on it, a harder form of the test used to validate grasps in Wang et al. (2022). The direction of the force changes at random every 3 seconds, and its strength increases every 3 seconds.

Table 5: Reward terms of the pick task.

Term	Equation	Weight
approach	$\text{mean}_i\ \exp(-d_i/0.02)$, before first contact	0.4
contact	$\text{mean}_i\ f_i$	0.8
multiple contacts	at least two appendages touch the object	0.8
lift	$\text{clip}\big((h - 0.022)/0.008,\, 0,\, 1\big)$	8
near the root	$\exp(-\lvert\mathbf{p} - \mathbf{p}^*\rvert/0.05)$	2
stable grasp	held, at least two contacts, and nearly at rest	2
sideways drift	$\max(r - 0.04,\, 0)$	-5
linear speed	v	-0.25
angular speed	ω	-0.02
action magnitude	$\lvert\mathbf{a}_t\rvert^2$	$-2\text{e}{-4}$
self-collision	f_{self}	-0.02
drop	object dropped	-10

Table 6: Real-world performance on the pick task across diverse objects. Results indicate successful picks out of 10 trials per object.

Wooden Cube	Rubber Cube	Rubik's Cube 3x	Rubik's Cube 4x	Fabric Cube	Cube Toy	Lemonade Mix Box	Tennis Ball	Moon Ball	Orange	Avocado	Donut Toy
10/10	10/10	8/10	8/10	10/10	7/10	10/10	6/10	5/10	9/10	5/10	4/10
Water Can	**Candle Jar**	**Pudding**	**Polygon Toy**	**Jam Jar**	**Energy Stick**	**Meat Can**	**Toy Bear**	**Disney Toy**	**Baby Bottle**	**Motor**	**Salt Container**
10/10	6/10	10/10	8/10	6/10	10/10	10/10	10/10	10/10	10/10	10/10	0/10

The policy was distilled into a proprioceptive policy and fine-tuned as in the rotate task. We then deployed the policy zero-shot on the physical design and tested 24 objects, with 10 attempts each (Table 6). In each attempt the design was lowered onto the object, closed its appendages, and was lifted. An attempt counts as a success if the object was lifted off the table and stayed in the hand.

The design picked up 23 of the 24 objects at least once and succeeded in 80% of all attempts. It failed only on the salt container, the heaviest object, and was least reliable on round objects such as the moon ball and the avocado.

C.3 HANG

The design starts next to a horizontal bar, with a 500 g weight hanging below its root. Each episode has two phases. In the first, gravity is off and the root is held in place while the design closes its appendages around the bar. Then the root is released and gravity is switched on, so the design must carry its own weight and the load. Throughout, the bar twists back and forth about its axis. The design counts as holding on when at least two appendages touch the bar and the root has not slipped away from it. The reward (Table 7) pays for approaching and touching the bar and, after release, for holding on without slipping, and penalizes the torque the joints use. An episode ends early when the design loses contact with the bar, slips too far, or falls. Fitness is the time the design held on over a 20 second window in which the bar twists twice as far as in training. Each second held counts for less the more torque the joints used, down to half.

We deployed the policy zero-shot on the physical design, which gripped a stationary bar and hung from it under the 500 g load (Fig. 5E-H).

Table 7: Reward terms of the hang task. Terms marked with † are active only after release. τ_{ref} is a hyperparameter.

Term	Equation	Weight
approach	$\mathrm{mean}_i\ \exp(-d_i/0.02)$, before first contact	0.4
contact	$\mathrm{mean}_i\ f_i$	0.8
multiple contacts	at least two appendages touch the bar	0.8
stability†	$\mathrm{mean}_i\ f_i \cdot \exp(-s/0.0075)$, if at least two contacts	8
stable grasp†	at least two contacts and $s \leq 0.0075$	2
joint torques	$\lvert\boldsymbol{\tau}\rvert^2/\tau_{\mathrm{ref}}$	-0.5
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$	-0.001
self-collision	f_{self}	-0.02
failure	episode ends	-10

C.4 RETRIEVE

A cube rests inside a narrow horizontal pipe. The design is mounted at the opening of the pipe with its root fixed, so only its appendages can enter. It must reach in, hook the cube from behind, and drag it out. Every episode starts from a pose, searched for before training, in which one link reaches behind the cube. The policy observes the joint positions, the joint position targets, and the position of the cube inside the pipe. The reward (Table 8) pays for moving the cube toward the opening, with bonuses for passing milestones along the way and for getting it out. An episode ends when the cube is out of the pipe.

Table 8: Reward terms of the retrieve task.

Term	Equation	Weight
new progress	$\min(\Delta g^\star,\ 0.1)$	40
new progress in contact	$\min(\Delta g^\star,\ 0.1)$	40
new progress, unclipped	$\Delta g^\star$	80
new progress, squared	$\Delta (g^\star)^2$	80
net progress	$\mathrm{clip}(\Delta g,\ \pm 0.1)$	10
proximity	$\mathrm{clip}(\Delta p,\ \pm 0.25)$	5
contact	in contact	0.005
milestones	$g^\star$ passes $0.25,\ 0.5,\ 0.65,\ 0.8,\ 0.95$	2, 4, 8, 12, 20
retrieved	retrieved	400
escape	escaped	-120
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$	-0.001

Most designs cannot move the cube at all, so a fitness that only counted retrievals would be zero for almost every design. Fitness therefore ranks designs in stages. A design that cannot touch the cube is scored by how close it gets, one that touches it by how far behind it it reaches, and one that hooks it by how far it can drag it. Only designs that can drag the cube are trained, and they are scored above all others by the fraction of episodes in which they retrieve it. The policy was distilled into a proprioceptive policy and fine-tuned, then deployed zero-shot on the physical design, it retrieved the cube from a real pipe (Fig. 5I-K).

C.5 FLIP

A cube rests on a table, and the design is mounted above it with its root facing downward. The design must flip the cube over its edge, again and again, in one fixed direction. It cannot hold the cube in the air, and must use the table, gravity, and its own body together to turn it, which is known as extrinsic dexterity (Dafle et al., 2014; Zhou & Held, 2023). Each flip moves the cube one edge length along the table, so the design must also pull it back to keep flipping it. The policy observes the joint positions, the joint position targets, the position and rotation rate of the cube, and the current stage. Each episode alternates between two stages. In the flip stage, a goal orientation advances by $22.5°$ in the rolling direction every time it is reached, so a policy that keeps reaching goals rolls the cube continuously. After four goals, that is, one face, the return stage begins, in which the design must bring the cube back to where it started. The reward (Table 9) pays for progress toward the goal in the flip stage and toward the start position in the return stage, and penalizes rolling the cube backward, lifting it, turning it off its rolling axis, or losing contact with it. An episode ends early when the cube strays too far from its start position, is lifted, or leaves the table. Fitness is the number of full turns of the cube over a 60 second window.

Table 9: Reward terms of the flip task.

Term	**Equation**	**Weight**
Flip stage		
progress toward goal	$\max(-\Delta\theta,\, 0)$	10
goal reached	$\theta \leq 0.1$	250
steady rolling	$0.25 < \omega < 1.5$	1
rolling backward	$\max(-\omega,\, 0)$	-1
Return stage		
progress toward start	$-\Delta \max(r - 0.03,\, 0)$	5000
returned	$r \leq 0.03$	250
rolling backward	$\max(-\omega,\, 0)$	-5
Both stages		
displacement	$\max(r - 0.03,\, 0)$	-10
lift	$\max(h - 0.015,\, 0)$	-10
off axis	$1 - \cos\phi$	-2
no contact	no appendage touches the cube	-1
action magnitude	$\lvert\mathbf{a}_t\rvert^2$	$-2\text{e}{-4}$
pose deviation	$\lvert\mathbf{q}_t - \mathbf{q}_0\rvert^2$	-0.3
failure	episode ends	-10

C.6 HOLD

The design is mounted on the end of a robot arm, and a hammer is held in place next to its root. Each episode has two phases. In the first, the design closes its appendages around the handle. Then the hammer is released, and the arm swings it up and down while a force pulls on it in a direction that changes at random. The design must therefore hold the hammer so that it stays put while it is used, called functional grasp (Agarwal et al., 2023). The design counts as holding the hammer when at least two appendages touch the handle. The policy observes the joint positions, the joint position targets, and the pose and velocity of the hammer relative to the root. The reward (Table 10) rewards approaching and touching the handle and, after release, holding it without letting it slip along, across, or about the handle. An episode ends early when the hammer drops or moves too far from the root. Fitness is how still the hammer stays relative to the root over a 30 second window of

swinging. It is highest when the hammer does not move at all, falls with the speed of the hammer relative to the root and with how far it has shifted from its pose at release, and is zero if the hammer drops.

Table 10: Reward terms of the hold task. The grasp score is $G = 0.2\,\mathrm{mean}_i\, e^{-d_i/0.02} + 0.4\,\mathrm{mean}_i\, f_i + 0.4$, and the slip cost is $S = 0.25\,(s_\mathrm{a}/0.02)^2 + 0.25\,(s_\mathrm{l}/0.015)^2 + 0.5\,(s_\mathrm{r}/0.35)^2$, where s_a, s_l and s_r are the slip along, across, and about the handle. Terms marked with † are active only after release; those marked with ‡ are, in addition, multiplied by $\mathrm{mean}_i\, f_i$ when at least two appendages touch the handle, and by 0 otherwise.

Term	Equation	Weight
grasp progress	$G_t - G_{t-1}$, before release	0.5
contact†	$\mathrm{mean}_i\, f_i$	0.25
multiple contacts†	at least two appendages touch the handle	0.25
hold‡	1	1
no slip‡	$1/(1+S)$	8
action smoothing	$\lvert\mathbf{a}_t - \mathbf{a}_{t-1}\rvert^2$	-0.005
self-collision	f_self	-0.02
drop	hammer dropped	-10

C.7 MULTITASK

Each design was evaluated in the rotate, flip, pick, hold, and hang environments, with a separate policy trained for each task as in the single-task trials. The five task scores were divided by the best fitness of the corresponding specialist available when the trial began, then summed to obtain unified fitness. We ran a single trial of 70 generations (Fig. 14).

Early improvements came mainly from rotate and hang, followed by a sharp increase in flip fitness around generation 11. The resulting design remained the leader until generation 55. It performed well at flipping and hanging but rotated slowly and could neither pick up objects nor hold the hammer. During this plateau, both unified fitness and the task scores of the leading design remained unchanged.

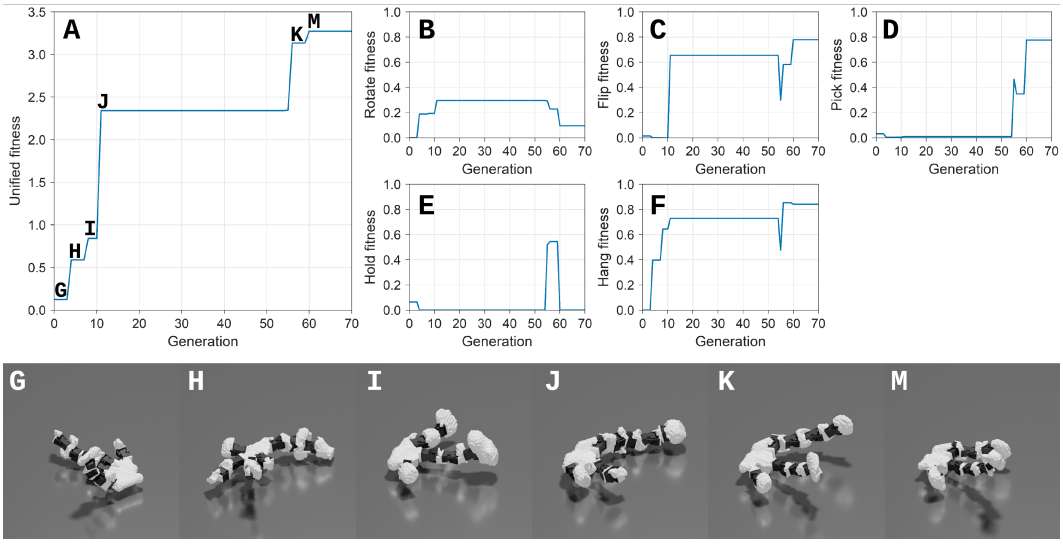

Figure 14: **The evolution of multitaskers.** Robots were evolved to rotate, flip, pick, hold, and hang. Unified fitness (A) was the sum of task normalized scores. The task scores of the best design found so far (B-F) reveal trade-offs: gains in unified fitness did not always improve performance on every task. Selected designs (G-M) correspond to the labeled points in A.

Around generation 55, designs emerged with nonzero scores on all five tasks, producing a sharp increase in unified fitness. The design from generation 56 differed from the earlier leader by only two topology edits: one link was removed from its long appendage and another added to its shorter appendage, making the two working appendages more alike. This change was accompanied by the ability to lift almost every object and hold the hammer through most of the swing, while retaining strong flip and hang performance. Further gains in unified fitness, however, did not preserve all five abilities. The highest-scoring design, found in generation 60, improved flip and pick performance but dropped the hammer in almost every episode and rotated more slowly.

To select the final generalist, we divided each task score by the best score reached on that task within this trial and summed again. This changed the relative weighting of the tasks and ranked the design from generation 56 first. We selected this design, which achieved nonzero fitness on all five tasks (Fig. 6).

Compared with the specialists, the selected generalist flipped the cube more often and hung from the bar nearly as long. It lifted almost every object but maintained its grasp under the pulling force for only a quarter of the evaluation window, whereas the pick specialist maintained its grasp for nearly the entire window. It dropped the hammer in one episode in five, compared with one in thirty for the hold specialist, and rotated objects at a quarter of the rate of the rotate specialist. Rotate and hold showed the largest performance gaps; their specialists also had the most appendages.

C.8 Real-World Deployment

Every joint of a physical design is driven by a Dynamixel XL330-M288-T servo. The servos are chained on one bus and driven through a U2D2 controller from a computer running ROS2. Each servo runs its built-in position controller, with the same stiffness and damping as in simulation. At every step of the 30 Hz loop. Figs. 15-19 show the didactyl and tetradactyl designs rotating, and the pick design lifting, each of the test objects.

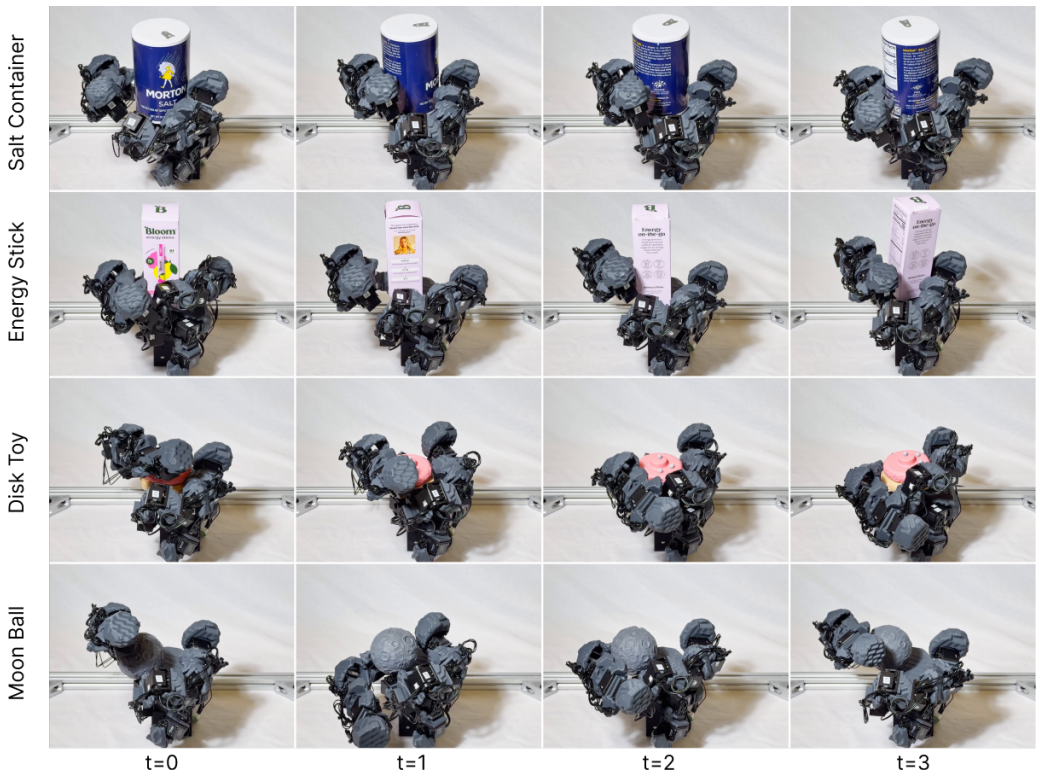

Figure 15: **Real-world in-hand rotation across diverse objects for didactyl design.**

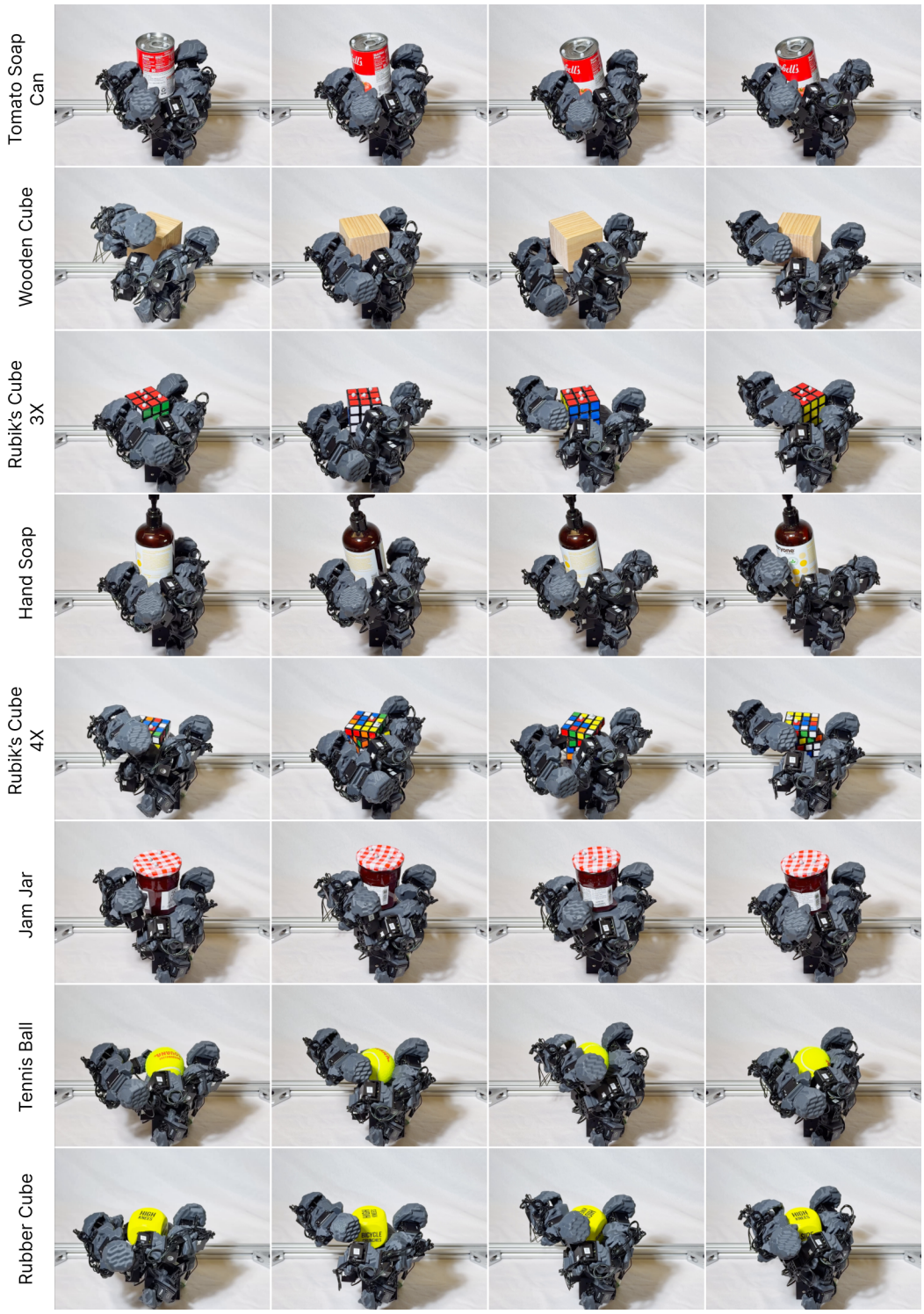

Figure 16: **Real-world in-hand rotation across diverse objects for didactyl design.**

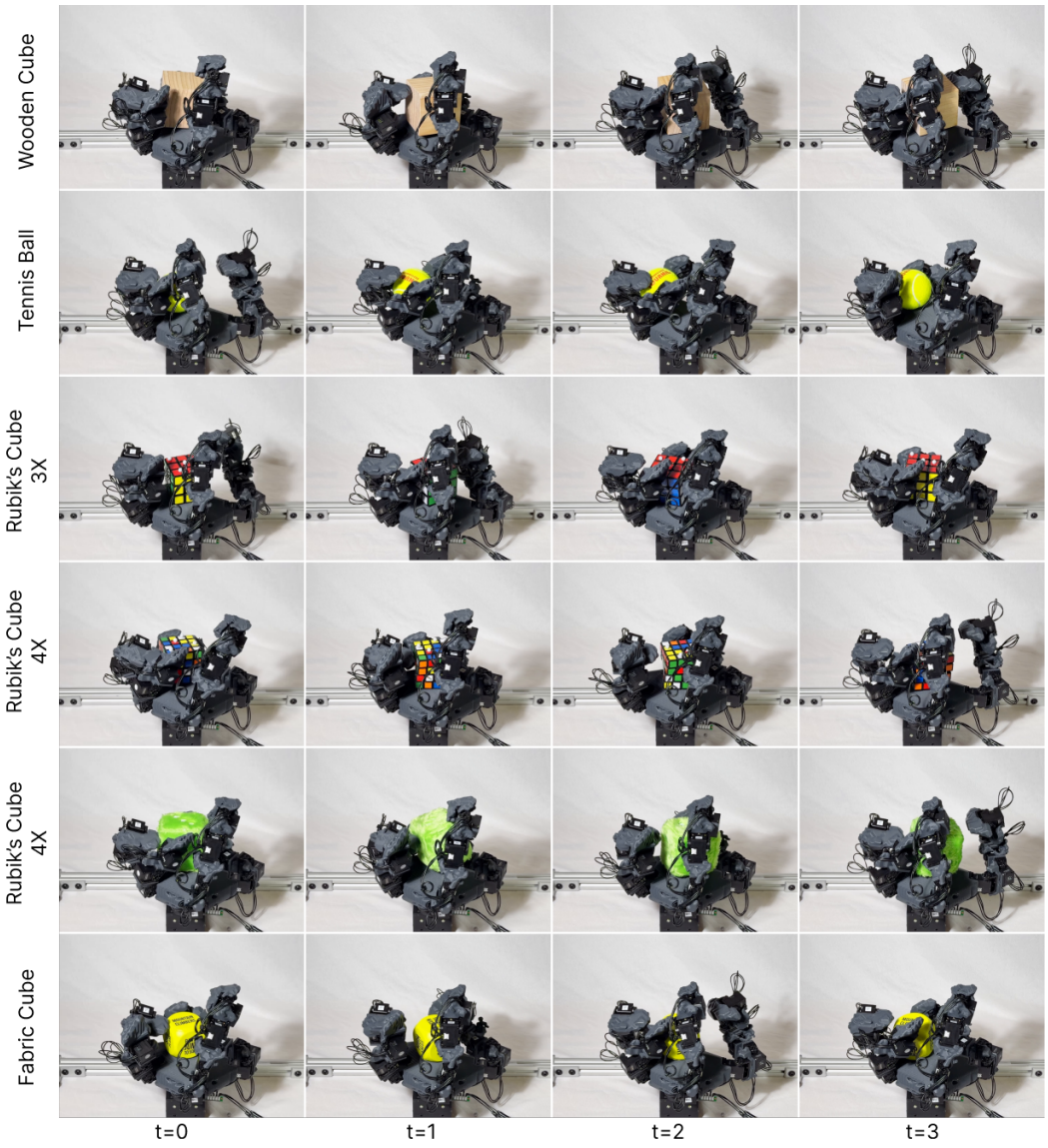

Figure 17: **Real-world in-hand rotation across diverse objects for tetradactyl design.**

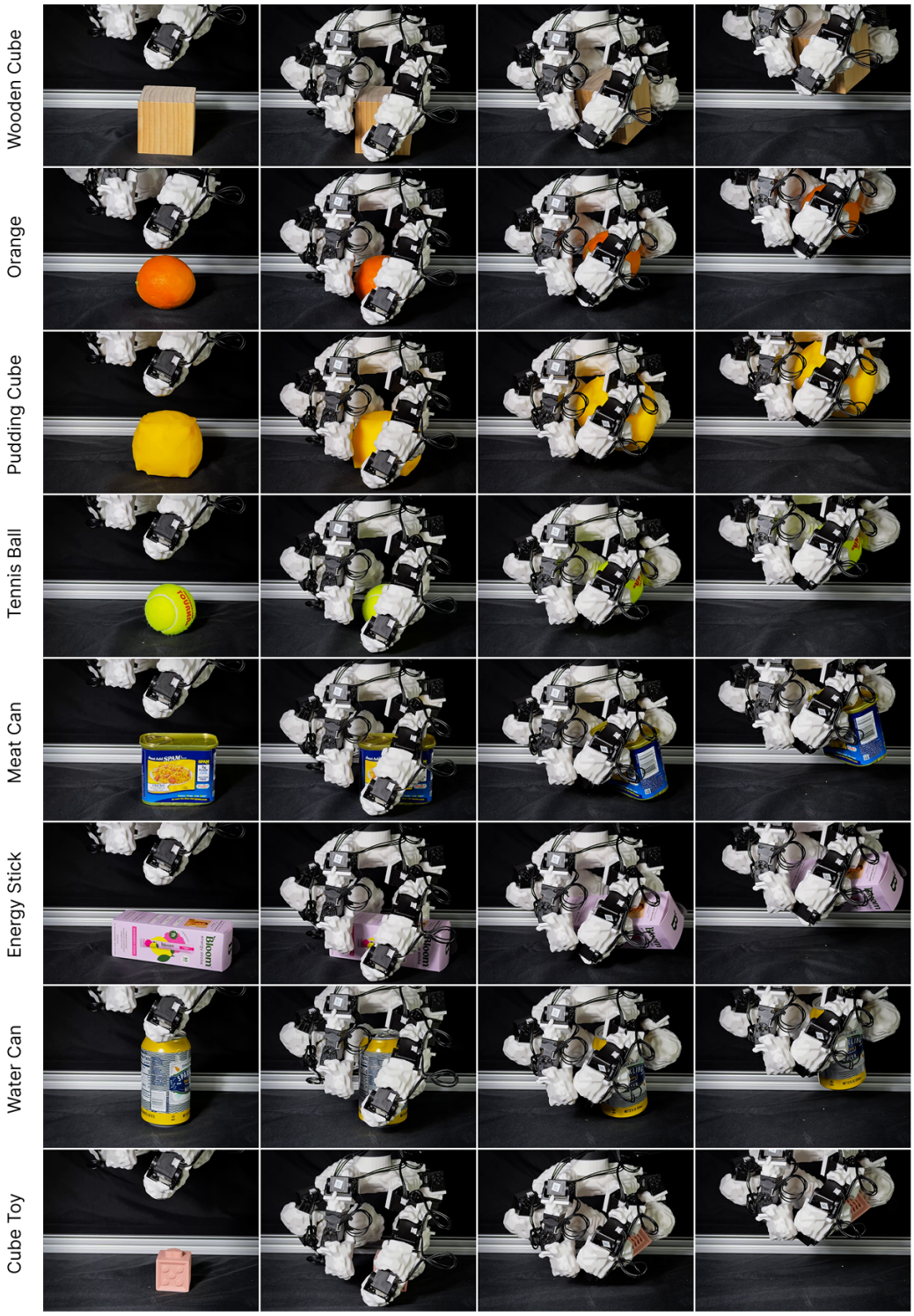

Figure 18: **Real-world picking across diverse objects using the four-appendage design.**



Figure 19: **Real-world picking across diverse objects using the four-appendage design.**



Figure 20: **Real-world test objects.** The everyday objects used in the real-world rotate and pick experiments: rigid and soft, smooth and textured, from a 16 g fabric cube to a 687 g salt container.



Figure 21: **The five physical designs.** From left to right: the retriever (purple), the didactyl rotator (grey), the picker (white), the 4-appendage rotator (grey) and the hanger (tan). Each is 3D-printed in PLA with one actuator per joint; parts are listed in Table 17. Their real-world results are in Tables 3 and 6 and Figs. 4–5.

D ABLATION ANALYSIS

Our embedding performed well in evolution, the embedding built without contrastive loss performed worse, and the embedding generated by the VAE performed far worse (Fig. 7B).

In this section, we ask why: what properties of the latent genome produced by contrastive learning allow a black-box optimizer such as CMA-ES to search it efficiently?

CMA-ES samples genes around its current estimate and updates that estimate using the fitness of the resulting designs. For this search to be effective, the latent genome must satisfy two requirements: its genes must decode to useful designs, and distances between genes must reflect meaningful changes in design.

We define the first requirement as validity. The gene of an existing design should reconstruct its body plan, while genes sampled during evolution should decode to designs that pass the checks required for evaluation. In evolution, these checks extend from basic geometry to motor fit, mesh generation, and finding a feasible initial pose. Failures at different stages receive different fitness values, but samples that fail cannot be evaluated on the manipulation task. Here we test reconstruction fidelity and basic geometric validity; These tests probe the requirement, but do not establish that a design will pass the full evaluation pipeline (see Table 14 for the rejection stages and their fitness values).

We call the second requirement smoothness. Different poses of the same body plan should map to the same part of the genome, so that pose does not become an extraneous direction of search. Increasingly large edits to a body plan should move its gene increasingly far, and nearby genes should decode to body plans with similar structure and size. Together, these properties make local sampling more likely to explore related designs and allow small changes to accumulate during search.

Table 11: Ablation of the latent genome. All validity metrics, Recall@1, and Edit ρ use the 2,000-design test set; all three 10-NN metrics use a separate set of 4,500 designs. The best and second best values in each column are shaded.

Model	Validity			Smoothness				
	App. acc.	Joint acc.	Valid	Recall@1	Edit ρ	Topology 10-NN acc.	Length 10-NN r	Volume 10-NN r
Ours (2M)	1.00	0.96	0.82	1.00	0.97 ± 0.10	0.92	0.97	0.91
Ours (200k)	1.00	0.84	0.65	1.00	0.99 ± 0.05	0.76	0.96	0.90
w/o $z \in \mathbb{S}^{31}$	1.00	0.80	0.64	1.00	0.99 ± 0.05	0.71	0.95	0.88
w/o $\mathcal{L}_{\text{con}}$	1.00	0.95	0.67	0.03	0.97 ± 0.09	0.59	0.82	0.73
VAE	0.44	0.06	0.33	0.01	0.93 ± 0.13	0.02	0.79	0.79

D.1 VALIDITY

We compared five models (Table 11). Ours (2M) is our full model trained on all 2M designs. Ours (200k) is the same model trained on the 200k-design subset used for the ablations and VAE, allowing a fair comparison (Sect. 3.9). We encoded and decoded the 2,000 test designs with each model. **App. acc.** is the appendage accuracy rate of designs decoded with the same number of appendages as the original, and **Joint acc.** is the joint accuracy rate of designs decoded with the same number of joints. **Geom. val.** is the fraction of decoded designs satisfying geometric validity with no overlapping appendages and no geometry extending beyond the 128^3 voxel grid.

Ours (2M) reconstructed designs best and had the highest geometric validity rate. Most designs decoded from our latent genome passed the basic geometric checks, though further checks are required before they can be built and simulated. Among the ablations, training on reconstruction alone improved joint-count accuracy relative to ours (200k), which is reasonable since reconstruction is its only objective. The VAE performed poorly. It recovered the number of appendages only for designs that happened to have five appendages, almost never recovered the number of joints, and produced valid designs less than half of the time.

Despite higher joint accuracy and geometric validity rate than Ours (200k), the reconstruction-only model performed poorly under CMA-ES (Fig. 7B), suggesting that genome smoothness also matters.

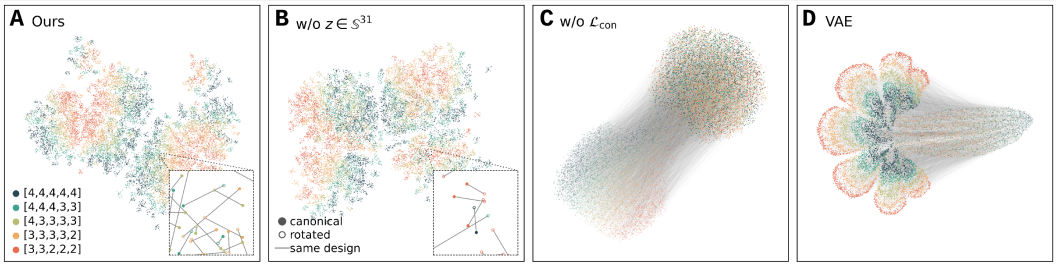

Figure 22: **Genetic coherence.** Ten thousand unique, pentadactyl morphologies were generated, copied, and the copy was rotated by a randomly selected angle. The resulting design pairs are visualized by UMAP (Healy & McInnes, 2024) along 2D projections of four different latent embeddings (**A-D**). A gray line connects the original (filled circle) to its rotated (open circle) counterpart. Five design variants were considered for this analysis, those with a pair of three-jointed appendages and three two-jointed appendages (3,3,2,2,2), four three-jointed and one two-jointed (3,3,3,3,2), one four-jointed and four three-jointed (4,3,3,3,3), three four-jointed and two three-jointed (4,4,4,3,3), and five four-jointed appendages (4,4,4,4,4). In the VAE, rotated copies can appear far from their originals in the UMAP projection (D). With the contrastive loss, rotated copies remain close to their originals in the UMAP projection (A–B); without it, the two views separate (C). Removing the spherical projection, $z \in \mathbb{S}^{31}$, had little effect on this pairing (B).

D.2 Smoothness

We assessed whether the genome is invariant to pose and whether genetic distance reflects changes in body plan.

To test pose invariance, we encoded augmented copies of the test designs using the methods in Fig. 12, excluding segment removal. **Recall@1** is the fraction of augmented copies whose nearest neighbor among the encoded originals of the 2,000 test designs is its own original. Both Ours (2M) and Ours (200k), as well as the model without the spherical projection, achieved 1.00. The reconstruction-only model and the VAE achieved 0.03 and 0.01, respectively.

Fig. 22 visualizes one of these augmentations, rotation about the vertical axis. We generated 10,000 designs with five appendages from five topologies that differ by two segments from one to the next, encoded each design before and after rotating it about the vertical axis, and projected the resulting genes into two dimensions using UMAP (Healy & McInnes (2024)). Each design appears as two points joined by a line and colored by its topology. With the contrastive loss (Fig. 22A-B), the two points of a design appear close together and designs of the same topology form patches that blend into those of neighboring topologies. Without it (Fig. 22C), the original and rotated designs occupy separate regions. In the VAE (Fig. 22D), the original designs form a flower-like shape. Designs with fewer joints (red) lie along the outer petals, while designs with more joints (green) lie closer to the center. After rotation, their points move away from the petals into the stem-like region to the right.

We next tested whether genetic distance tracks controlled edits to a body plan. For each test design, we removed up to four distal segments, one at a time, and encoded each edited copy. **Edit** ρ is the Spearman correlation between the number of segments removed and the genetic distance from the original, averaged over designs. It was high for all five models (Table 11), indicating that genetic distance generally increased as more segments were removed. We next measured whether independently sampled designs with nearby genes have similar body plans. We encoded a separate set of 4,500 designs from 17 topologies, with 900 designs for each number of appendages. **Topology 10-NN acc.** is the fraction of designs whose topology matches the most common topology among the decoded designs of their 10 nearest genetic neighbors. For each design in this set, we estimated total appendage length and body volume by averaging the corresponding properties of its 10 nearest genetic neighbors. **Length 10-NN** r and **Volume 10-NN** r are the Pearson correlations, across these 4,500 designs, between the estimates and the true values.

Ours (2M) scored highest on all three neighborhood metrics, and the VAE scored lowest. Removing the contrastive loss improved joint accuracy but reduced pose invariance and all three neighborhood metrics. This pattern is consistent with the reconstruction-only model's poorer CMA-ES

performance (Fig. 7B), suggesting that coherent genetic neighborhoods support latent search beyond accurate reconstruction.

E HYPERPARAMETERS AND BOM

Table 12: Encoder architecture.

Layer	Description
Voxel input	$128^3 \times 2$
Stem	Conv, kernel size 3^3, stride 2, to 64^3, 32 channels, batch normalization, ReLU
Stage 1	$2\times$ ResBlock, downsample to 32^3, 64 channels
Stage 2	$2\times$ ResBlock, downsample to 16^3, 128 channels
Stage 3	$2\times$ ResBlock, downsample to 8^3, 256 channels
Stage 4	$2\times$ ResBlock, downsample to 4^3, 512 channels
Average pool	out: 512
ResBlock	$2\times$ (Conv, kernel size 3^3, batch normalization), skip connection, ReLU; the first ResBlock of each stage downsamples with stride 2
Graph input	up to 26 nodes (root and joints) $\times$ 6 (location, hinge axis)
Linear	in: 6, out: 128, ReLU
Message passing	$3\times$ (mean over neighbors, Linear in: 256, out: 128, ReLU, skip connection)
Readout	mean and max over nodes, Linear in: 256, out: 128
Fusion	in: 512 + 128, out: 512, batch normalization, ReLU; Linear in: 512, out: 512
Projection	Linear in: 512, out: 32, normalized to unit length
Output	$z \in S^{31}$

Table 13: Decoder architecture.

Layer	Description
Input	$z \in S^{31}$, and the tokens predicted so far
Memory	Linear in: 32, out: 8×384 (eight vectors)
Embedding	token embedding (137) + learned position embedding (368), 384 channels
Layers 1–4	Transformer decoder layer: causal self-attention, attention to the eight vectors, feed-forward (1536, GELU); 8 heads, layer normalization before each part
Head	Linear in: 384, out: 137
Output	logits of the next token: 137

Table 14: CMA-ES hyperparameters.

Parameter	Value
CMA-ES	
Population size	32
Initial step size (σ_0)	0.25
Fitness of Rejected Designs	
Failed to compile	-1000
Motors do not fit	-950
No initial pose found	-900
Object dropped before training	-850

Table 15: Domain randomization used for fine-tuning.

Parameter	**Range**
Observation	
Joint position noise (per step)	0 to 0.02 rad
Joint position offset (per episode)	$\pm$0.005 rad
Action	
Action noise (per step)	0.05 to 0.1
Action delay	0 to 2 control steps
Initial State	
Joint position	$\pm$0.02 rad
Joint velocity	$\pm$0.01 rad/s
Object position	$\pm$0.005 m
Object pitch and roll	$\pm$0.1 rad
Design	
Joint stiffness	2.8 to 3.1 N m/rad
Joint damping	0.08 to 0.12 N m s/rad
Restitution	0 to 0.2
Object	
Friction coefficient	0.3 to 1.5
Restitution	0 to 0.2
Mass scale	0.9 to 1.1
Size scale	1.1 to 1.2

Table 16: Reinforcement learning hyperparameters.

Parameter	Value
Parallel environments	4096
PPO iterations	500 for rotate and flip, 200 otherwise
Steps per iteration	32
Policy and value network	GRU (256) + MLP (512, 256, 128), ELU
Learning rate	5×10^{-4} (3×10^{-4}), adaptive
Target KL divergence	0.02
Discount (γ)	0.99
GAE (λ)	0.95
Clip range	0.2
Value loss weight	4.0 (2.0)
Entropy weight	0 (0.002)
Learning epochs per iteration	5
Mini-batches per epoch	1 (4)
Gradient clipping	1.0

Table 17: Bill-of-materials (BOM) of the five physical designs. All designs share the same parts: the base and every segment are 3D-printed in PLA, and each actuated joint is one DYNAMIXEL XL330-M288-T that drives the child link through an FPX330-H101 hinge frame and is seated in the parent link on an FPX330-S102 (radial mount) or FPX330-S101 (lateral mount) frame. The first column gives the unit cost (ROBOTIS US list price, September 2026) and the remaining columns the quantity used by each design; PLA is counted by mass. Totals are rounded to the nearest dollar. [a]Sold in sets of four; unit cost is one quarter of the set price. [b]One cable ships with every actuator, so cables add no cost. [c]M3 heat-set insert cost is based on a 100-pack of Uxcell M3×6 mm brass heat-set inserts (5 mm OD) at $12.69. PLA cost is estimated at $0.02/g, with mass estimated from the printed volume at the effective printed density.

Item	Cost (US)	Rotate	Rotate-4	Pick	Hang	Retrieve
XL330-M288-T Actuator	$27.49	15	15	10	10	9
FPX330-H101 Hinge Frame with Idler[a]	$2.82	15	15	10	10	9
FPX330-S102 Base Frame (Radial)[a]	$2.50	7	7	10	5	1
FPX330-S101 Side Frame (Lateral)[a]	$2.50	8	8	0	5	8
Robot Cable-X3P 180 mm[b]	incl.	15	15	10	10	9
U2D2 USB Converter	$36.92	1	1	1	1	1
U2D2 Power Hub Board Set	$21.85	1	1	1	1	1
M3 Heat-Set Insert[c]	$0.13	180	180	120	120	108
PLA Filament (g)[c]	$0.02	223	245	219	103	99
Total		**$578**	**$579**	**$406**	**$404**	**$370**

GENERATIVE AI USAGE

We used generative AI tools to assist in implementing methods, creating figures, and editing some parts of the Appendix for readability. For all other tasks, generative AI was not used. We reviewed all AI-assisted contributions for correctness and take responsibility for the final contents of this paper.