2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

# An approach to representation learning in morphological robot evolution

Aart C. Stuurman
*Vrije Universiteit Amsterdam*
a.stuurman@vu.nl

Anil Yaman
*Vrije Universiteit Amsterdam*
a.yaman@vu.nl

A.E. Eiben
*Vrije Universiteit Amsterdam*
a.e.eiben@vu.nl

***Abstract*—A key challenge for evolving complex physical objects is to design a representation, that is, to devise suitable genotypes and a good mapping from genotypes to phenotypes (the objects to be evolved). This paper outlines a new approach to address this challenge for evolving robot morphologies and presents a proof-of-concept study to assess its feasibility. The key idea is to design genotype-phenotype mappings using variational autoencoders. This idea is implemented and tested for the evolution of modular robots for a locomotion task. The experiments show the practicability of this idea where the representation is not hand-designed, but algorithmically generated. This indicates a great future potential for the evolution of complex objects where there are no straightforward representations to use.**

***Index Terms*—evolutionary robotics, morphological evolution, representation, genotype-phenotype mapping**

## I. Introduction

Evolutionary Algorithms (EA) have been used to optimize the morphology, the controller or both of simulated or real robots for over two decades, with several approaches regarding the EA setup [1]–[3]. Unfortunately, the details of the EA configuration are seldom discussed and, to date, there is little known about how to set up an effective EA for robot evolution. The EA feature that probably has received the least attention is the representation or genotype-phenotype mapping, that is, the encoding of phenotypes (robots) into genotypes (data structures). The usual practice is to use some common representation, based on the researchers' habits, subjective preference, or ease of implementation. This is very questionable, because it is well-known in the EA research community that the representation is a crucial factor in the performance of EAs [4].

Zooming in on morphological robot evolution, it can be noted that available knowledge about representations is scarce. There are a number of options in the literature for encoding the morphologies of evolving robots, including parse trees, L-systems, compositional pattern-producing networks (CPPNs), but we are only aware of a few studies that explicitly address the effects of different morphology encodings [5], [6]. Research on this subject is in its infancy and the only thing we seem to know is that "the representation is at least as important as the optimization method for effectively creating robots" [7], [8].

The goal of this paper is to outline a new approach to handle representations for morphological robot evolution and to conduct a proof-of-concept study to assess its feasibility. The key idea is to design genotype-phenotype mappings in a systematic way, using variational autoencoders (VAE) [9]. The corresponding workflow consists of the following steps: 1) specify the phenotype space (the robots to be evolved), 2) specify the genotype space (the desired data structure to be used in the EA), 3) identify the criteria that define when a genotype-phenotype mapping is considered good, 4) use an applicable machine learning algorithm to find a good mapping.

Our main research question concerns the feasibility of learning good representations for morphological robot evolution by this approach. To answer this question we consider the evolution of morphologies of modular robots, whose phenotypes are defined by the RoboGen system [10], and aim at a vector representation, where genotypes are real valued vectors. Regarding the desired properties of the learned mapping we choose to be minimalistic here. Although there are several sensible criteria to define when a genotype-phenotype mapping is good (these will be discussed in Section IV), for this proof of concept, we only apply the standard reconstruction and KL-divergence loss [11] of the mapping as used in VAEs.

Note, that by this approach we not only get a genotype-phenotype mapping, but also a phenotype-genotype mapping and that the phenotype-genotype mapping is the (approximate) inverse of the genotype-phenotype mapping because of the use of the reconstruction loss in the learning algorithm. In EA terms, this means that the representation is (approximately) invertible. This is a seldom feature that can be advantageous. Another aspect to note is the locality of the representation. In EAs, locality stands for the property that nearby genotypes map to nearby phenotypes and it is a very desirable feature that makes a problem easier to solve. We expect that the genotype-phenotype mapping we obtain by the learning algorithm scores well on the locality criterion, even though it is not explicitly included among the optimization criteria.

## II. Robot evolution system

The most general case of robot evolution concerns joint evolution of morphologies (bodies) and controllers (brains) with lifetime learning right after birth as captured in the Triangle of Life model [12]. This implies two important things. Firstly, a phenotypic dichotomy (a robot consists of a body and a brain) and a genotypic dichotomy (the genotype of a robot consists of a body-coding segment and a brain-coding segment). Secondly, that the fitness of the robots is not

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

Fig. 1. The modules considered in the morphological design space. From left to right: *core*, *brick*, *vertical active hinge*, and *horizontal active hinge*.

calculated with the inherited brain, directly after 'birth', but with the learned brain, when the lifetime learning is finished in the so-called infancy period.

### A. Robotic phenotypes

*1) Morphologies (bodies):* The morphologies of the modular robots in our setup are composed of a subset of the modules in the RoboGen system. Each module has specific areas ("slots") where it can be attached to another module. We consider four distinct modules, as depicted in Fig. 1:

- *Core*: The cuboid, rigid centerpiece of the robot, of which there is always exactly one in each configuration. It has slots on four of its sides, excluding the top and bottom.
- *Brick*: A passive component similar to the *core*, but with a slightly smaller size. There can be zero or more *bricks* in a robot. Practically, a *brick* has three available slots, as one is used for attaching it to the parent module.
- *Active Hinge*: This module is an actively-controlled rotary joint that can rotate up to 60 degrees in either direction, which enables the robot to move. There are two variants of the *active hinge*, allowing for either vertical or horizontal rotation. Each *active hinge* has two slots, one on each end of the joint, with one slot being utilized to connect the joint to its parent module. Like the brick, a robot can have zero or more *active hinges*.

Throughout this paper, we discuss locality, a property that connects phenotypic and genotypic distances. To work with this concept, it is essential to define a metric for modular robots. A logical selection is the tree edit distance, which represents the number of modules that need to be added or removed from one tree to transform it into another. For this purpose, we use the APTED algorithm [13], [14].

*2) Controllers (brains):* Each robot is controlled through a network of central pattern generators (CPGs) [15]. Every active hinge $i$ has a corresponding CPG consisting of state variable $x_i$, representing the desired angle, and hidden state $y_i$. The system changes over time according to the following set of differential equations:

$$\begin{aligned} \dot{x}_i &= w_i y_i + \sum_{j \in \mathcal{N}_i} w_{x_j x_i} x_j \\ \dot{y}_i &= -w_i x_i \end{aligned} \tag{1}$$

where $w_{x_j x_i} = -w_{x_i x_j}$. The controller parameters $w$ determine the interaction between state variables. If more than one module is in between active hinge $a$ and $b$ we set $w_{x_a x_b} = w_{x_b x_a} = 0$. Initially all state variables are set to $\frac{\sqrt{2}}{2}$, leading to a system generating sine and cosine signals that control the robot.

### B. Robotic representations

In this work, we focus on the morphological representation and evolution. Therefore, we do not co-evolve the brains with the bodies. Thus the genotype of a robot encodes only the morphology and the brain of each 'newborn' robot is learned from scratch as discussed in Section III-B. We consider two types of representations for robot morphologies: (1) CPPN representation and (2) vector representation.

*1) CPPN representation:* This approach uses CPPNs, a type of artificial neural network, to represent the robot morphologies [16]–[18]. Our CPPN representation has four inputs and five outputs. The inputs include the x, y, and z coordinates of a component and the number of modules between that component and the core. The outputs determine the probabilities of module types (brick, active hinge, or empty) and rotations (0 or 90 degrees). Rotations are only applied to active hinges.

The body's genotype to phenotype mapping treats the morphology as a 3D grid. Starting from the core, we move outward querying the CPPN network to decide the type and rotation of each module until no open slots remain or a maximum number of modules is reached. If a grid location is already occupied, the module is not placed and that branch ends.

*2) Vector representation:* Our goal is to represent morphologies using real-valued vectors, as this enables the use of many vector-based optimization techniques. A key insight is that our system can be described by an unambiguous regular tree grammar, and there exist readily accessible methods for learning good representations from a set of trees. The formal definition of the tree grammar describing our robots is as follows:

$$\begin{aligned} \text{Alphabet} &= \{\text{core}, \text{brick}, \text{hinge\_h}, \text{hinge\_v}, \text{empty}\} \\ \text{Nonterminals} &= \{\text{start}, \text{child}\} \\ \text{Start symbol} &= \text{start} \\ \text{Production rules} &= \text{start} \rightarrow \text{core}(\text{child}, \text{child}, \text{child}, \text{child}) \\ &\quad\ \ \text{child} \rightarrow \text{brick}(\text{child}, \text{child}, \text{child}) \\ &\quad\ \ \text{child} \rightarrow \text{hinge\_h}(\text{child}) \\ &\quad\ \ \text{child} \rightarrow \text{hinge\_v}(\text{child}) \\ &\quad\ \ \text{child} \rightarrow \text{empty} \end{aligned}$$

Where *hinge_h* and *hinge_v* are horizontal and vertical active hinges respectively, and *empty* represents a slot with no module attached.

Specifically, the recursive tree grammar autoencoder (RT-GAE), a type of VAE, learns an approximate encoder and decoder between tree and vector space from an unambiguous regular tree grammar and an array of sample trees [19]. Note that in contrast to the CPPN genotype, which represents morphologies as non-overlapping grids, this approach directly represents them as trees, resulting in the two representations capturing slightly different morphology spaces.

The encoder consists of neural networks, each associated with a specific production rule. When encoding a tree, the

network corresponding to the root node's matching production rule is evaluated using representations of its child nodes, which are recursively encoded before the parent. Notably, production rules leading to terminals are encoded as a constant. A final neural network layer maps the encoded root node to the parameters of the VAE's latent distribution.

Similarly, the decoder comprises neural networks not only for production rules but also for nonterminals. Beginning with the initial symbol, it iteratively substitutes a nonterminal by feeding its representation into the neural network tied to the symbol to derive a production rule, leading to a terminal. Subsequently, it evaluates the neural network associated with that production rule to generate representations for the terminal's children, if any.

It is worth highlighting that genotype-phenotype mappings generally do not have an (approximate) inverse and that this method provides it without additional effort. Additionally, locality, the property ensuring that similar genotypes map to similar phenotypes, is crucial for an effective representation [4]. We anticipate that the compression achieved by the RTGAE using standard reconstruction and KL-divergence loss will partially preserve this relationship, even without explicit training for it.

## III. Experiments

The source code for our experiments is available at[1].

### A. Representation learning

The objective of our first experiment is to generate a vector representation using the methods described earlier. We train the RTGAE for 200 epochs with a batch size of 200. The training set consists of 10,000 trees uniformly sampled from a bounded domain of 1 to 10 nodes, including the core. We use a sampling algorithm similar to [20] where trees are constructed by continuously adding modules to available slots. Each iteration randomly selects a module type to append or leaves the slot empty. The selection weight of each type is based on the frequency of its appearance in trees that have the desired number of modules. Tree sizes are randomized to encounter both large and small trees during training. The network was trained using reconstruction and KL divergence losses, following the original RTGAE paper.

The RTGAE has two main hyperparameters to consider: the "tree dimensionality", which refers to the size of the encoding used by the encoder and decoder neural networks, and the "VAE dimensionality", indicating the dimensionality of the VAE latent space and corresponding to the size of the genotype. The model must learn to represent the complete tree space, although bounded, in contrast to the original RTGAE paper that focused on a specific subset of trees. Consequently the compression will appear low in comparison and a high model dimensionality is required. Fig. 2 shows the combined reconstruction and KL divergence losses during training for varying hyperparameter values. From this plot, we deduce

[1]https://github.com/surgura/morphology_representation

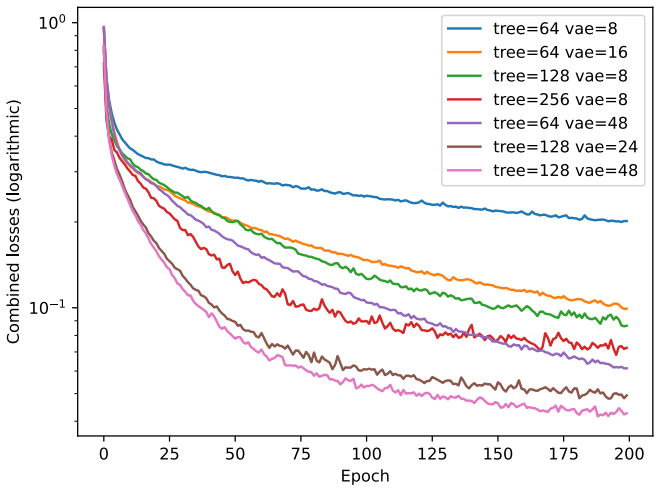



Fig. 2. Combined reconstruction and KL divergence losses over epochs for varying RTGAE hyperparameters. Dimensions in the legend are in order of the losses at the final epoch.

that 128 is a reasonable value for tree dimensionality, and 24 is preferable to 48 for VAE dimensionality. We favor the representation of 24 because a lower dimension is generally more beneficial for evolutionary algorithms and the difference in training loss is small. Additionally, we aim to limit the number of nodes in the generated trees to ten, aligning it with the training set. However, the RTGAE implementation we use counts empty nodes as well. Consequently, we adjusted the node limit to 30, allowing for robots with ten modules, but sometimes resulting in an excessive number of active hinges.

To assess adherence to the tree edit distance we randomly sample pairs of genotypes, map them to phenotype space, and compare genotypic (Euclidean) distance to phenotypic (tree edit) distance. Fig. 3 depicts this as a scatterplot, focusing on a range of small genotypic distances. A slight positive correlation can be observed between genotypic and phenotypic distances. The full plot, which covers up to a genotypic distance of 5, maintains this trend, extending to a phenotypic distance of 35. From this, we conclude that the RTGAE produces a mapping with modest locality, at no extra cost.

Additionally, for added validation, we randomly sample genotypes, pair each with a neighbor by applying Gaussian perturbation, and then visualize the resulting robots in Fig. 4. As expected, neighboring genotypes result in similar robots.

### B. Evolutionary experiments for validation

In our second experiment, we aim to validate the effectiveness of the learned vector representation by using it to evolve robots for a locomotion task. To this end, use a flat terrain and measure fitness by displacement over an evaluation period of 30 seconds by

$$fitness = \sqrt{x_{end}^2 + y_{end}^2}$$

where the initial position is $(0, 0)$ and $(x_{end}, y_{end})$ denotes the robot's final position in the x-y plane. Exploiting the fact

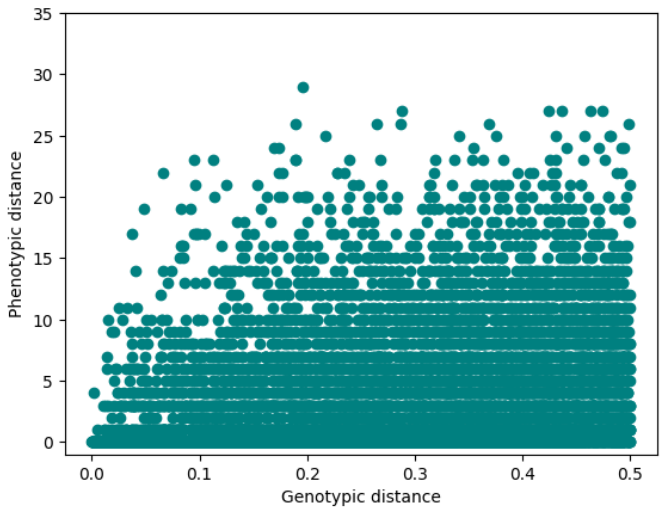



Fig. 3. The vector representation demonstrates modest locality even though it was not explicitly optimized for this. This property entails that small changes in the genotype lead to small changes in the phenotype. Each dot in the figure shows the genotypic and phenotypic distances of a pair of robots measured by Euclidean and tree edit respectively. Note that this is a zoomed-in view. The full plot extends to maximum distances of 5 and 35, showing the same effect.

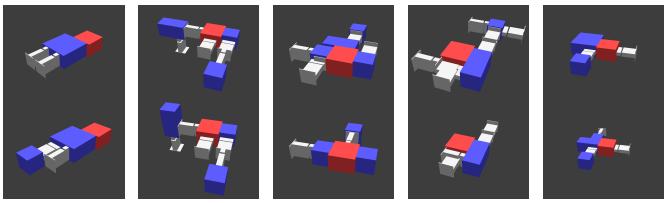


Fig. 4. Illustration of the effects of small mutations of genotypes shown in phenotypes. Top: morphologies of randomly sampled vector representations. Bottom: morphologies resulting from applying small Gaussian perturbations to the representations on top.

that bodies are represented by vectors, we simply use the covariance matrix adaptation evolution strategy (CMA-ES), a popular gradient-free vector optimization algorithm [21] to evolve good robots. We utilize the CMA-ES Python package developed by the original paper's author, with default settings and an initial standard deviation set to 0.5, terminating after 1100 fitness evaluations.

As explained in Section II-B, robot brains are not evolved together with the bodies, but learned for each 'newborn' robot from scratch in the infancy stage of the Triangle of Life model. Because brain configurations are specified by vectors (the weights of the CPG network) we choose to use the CMA-ES as the learning algorithm, with default settings and an initial standard deviation set to 0.5, running for 20 generations [22]. Note that it is important to distinguish two instances of the CMA-ES: one for evolving the bodies and one for learning the brains.

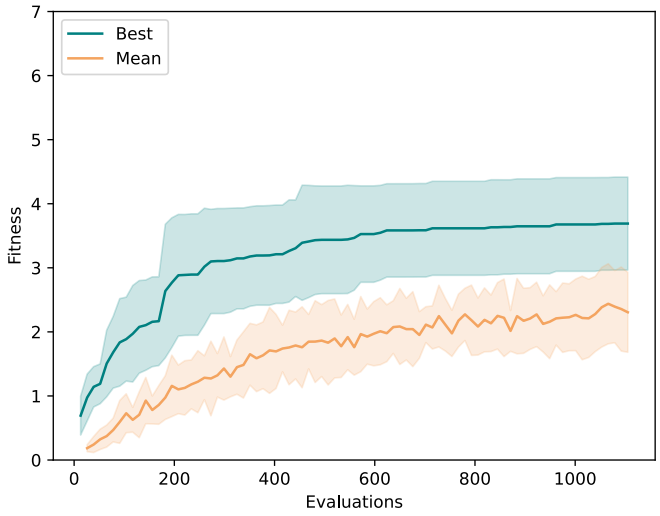
Further to the validation experiments, we also perform comparative experiments with the CPPN representation using crossover and mutation operators from the original CPPN paper but without the speciation mechanism with settings after [23]. The experiment is conducted over 20 generations with a population size of 100 and an offspring size of 50, leading to a total of 1100 evaluations. Similarly to the validation experiments, brains are not co-evolved with the bodies, but learned by the CMA-ES using the same settings as above.

Fig. 5a illustrates the fitness trend during the evolutionary experiment using the learned vector representation. The curves show the typical behavior of ascending at a diminishing rate, as generally seen in EAs. This leads us to conclude that our system for learning a vector representation is effective in practice. As an additional sanity check, we consider the learning delta, a specific measure for morphological robot evolution systems with individual learning after 'birth'. The learning delta quantifies the learning capacity of a morphology by measuring the difference in fitness before and after learning. While the concept of learning delta is relatively new, several papers have reported the same behavior in different systems: namely, that the learning delta grows over the course of morphological evolution [24]. Fig. 5b shows that the evolutionary dynamics using the learned vector representation are fully in line with the general pattern.

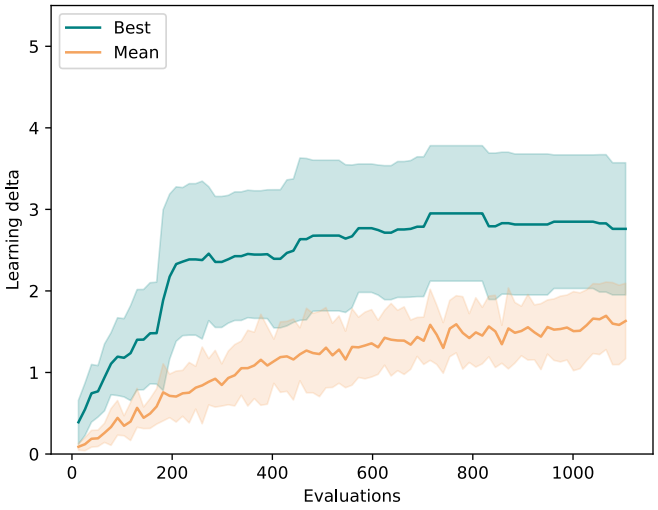
The plots belonging to the CPPN-based evolutionary runs (Fig. 5c and Fig. 5d) show that CPPN outperforms the vector representation. Closer inspection reveals a strange effect: the presence of highly fit individuals in the initial populations of the CPPN experiment. We aim to illustrate this effect in Fig. 6a and Fig. 6b, where we show the nine fittest robots from a random sample of 100 robots for the CPPN and vector representations, respectively. Our vector representation yields a diverse array of morphologies, while the CPPN representation predominantly generates snake-like shapes. Intriguingly, the snake is also the best performing robot at the end of evolution. This is evidenced in Fig. 7b, which displays the fittest morphology from each CPPN experiment. These robots execute a distinct curling-and-uncurling movement pattern that does not transfer well to other morphologies, even if they are similar. We suspect that the snake is located on a steep local maximum surrounded by strong attractors, which usually makes it hard to find by evolutionary algorithms. On the other hand, while the experiments using the learned vector representation also converge to a uniform behavior (rolling), the resulting morphologies are significantly more diverse, suggesting a wider fitness peak that comprises many morphologies. This leads us to believe that CPPN's superior performance is primarily attributed to its inherent bias towards the snake morphology.

## IV. Discussion

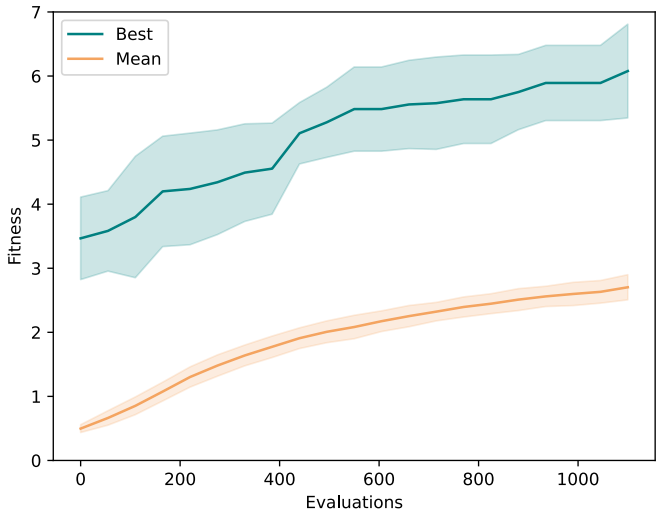
The experiments provided a positive answer to our main question: is learning a genotype-phenotype mapping using VAEs for morphological robot evolution possible? The core evidence is in the curves of Fig. 5a showing that evolution works as expected. To appreciate this, recall that the representation used here was not hand-designed, but algorithmically



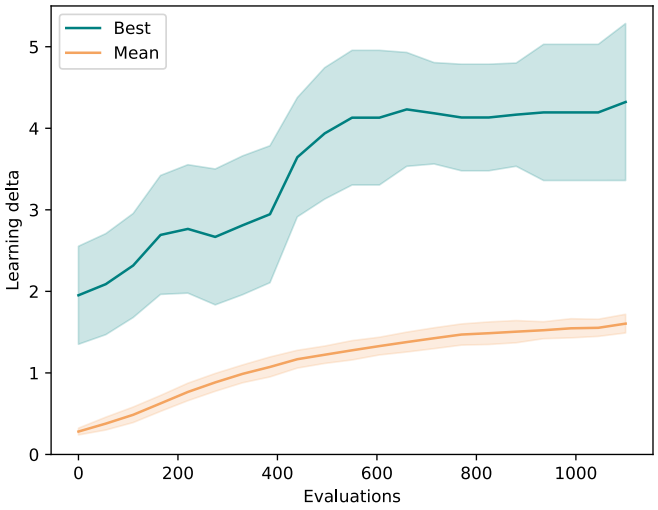

(a) Fitness during evolution using learned vector representation




(b) Learning delta during evolution using learned vector representation

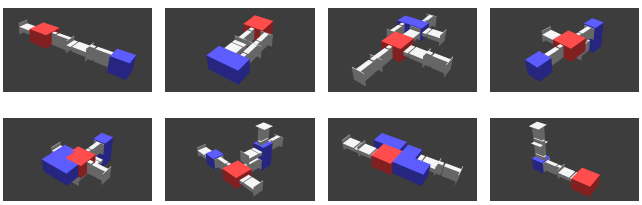



(c) Fitness during evolution using CPPN representation

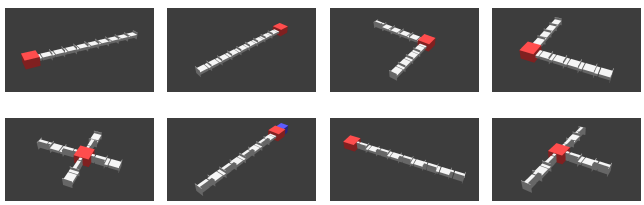



(d) Learning delta during evolution using CPPN representation

Fig. 5. (a & b) An evolutionary process using the learned vector representation yields morphologies with progressively higher fitness and learning ability. (c & d): The CPPN representation can already produce morphologies with high fitness and learning ability in the initial population (see Section III-B for further analysis). "Best" refers to the most fit robot encountered up to the current number of evaluations. "Mean" represents the average among the individuals in the current population or, in the case of CMA-ES, the pool of candidate solutions.

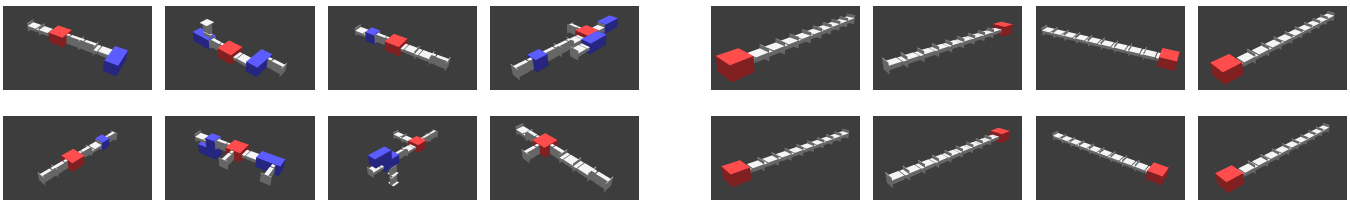


(a) Best morphologies out of 100 random vector representations



(b) Best morphologies out of 100 random CPPN representations

Fig. 6. The best performing robots out of 100 randomly sampled genotypes. (a) The learned vector representation yields a diverse array of robots. (b) The CPPN representation predominantly generates snake-like shapes.



(a) Best morphologies from each experiment using the learned vector representation

(b) Best morphologies from each experiment using the CPPN representation

Fig. 7. The best robots from multiple independent evolutionary experiments. (a) The learned vector representation led to a uniform behavior, but with diverse morphologies. (b) The CPPN representation produced a snake morphology in all trials.

generated; all the user has to do for this is to specify the length (dimensionality) of the genotype vectors, then apply an off-the-shelf learning method. This is a potentially very impactful approach, especially for evolving complex objects (like a robot body), where there are no straightforward representations to use.

In Introduction, we expressed our expectation that the learned genotype-phenotype mapping scores well on the locality criterion. Fig. 3 confirmed this. While the dot cloud does not form a clear band along the diagonal, it shows that close neighbors in the genotype space are also close in the phenotype space.

We see several possibilities to extend the main idea behind this study. The most promising line of research concerns the addition of different criteria to define the quality of a genotype-phenotype mapping and include these in the learning process. Specifically, we are thinking of locality (how well do nearby genotypes map to nearby phenotypes), coverage (how much of the phenotype space can be accessed through the mapping), and representation bias (how much does the mapping steer towards certain regions in phenotype space).

## References


[1] S. Nolfi and D. Floreano, *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[2] P. A. Vargas, E. A. Di Paolo, I. Harvey, and P. Husbands, *The Horizons of Evolutionary Robotics*. The MIT Press, 2014.

[3] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. Eiben, "Evolutionary robotics: what, why, and where to," *Frontiers in Robotics and AI*, vol. 2, p. 4, 2015.

[4] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms, 2nd edn*. Springer Berlin, 2006.

[5] *How Different Encodings Affect Performance and Diversification when Evolving the Morphology and Control of 2D Virtual Creatures*, ser. Artificial Life Conference Proceedings, vol. ALIFE 2020: The 2020 Conference on Artificial Life, 07 2020.

[6] K. Miras, "Constrained by design: Influence of genetic encodings on evolved traits of robots," *Frontiers in Robotics and AI*, vol. 8, p. 177, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2021.672379

[7] F. Veenstra, M. H. Olsen, and K. Glette, "Effects of encodings and quality-diversity on evolving 2d virtual creatures," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '22. Association for Computing Machinery, 2022, p. 164–167.

[8] M. Angus, E. Buchanan, L. K. Le Goff, E. Hart, A. E. Eiben, M. De Carlo, A. F. Winfield, M. F. Hale, R. Woolley, J. Timmis, and A. M. Tyrrell, "Practical hardware for evolvable robots," *Frontiers in Robotics and AI*, vol. 10, 2023.

[9] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[10] *RoboGen: Robot Generation through Artificial Evolution*, ser. Artificial Life Conference Proceedings, vol. ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems, 07 2014.

[11] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[12] A. Eiben, N. Bredeche, M. Hoogendoorn, J. Stradner, J. Timmis, A. Tyrrell, and A. Winfield, "The triangle of life: Evolving robots in real-time and real-space," in *Proc. of the 12th European Conference on the Synthesis and Simulation of Living Systems (ECAL 2013)*, P. Lio, O. Miglino, G. Nicosia, S. Nolfi, and M. Pavone, Eds. MIT Press, 2013, pp. 1056–1063.

[13] M. Pawlik and N. Augsten, "Efficient computation of the tree edit distance," *ACM Trans. Database Syst.*, vol. 40, no. 1, mar 2015.

[14] ——, "Tree edit distance: Robust and memory-efficient," *Information Systems*, vol. 56, pp. 157–173, 2016.

[15] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[16] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, p. 131–162, jun 2007.

[17] S. Kriegman, D. Blackiston, M. Levin, and J. Bongard, "A scalable pipeline for designing reconfigurable organisms," *Proceedings of the National Academy of Sciences*, vol. 117, no. 4, pp. 1853–1859, 2020.

[18] J. Luo, A. C. Stuurman, J. M. Tomczak, J. Ellers, and A. E. Eiben, "The effects of learning in morphologically evolving robot systems," *Frontiers in Robotics and AI*, vol. 9, 2022.

[19] B. Paaßen, K. I., and K. Yacef, "Recursive tree grammar autoencoders," *Mach Learn*, vol. 111, pp. 3393–3423, 2022.

[20] W. Böhm and A. Geyer-Schulz, "Exact uniform initialization for genetic programming," in *Foundations of Genetic Algorithms 4*, R. K. Belew and M. Vose, Eds. San Francisco: Morgan Kaufmann Publishers, 1997, ch. 19, pp. 379 – 407.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] N. Hansen, "Cma python package," version 3.3.0. [Online]. Available: https://github.com/CMA-ES/pycma

[23] G. Lan, M. De Carlo, F. van Diggelen, J. M. Tomczak, D. M. Roijers, and A. Eiben, "Learning directed locomotion in modular robots with evolvable morphologies," *Applied Soft Computing*, vol. 111, p. 107688, 2021.

[24] K. Miras, M. De Carlo, S. Akhatou, and A. Eiben, "Evolving-controllers versus learning-controllers for morphologically evolvable robots," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2020, pp. 86–99.