# Directed Locomotion for Modular Robots with Evolvable Morphologies

Gongjin Lan, Milan Jelisavcic, Diederik M. Roijers, Evert Haasdijk, A.E. Eiben

Department of Computer Science, VU University Amsterdam, The Netherlands
g.lan@vu.nl, m.j.jelisavcic@vu.nl, d.m.roijers@vu.nl, a.e.eiben@vu.nl

**Abstract.** Morphologically evolving robot systems need to include a learning period right after 'birth' to acquire a controller that fits the newly created body. In this paper, we investigate learning one skill in particular: walking in a given direction. To this end, we apply the HyperNEAT algorithm guided by a fitness function that balances the distance travelled in a direction and the deviation between the desired and the actually travelled directions. We validate this method on a variety of modular robots with different shapes and sizes and observe that the best controllers produce trajectories that accurately follow the correct direction and reach a considerable distance in the given test interval.

**Keywords:** Evolutionary Robotics, Evolvable Morphologies, Modular Robots, Gait Learning, Directed Locomotion.

## 1 Introduction

While it can already be hard to design robots for known environments, it is considerably harder for (partially) unknown environments, like the deep sea or Venus. In unknown environments, robots should be able to respond to the circumstances they encounter. The problem with this however, is that there is no way to predict what the robots will encounter. Therefore, in such environments, it would be highly useful to have robots that evolve over time, changing their controllers and their morphologies to better adapt to the environment.

The field that is concerned with such evolving robots is Evolutionary Robotics [6,10]. To date, the research community has mainly been focussing on evolving only the controllers in fixed robot bodies. The evolution of morphologies has received much less attention even though it has been observed that adequate robot behaviour depends on both the body and the brain [3,4,30]. To unlock the full potential of the evolutionary approach one should apply it to both bodies and brains. At present, we can only do this in simulation, as there are still key obstacles to overcome for evolving robots in real hardware [12,13,21].

One of the challenges inherent to evolving robot bodies – be it simulated or real – is rooted in the fact that 'robot children' are random combinations of the bodies and brains of their parents. In general it cannot be assumed that simply recombining the parents' controllers results in a controller that fits the

recombined body. Hence, a 'robot child' must learn how to control its body, not unlike a little calf that spends the first hour of its life learning to walk. It is vital that the learning method is general enough to work for a large variety of morphologies and fast enough to work within practical time intervals.

A generic architecture of robot systems, where both morphologies and controllers undergo evolution has been introduced recently [11,14]. The underlying model, called the Triangle of Life (ToL), describes a life cycle that runs from conception (being conceived) to conception (conceiving offspring) through three principal stages: Birth, Infancy, and Mature Life. Within this scheme, the learn-to-control-your-own-body problem can be positioned in the Infancy phase, where a newborn robot acquires the basic sensory-motor skills. Formerly, we have investigated the most elementary case: gait learning [21,22,23,35]. However, although gait learning is a popular problem in evolutionary robotics, in practice we are not really interested in a robot that just walks without purpose. For most cases, a robot has to move in a given direction, e.g., to move towards a destination. Here we focus on the task of directed locomotion, where the robot must follow a given direction, e.g. "go left". Our specific research goals are the following:

1. Develop a dedicated evaluation function that balances the distance travelled in a direction and the deviation between the desired and the actually travelled directions.
2. Provide a method to learn a controller for directed locomotion in different modular robots.
3. Evaluate the method on a test suite consisting of robots with different shapes and sizes.

## 2 Related work

The design of locomotion for modular robots is a difficult task. Several approaches based on various types of controllers and algorithms for locomotion of robots have proposed in [1,32]. An early approach is based on gait control tables that in essence are a simple cyclic finite state machines [5]. A second major approach is based on neural networks, for instance, HyperNEAT. In previous work we have implemented evolutionary controllers for locomotion in modular robots [16,35] using HyperNEAT. Other studies also have shown that HyperNEAT can evolve the good controllers for the efficient gaits of a robot [8,36]. Other successful approaches that have been extensively investigated for robot locomotion are based on Central Pattern Generators (CPGs) [19]. CPGs are neural networks that can produce rhythmic patterned outputs without rhythmic sensory or central input [17]. The use CPG-based controllers reduces the dimensionality of the locomotion control problem while remaining flexible enough to continuously adjust velocity, direction, and type of gait depending on the environmental context [20]. This technique has been shown to produce well-performing and stable gaits for modular robots [24,25,27]. Last, an alternative approach based on machine learning for adaptive locomotion was proposed by Cully et al., to account for changes in body properties [9].

Although there are extensive existing studies on the locomotion of robots, most of them focus on the controllers in fixed robot bodies for gait learning, and only the research described in [24,32] tested on multiple shapes. Our own previous work [21,22,23,35] focussed on gait learning for modular robots with evolvable morphologies. For directed locomotion, most related studies with robots concern the control of vertebrates with fixed shapes, such as a bipeds. The different neural control systems involved in directed vertebrates locomotion are reviewed in [15]. A CPG approach based on phase oscillators towards directed biped locomotion is presented in [28]. A special snake-like robot with screw-drive units is presented in [7] for directed locomotion using a reinforcement learning approach. There are few studies on the directed locomotion of the modular robots, and they focus on fixed morphologies or the special structures.

## 3 Experimental Set-up

In this study, the controllers for all modular robots are learned in an infinite plane environment [18], using our Gazebo-based[1] custom simulator *Revolve*.

### 3.1 Robots

Our robot design is based on RoboGen [2]. We use a subset of those 3D-printable components: *fixed bricks*, a *core component*, and *active hinges*. The fixed bricks are cubic components with slots that can attach other components. The core component holds a controller board. It also has slots on its four lateral faces to attach other components. The active hinge is a joint moved by a servo motor. It can attach to other components by inserting its lateral faces into the slots of these other components. Each robot's genotype describes its layout and consists of a tree structure with the root node representing a core module from which further components branch out. These models are used in simulation, but also could be used for 3D printing and the construction of the real robots.

As a test suite we chose nine robots in three different shapes and sizes, to examine the generality and scalability of our method, see Fig. 1. We refer to these three shapes as *spider*, *gecko*, and *baby*. The 'baby' robots were created through recombination of the 'spider's' and 'gecko's' [22] morphological genotypes.

### 3.2 Controllers

Controllers based on Central Pattern Generators (CPGs) have been proven to perform well for modular robots. In this work, we use CPGs whose main components are differential oscillators. Each oscillator is defined by two neurons that are recursively connected as shown in Fig. 2a. These generate oscillatory pat-

[1] http://gazebosim.org/

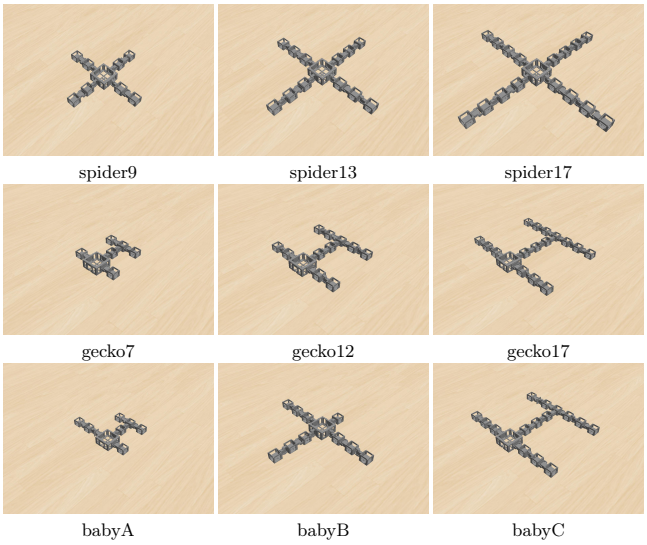
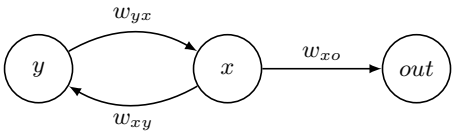
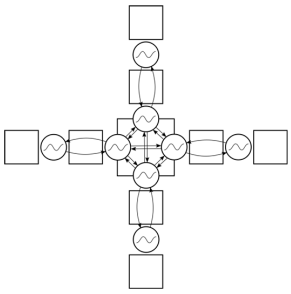



Fig. 1: Images of the used robots. Note that the top leg of gecko17 and babyC are different; babyC has one more active hinge where gecko17 has a brick.




(a) A differential oscillator with output node as used in the CPG controller.



(b) Schematic view of the CPG network generated for a specific morphology.

Fig. 2: Controller concept used in the robots. In (b) the rectangular shapes indicate passive body parts, the circles show active hinges, each with their own differential oscillator, and the arrows indicate the connections between the oscillators for the body shown in the top-left panel of Fig. 1.

terns by calculating their activation levels $x$ and $y$ according to the following differential equation:

$$\dot{x} = w_{yx}y + bias_x$$
$$\dot{y} = w_{xy}x + bias_y$$

with $w_{xy}$ and $w_{yx}$ denoting the weights of the connections between the neurons; $bias_x$ and $bias_y$ are parameters of the neurons. If $w_{yx}$ and $w_{xy}$ have different signs the activation of the neurons $x$ and $y$ is periodic and bounded.

We used Compositional Pattern-Producing Networks (CPPNs) to generate the weights of the CPG controller. CPPNs are a variation of artificial neural networks (ANNs) that have an architecture whose evolution is guided by HyperNEAT algorithm [33], so that the substrate network's performance is optimised [34]. The CPG nodes are positioned in a three-dimensional space. Such *modular differentiation* allows specialisation of the active hinge's movements depending on its relative position in the robot. The hinge coordinates are obtained from a top-down view of the robot body. Thus, two coordinates of a node in the CPG controller correspond to the relative position of the active hinge it is associated with. The third coordinate depends on the role of the node in the CPG network: output nodes have a value of 0 and differential nodes have values of 1 for $x$ and $-1$ for $y$ nodes. Therefore the CPPNs have six inputs denoting the coordinates of a source and a target node when querying connection weights or just the position of one node when obtaining node parameters with the other three inputs being initialised as zero. The CPPNs have three outputs: the weight of the connection from source to target as well as the bias and gain values when calculating parameters for a node.

The CPPNs return the connection weights for the CPG network that in turn constitutes the controller that induces the behaviour for directed locomotion. The behaviour is evaluated by a fitness function (Section 4) and the fitness value is fed to HyperNEAT which in turn generates new CPPNs. The CPPNs evolve until a termination condition is triggered; in our experiments this is reaching a maximum number of generations.

### 3.3 Experimental Parameters

An initial population of 20 CPPNs are randomly generated in the first generation. Each CPPN generates the weights of a CPG network whose topology is based on a robot's morphology. The fitness of the CPG is evaluated in Revolve for a given evaluation time. We set this evaluation time to be 60 seconds to balance computing time and accurately evaluating a complex task as directed locomotion. We found this 60s to be a suitable value empirically. Each EA run is terminated after 300 generations, that is, $300 * 20 = 6000$ fitness evaluations – this amounts to 100 hours of (simulated) time.

The robots used in the experiments include three small robots (spider9, gecko7, babyA), three medium size robots (spider13, gecko12, babyB) and three large robots (spider17, gecko17, babyC). For each robot we tested the EA on five target directions ($-40°$, $-20°$, $0°$, $20°$, and $40°$ relative to the robot) to simulate the robot's limited field of view in the real-world. This resulted in 45 test cases. For each test case the EA runs were repeated five times. All together, we performed 225 HyperNEAT runs per 100 hours of simulated time each.

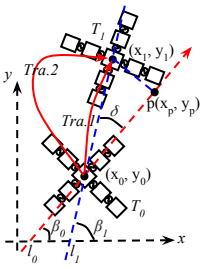
| Parameter | Value | Description |
|---|---|---|
| Population size | 20 | Number of individuals per generation |
| Generations | 300 | Termination condition for each run |
| Tournament size | 4 | Number of individuals used in tournament selection |
| Mutation | 0.8 | Probability of mutation for individuals |
| Evaluation time | 60 | Duration of the test period per fitness evaluation in seconds |

Table 1: Experimental Parameters

## 4 Fitness Function

In this section, we propose a fitness function for directed locomotion and illustrate how the performance of a controller is evaluated. We provide a step-by-step derivation that leads to our final fitness function shown in Equation 5.

Fig. 3: Illustration of the fitness calculation for each evaluation. $T_0$ is the starting position of the robot, with coordinate $(x_0, y_0)$. $T_1$ is the end position of the robot, with coordinate $(x_1, y_1)$. $l_0$ is a given target direction. The point $p$ is the projected point on the target direction $l_0$. The red lines $Tra.1$ and $Tra.2$ show two different trajectories of the robot.




The scenario for an evaluation in our experiments is illustrated in Fig. 3. We can collect the following measurements from the Revolve simulator:

1. $c_0 = (x_0, y_0)$ is the coordinate of the core component of the robot at the start of the simulation, i.e., time $T_0$.
2. $c_1 = (x_1, y_1)$ is the coordinate of the core component of the robot at the end of the simulation, $T_1$.
3. The orientation of the robot in $T_0$ and $T_1$.
4. The length of the trajectory that the robot travelled from $c_0$ to $c_1$

The target direction, $\beta_0$, is an angle with respect to the initial orientation of the robot at $T_0$. In Fig. 3 we drew lines in the target direction, $l_0$, and the line through $c_0$ and $c_1$, $l_1$. The angle between $l_1$ and $x-$axis, $\beta_1 = atan2((y_1 - y_0), (x_1 - x_0))$, is the actual direction of the robot displacement between $T_0$ and $T_1$.

The absolute intersection angle between $l_0$ and $l_1$, $\delta$, is the deviation between the actual direction of the robot locomotion and the target direction. It can be calculated as:

$$\delta = \begin{cases} 2 * \pi - |\beta_1 - \beta_0| & (|\beta_1 - \beta_0| > \pi) \\ |\beta_1 - \beta_0| & (|\beta_1 - \beta_0| <= \pi) \end{cases} \quad (1)$$

Note that we pick the smallest angle between the two lines. To perform well on a directed locomotion task, $\delta$ should be as small as possible. However, just minimizing $\delta$ is not enough to for successful directed locomotion.

In addition to moving in the right direction, i.e., minimizing $\delta$, the robot should move as far as possible in the target direction. Therefore, we calculate distance travelled by the robot in the target direction by projecting the final position at $T_1$, $(x_1, y_1)$, onto $l_0$; we denote this point as $p = (x_p, y_p)$. The distance travelled is then

$$distProjection = sign\ |p - c_0|, \tag{2}$$

where $|p - c_0|$ is the Euclidean distance between $p$ and $c_0$, and $sign = 1$ if $\delta < \frac{\pi}{2}$ (noting that $\delta$ is an absolute value) and $sign = -1$ otherwise. The $distProjection$ is thus negative when the robot moves in the opposite direction.

To further penalize deviating from the target direction we calculate the distance between $(x_1, y_1)$ and $(x_p, y_p)$ :

$$penalty = f_p * |c_1 - p|, \tag{3}$$

where $|c_1 - p|$ is the Euclidean distance between $c_1$ and its projection on the target direction line $l_0$, $p$. $f_p$ is a constant scalar penalty factor, determining the relative importance of the deviation. In our experiments we use $f_p = 0.01$.

A naive version of the fitness would be:

$$fitnessPro = \frac{distProjection}{\delta + 1} - penalty, \tag{4}$$

where $(\delta + 1)$ aims to guarantee that the denominator does not equal zero.

While $fitnessPro$ is proportional to $distProjection$, and inversely proportional to $\delta$ and $penalty$, this does not yet entirely express all desirable features of a good trajectory for the robot. Specifically, we not only care about the final position of the robot, but also about how the robot moves to the end point. To illustrate this please compare the trajectories marked $Tra.1$ and $Tra.2$ in Figure 3. Although the robot has the same starting and end position for both trajectories, $Tra.1$ is a more efficient way of moving between the two points. Therefore, we would want the controller of $Tra.1$ to have a higher fitness than that of $Tra.2$. In general, we aim to evolve a controller to move from start to finish as efficiently as possible, i.e., in a straight line. Therefore, we make the fitness function inversely proportional to the length of the trajectory (noted as $lengthTra$) that the robot performs. We thus propose the following fitness function to measure the performance of controllers for directed locomotion:

$$fitness = \frac{|distProjection|}{lengthTra + \varepsilon} * (\frac{distProjection}{\delta + 1} - penalty) \tag{5}$$

where $\varepsilon$ is an infinitesimal constant. The fitness function is proportional to $distProjection$, but inversely proportional to $lengthTra$ and $\delta$. That is, the fitness function rewards higher speeds in the target direction (as measured through $distProjection$), and punishes the length of trajectories, $lengthTra$, and deviations from the target directions.

## 5 Experimental Results

Inspecting the usual fitness vs. time curves (omitted here because of space limitations) we observe that the controllers of small size robots have the highest average fitness. The controllers of medium and large size robots reach significantly lower values. This is in line with our previous work [22] suggesting that the parameter settings for the larger robots are more difficult to learn, irrespective of the algorithm, such as HyperNEAT or RL PoWER.

An important metric for directed locomotion is the deviation from the target direction, $\delta$. The progression of the learning process is shown in Fig. 4 for each of the nine robots. Each sub-figure shows the average $\delta$ for the 20 controllers in a population over five repetitions. The five target directions are represented by the colours. These curves show that in all cases $\delta$ gradually decreases. Interestingly, the $\delta$ of small size robots is higher than for the larger robots. This means that small size robots are easier to evolve for speed (as they have higher fitness), but do worse in terms of deviation. Similar results were shown in our previous work [22]. We hypothesize that this is because larger robots have more joints, they have more flexibility, and can control their direction more precisely.

To see the outcome of the learning process we select the best controllers from the 30000 controllers (6000 evaluations per run, 5 repetitions) for each robot in each target direction and inspect the trajectories these controllers induce. The best three trajectories for each robot and direction are shown in Fig. 5.

In general, the trajectories follow the target directions well. For example, the trajectories of spider9 are almost exactly on the target directions and they display faster speed than other robots. Because maximizing the distance in the target directions, $distProjection$, is rewarded in the fitness function, as well as minimizing the deviation from the target directions, evolution can lead to different trade-offs between these two preferences. For example, one of the trajectories (purple point-line) for $-40°$ of spider13 deviates quite far from the target direction but travels a long distance, while the other trajectories for this robot and direction get less far but stick more closely to the target direction. In addition, although the trajectories (black point-line) for $0°$ of babyA have high values for $lengthPath$, and thus receive a punishment in the fitness function for the deviation from the straight line in the target direction of $0°$, they have top fitness because of the high speed ($distProjection$) and a good final $\delta$. The small size robots have the better trajectories, especially in terms of speed. The medium size robots have the second-best trajectories. The large size robots also have good trajectories but not as good as the small and medium size robots, especially in terms of speed. In summary, we conclude that using our method, successful controllers can be evolved for directed locomotion for modular robots with evolvable morphologies. Furthermore, the small-sized robots have the better performance for directed locomotion, especially in terms of speed in the target direction.

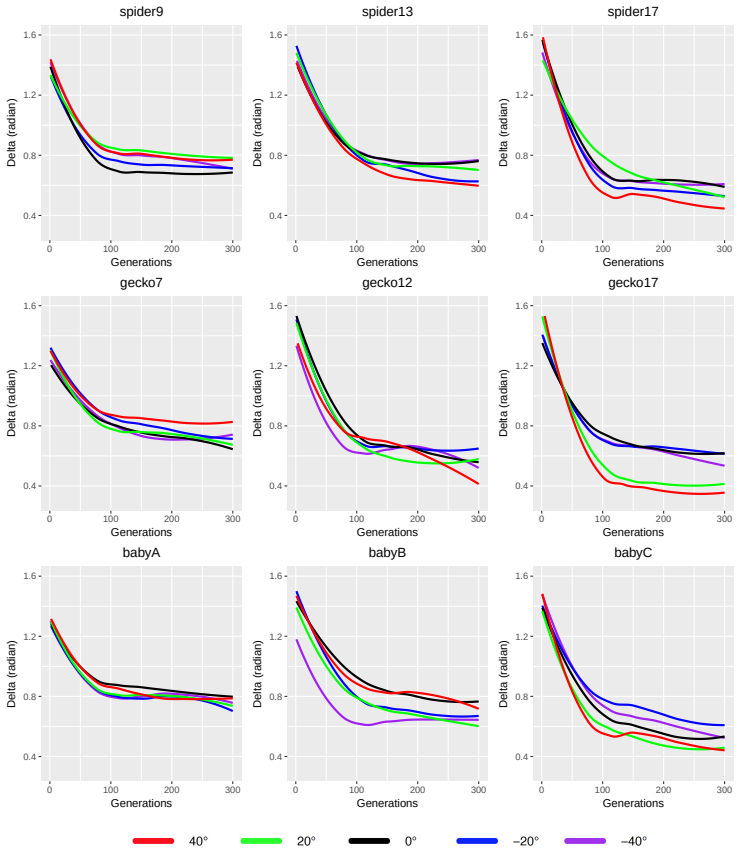



Fig. 4: Deviation ($\delta$) from the target direction during the learning process.

## 6 Concluding remarks

We addressed the problem of learning sensory-motor skills in morphologically evolvable robot systems where the body of newborn robots can be a random combination of the bodies of the parents. In particular, we presented a method to learn good robot controllers for directed locomotion based on HyperNEAT and a new fitness function that balances the distance travelled in a desired direction and the angle between the desired direction and the direction actually travelled. We tested this method on nine modular robots for five different target directions and found that the robots acquired good controllers in all cases. From the resulting trajectories it is apparent that our fitness function adequately balances the speed and direction of the robots.

These experiments were, while well-performing, not too efficient, as the learning speed of HyperNEAT is not very high. Currently we are comparing HyperNEAT to other methods for training the controllers, such as reinforcement

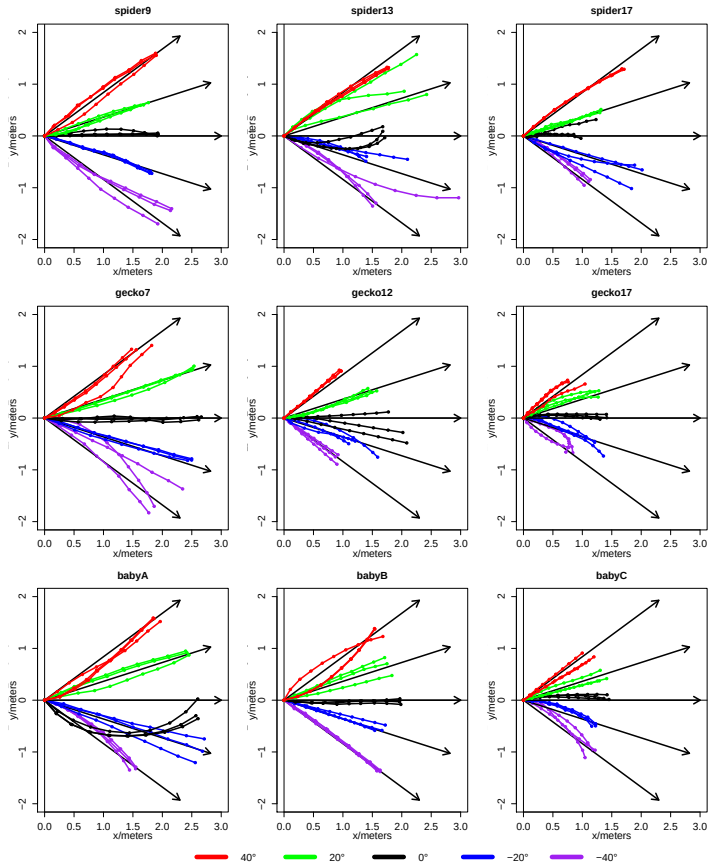



Fig. 5: The best three trajectories for each robot and each direction. The black arrows show the five target directions.

learning [26] and Bayesian optimisation [29]. Furthermore, we aim to investigate which other trade-offs between deviation from the target direction and the speed exist by using a vector-valued, i.e., multi-objective, rather than a scalar fitness function [31]. Finally, we aim to validate our results by replicating the experiments in real hardware and consider more scenarios and other skills.

## References

1. Aoi, S., Manoonpong, P., Ambe, Y., Matsuno, F., Wörgötter, F.: Adaptive control strategies for interlimb coordination in legged robots: A review. Frontiers in Neurorobotics 11, 39 (2017)
2. Auerbach, J., Aydin, D., Maesani, A., Kornatowski, P., Cieslewski, T., Heitz, G., Fernando, P., Loshchilov, I., Daler, L., Floreano, D.: RoboGen: Robot Generation through Artificial Evolution. In: Sayama, H., Rieffel, J., Risi, S., Doursat, R.,

Lipson, H. (eds.) Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems. pp. 136–137. The MIT Press, New York, New York, USA (jul 2014)
3. Auerbach, J.E., Bongard, J.C.: On the relationship between environmental and morphological complexity in evolved robots. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. pp. 521–528. GECCO '12, ACM, New York, NY, USA (2012)
4. Beer, R.D.: The dynamics of brain– body–environment systems: A status report (2008)
5. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. Science 314(5802), 1118–1121 (2006)
6. Bongard, J.C.: Evolutionary robotics. Communications of the ACM 56(8), 74–83 (2013)
7. Chatterjee, S., Nachstedt, T., Wörgötter, F., Tamosiunaite, M., Manoonpong, P., Enomoto, Y., Ariizumi, R., Matsuno, F.: Reinforcement learning approach to generate goal-directed locomotion of a snake-like robot with screw-drive units. In: 2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD). pp. 1–7 (Sept 2014)
8. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: 2009 IEEE Congress on Evolutionary Computation. pp. 2764–2771 (May 2009)
9. Cully, A., Clune, J., Tarapore, D., Mouret, J.B.: Robots that can adapt like animals. Nature 521, 503 (may 2015)
10. Doncieux, S., Bredeche, N., Mouret, J.B., Eiben, A.: Evolutionary robotics: what, why, and where to. Frontiers in Robotics and AI 2(4) (2015)
11. Eiben, A., Bredeche, N., Hoogendoorn, M., Stradner, J., Timmis, J., Tyrrell, A., Winfield, A.: The triangle of life: Evolving robots in real-time and real-space. In: Liò, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (eds.) Advances In Artificial Life, ECAL 2013. pp. 1056–1063. MIT Press (2013)
12. Eiben, A., Kernbach, S., Haasdijk, E.: Embodied artificial evolution. Evolutionary Intelligence 5(4), 261–272 (2012)
13. Eiben, A., Smith, J.: From evolutionary computation to the evolution of things. Nature 521(7553), 476–482 (May 2015)
14. Eiben, A.E.: In vivo veritas: Towards the evolution of things. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) Parallel Problem Solving from Nature – PPSN XIII. pp. 24–39. Springer International Publishing, Cham (2014)
15. Grillner, S., Wallén, P., Saitoh, K., Kozlov, A., Robertson, B.: Neural bases of goal-directed locomotion in vertebrates—an overview. Brain Research Reviews 57(1), 2 – 12 (2008)
16. Haasdijk, E., Rusu, A.A., Eiben, A.E.: Hyperneat for locomotion control in modular robots. In: Tempesti, G., Tyrrell, A.M., Miller, J.F. (eds.) Evolvable Systems: From Biology to Hardware. pp. 169–180. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
17. Hooper, S.L.: Central pattern generators. Encyclopedia of Life Sciences pp. 1–12 (April 2001), https://onlinelibrary.wiley.com/doi/10.1038/npg.els.0000032
18. Hupkes, E., Jelisavcic, M., Eiben, A.E.: Revolve: A versatile simulator for online robot evolution. In: Sim, K., Kaufmann, P. (eds.) Applications of Evolutionary Computation. pp. 687–702. Springer International Publishing, Cham (2018)

19. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: A review. Neural Networks 21(4), 642 – 653 (2008), Robotics and Neuroscience
20. Ijspeert, A.J., Crespi, A., Ryczko, D., Cabelguen, J.M.: From swimming to walking with a salamander robot driven by a spinal cord model. Science 315(5817), 1416–1420 (2007)
21. Jelisavcic, M., de Carlo, M., Hupkes, E., Eustratiadis, P., Orlowski, J., Haasdijk, E., Auerbach, J.E., Eiben, A.E.: Real-world evolution of robot morphologies: A proof of concept. Artificial Life 23(2), 206–235 (May 2017)
22. Jelisavcic, M., Carlo, M.D., Haasdijk, E., Eiben, A.E.: Improving RL power for on-line evolution of gaits in modular robots. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8 (Dec 2016)
23. Jelisavcic, M., Haasdijk, E., Eiben, A.: Acquiring moving skills in robots with evolvable morphologies: Recent results and outlook. GECCO 2017 - Proceedings of the Genetic and Evolutionary Computation Conference Companion (2017)
24. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a modular robotic system. IEEE/ASME Transactions on Mechatronics 10(3), 314–325 (June 2005)
25. Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S., Murata, S.: Distributed adaptive locomotion by a modular robotic system, m-tran ii. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). vol. 3, pp. 2370–2377 vol.3 (Sept 2004)
26. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. vol. 3, pp. 2619–2624 (2004)
27. Marder, E., Bucher, D.: Central pattern generators and the control of rhythmic movements. Current Biology 11(23), R986 – R996 (2001)
28. Matos, V., Santos, C.P.: Towards goal-directed biped locomotion: Combining cpgs and motion primitives. Robotics and Autonomous Systems 62(12), 1669 – 1690 (2014)
29. Paul, S., Chatzilygeroudis, K., Ciosek, K., Mouret, J.B., Osborne, M.A., Whiteson, S.: Alternating optimisation and quadrature for robust control. In: AAAI 2018-The Thirty-Second AAAI Conference on Artificial Intelligence (2018)
30. Pfeifer, R., Bongard, J.C.: How the Body Shapes the Way We Think: A New View of Intelligence (Bradford Books). The MIT Press (2006)
31. Roijers, D.M., Whiteson, S.: Multi-objective decision making. Synthesis Lectures on Artificial Intelligence and Machine Learning 11(1), 1–129 (2017)
32. Sproewitz, A., Moeckel, R., Maye, J., Ijspeert, A.J.: Learning to move in modular robots using central pattern generators and online optimization. The International Journal of Robotics Research 27(3-4), 423–443 (2008)
33. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines 8(2), 131–162 (Jun 2007)
34. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation 10(2), 99–127 (2002)
35. Weel, B., D'Angelo, M., Haasdijk, E., Eiben, A.: Online gait learning for modular robots with arbitrary shapes and sizes. Artificial life 23(1), 80–104 (2 2017)
36. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J.C., Lipson, H.: Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In: In Proceedings of the 20th European Conference on Artificial Life. pp. 890–897 (2011)