Crawling Out of the Simulation: Evolving Real Robot Morphologies Using Cheap, Reusable Modules

Ian Macinnes[1] and Ezequiel Di Paolo[2]
Centre for Computational Neuroscience and Robotics, University of Sussex, Brighton, U.K.
[1]I.A.Macinnes@sussex.ac.uk [2]ezequiel@sussex.ac.uk

Abstract

A current issue in evolutionary robotics involves the co-evolution of robot controllers and body morphologies built from modular parts. As part of ongoing research, a model for the evolution of the morphologies and neural network controllers of robots is described. Several robots are evolved for locomotion in simulation built from modules representing cheap, preexisting parts and one is physically built that has comparable behaviour with its original simulated version. The behaviour in simulation of such example robots is described. A brief comparison is made between the behaviour of a simulated robot whose design and behaviour has been evolved and its physically instantiated counterpart.

Introduction

A current line of research in evolutionary robotics involves the co-evolution of robot controllers and body morphologies. Much of the existing research, going back to the work of Sims (Sims, 1994b; Sims, 1994a), is either concerned with exploring arbitrary evolved structures in simulation (Komosinski and Ulatowski, 1999) or evolving controllers for a given complex morphology (Ijspeert, 2001), or building self-repairing modular robots that alter their morphology during the lifetime of the agent within constraints (Murata et al., 2001), or exploring appropriate generative mappings to develop body structures out of genotypic data, some of which are then tested in the real world (Pollack et al., 2001; Hornby, 2003; Funes, 2001). The different (but related) research issues of these lines involve questions of modular design, evolvability, and the relation between embodied structures and behavioural performance.

The present paper is part of a research direction within this area with two central objectives: the development and understanding of more complex behavioural capabilities beyond basic movement and locomotion using co-evolved robot bodies and controllers, and the achievement of this first aim in the real world by means of cheap and re-usable components. The long term aim is to explore the feasibility of machines capable of engaging in simple tasks of self-repair and partial self-assembly for which the two previous objectives are prerequisites.

We have started exploring the first objective using physical simulations and involving tasks such as orientation, positive and negative taxis, and pushing (Macinnes, 2003). Our purpose in this paper concerns now the second objective: the development of a feasible evolutionary scheme capable of handling evolved structures made out of cheap, re-usable real-world components using a standardised encoding itself capable of re-use in a variety of contexts. Such components cannot be subject to arbitrary changes (e.g., elongations, continuous displacements) and will constrain the evolutionary search by their fixed properties and discrete spatial arrangements.

The idea is to find an appropriate combination of the power of evolutionary search and well-established engineering principles such as standardisation of modules, reusability and low part and assembly costs. We return to simpler behaviours like locomotion because our emphasis is on solving the issues involved in encoding components using a genotype that contains information about their physical properties and potentiality of assembly. This genotype must also be evolvable under the constraints of using modules with fixed properties and must generate a convenient plan for the resulting structures so that they can be assembled and tested in the real world.

This work represents a change in emphasis from previous work in evolvable morphology for various reasons. Firstly, the use of a flexible library of pre-existing components from which the robot is to be constructed. This describes both the physical properties and other constraints such as the maximum number of modules. An advantage of this is that we can easily restrict the robot to evolve from and utilise the parts we have currently available. It also allows specification of a minimal subset of essential parts that the robot must contain to be viable, such as a controller block and portable power source. Another advantage is that parts can be added or removed from the library according to changing availability, or the set of components can be replaced to produce a completely different kind of robot.

Secondly, the physical properties of the actuators used are modelled as part of the dynamics of the robot, along with the

other essential parts such as the controller block that are necessary to build a self-contained autonomous robot. We hope that this will allow the evolutionary process to take advantage of the physical dynamics of all its constituent parts and hence build more adapted robots.

Thirdly, sensory feedback from the actuators are fed into the controller, creating a closed sensorimotor loop and allowing the robot to modulate its behaviour. This is preliminary to adding more sophisticated sensors such as cameras that has already been done in simulation to direct and modulate behaviour (Macinnes, 2003).

In summary, the morphologies and controlling neural networks of a population of thirty robots are co-evolved over time via the use of an evolutionary algorithm. This is performed in simulation by a cluster of PC's. An example robot is then physically instantiated using the design provided by its genotype. The following sections describe details regarding the real robots, the simulated robots, the neural network controller, the genotype and evolutionary algorithm, and an example of an instantiated robot.

Real Robots

We define a robot to consist of a controller block (Figure 1), a variable number of Lego bricks, a variable number of servomotors (Figure 2), and a servo-driver block for each servomotor. The on-board controller is a Motorola 68332 microprocessor running at 25MHz. The servomotors have been adjusted to provide the position of the disk back into the controller. The controller runs a continuous time recurrent neural network (CTRNN) that has a genetically determined structure. The feedback from the servomotors is fed as input into the network. The output generated by the network is used to specify the power supplied to the servomotors. The control loop for the robot is:

1. Feed the disk positions of the servomotors into the neural network.
2. Perform an iteration of the neural network.
3. Set the positions of the servomotors as specified by the neural network.

Simulated Robots

The physical parts from which the real robots are constructed need to be simulated by a physics engine[1] so we can have an estimation of how the robot would behave if it were physically constructed. The approach taken is to describe the shape and other physical properties of preexisting parts such as servomotors together with the various ways they can be connected together. This information is stored

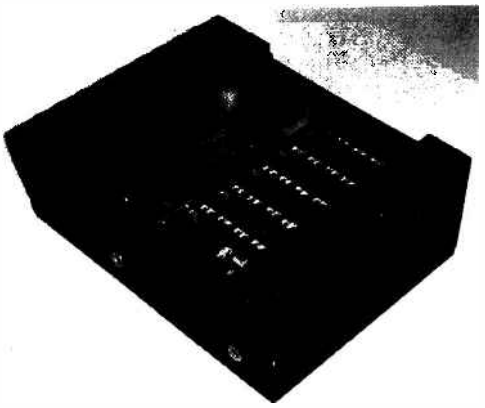

Figure 1: The controller has Lego parts glued to it allowing it to be attached to other parts.

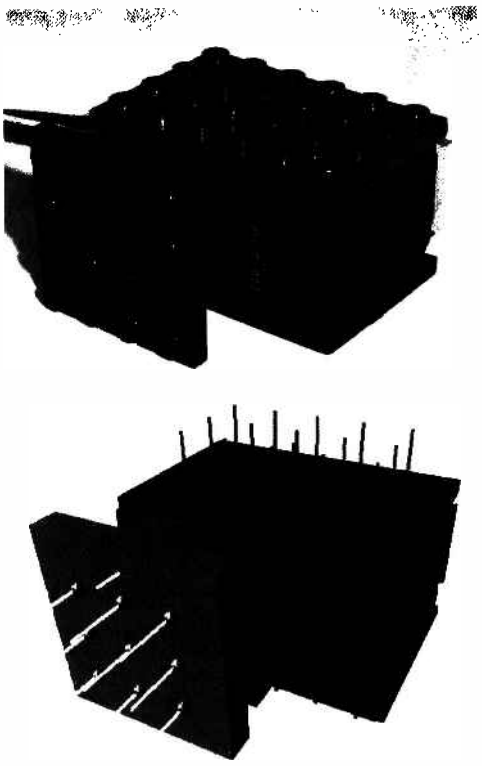

Figure 2: The servomotors have Lego parts glued onto them which allow them to be connected to other parts. They are simulated by approximating their shape with blocks. The anchor point positions are indicated by spheres and the anchor point directions are indicated by lines. It takes four Lego studs to make a connection with another block as can be seen by comparing the above servomotor with its simulated representation below it.

[1]The Open Dynamics Engine (ODE) is used, available from http://opende.sourceforge.net

... physics library. An XML schema was developed to fa-

cilitate this[2]. The shape and other physical attributes of each part was approximated by reducing it into a series of blocks. Each block has two principal properties, the block dimensions and density. The possible variations for these values are defined in the parts library.

Each block can have associated an unlimited number of possible *anchor points*. An anchor point describes a possible location for a connection with another block. When two blocks are to be connected, an anchor point is chosen on each and the blocks are moved and rotated so that both anchor points occupy the same position and opposing directions. The anchor point properties used for this are:

1. Coordinates specifying a position, usually lying on the edge of the block although it need not be within the body of the block at all. It may be relative either to the centre of the block or to the block edge.

2. A vector specifying the direction of the anchor point allowing the relative orientation of each block to be specified in two dimensions.

3. A set of angles specifying the rotations of two connected blocks, allowing the relative orientation in a third dimension.

Neural network controller

The controller used is a continuous time recurrent neural network (CTRNN) (Beer, 1996). This is co-evolved in conjunction with the morphology - that is, over time, the neural network and physical morphology co-evolve together as a single integrated entity. This means that the controller and the robot with be closely coupled together as a single dynamic system, with feedback going both ways between the controller and the servomotors. The potential in each neuron is given by:

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_j w_{ji} z_j \qquad (1)$$

where τ_i is the time constant of neuron i, y_i is its potential, and w_{ji} is the weight from i to j. The activation of the neuron z_j is given by:

$$z_j = tanh(b_i - y_i) \qquad (2)$$

where b_i is the genetically determined bias value for neuron **i**.

At each time-step, the input neurons of the controller are set to the positions of the servomotors, which are normalised to the range -1..1. An iteration of the neural network is performed. Then output neurons are normalised to a range 0..255, and are used to set the desired position of the servomotors.

[2]Both XML schema and library in XML format are available from *http://www.cogs.susx.ac.uk/users/ianma/alife9*

The Genotype

Each robot has a genotype describing how it is constructed. The genotypes that construct the most successful robots are more likely to be selected to have offspring in the next generation. Rank selection is used to pick genotypes.

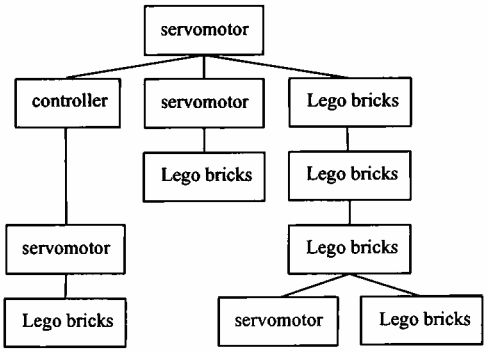

Figure 3: The genotype is constructed as a tree structure where each node represents a modular part and each branch represents a joint.

A direct genotype-to-phenotype mapping is used. For each attribute in the genotype there is a corresponding attribute in the robot. The parts library describes how each block can vary. A block's dimensions and density may be limited to a set of discrete values or be constrained within a range. Offspring are copies of their parents with numerical values altered and/or small structural changes (Tables 2 and 3). Recombination is not used so offspring have only a single parent.

Block	Anchor point	Neural network
dimensions density	position axis direction	neuron bias neuron time constant neuron weight

Table 1: There are various numerical attributes within the genotype that can mutate during the production of offspring. There are also a number of structural mutations such as the random addition or deletion of neurons or modular parts such as servomotors or Lego bricks.

The Trial

The fitness of a robot is provided by the minimum distance moved by any of its constituent blocks during a period of one minute. The measuring period starts after thirty seconds to avoid the strategy of the robot growing top heavy and falling over, hence unfairly gaining a respectable fitness (Sims, 1994b; Macinnes, 2003).

Minimal simulations is used to facilitate the transfer to reality (Jakobi, 1998). It requires that the block dimensions

Neural network mutation	probability of occurring
discrete numerical mutation	0.5
adding a random weight	0.1
randomly removing a weight	0.1
moving a weight's endpoints	0.1
adding a random neuron	0.1
randomly removing a neuron	0.1

Table 2: Different mutations can occur to the neural network controller with various probabilities during reproduction of the genotype.

Morphology mutation	probability of occurring
inserting a random part	0.05
adding a random part	0.05
randomly removing a part	0.05
dimension or density	0.3
joint moving	0.3
change in feedback sensor	0.3

Table 3: The morphology can mutate in various ways. Offspring are tested to make sure they can be properly constructed, for example that no block occupies the same space. If their offspring is not viable, its genotype is discarded and another offspring is produced. This is repeated until a viable offspring is generated.

and densities specified by the genotype are subject to noise when the robot is constructed in simulation. The parts library specifies the magnitude of the noise for each block and it often in the order of plus or minus ten percent. Therefore the same genotype will construct differing robots for each trial. This is performed to help cope with inaccuracy in the simulation and to help produce more robust robots. It should be noted then that we do not attempt to simulate the robot as it will exist in reality, but rather a noisy artifact that we will use to guide the evolution of the population of genotypes.

Two robots are created using the genotype per trial. The fitness assigned to the genotype is the average of the fitness of both robots.

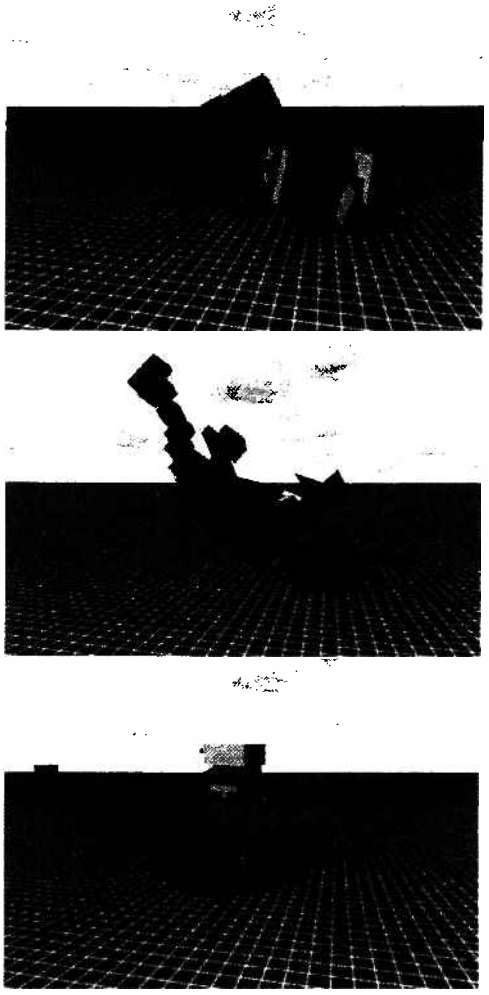

Figure 4: A robot evolved after 1869 generations that moved by throwing its two limbs up in the air and across. One of its limbs developed a larger moment of inertia by increasing its length and mass. It used its servomotors to fling both limbs so the body tilted forward. The smaller limb changed its angle so the robot fell in a different position from when it started. The robot then fell forward and by the end of the movement cycle had changed its position.

Results

Several robots were evolved in simulation (Figures 4 and 5) and one was transfered to reality. A robot (Figure 6) was built from a genotype in generation 2149. It evolved a locomotion strategy of actively using two of its four servomotors, the other two being locked into fixed positions[3]. It uses servomotor *c* to extend limb *a*, resulting in a redistribution of its weight causing it to lean forward. Servomotor *b* has an edge on the ground. It turns, pushing the robot forward. The robot then draws in limb *b*, shifting the weight causing the robot to lean back. The servomotors other edge is now on the ground and it turns in the other direction, again pushing the robot forward.

This strategy takes advantage of the movement of servomotor *b* when it turns in either direction. It depends on coordinating the movement of servomotors *b* and *c*, varying its centre of mass and continuously redistributing its weight.

There are obvious differences in the behaviour of the simulated and real robot. The motions of the real robot are far more jerky that than the simulated robot. Although the

[3]Quicktime movies of this robot moving may be downloaded from http://www.cogs.susx.ac.uk/ianma/alife9.html

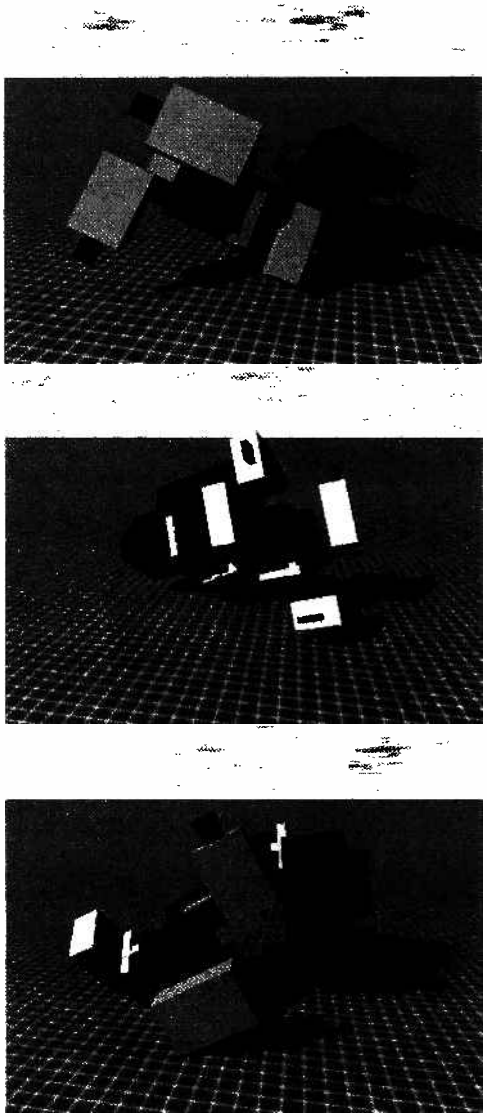

Figure 5: A robot evolved after 2166 generations that tumbles and rolls along. The robot starts perched on its servomotors. It moves its controller up above, which unbalances it, causing it to fall over onto its back. Then it twists the servo connecting the controller and tumbles over, moving its position along.

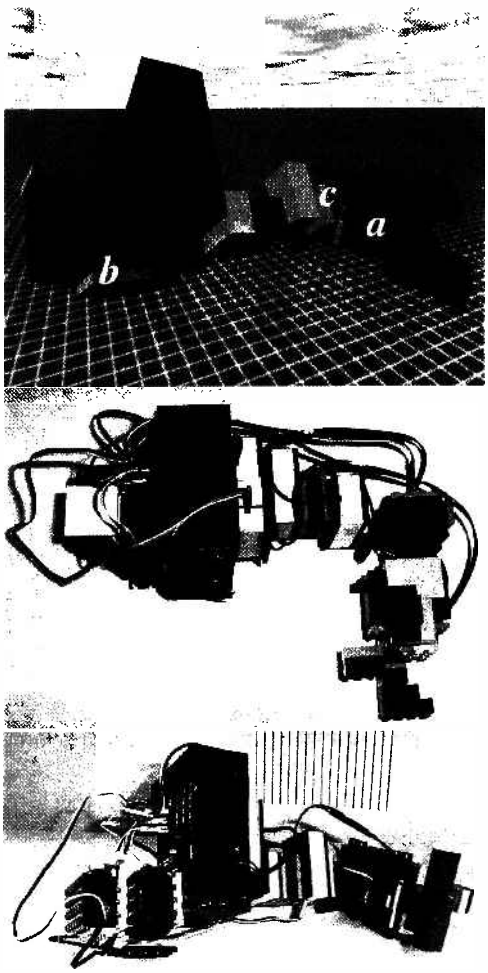

Figure 6: The locomotion strategy of this robot depends closely on its close coupling between its morphology and controller. Above is the simulated robot and below the view of its real counterpart from above and the front. The real robot's behaviour is comparable with its original simulation.

strategy used is clearly the same, the simulated robot travelled approximately 55cm per minute whereas the real robot moved approximately 14cm per minute. One reason was found to be because updating the neural network in the simulation was always assumed to be done within a single timestep of 0.05 seconds. The microprocessor in the real robot's controller was set to use this as a minimum period but took longer than this to perform a single update of the neural network. As a result, it took one minute forty seconds for the real robot to perform the same number of updates. For the measuring period of one minute, there are (60 / 0.05 seconds) 1200 discrete neural network updates. The real robot moved a distance of 23cm using a measuring period defined by the number of updates rather than time. It is expected that the time taken for the real robot to perform a neural network update will vary depending upon the number of neurons and speed of the controller. This may result in it extending beyond any fixed update time used in the simulated robot. We conclude that the neural network update time should be subject to *minimal simulation* noise to result in a better reality transfer.

Conclusions

It is possible to evolve the morphologies and controllers of simulated robots to perform locomotion and successfully

transfer them to reality using cheap, reusable, and modular parts. The model shown does not use continuous values within the physical morphology section of the genotype but discrete values from a predefined set of preexisting components. The resulting robots successfully employ a variety of strategies that depend on the close coupling of their physical body and their controller. This is facilitated via careful use of *minimal simulations* (Jakobi, 1998), which requires considerable inter-trial noise.

The next step is reducing the difference between the simulated and real robots, and to produce the more complicated sequential behaviour that has already been produced in robots with evolved morphologies made from genotypes with continuous values (Macinnes, 2003). It is probable that an indirect genotype-to-phenotype mapping would improve the evolvability of the methodology. More complicated mappings have previously been explored (Hornby and Pollack, 2001a; Hornby and Pollack, 2001b) and would seem to be a fruitful area of study combined with the technology described here.

Acknowledgements

We'd like to thank Bill Bigge of the Autonomous Systems Laboratory at the University of Sussex for help at various times on the hardware of the robots.

References

Beer, R. (1996). Towards the evolution of dynamical neural networks for minimally cognitive behaviour. In *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour*, pages 421–429. MIT Press.

Funes, P. (2001). *Evolution of Complexity in Real-World Domains*. PhD thesis, Brandeis University, Waltham, Massachusetts, USA.

Hornby, G. (2003). *Generative Representations for Evolutionary Design Automation*. PhD thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA.

Hornby, G. and Pollack, J. (2001a). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 600–607, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea. IEEE Press.

Hornby, G. and Pollack, J. (2001b). Body-brain co-evolution using L-systems as a generative encoding. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 868–875, San Francisco, California, USA. Morgan Kaufmann.

Ijspeert, A. (2001). A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 85(5):331–348.

Jakobi, N. (1998). *Minimal Simulations for Evolutionary Robotics*. PhD thesis, University of Sussex.

Komosinski, M. and Ulatowski, S. (1999). Framsticks: Towards a simulation of a nature-like world, creatures and evolutions. In Floreano, D., Nicoud, J., and Mondada, F., editors, *Advances in Artificial Life - Proceedings of the 5th European Conference on Artificial Life (ECAL)*, pages 261–265. Springer-Verlag.

Macinnes, I. (2003). Visually guided physically simulated agents with evolved morphologies. In W.Banzhaf, T.Christaller, P.Dittrich, J.T.Kim, and J.Ziegler, editors, *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 821–828. Springer Verlag Berlin, Heidelberg.

Murata, S., Yoshida, E., Kurokawa, H., Tomita, K., and Kokaji, S. (2001). Self-repairing mechanical systems. *Autonomous Robots*, 10(1):7–21.

Pollack, J., Lipson, H., and Funes, P. (2001). Three generations of automatically designed robots. *Artificial Life*, 7(3):215–233.

Sims, K. (1994a). Evolving 3D morphology and behaviour by competition. In Brooks, R. and Maes, P., editors, *Artificial Life IV Proceedings*, pages 28–39, MIT, Cambridge, MA, USA. MIT Press.

Sims, K. (1994b). Evolving virtual creatures. In *Computer Graphics, Annual Conference Series, (SIGGRAPH 1994 Proceedings)*, pages 15–22.