Before discussing the details of previous and similar systems it is useful to establish a framework for the type of systems in this thesis. A general scheme for traditional evolutionary robotics is displayed in Figure 2.1a. A population of $\mu$ candidates is maintained on an external computer. One by one each candidate genome is decoded into a controller and allowed to drive the robot a certain amount of time, the fitness evaluation period ($\tau$). After evaluation the fitness is reported back to the computer. When all candidates have been evaluated survivors are selected and from these survivors parents are selected. From these parents offspring is created through recombination (crossover) and mutation. This new population is then evaluated, which completes the evolutionary loop.
Chapter 2. Embodied Evolution and Evolution of Organisms

(a) General scheme of evolutionary robotics Floreano et al. [49]

(b) Taxonomy of embodied evolution as presented by Watson et al. [155]

In contrast to the methodology described above we apply a form of evolution which is embodied. Embodied evolution distinguishes itself from traditional evolutionary robotics in that evolution takes place on the robots themselves during their deployment in the real world. This means that adaptation takes place on the robots themselves, in contrast to adaptation on an external computer. Second, the adaptation is done during the robots deployment period in the environment, in contrast to adaptation beforehand.

As in the classification of Watson et al. [155] embodied evolution is when the trials are on the robots themselves, using multiple robots in parallel and the Evolutionary Algorithm (EA) is distributed over the robots (cf. Figure 2.1b). From an evolutionary algorithm perspective this means each robot is an island and candidates or parts of candidates’ genomes are migrated to other islands. Please note that the most important distinguishing feature of embodied evolution is that the evolutionary algorithm is distributed over the robots and the decision for reproduction is done locally, rather than the fact that it is running on real hardware. A local reproduction decision, using local information and based on local interactions, cannot use global information and is therefore fundamentally different.
We use four major features, inspired by island model EA’s, that distinguish different methods for embodied evolution:

**Local population**
- The size and structure of the local population;

**Migration**
- The method of migration, this includes the choice of the migrant and to which robot it is sent;

**Integration**
- The method of integration of a newly received migrant, e.g. whether it is reevaluated or whether it is always used as a parent;

**Reproduction Mode**
- Whether reproduction takes place concurrently with any task that is performed or there is a special mode for only reproduction and nothing else.

In embodied evolution adaptation often acts on the minds and is performed during the operational period of the robots, therefore, we could call this process learning, which is traditionally done using machine learning algorithms, so why call it evolution? A clear distinction between what is evolution and what is learning is made by Eiben, Haasdijk, and Bredeche [42]. They give an schematic for a system where both evolution and learning occurs. First we should note that in evolutionary systems genotypes and phenotypes are conceptually different entities. They define phenotypes as the controllers of the robots including all their structural and functional complexity, while genotypes are defined as a, usually simpler, representation of these controllers. A mapping from genotype to phenotype is defined, this mapping can be very simple (such as a direct encoding of a neural network) or very complex (e.g. a gene regulatory network). Learning and Evolution are then easily defined as: Learning acts on the phenotypic level and Evolution operates on the Genotypic level. This results in a scheme with two feedback loops (shown in Figure 2.2) that is more sophisticated than the one shown before. This scheme also holds if the genotype also encodes the shape of a robot body, the learning operators then act on only part of the phenotype, namely the control scheme.
Chapter 2. Embodied Evolution and Evolution of Organisms

![Diagram of evolutionary process]

**Figure 2.2:** General scheme of evolution and learning based on the genotype - phenotype distinction.

Keeping the above in mind we provide an overview embodied evolution in Section 2.1. In Section 2.2 we provide a short overview of evolution of organisms. An overview and classification of all the discussed literature can be found in Table 2.1 at the end of this chapter.

### 2.1 Embodied Evolution

The number of papers on embodied evolution in the past decades is surprisingly low, therefore we discuss each paper separately. We describe them using the four major features defined above, as well as the tasks that the algorithm is applied to and the environment used if known.
Embodied evolution: Distributing an evolutionary algorithm in a population of robots

The most well known example of embodied evolution is the work of Watson et al. [155]. They present a system of embodied evolution where neural network controllers’ weights are evolved for a phototaxis task on a group of eight robots. Each robot is controlled by a neural network of 4 weights that drive two motors based on a left and right sensor. This neural network is the single candidate in the local population. In this system single mutated genes (e.g. one mutated connection weight) are periodically broadcast to the other robots in a rate proportional to the fitness of the candidate. When received this gene is integrated into the local candidate probabilistically, again proportional to the fitness of the local candidate. The broadcast of genes is concurrent with the phototaxis task. The main point of their work is to show that embodied evolution is possible and argue that it is a candidate solution to a number of issues in robotics, such as the reality gap and the slowness and inaccuracy of simulations. They note, however, that any embodied evolution system introduces an inherent level of interaction between the robots that may not always be beneficial for the situation.

Situated and embodied evolution in collective evolutionary robotics

Usui and Arita [147] present an embodied evolution system with which they evolved neural network controllers for obstacle avoidance on a group of six Khepera robots. The neural network has 6 sensory inputs and a bias that control the left and right wheels. The local population is varied in their experiments, trying values of 1, 5 and 10 candidates. Migration between robots is done by periodically broadcasting a random candidate selected from the local population.
using roulette wheel selection. This broadcast is also probabilistic proportional to the fitness of the selected candidate. Received candidates are added to a queue of candidates queue of candidates. The migrant is thus re-evaluated before being added to the local population. They ran their system once with each of the three local population sizes mentioned before with migration and one did one run with a population size of 5 without migration. As the number of runs is very low they do not make any conclusions, however they do also show that it is possible to run an evolutionary algorithm in an embodied fashion.

**Figure 2.4:** Obstacle avoidance experiment by Usui and Arita [147]. Experimental setup (A: Continuous power supply, B: Infra-red emitter/receiver unit, C: Power supply

**Figure 2.5:** Robot used in the experiments by Nehmzow [109].

Physically embedded genetic algorithm learning in multi-robot scenarios

Nehmzow [109] did work using two robots which learn several different tasks in collaboration. The robots used are small mobile robots with a number of sensors and the ability to communicate using a IR transmitter and receiver. The tasks are: phototaxis, obstacle avoidance and robot seeking. Each task is first evolved separately, afterwards a combination of phototaxis and obstacle avoidance is tried and finally all three tasks at the same time. The robots are controlled by a behavioural strategy chosen using a table based on the front, right, back and left sensors, in the phototaxis task these are light sensors, for obstacle avoidance they are infrared sensors. Each robot maintains a current individual and a historical best individual. After each evaluation the robots exchange their genomes with their respective fitness and
use those in crossover if the received genome has a higher fitness than the local one. If the received genome has lower fitness a random bit of the best genome is integrated into the local individual in 30\% of the cases, otherwise mutation is applied. While they are exchanging genomes the robots are not performing any tasks, so this is an example where there is a special reproduction mode.

**Figure 2.6:** Obstacle avoidance experiment by Simoes and Dimond [133]  

Simoes and Dimond [133] introduce a system where both controller and part of the morphology evolve for obstacle avoidance. A neural network is evolved as well as which sensors are active using diploid genes. The neural network then chooses between several available actions like: slow forward, fast forward, fast rotate right, fast rotate left etc. Each individual is evaluated for several minutes on 5 mobile robots of 20cm diameter with collision and proximity sensors. After the evaluation phase there is a ”breeding” phase in which the robots broadcast their genome and its fitness. The best performing genome survives and continues on the respective robot. The other 4 robots select a partner for crossover with their local genome, 80\% of the time this is the best genome, the remaining 20\% of the time this is a random other genome. After this a uniform crossover and a 3\% bit flip mutation is applied.

**Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents**

Bredeche et al. [19] introduce mEDEA: a minimal environment drive evolutionary algorithm. This algorithm each robot has a local population of 1 genome that encodes a neural network. Robots broadcast this genome, without supplying any fitness function, to the other robots. In fact, no fit-
ness in the traditional sense is measured at all in their system. After a
certain operational period and having gathered a number of genomes from
others the robot randomly selects a partner from these and applies muta-
tion, this new controller then replaces the local one for the next generation.
This means that the local genome can only survive by spreading itself to
other robots, this achieves a selection pressure on the global level. They
test their algorithm first in simulation in a simple free-roaming environment
and then in a foraging tasks where collecting ‘food items’ lengthens the cur-
rent individuals lifespan. Then they apply their algorithm on a population
of 20 e-puck robots in an open arena.
In this arena they also place a ‘sun’, a repurposed e-puck that the other
robots can recognise. Again there is no explicit fitness function or task the
robots are trying to solve, however they show that with their setup the
robots learn to aggregate near the sun for easier spreading of their genomes.

Evolution, Individual Learning, and Social Learning in a Swarm of
Real Robots

Heinerman et al. [65] combine evolu-
tion with learning on a swarm of
Thymio II robots. In their system
both the evolutionary algorithm as
well as the learning algorithm are dis-
tributed. The key innovation of their
system lies in the combination of ev-
olutionary adaptation, individual learn-
ing and social learning. The genome
encodes the sensory layout which is
inheritable in the evolutionary process, while the individual and social learning adapt the weights of a neural network and are adaptable during the lifetime and not inherited. Both evolutionary and social learning algorithms have a local population of 1 genome and memome which are transmitted independently to all other robots. Received genomes are used for reproduction using binary tournament selection and applying uniform crossover and mutation. Received memomes are used for lifetime learning, the current champion and the last received memome are combined by copying the weights corresponding to the sensory layout from the received memome to the champion. All reproduction is concurrent with the task. They apply their system to the task of obstacle avoidance using 6 Thymio II robots in a 2 by 2.5 meter arena with obstacles. They show that social learning increases the learning speed and leads to better controllers.

2.2 Evolution of Organisms

There are many papers on the evolution of organisms with various levels of embodiment, ranging from very unrealistic to realistic enough to be transferred to hardware. Below is an overview of the most notable examples of evolution of organisms, grouped by their level of realism.

Virtual entities as Organisms

One type of evolutionary system is where the organisms are programs in computer memory. This type of evolution has been studied by several people: Rasmussen et al. [121]; Ray [122]; Lenski et al. [92]; Batut et al. [7]. With this type of system a population of (virtual) computer programs are evolved to survive and replicate in a digital environment. This type of system is the bodies of the organisms are
not realistic at all, if they have bodies at all and therefore will not be discussed in depth.

A slightly more realistic type of evolution is the work of Yaeger [163]. In his Poly World each organism is a simple polygon on a 2D world. Organisms in this system can move, feed, fight, and look around, they expend energy while performing these tasks and need to replenish their energy by feeding. When organisms overlap and they both agree they can reproduce creating a new polygonal child on the world.

These types of evolutionary systems are called artificial life (ALife) systems. They aim to recreate parts of biological life in a digital setting as a tool for experimenting with evolution and life and gaining a deeper understanding of biological life and evolution. Therefore the lack of realistic embodiment is not an issue for many of the ALife research questions, however for our goal the (lack of) embodiment does not suffice.

**Simulated organisms**

![Figure 2.10: Evolution of virtual organisms by Sims [134, 135]](image)

The well known work of Sims [134, 135], acts as an inspiration for many systems of evolution of organisms, this thesis included. He evolves virtual creatures in simulation using a centralised evolutionary algorithm for the task of locomotion and for the task of fighting over a block placed between the two organisms. Both the morphology of the organisms as well as their neural network are evolved as a graph. The organisms evolved consist of blocks of different sizes which are connected through actuators. They showed several different strategies to solve the tasks, including very creative and surprising strategies, especially for the fighting task.
2.2. Evolution of Organisms

In similar fashion Komosinski [84]; Bongard and Pfeifer [16]; Auerbach and Bongard [6] have also evolved organisms and their control structures in simulation using centralised evolutionary algorithms. Again these creatures are not very realistic, as they cannot be manufactured in real hardware. In the case of Komosinski [84] the organisms consist of bars of different length connected together. The bars can be actuated by muscles and house various types of sensors. In Bongard and Pfeifer [16] the organisms consist of spheres which are connected through joints that can be rotationally actuated. Last the organisms in Auerbach and Bongard [6] consists of a central pill shaped structure and two actuated structures that are evolved. These actuated structures can be of any shape and are defined by a convex triangle mesh. In these three papers the task was always to move as quickly as possible in the provided terrain.

Lund [97] evolved the brains and morphology of LEGO robots in simulation for the task of obstacle avoidance. A genetic algorithm evolved the robots brains, which was a linear perceptron, as well as the morphology. The morphological evolution controlled the wheel base and size of the wheels used and the position of the sensors on the robot. The robots were simulated using a minimal simulation and lookup tables. In contrast to the papers above the evolved robots are in principle constructible, even though the simulation is not particularly realistic.

Zykov et al. [171] evolved artificial organisms to self-replicate in simulation. They use a 2D approximation of their cubic Molecube robot for the simulation. First they evolve the morphology of their organisms consisting of multiple Molecubes to reach a space large enough to be able to hold a detached copy of itself. Then in a second stage they evolve instruction sequence for the organism to build a copy of itself. Both evolutionary stages are done using a centralised evolutionary algorithm. Although the simulation is an approximation they showed to be able to execute hand-made replication sequences in real hardware. It is therefore likely that evolved sequences, perhaps with a higher fidelity simulator, can be replicated in real hardware.
Simulate and transfer

The next step in realism is the case where controllers and morphologies are first evolved in simulation and then manufactured and tested in hardware.

Lipson and Pollack [94] evolved the controllers and morphologies of machines that consist of bars and joints. Bars can be actuated with linear actuators which are controlled by recurrent artificial neural networks. The genome encodes the shape of the robot by specifying how bars are connected through vertices, their length and stiffness. Actuators can be linked to these bars and are also linked to a single neurone. Neurones can be connected together in arbitrary ways and their weights are specified as well. A simulator using quasi-static motion was used to compute the fitness of each individual on a forward locomotion task. Several of the best performing robots were then selected to build using rapid prototyping (3D-printing). The evolutionary process resulted in many different shapes that performed well, they also noted that in some cases the performance in hardware was quite similar to the simulation, while in others the physical performance was much lower.

Hornby et al. [70] used an L-system as a generative encoding for evolving stick-like organisms. Each organism consists of multiple sticks that are connected either at fixed angles, or through oscillators that allow the organism to use. The L-system is a program like language that describes both the morphology of the organism, as well as the parameters for the oscillators. They evolved several different organisms and showed that the generative encoding was better
2.2. Evolution of Organisms

at evolving larger and faster moving organisms compared to a non-generative encoding. Several of the organisms were then replicated in hardware using prefabricated- and 3D-printed parts.

Rieffel et al. [123] evolved the morphologies of tensegrity robots. A tensegrity robot is a self-supporting structure consisting of rigid elements connected using a number of strings. These tensegrities are evolved using an L-system as a generative encoding and compared their method to a standard GA with a direct encoding. The robots are then reproduced in the ODE physics engine and evaluated by the volume that they encompass. They show that their L-System performs significantly better than the direct encoding, furthermore they show that the L-system encodes a larger region of the search space and shows more regularity in the shapes. Finally they produced some of the evolved tensegrities using a rapid prototyping machine.

Cheney et al. [24] evolved soft robots in simulation that consist of cubes (called voxels) of different material. The voxels can be ones that are rigid, soft, expanding and then contracting or contracting and then expanding. The genome for such a creature is a Central Pattern Producing Network (CPPN) after HyperNEAT. The CPPN is queried on a box of 10 by 10 by 10 voxels and encodes 1) whether a voxel is filled and 2) which of the 4 types the voxel is made of. Therefore the genome encodes both the morphology and the controller at the same time. The creatures are simulated using the VoxCAD soft-robot simulator. With the same simulator Methenitis et al. [102] has shown that different gravity on different planetary surfaces resulted in different optimal shapes and locomotions. In Hiller and Lipson [67] these type of soft robots are shown to be physically feasible, albeit in a pressure chamber for now.

Auerbach et al. [5] evolved the controllers and morphologies of robots consisting of 3D-printable parts. They defined multiple parts, some have no evolvable parameters like blocks, hinges, cardan joins, etc. Others have evolvable parameters, such as wheels and their diameters, connector bars with their length and angles under which they are attached. The genome encodes both the morphology, as a tree structure describing which parts are used and how they are attached, as well as a neural network controller. The neural
network is a fully connected, recurrent artificial neural networks of which the weights are encoded in the genome. The robots resulting from evolution are printed using a 3D-printer and actuated by the neural network using a Arduino micro-controller. For now the system has only been used as an educational tool, but is designed to be a experimentation platform as well.

**Real Hardware**

![Figure 2.12: Robot being constructed by an assembly arm Brodbeck et al. [21]](image)

Very recently, in June 2015, Brodbeck et al. [21] published an experimental study about “Morphological Evolution of Physical Robots through Model-Free Phenotype Development”. The overall objective is to demonstrate a “model-free implementation for the artificial evolution of physical systems, to stochastically optimize the design of real-world machines”. Being model-free means that the system does not employ simulations, all robots are physically constructed. As noted by the authors this avoids the reality gap but raises two new problems: the speed problem and the birth problem (challenge 4 and challenge 2 in [44]). The system demonstrates a solution to the birth problem in real hardware based on modular robot morphologies. Two types of cubic modules (active and passive) form the ‘raw material’ and robot bodies are constructed from a handful (two to eight) of such modules. The robots have an on-board controller, running on an arduino microcontroller, which operates the servos. The task is locomotion, which is achieved by oscillating the servos at a certain frequency and amplitude determined by the genome. The evolutionary process is induced by a classic generational evolutionary algorithm running on the PC using populations of size 10 and fitness proportional selection where fitness is the travelled distance in a given time interval. Robot genomes encode the bodies implicitly by specifying the sequence of operations to build
them by a robotic arm, dubbed the mother robot. The system was designed to construct new robots autonomously.

This paper represents a very important milestones towards the Evolution of Things. It demonstrate the feasibility of such systems by showcasing a genuine hardware implementation, where the robotic manipulator (mother robot) and the given collection of modules form a real-world `birth clinic’. However, the robots do not have their own onboard controller, and the EA runs in an offline, centralised fashion, position this work into the embodied serial trials category in Watson’s taxonomy (cf. Figure 2.1b).

**Controller & Model**

When evolving the morphology and controllers of robotic organisms there will always be a mismatch between the body and its controller. This mismatch can also occur when the controller has been evolved in simulation and transferred to a real robot or when the robot is damaged during its operational period. We chose to deal with mismatch by using lifetime learning (see Chapter 8), however there have been several papers which investigate ways to deal with this mismatch using a form of modelling. These papers all combine simulation with trials in hardware.

Bongard et al. [15] developed a system where the robot runs an internal simulator with a model of itself. This model is updated regularly and is used for the synthesis of new candidate controllers. This internal model allows their system to cope with hardware failures of the robot, such as the loss of limbs.

Zagal and Ruiz-Del-Solar [169] present a system where the controllers and the simulation are co-evolved. They first evolve controllers in simulation, which are then transferred to hardware and tested again. The controllers that performed best on the real hardware are then used as a target for evolution of the simulator to minimise the difference in fitness between the simulation and hardware. In this manner the fidelity of the simulation improves over time and therefore the quality of the controllers.

Cully et al. [30] evolved controllers in simulation for a six legged robot. They also created a behaviour-performance map in simulation which is a surrogate
model for the performance of each controller based on its parameters. This best controller is then transferred to the real robot which has a hardware fault (a broken leg). The fitness of the controller on this robot is of course lower than the behaviour-performance indicates and therefore the map is updated using the new fitness. This new map is used to quickly develop a new controller which is then tested on the robot and its fitness used to update the map again. This loop continues until the map predicts the best controller has already been found.

Koos et al. [87] use a multi-objective evolutionary algorithm with three objectives: the fitness, the transferability and a diversity measure. This transferability objective is estimated by training a surrogate model of the actual disparity between the simulation and reality using an interpolation technique. The combination of these three objectives allows the evolutionary algorithm to find those controllers that perform well in simulation and will perform well in hardware to be found. The authors note however that if the best solution in hardware is not the best in simulation this solution will not be found using their approach. In their experiments they did not find a situation where this occurred though. Their technique is applied on an e-puck robot and the task of solving a puzzle T-maze where the e-puck needs to go either left or right based on a visual queue. Second they apply it on a four legged robot for the task of locomotion.

2.3 Conclusion

Embodied evolution is still a very unexplored field of research. Furthermore, all research in embodied evolution is on fixed hardware that does not change through evolution. The closest examples of evolving morphology in hardware
2.3. Conclusion

is that of Simoes and Dimond [133], where the active sensor layout is evolved. To truly achieve embodied evolution of organisms hardware limitation need to be overcome. Modular robot systems like the Molecubes of Zykov et al. [171] may provide a solution, otherwise we may need to look at new manufacturing techniques like 3D printing.

In the field of evolution of organisms we see a lot of work in simulation with various levels of realism. Almost all of the simulations are realistic in the sense that they use a good physics simulator; however, they are not all feasible to actually manufacture in real hardware. Several papers show work where evolution is done in simulation and the results are manufactured in hardware. In these cases however, the evolutionary system is still centralised and work in the traditional evolutionary robotics sense. As of yet there has not been a system where evolution of organisms has been achieved in an embodied fashion, i.e. robotic organisms in the real world that run an embodied evolutionary algorithm that evolves both the shapes and the minds of the organisms.

Table 2.1: Overview of two decades of embodied evolution and evolution of organisms.

<table>
<thead>
<tr>
<th>Reference</th>
<th>What evolves</th>
<th>Controller</th>
<th>Lifetime learning</th>
<th>Evolution</th>
<th>Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embodied Evolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Watson et al. [155]; Nehmzow [109]; Usui and Arita [147]; Bredeche et al. [19]</td>
<td>Controller</td>
<td>Neural network</td>
<td>None</td>
<td>Distributed; objective function</td>
<td>Fixed hardware</td>
</tr>
<tr>
<td>Simoes and Dimond [133]</td>
<td>Controller &amp; Sensor Layout</td>
<td>Neural network</td>
<td>None</td>
<td>Distributed; objective function</td>
<td>Fixed hardware</td>
</tr>
</tbody>
</table>
## Chapter 2. Embodied Evolution and Evolution of Organisms

<table>
<thead>
<tr>
<th>Reference</th>
<th>What evolves</th>
<th>Controller</th>
<th>Lifetime learning</th>
<th>Evolution</th>
<th>Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bredeche et al. [19]</td>
<td>Controller</td>
<td>Neural network</td>
<td>None</td>
<td>Distributed; no objective function</td>
<td>Fixed hardware</td>
</tr>
<tr>
<td>Haasdijk et al. [59]</td>
<td>Controller &amp; Sensor Layout</td>
<td>Neural network</td>
<td>None</td>
<td>Distributed; objective function</td>
<td>Simulated realistic</td>
</tr>
<tr>
<td>Heinerman et al. [65]</td>
<td>Virtual program</td>
<td>n/a</td>
<td>n/a</td>
<td>Distributed; no objective function</td>
<td>Fixed hardware</td>
</tr>
<tr>
<td>Rasmussen et al. [121]; Ray [122]; Lenski et al. [92]; Batut et al. [7]</td>
<td>Morphology &amp; controller</td>
<td>Neural network</td>
<td>Hebbian learning</td>
<td>Distributed; no objective function</td>
<td>Virtual memory cells</td>
</tr>
<tr>
<td>Yaeger [163]</td>
<td>Morphology &amp; controller</td>
<td>Neural network</td>
<td>None</td>
<td>Distributed; objective function</td>
<td>Simulated non-realistic</td>
</tr>
<tr>
<td>Sims [134]; Komosinski [84]; Lund [97]; Bongard and Pfeifer [16]; Auerbach and Bongard [6]</td>
<td>Morphology &amp; controller</td>
<td>Neural network</td>
<td>None</td>
<td>Centralised; objective function</td>
<td>Simulated non-realistic</td>
</tr>
<tr>
<td>Pollack et al. [120]; Auerbach et al. [5]</td>
<td>Morphology &amp; controller</td>
<td>Neural network</td>
<td>None</td>
<td>Centralised; objective function</td>
<td>Simulated, realistic. End results manufactured</td>
</tr>
</tbody>
</table>
### 2.3. Conclusion

<table>
<thead>
<tr>
<th>Reference</th>
<th>What evolves</th>
<th>Controller</th>
<th>Lifetime learning</th>
<th>Evolution</th>
<th>Embodiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zykov et al. [171]</td>
<td>Morphology &amp; controller</td>
<td>Instruction sequences</td>
<td>n/a</td>
<td>Centralised; objective function</td>
<td>Simulated, realistic.</td>
</tr>
<tr>
<td>Bongard et al. [15]</td>
<td>Internal model of self and controller</td>
<td>Neural network</td>
<td>None</td>
<td>Centralised; objective function</td>
<td>Fixed hardware, with fault introduced</td>
</tr>
<tr>
<td>Zagal and Ruiz-Del-Solar [169]; Glette et al. [54]</td>
<td>Controller</td>
<td>Neural network</td>
<td>None</td>
<td>Centralised; objective function</td>
<td>Simulated realistic, Validated on hardware</td>
</tr>
<tr>
<td>Hornby et al. [70]; Rieffel et al. [123]; Hiller and Lipson [67]; Cheney et al. [24]; Methenitis et al. [102]</td>
<td>Morphology</td>
<td>Fixed oscillators</td>
<td>n/a</td>
<td>Centralised; objective function</td>
<td>Simulated realistic. End results manufactured</td>
</tr>
<tr>
<td>Koos et al. [87]</td>
<td>Controller</td>
<td>Neural network</td>
<td>None</td>
<td>Centralised; objective function</td>
<td>Interleaved simulations and hardware trials</td>
</tr>
</tbody>
</table>