A Hox Gene Inspired Generative Approach to Evolving Robot Morphology

Eivind Samuelsen
Institute of Informatics,
University of Oslo
Postboks 1080 Blindern
0316 Oslo, Norway
eivinsam@ifi.uio.no

Kyrre Glette
Institute of Informatics,
University of Oslo
Postboks 1080 Blindern
0316 Oslo, Norway
kyrrehg@ifi.uio.no

Jim Torresen
Institute of Informatics,
University of Oslo
Postboks 1080 Blindern
0316 Oslo, Norway
jimtoer@ifi.uio.no

ABSTRACT

This paper proposes an approach to representing robot morphology and control, using a two-level description linked to two different physical axes of development. The bioinspired encoding produces robots with animal-like bilateral limbed morphology with co-evolved control parameters using a central pattern generator-based modular artificial neural network. Experiments are performed on optimizing a simple simulated locomotion problem, using multi-objective evolution with two secondary objectives. The results show that the representation is capable of producing a variety of viable designs even with a relatively restricted set of parameters and a very simple control system. Furthermore, the utility of a cumulative encoding over a non-cumulative approach is demonstrated. We also show that the representation is viable for real-life reproduction by automatically generating CAD files, 3D printing the limbs, and attaching off-the-shelf servo motors.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Design, experimentation

Keywords

Evolutionary robotics, morphological evolution, embryogenic encoding

1. INTRODUCTION

Currently, both rigid body simulators and prototyping tools such as 3D-printers are becoming more advanced, and at the same time more accessible. This opens up new opportunities in automated robot design, as simulations are quicker to do and real-world verification becomes much easier. This is certainly the case when designing robot morphology using evolutionary algorithms, which requires a large number of evaluations, either through simulation or prototypes, or both.

Automated robot design without a fixed topology introduces an encoding challenge, as more complex data structures are needed to describe the space of possible solutions. Thus, the research into automated robot design using evolutionary algorithms has produced a wide variety of coding schemes for describing morphology.

In [14] a tree-based structure is used, where each node represents one of several possible standard modules like joints or variable sized rigid elements. A wide variety of tasks involving both fixed-position robots and mobile robots are optimized for, starting from initial populations of hand-made designs. Here, control is not evolved, but rather handled using hand-written inverse kinematics routines in combination with task-specific logic.

In Lipson and Pollack's groundbreaking work [15], simulated locomoting robots with co-evolved morphology and control were reproduced and tested in real life. The reproduction was done using 3D-printing and standard actuator components. The encoding used is direct, and very low-level: The control system is a fully connected, variable size artificial neural network and the morphology is represented as a number of straight bars connecting a variable number of points in space. All bars could form either rigid elements or linear actuators. While this approach succeeded in creating real-life robots capable of forward locomotion, the encoding lacked support for the kind of modularity and structural regularity seen in nature.

In generative and developmental systems, complex phenotypes are generated from simpler genotypes through some sort of indirect encoding. This encoding is used as a developmental program, or set of instructions that can be evaluated to produce the phenotype. These encodings are highly productive - they can produce highly complex and structured results from very terse information [9] and are known to outperform direct encoding in simple morphology optimization tasks [13].

In the first and perhaps most well-known example of generative co-evolution of morphology and control [18], Sims evolves animal-like creatures in a virtual environment using a graph-based representation where a directed graph is used to describe morphology. Each node in the graph represents a part design and each edge represents an instantiation of the part the edge points to. By starting from a root node and

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

following the edges repeatedly, allowing cycles in the graph, advanced morphologies with repeating or fractal structures are achieved. The control system is explained in terms of a neural network, but with nodes that can represent a wide range of mathematical functions. A variety of behaviors like walking, swimming and following are evolved.

Much research has been done on how DNA specifies the growth and shape of living creatures. This research has, for practical reasons, primarily been focused on a species of fruit fly, the *drosophila melanogaster*. An *drosophila* egg is polarized during creation by a gradient of different proteins along the anterior-posterior (head-tail) axis, forming the basis for segmenting the body into different sections [17]. The growth of these sections is then regulated by Hox genes. These genes lie in a handful of tight clusters in the DNA, and are usually colinear - they appear in the same order in the chromosome as they are expressed in the embryo from anterior to posterior.

Hox genes and other development-related genes have previously inspired artificial genetic regulatory networks (GRNs), a developmental process emulating gene regulation on a very fine level that has evolved complex robots capable of locomotion [4]. An L-system/GRN hybrid, also inspired by how Hox genes regulate segment development, was used in [1] to evolve insect-like robots with fixed limb morphologies and a simple co-evolved control system.

One of the most famous Hox gene studies is that of the Bithorax Complex, the gene cluster that regulates the rear end of the *drosophila* body. It has been shown that the effect of section specific regulators in this gene cluster stack up as one moves backwards along the body: in addition to the regulators associated with the section itself, the regulators associated with all anterior sections are also active [16].

It is also expected that the Hox genes themselves do not contain the entire blueprint of the section, but rather regulate control parameters or switches that affect more detailed developmental programs stored elsewhere in the genome [16, 6]. Intuitively this makes sense, because it allows the same basic design, of a wing or a leg for example, to be repeated with some variation down along the body.

The default, then, would be for each section to be identical, and then become differentiated later as the developmental program grows more complex, as seen in [6]. This seems similar to the concept of symmetry breaking [22]: left and right side would be mirrors of each other until the developmental program takes a parameter varying from left to right into use, and front and back limbs would be identical until otherwise specified.

This paper proposes an approach for encoding animal-like robot morphologies in a generative, biologically inspired manner by separating the genotype information into two separate parts. The two parts can be thought of as providing a separate high- and low-level description of the morphology and control system, with all sections being constructed by reusing the same developmental program with different parameters. Based on biological precedent and supported by results in evolutionary robotics, bilateral symmetry is implicitly enforced in the encoding. Altogether, this enables a compact and structured method of encoding the morphology of a limbed robot.

It is then demonstrated that this approach can be implemented in a fairly simple encoding scheme that can be used to evolve robots capable of stable forwards locomotion. The

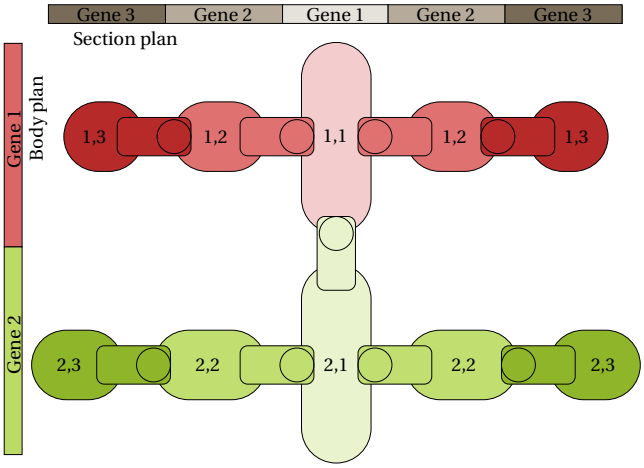


Figure 1: Influence of section plan and body plan genes, with the section plan encoded as a list of per-segment parameters. As the numbering and shading illustrates, the right and left limb segments are mirror images created from the same genetic information.

encoding is also designed with real-life reproduction in mind, and generates designs that are restricted to realistic configurations and physical dimensions for reproduction using only 3D-printing and commercial off-the-shelf components.

The remainder of this paper is organized as follows: In Section 2 the general approach is explained, and the specific encoding used in the experiments is described along with the experimental setup. The results of the experiments are shown in Section 3. Section 4 contains discussion of the results, before a final conclusion is given in Section 5.

2. METHODS

The interpretation of Hox genes in the previous Section suggest a simple bioinspired model for encoding morphological information: the body is divided into a linear sequence of sections, each of which is developed from the same developmental program, or *section plan*, but differentiated by a high-level *body plan*, as shown in Figure 1. The body plan has a gene for each section in the body. Each of these genes specifies control parameters that decide the outcome of evaluating the section plan when located in the section corresponding to the gene.

For each section, all genes up to and including the corresponding gene are evaluated in order, each gene modifying some parameters, while completely overwriting others. In this way, the parameters may be differentially coded. Overwriting can be thought of as the last gene repressing the expression of the previous ones. Modification can be likened to having each gene make a contribution to the production of some protein, which then regulates the expression of the parameter in question. When a parameter is modified and not overwritten, we say that it is coded *cumulatively*.

For example, if the physical dimensions are coded this way, then the dimensions of a segment or a whole section would depend on the previous one, and the physical dimensions of the first one would decide the physical scale of the entire robot. This may be an evolutionary advantage, as it would make single mutations affect shape more smoothly.

Genotypes that exhibit this kind of two-stage organization could certainly develop during evolution in a variety of de-

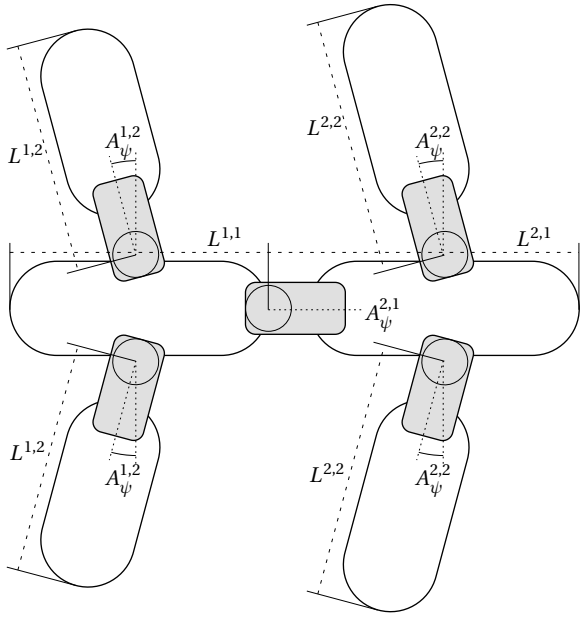


Figure 2: Schematic view of the physical parameters of a simple robot. The axis-of-rotation angle $A_\phi^{i,j}$ is not shown, but would have the effect of rotating the segment along with its joint around its major axis. The rotation of the rest angle $A_\psi^{i,j}$ is then applied around the resulting axis of rotation. The shaded areas symbolize the actuators.

velopmental encodings such as GRNs, or L-system or GP-based methods. However, given the prevalence of Hox genes in nature, designing the genome with two explicit levels of description might be a reasonable trade-off between generality and design efficiency when it is acceptable to restrict the search to animal-like topologies.

For the purpose of providing a minimal proof of concept of this encoding model, a minimal realization, to be described in the Subsection below, is tested in a gradient following optimization task. Control system parameters are also embedded in the encoding so that the control system is co-evolved with the morphology. These parameters will be discussed in detail in Subsection 2.2.

Here, the section is encoded as a simple list of segments, similar to the scheme seen in Figure 1. It has a gene for the core of the section (the "spine" of the segment), plus one for each segment in the limb, extending from both sides of the spine. Like in the body plan, the parameters stored in the section plan genes can be cumulative.

Both the top-level body plan and the section plan is represented as a variable-length vector. Each gene contains a historical marker and a set of parameters which are different for the body plan and the section plan. The historical markers are used to implement an efficient crossover operator in the same way as in the NEAT algorithm [19], and are discussed further in Subsection 2.4.

2.1 Physical Parameters

The physical configuration of each segment in the robot is decided by three variables: The length $L^{i,j}$, the angle of the axis of rotation $A_\phi^{i,j}$, and the rest angle $A_\psi^{i,j}$, where i is the section number and j is the segment number ($i = 1$ is the first section, and $j = 1$ is the spine). The effect of these parameters will be described in detail below. A schematic view of a simple robot is shown in Figure 2.

For each segment, a major axis is defined. If it is the first segment of a limb ($j = 2$), this axis is perpendicular to the major axis of the previous segment; otherwise it is parallel to it. $L^{i,j}$ is measured along the major axis of the segment, from the tip of the segment to the tip of the previous segment, or from a certain offset from the center of the previous segment for $j = 2$.

The previous segment of the spine of a section is defined to be the spine of the previous section. In that way the body of every robot consists of a connected tree of segments with the spine as the main branch and the rest of the genes in the section plan creating two branches shooting off from the main branch for each section. Each node in this tree represents a rigid body part and each edge represents a single revolute actuator in the robot.

The axis of rotation of the revolute joint connecting the segment to the previous segment is perpendicular to the major axis and has an angular offset of $A_\phi^{i,j}$ to the axis of rotation of the previous segment, or to the axis that is orthogonal to the major axis of both segments for $j = 2$. To avoid twisting of the spine that would interfere with the bilateral symmetry, $A_\phi^{i,j}$ is always zero for $j = 1$. The rest angle $A_\psi^{i,j}$ adds a constant offset to the actuator position as seen by control system, affecting the midpoint of oscillations as well as the initial position.

While A_ϕ and $A_\psi$are coded directly in the section plan, L is coded indirectly across both body and section plans. Each gene in the section plan hold a parameter L_s that represents the base length of the segment, the two angle parameters and a set of parameters for the control system of the corresponding actuator, which will be detailed in the following subsection.

Each gene in the body plan contains two parameters L_b and L_l that affect the segment length. L_s affects the length of all parts of the section, including the spine, while L_l only affects the limb segments. The physical length of the jth segment on section i is calculated recursively as

$$L^{i,j} = \begin{cases} L^{i-1,j} \times L_b^i \times L_s^j & j = 1 \\ L^{i,j-1} \times L_l^i \times L_s^j & j = 2 \\ L^{i,j-1} \times L_s^j & j \geq 3 \end{cases} \quad (1)$$

The recursion terminates to $L^{0,1}$, which is used as a global parameter that sets the base scale of the robots.

To test the effect of the cumulativeness of the length formula, an alternative method of calculation, where cumulative effects are minimized, is also tested. This method uses the exact same parameters, and is given as

$$L^{i,j} = \begin{cases} L^{0,1} \times L_b^i \times L_s^j & j = 1 \\ L^{0,1} \times L_b^i \times L_l^i \times L_s^j & j \geq 2 \end{cases} \quad (2)$$

There are no parameters in the genotype for introducing left-right asymmetry to the body - the left and right sides are simply mirror images of the same section plan. Bilateral symmetry is all but universal among animals, so there is substantial biological basis for ignoring left-right symmetry breaking. There are also results in evolutionary robotics indicating that bilaterally symmetric robots have more efficient movement patterns [5]

In the experiment, the physical dimensions as well as the stall torque and maximum angular velocity of the actuator

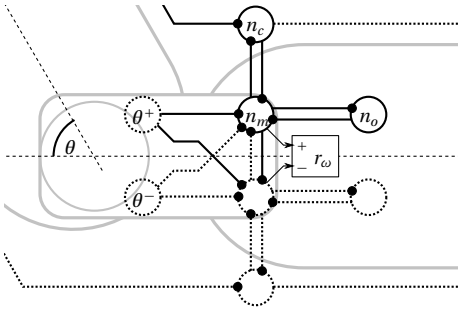


Figure 3: The central pattern generator between two links. The neurons and synapses that are directly encoded for that actuator is shown in solid lines. Dashed neurons and synapses are generated by mirroring or by connecting the network together with the previous segment. The dashed nodes marked θ^+ and θ^- are sensor nodes that read positive and negative actuator angles respectively. The node n_m and its mirror node together control the angular velocity setpoint r_ω of the actuator.

box is modeled by the ROBOTIS AX-18A servo motor with dimensions 32 mm × 50 mm × 26 mm, stall torque of 1.8 Nm and maximum angular velocity of 10.15 rads^{-1}. The capsules have a radius of 20 mm and $L^{0,1}$ was set to 100 mm.

2.2 Control System

The control system used for the robots is a recurrent neural network based on central pattern generator (CPG) modules. Chains of symmetric CPGs with parameters optimized by evolutionary algorithms have been shown to effectively replicate gaits of certain reptiles [11] and such networks have also been successfully applied to control robots of various topologies [7, 21].

The neuron model used here is the average firing rate model given by the equation

$$\tau_i \frac{d\,m_i}{d\,t} = -m_i + \sum_j w_{i,j}\, x_j\,, \quad x_i = \left(1 + e^{-(m_i+b_i)}\right)^{-1}$$

Here m_i and x_i are the average membrane potential and firing rate of neuron i, and τ_i and b_i are the decay time and membrane potential bias of that neuron. $w_{i,j}$ is the strength of the synapse from neuron j to neuron i. This model is used with a network with one module per actuator, each having a fixed topology containing feedback paths that control actuator velocity based on actuator angle and an input signal, as shown in Figure 3.The input neurons of each module are connected to those of the parent module, transmitting a control signal from the head throughout the body. The input signals of the first section come from two "antennae" that encode the target heading as

$$a_l = \frac{1}{2} + \frac{\alpha}{2\pi}, \quad a_r = \frac{1}{2} - \frac{\alpha}{2\pi}$$

where $\alpha \in (-\pi, \pi]$ is the angle between the direction of the spine segment of the first section and the target heading, both projected to the ground plane. If no target heading is set, α is set to zero.

The parameters for each module are stored in the genotype along with the physical parameters of the corresponding gene in the section plan. Like in many CPG based controllers (see for example [11, 21]), the actuator control and feedback signals are coded differentially and interact with a symmetric control network, as shown in Figure 3. This emulates the way most single-axis joints in animals are actuated by a pair of muscles that work in opposite directions. Each half of the control network consists of three neurons: a control neuron (n_c), a motor neuron (n_m) and an oscillator neuron (n_o).

The motor neuron has incoming synapses from both control and oscillator neurons, as well as from the opposing motor neuron and positive and negative angle state feedback from the actuator. The control neuron has incoming synapses from the motor neuron and the control neuron on the same side on the previous segment, while the oscillator neuron only has a single incoming synapse from the motor neuron. Since each neuron has two parameters (time delay τ and bias b) and each synapse has a weight parameter w, each network module has a total of 14 real-valued parameters.

In the initial population, all networks are configured with a strong negative feedback with a high time delay between motor and oscillator neurons to create oscillations and strong inhibiting synapses between the two motor neurons to make the two sides of the module act antisymmetrically. The motor and control neurons have weak excitatory connections to each other so that they influence each other. This enables the control signal to be modified by the motor neurons as it spreads throughout the tree of CPGs.

All parameters in the network are then allowed to mutate freely, which might change the amplitude, phase or frequency of the oscillations, or even break the oscillatory properties of the network entirely.

2.3 Evaluation

Each robot is evaluated by measuring the robot's performance in following a path defined by a sequence of points, a typical use case for a mobile robot. The need for more advanced skills is gradually introduced, and in this aspect the evaluation is conceptually similar to the scaffolding schedules used in [2].

The robot is placed on a flat surface with the axis of the first spine segment aligned with the x-axis. The robot is given 0.5 s to settle before a target point four meters in front of it along the x-axis is put in place. The robot then has 16 s to get as close as possible to that point.

The performance is calculated as how far the robot has progressed along the route to the target point. Since the first target point is 4 m away, it is calculated as $4-d$, where d is the euclidean distance from the tip of the robot to the next target point. If it reaches within 0.1 m of the point, a new goal is set an additional two meters along the x-axis and two meters along the y-axis, and the robot given a score of $4+\sqrt{8}-d$.

This way, robots are first encouraged to achieve forward locomotion. As the locomotion gets more effective, greater precision in heading are required in order to keep closing in on the target point. After a certain level of efficiency and precision have been achieved in moving in the direction the body is initially aligned in, the robot will have to take a controlled turn at a certain point in order to keep improving its score substantially.

Since each individual is only simulated for a few seconds, it is not expected that the robots evolved in this experiment should make it past the first target point. The second point is added primarily as a contingency in the evaluation procedure, and also makes it more difficult for chaotic behavior

to propel robots quickly straight forward and thus dominate the population.

The rigid body physics are simulated using the Nvidia PhysX engine. The simulation is run with a fixed timestep of $1/128$ s $\approx$ 8 ms. The joint constraint solver is set to 100 iterations with an extrapolation factor of 0.75 to ensure good accuracy in the joint constraints. All surfaces are treated as having the same material, with a static and kinetic friction coefficient of 0.60 and 0.54 respectively.

Each segment is modeled as a fixed-size box-shaped actuator and a variable length capsule. The box is assigned a mass of 55 g and the capsule is modeled as hollow on the inside, with the outer 20% of its volume made of a PVC-like material with a specific gravity of 1.2 kgl^{-1}. Skin width, the amount rigid shapes are allowed to interpenetrate to improve simulation stability, is set to 2 mm for the capsules and 1 mm for the actuator boxes and the ground plane.

Using the distance performance evaluation alone would result in evolutionary pressure towards increasingly large robots with an increasing number of actuators, since larger bodies enable the robot to travel farther with each swing of a limb, and more actuators can give the robot more power. To counteract that, two secondary objectives are introduced, one minimizing the number of actuators and another minimizing the size of the robot, measured as the largest dimension of the axis-aligned bounding box of the robot in its initial pose.

To avoid emphasizing these measures on very simple robots, any robot with fewer than four actuators is treated as having four, and any robot with a size less than 0.2 m is treated as being of that size. Robots with more than 21 actuators or a size greater than 1 m is not simulated but get a performance score of -10 assigned to them instead, marking them as infeasible.

Additionally, if any part of the robot is subjected to a contact force greater than 1000 N the evaluation is immediately terminated, and a performance score of -20 is given. Any force of that magnitude is assumed to be caused by the robot trying to exploit weaknesses in the simulator to propel itself around, and thus the result is invalidated. Different negative scores are chosen for different reasons for disqualification so that the reason can be easily read out from fitness output.

2.4 Evolution

As explained above, the evolved designs are evaluated using three objective functions, making the design problem a multi-objective optimization problem. The multi-objective evolutionary algorithm NSGA-II [8] is used to execute the evolutionary process. NSGA-II ranks the parent and offspring population together using non-dominated sorting to separate the population into ranks, and then differentiates within ranks based on a diversity measure called crowding distance. Survival selection and parent selection is then done by truncation and binary tournaments respectively.

As explained in subsections 2.1 and 2.2, the genotype consists of two variable length vectors: the body plan and the section plan. The body plan contains two parameters per gene and the section plan contains 17 parameters per gene: three physical parameters and 14 parameters for the control system.

Each gene of both vectors also contains a historical marker. This is a unique number identifying the evolutionary origins of the genes; it never changes during mutation or crossover. After parent selection crossover is done by first cloning a pair of parents. The genes of the two parent clones are matched by their historical markers. All genes that have a match are crossed using discrete crossover: each parameter is swapped with a probability $p = 0.1$. Unmatched genes are kept in the offspring in which they were found.

Table 1: Parameter mutation settings. Initial values marked * are detailed in Subsection 2.2.

	Parameter	Mutation	Initial value	Range	σ
Section	L_b	$\times e^{\mathcal{N}(0,\sigma^2)}$	1	-	0.3
	L_l	$\times e^{\mathcal{N}(0,\sigma^2)}$	1	-	0.1
Segment	L_s	$\times e^{\mathcal{N}(0,\sigma^2)}$	1	-	0.3
	A_ϕ	$+\mathcal{N}(0,\sigma^2)$	0	$[-\pi,\pi)$	0.1
	A_ψ	$+\mathcal{N}(0,\sigma^2)$	0	$[-\frac{\pi}{3},\frac{\pi}{3}]$	0.1
	$w_{i,j}$	$+\mathcal{N}(0,\sigma^2)$	*	-	0.1
	b_i	$+\mathcal{N}(0,\sigma^2)$	*	-	0.1
	τ_i	$\times e^{\mathcal{N}(0,\sigma^2)}$	*	-	0.1

After crossover, mutation is then performed on every new individual as follows: Each preexisting gene in both vectors is either deleted ($p = 0.03$), duplicated ($p = 0.04$) or has its parameters mutated ($p = 0.93$). If the gene is duplicated, then a new gene with a new unique history marker is inserted after this gene with all parameters copied (except for A_ϕ and A_ψ which are set to zero). This is the only time new historical marker values are introduced.

When a gene is mutated each parameter in it has a probability $p = 0.1$ of being changed. Since the various parameters are on different scales and intervals, they are mutated in slightly different ways, as summarized in Table 1. In essence, the parameters encoding angles have a fixed range, and either wraps or clamps to that range, depending on whether it corresponds to a full circle or not. Some of the parameters have a normal random variable added to them, while others are multiplied by a lognormal random variable, depending on whether they represent a property which is assumed to have geometrical growth or not.

The initial population consists of identical designs with one section and one segment, with initial values as shown in Table 1. This gives a neutral and maximally simple starting point for the algorithm. According to complexification [20], the algorithm should then be able to introduce the complexity needed to evolve efficient designs.

3. RESULTS

The multi-objective evolutionary algorithm was run 160 times, each time with a population of 200 individuals over 80 generations. 80 of the runs were done with the cumulative length encoding (equation (1)) and the remaining 80 were done with the noncumulative encoding (equation (2)).

Figure 4 maps the distribution of performance scores and sizes in all final populations with the cumulative encoding. Each individual is counted in one of 24×24 equally sized bins depending on their performance and size, creating a two-dimensional histogram. The count in each bin is then divided by the number of samples in total to get the proportion of the individuals counted in each bin.

This number can be seen as an estimate of the probability of a randomly selected individual from any run of the algorithm being situated in that area of the objective space. It

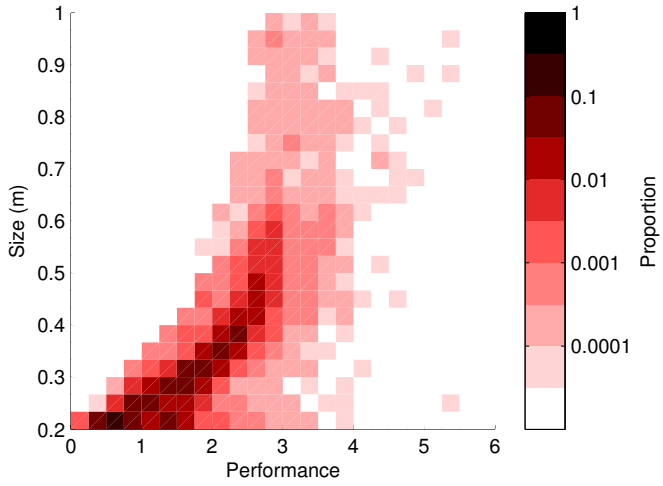


Figure 4: Distribution of performance scores and sizes of all individuals in the final populations of all runs with cumulative length encoding. The proportions are color-coded logarithmically to visualize the full dynamic range of the distribution.

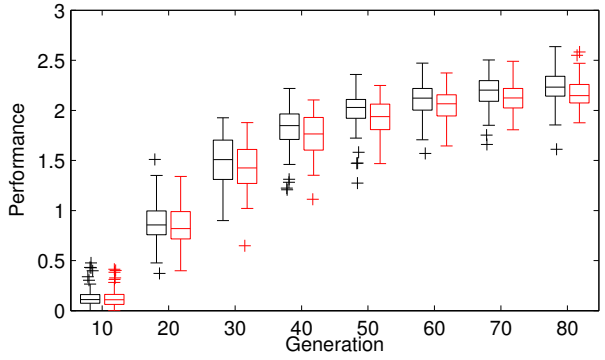
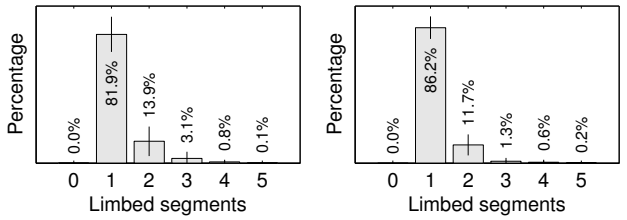


Figure 5: Box plot of the fourth quintile performance score of each population across all runs with cumulative encoding (black) and noncumulative encoding (red).

can be seen that most solutions are relatively small in size and lie in a oblong cluster that extends further along the performance axis as the size increases. The corresponding plot for the noncumulative runs, which is not shown, displays the same general pattern.

Figure 5 shows the distribution of locomotion performance scores of the 16th best ranked individual (the fourth quintile) in that objective in different generations. The reason for not choosing the best performing individual is that it will tend too much towards extreme sizes and numbers of actuators, and will have a variation between runs that is not representative of the population in general. The box plot can also be read as illustrating how large proportion of the runs where 20% of the final population had a performance greater than or equal to a certain number.

It can be seen from the plot that the cumulative encoding performs slightly better with this metric, and t-tests between same-numbered generations with the two variants show a statistically significant difference for most generations after the 40th (the p-values average 0.0225 after this point, with three values above 0.05).



(a) Cumulative lengths (b) Noncumulative lengths

Figure 6: Distribution of the number of limb pairs of the 200×80 individuals after the 80th generation. Lines represent $\pm$ one standard deviation from the mean across runs.

The distribution of number of limbed sections are shown in Figure 6. Both cumulative and noncumulative encodings predominantly produce robots with a single pair of limbs, although roughly one eighth of the evolved robots are quadruped, and a small percentage are hexaped as well. The average proportion of both biped and hexaped solutions are significantly different between the two variants. Furthermore, the average number of limb pairs for a final population is also significantly different ($p = 0.0047$), with an average of 1.228 for the cumulative encoding, and 1.167 for the noncumulative encoding.

Most of the biped robots have evolved with a symmetrical dragging gait, as shown in Figure 7. The robots with more than one pair of limbs, on the other hand, have generally evolved antisymmetrical trotting gaits, where each limb is in opposite phase to the opposite limb, as well as the limb before and after it on the same side. An example of this is shown in Figure 8.

However, a significant portion of the robots do not have any gait at all, but rather rely on exploiting weaknesses in the simulator to propel themselves forward, like the robot in Figure 9. The exact proportion of simulator-exploiting robots is hard to quantify, precisely because we were unable to automatically detect them. However, a manual sample indicated that the results with eight or more limbs were almost exclusively of this type.

Figure 8 also shows a real-life reproduction of the same design. This reproduction was created by generating CAD files by automatically compositing hand-designed actuator mounting points[1] with parametrically generated capsule parts based on the phenotype parameters of each segment. Servos were then attached to the resulting 3D-printed parts.

4. DISCUSSION

As seen above, the algorithm used in the experiments produces predominantly biped designs, most using variants of the dragging gait shown in Figure 7, with most of the remaining population consisting of quadruped designs. The prevalence of these biped designs might be explained by the simplicity of the evaluation environment. The ground is composed of a single planar surface with no irregularities or changes in material, which might make evolution disregard the stability and robustness afforded by locomoting using more than one pair of limbs.

A different environment-related factor which might contribute to the predominance of the biped dragging robots

[1]The models that were used can be found at `http://folk.uio.no/eivinsam/data/hoxinspired/`

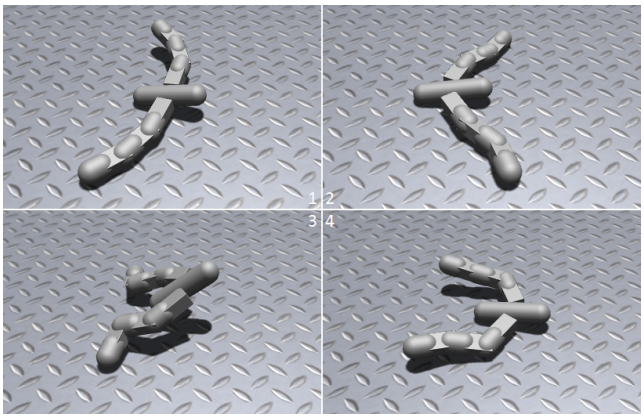


Figure 7: An evolved robot with a dragging gait.

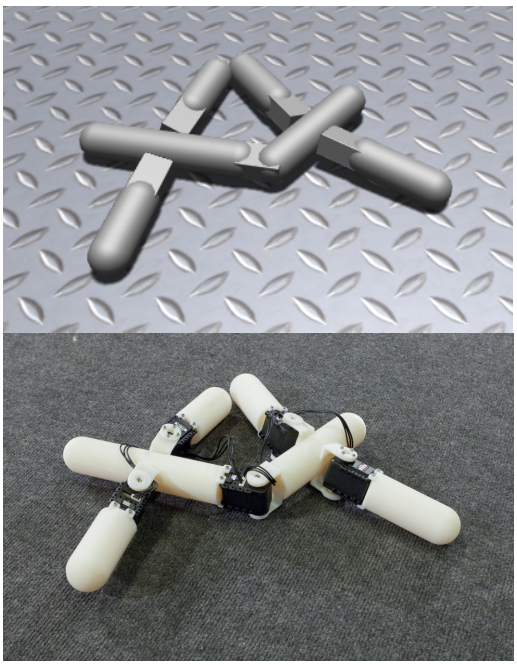


Figure 8: An evolved robot with a trotting gait, in simulation (above) and its real-world replica (below).

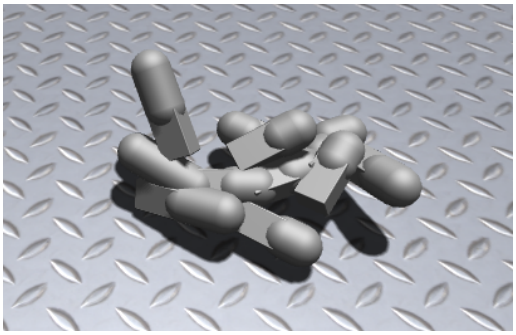


Figure 9: Typical robot evolved to exploit weaknesses in the simulator. Several limbs are locked inside each other, making the simulator apply large forces on the limbs, sometimes making the robot hover around chaotically.

is the material properties chosen. While the parameters for the robot design itself were quite detailed, some of the parameters for the environment were less certain. In particular, the friction coefficient between robot and floor might be too high, and thus distort the fitness landscape to a large degree. This would favor the designs with a dragging gait, which can use the high friction to latch on to the surface with the ends of their limbs, so that they can move forward in large strides.

Comparing the cumulative encoding with the noncumulative one, the results show that the cumulative encoding produces a small, but statistically significant increase in both performance, as well as the number of limbs on average. The increase in number of limbs seems to indicate that the cumulative encoding facilitates the evolution of complex designs better. In these experiments, the extent of the cumulative effects was limited to the physical size of the robot, and the evaluation scenario was of limited complexity. However, with more advanced scenarios, or cumulative effects on more parameters, the effects of this aspect of the approach presented here might become even more pronounced.

As Figure 4 shows, the proportion of final designs that are able to reach the first target point (performance score ≥ 4) is very small. However, some individuals are able to travel a considerable distance towards the second target point. Since both target points have a fixed position across all evaluations, it is unclear whether these solutions have evolved to actively use the heading sensor input, or the control system has evolved a fixed program that is followed regardless of sensory input.

Simulator instability is a common problem in evolutionary algorithms, since the evolutionary search is often very efficient at finding the weak points of the simulator. In this case, the rigid body simulation did not have continuous collision detection, and so the algorithm can discover configurations where the robot slips into impossible poses from one time step to the next, and then get locked there, as shown in Figure 9.

This would then create large forces, enabling robots to hover frictionlessly a small distance above the ground or suddenly jolt into the air. Some of these robots would hover in the direction of the target point, and thus distorting the population of that run. It was possible to disqualify the most explosive of these by enforcing a maximum limit on contact forces, but this tweak was not able to filter out all of the misbehaving robots.

5. CONCLUSIONS AND FUTURE WORK

In this paper, a general, biologically inspired approach for encoding co-evolved morphology and control is presented, with two main features: The separation of the genotype into two levels of description linked to two different physical axes of development, and a cumulative method of encoding, where the phenotype of each section of the robot body is a function of not only its own parameters, but that of all previous sections as well. A simple encoding based on this approach is used to evolve locomotion in a simulated environment.

The initial evolutionary results with basic forward locomoting robots are promising; however more work on the representation will be required for more efficient robots with morphologies not exploiting simulator instabilities. It is shown that the cumulative encoding gives better fitness than an alternative non-cumulative encoding, and it also tends to

produce more complex robot morphologies. By automatically generating the CAD files for one instance of the evolved robots, and then 3D printing and assembling it with servo motors, we show that the proposed encoding is suitable for transferring the results to real life robots.

For future experiments we will explore how the designs and control systems produced by the algorithm behave in the real world, extending the proof-of-concept given here. Given the known reality gap issue in evolutionary robotics [12], i.e. robots behaving differently when transferred to the real world due to shortcomings of the simulator, we would like to experiment with adaptation techniques after transferal. This would involve a form of lifetime learning where the real life robot adapts or re-learns the control system to the new environment, given the now fixed body.

In the general Hox-gene-inspired approach presented here, the section plan has intentionally been defined in very loose terms. In this proof-of-concept it is represented by a simple linear structure, but in principle any direct or indirect encoding capable of taking a set of additional control parameters can be used. Using techniques from genetic programming, the section plan could be coded as a developmental program, with control parameters from the body plan used both to control program flow and as parameters in numeric calculations. Likewise, genetic regulatory networks, or compositional pattern producing networks such as those evolved in [3] could be instantiated with different conditions or inputs to produce different section designs.

It also seems obvious that in order to evolve robots capable of locomoting robustly in the real world, one should introduce a more complex environment than the perfect plane used in the current experiments. We would like to introduce areas with obstacles and different friction to promote more robust gaits [10] or more complex designs [3].

6. REFERENCES

[1] S. Adams. Hox genes, diversity and adaptation in evolutionary robotics. Master's thesis, University of Plymouth United Kingdom, 2009.

[2] J. Auerbach and J. Bongard. How robot morphology and training order affect the learning of multiple behaviors. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 39 –46, may 2009.

[3] J. E. Auerbach and J. C. Bongard. On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 521–528, New York, NY, USA, 2012. ACM.

[4] J. Bongard. Evolving modular genetic regulatory networks. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1872–1877. IEEE, 2002.

[5] J. Bongard and C. Paul. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In *From animals to animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, volume 6, page 420. MIT Press, 2000.

[6] S. Carroll, S. Weatherbee, and J. Langeland. Homeotic genes and the regulation and evolution of insect wing number. *Nature*, 375:58–61, 1995.

[7] H. Cruse, C. Bartling, M. Dreifert, J. Schmitz, D. Brunn, J. Dean, and T. Kindermann. Walking: A complex behavior controlled by simple networks. *Adaptive Behavior*, 3(4):385–418, 1995.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182 –197, apr 2002.

[9] G. Hornby and J. Pollack. Evolving l-systems to generate virtual creatures. *Computers & Graphics*, 25(6):1041–1048, 2001.

[10] G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with aibo. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 3040 –3045 vol.3, 2000.

[11] A. Ijspeert. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological cybernetics*, 84(5):331–348, 2001.

[12] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life*, pages 704–720. Springer-Verlag, 1995.

[13] M. Komosinski and A. Rotaru-Varga. Comparison of different genotype encodings for simulated three-dimensional agents. *Artificial Life*, 7(4):395–418, 2001.

[14] C. Leger. *Automated synthesis and optimization of robot configurations: an evolutionary approach*. PhD thesis, Carnegie Mellon University, 1999.

[15] H. Lipson and J. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

[16] R. Maeda and F. Karch. Cis-regulation in the drosophila bithorax complex. *Hox Genes*, pages 17–40, 2010.

[17] R. Rivera-Pomar and H. Jäckle. From gradients to stripes in *Drosophila* embryogenesis: filling in the gaps. *Trends in Genetics*, 12(11):478–483, 1996.

[18] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 15–22, New York, NY, USA, 1994. ACM.

[19] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[20] K. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *J. Artif. Intell. Res. (JAIR)*, 21:63–100, 2004.

[21] G. Taga, Y. Yamaguchi, and H. Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological cybernetics*, 65(3):147–159, 1991.

[22] V. Valsalam and R. Miikkulainen. Evolving symmetry for modular system design. *Evolutionary Computation, IEEE Transactions on*, 15(3):368 –386, june 2011.