Computer Creativity in the Automatic Design of Robots

Jordan B. Pollack, Gregory S. Hornby, Hod Lipson and Pablo Funes

ABSTRACT

This article demonstrates the possibility that robotic systems can automatically design robots with complex morphologies and tightly adapted control systems at a low cost. These automatic designs are inspired by nature and achieved through an artificial coevolutionary process to adapt the bodies and brains of artificial life-forms simultaneously through interaction with a simulated reality. Through the use of rapid manufacturing, these evolved designs can be transferred from virtual to true reality. The artificial evolution process embedded in realistic physical simulation can create simple designs often recognizable from the history of biology or engineering. This paper provides a brief review of three generations of these robots, from automatically designed LEGO structures, through the GOLEM project of electromechanical systems based on "truss" structures, to new modular designs that make use of a generative, DNA-like representation.

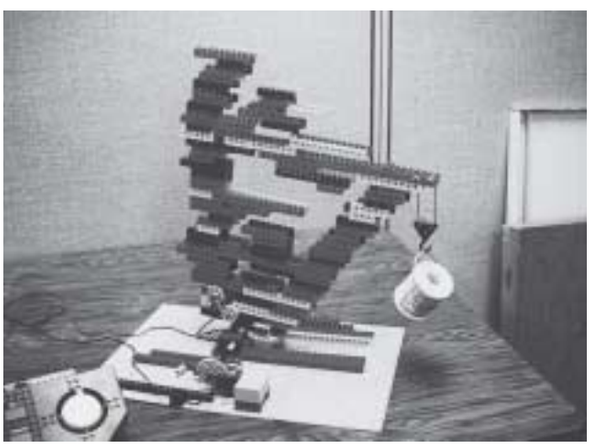
Traditionally, robots have been designed on a hardware-first, software-last basis: Mechanical and electrical engineers design complex articulated bodies with state-of-the-art sensors and actuators and multiple degrees of freedom; the next task is simply to "write the software." But robot designers have drastically underestimated the difficulty in writing control software and this, in addition to the high costs of designing and building the hardware, has led robot development to a stasis [1]. Modern commercial robots perform only simple and highly repetitive manufacturing tasks with very little decision-making, if any, by the onboard software [2]. The central issue addressed by our work is the capability to design robots of more complex structure and more onboard intelligence, yet at a lower cost. We suggest that this can be achieved only when robot design and construction are fully automatic.

In nature, the equivalent to fully automatic design software is the process of evolution. The body and brain of a creature are tightly coupled, the fruit of a long series of small mutual adaptations—like the chicken and the egg, neither one was designed first. There is never a situation in which the hardware has no software, or where a growth or mutation beyond the adaptive ability of the brain survives. Autonomous robots, like living creatures, require a highly sophisticated correspondence between brain, body and environment. Using natural systems as inspiration, we coevolve both the brain and the body, simultaneously and continuously, from a simple controllable mechanism to one of sufficient complexity for a particular specialized task [3]. We then create a representation scheme that allows for the hierarchical construction of more complex components from previously defined ones for the evolution of modular designs. Although we were not the first to propose brain/body coevolution [4–6], we are the first to have gone from computer simulation to reality.

The results of our work are predictive of a future in which humans are engaged in more complex and artistic forms of creativity, while the lower-level details of design and interactions between components are managed by computers. In this paper, we show that computer software, properly situated, can create functional forms and, further, that there are representational techniques that enable these systems to begin to scale to more complex forms in which evolved components are replicated and reused in

Fig. 1. Photographs of the FAD LEGO bridge (cantilever) and crane (triangle). (Photos: © Pablo Funes and Jordan Pollack)





Jordan B. Pollack (researcher), DEMO Laboratory, Computer Science Department, Brandeis University, Waltham, MA 02454, U.S.A. E-mail: <pollack@cs.brandeis.edu>.

Gregory S. Hornby (researcher), DEMO Laboratory, Computer Science Department, Brandeis University, Waltham, MA 02454, U.S.A. E-mail: <hornby@cs.brandeis.edu>.

Hod Lipson (researcher), School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, U.S.A. E-mail: <hod.lipson@cornell.edu>.

Pablo Funes (researcher), DEMO Laboratory, Computer Science Department, Brandeis University, Waltham, MA 02454, U.S.A. E-mail: <funes@cs.brandeis.edu>.

DEMO Web Site: <www.demo.cs.brandeis.edu>.

© J. Pollack, G. Hornby, H. Lipson, P. Funes

Project MUSE® http://muse.jhu.edu

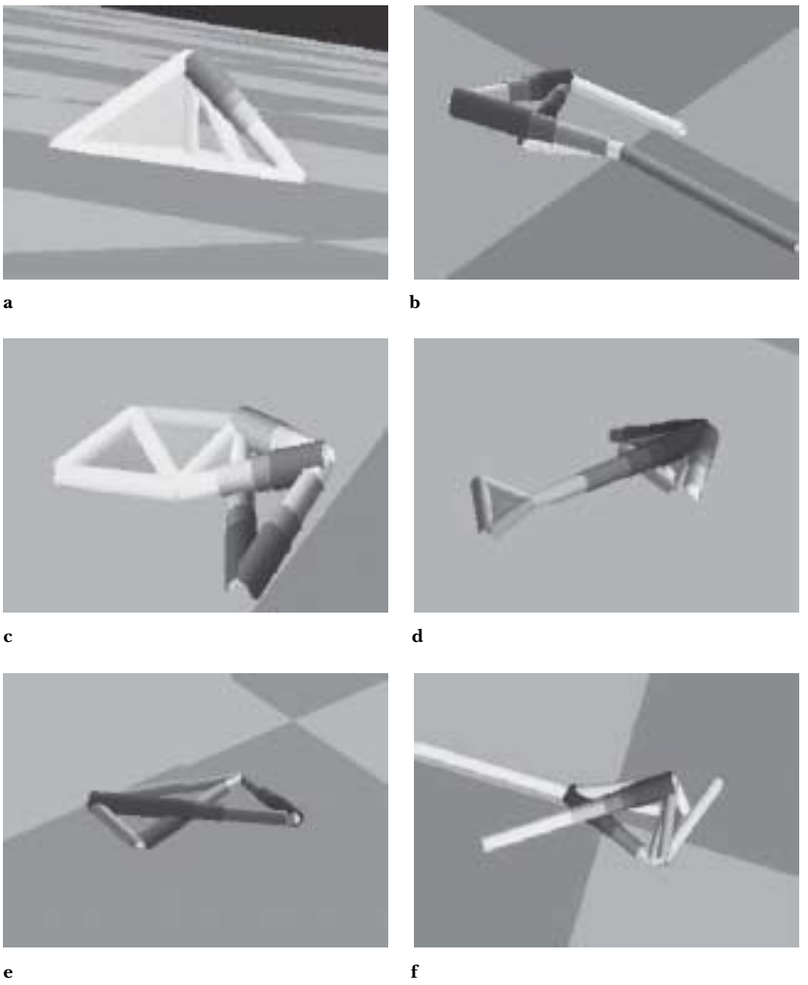


Fig. 2. (a) A tetrahedral mechanism produces hinge-like motion and advances by pushing the central bar against the floor. (b) Bipedalism: the left and right limbs are advanced in alternating thrusts. (c) Moving the two articulated components produces crab-like sideways motion. (d) While the upper two limbs push, the central body is retracted and vice versa. (e) This simple mechanism uses the top bar to delicately shift balance from side to side, shifting the friction point to either side as it creates oscillatory motion and advances. (f) This mechanism has an elevated body, from which it pushes an actuator down directly onto the floor to create a ratcheting motion. Redundant bars drag on the floor. (© Hod Lipson and Jordan Pollack)

hierarchical patterns [7,8]. We have been working for some time on refining our understanding of the dynamics of coevolutionary learning, which, if successful, could lead to more productive self-organization of complex and artificially engineered systems [9,10]. Even so, there remain artistic choices regarding the construction media, the design of fitness criteria and the selection of evolved designs to be translated into the real world.

Evolutionary Algorithms

Evolutionary algorithms (EAs), a technique inspired by biological evolution [11], have been used to generate computer images [12], shapes and objects [13,14], creature controllers [15–17] and creature morphologies [18]. An evolutionary algorithm works by keeping a population of candidate solutions and refining the population according to a given "fitness function," which can provide an absolute measurement of the quality of any candidate. Through successive iterations, the EA replaces poor-quality members of the population with individuals generated by applying variation to higher-quality members of the population. The fitness function is an automatic way for researchers or designers to specify their goals.

One form of EA is coevolution, in which a population is not ranked by an absolute fitness function, but by a relative measure such as a comparison between different candidates. Coevolution, when successful, dynamically creates a series of learning environments each slightly more complex than the last, and a series of artificial learners that are tuned to adapt in those environments. Sims's work [19] on body-brain coevolution and the more recent Framsticks simulator [20] demonstrated that the neural controllers and simulated bodies could be coevolved. The goal of our research in coevolutionary robotics is to replicate and extend results from virtual simulations like these to the reality of computer-designed and -constructed special-purpose machines that can adapt to real environments.

We are working on coevolutionary algorithms to develop control programs that operate realistic physical device simulators, both commercial off-the-shelf simulators and our own custom simulators, where we complete the evolution inside real embodied robots. We are ultimately interested in obtaining electromechanical structures that have complex morphology and place more degrees of freedom under control than anything that has ever been manually designed. We also expect that these machines will have low enough engineering costs to be produced in small quantities, because they employ minimal human design labor.

It is not feasible to evolve controllers for complete structures (in simulation or otherwise) without first evolving controllers for simpler constructions. Compared with the traditional form of evolutionary robotics [21–25], in which controllers are serially downloaded into a given piece of hardware, it is relatively easy to explore the space of body constructions in simulation. Realistic simulation is also crucial for providing a rich and nonlinear universe. However, while simulation creates the ability to explore the space of constructions far faster than real-world building and evaluation could, there remains the problem of transfer to real constructions and scaling to the high complexities used for real-world designs.

Results

The fundamental method of our work is evolution inside simulation, but in simulations more and more realistic, so that the resulting blueprints are not simply visually believable, as in Sims's work, but also buildable, either manually or automatically. Our first results involved automatic creation of structures with a large number of parts that could be transferred from simulation to the real world. In the second generation we evolved automatically buildable dynamic machines that are nearly autonomous in both their

design and manufacture, using rapid prototyping technology, then manually inserting motors. The third generation has begun to address scaling by handling complex structures through modularity. Even with these three demonstrations, we feel that the work is at a very early stage, with major issues only beginning to be addressed, such as the integration of sensors and automation of the feedback from "live" interactions.

Generation 1: LEGObots

Our first step towards fully evolved creatures was the evolution of static LEGO structures in simulation that could then be built in reality. A simulator for evolving such structures needs to satisfy the following constraints:

- Representativity—the simulator should cover a universal space of mechanisms
- Conservatism—because simulation is never perfect, it should preserve a margin of safety
- Efficiency—it should be quicker to test in simulation than through physical production and test
- Buildability—results should be convertible from a simulation to a real object.

The simulator we constructed considers the union between two bricks as a rigid joint between the centers of mass of each one, located at the center of the actual area of contact between them. This joint has a measurable torque capacity: more than a certain amount of force applied at a certain distance from the joint will break the two bricks apart. The fundamental assumption of our model is the idealization of the union of two LEGO bricks as a rotational joint with limited capacity.

Using this simulator, in conjunction with a simple evolutionary algorithm, we evolved complex LEGO structures, which we then manually constructed [26–28]. Our system reliably builds structures that meet fitness goals, exploiting physical properties implicit in the simulation. Building the results of the evolutionary simulation (by hand) demonstrated the power and possibility of fully automated design: the long bridge of Fig. 1 shows that our simple system discovered the cantilever, while the weight-carrying crane shows it discovered the basic triangular support.

Generation 2: Genetically Organized Lifelike Electromechanics

The LEGO machines, with computer-generated blueprints and manual construction, demonstrated that the interaction between simulated physics and evolution leads to a primitive form of discovery that can be transferred into reality. The next goal was to add motion to our designs and address the issue of manufacture. While LEGO kits have motion components, the design space is very broad and difficult to model, and no robot can match the manual dexterity of a 10-year-old human in assembly. To achieve actuated, automatically manufactured robots, we started with a whole new process in which robot morphology was constrained to be buildable by a commercial off-the-shelf rapid prototyping machine [29], allowing us to evolve bodies and controllers in simulation that were essentially transferrable automatically into reality [30].

These robots are composed of fixed bars, ball-joints and linear actuators controlled by artificial neurons. The configurations of the bodies are constrained to be buildable out of thermoplastic using our rapid prototyping machine. The entire configuration is evolved for a particular task, and selected individuals are printed pre-assembled (lacking only motors). In doing so, we have established for the first time a complete physical evolution cycle.

In this project, the evolutionary design approach assumes two main principles: (a) to minimize inductive bias, we must strive to use the lowest-level building blocks possible, and (b) we must coevolve the body and the control, so that they stimulate and constrain each other. We use arbitrary networks of linear actuators and bars for the morphology and arbitrary networks of sigmoidal neurons for the control. Evolution is simulated starting with a soup of disconnected elements and continues over hundreds of generations of hundreds of machines, until creatures that are sufficiently proficient at the given task emerge. The simulator used in this research is based on quasi-static motion. The basic principle is that

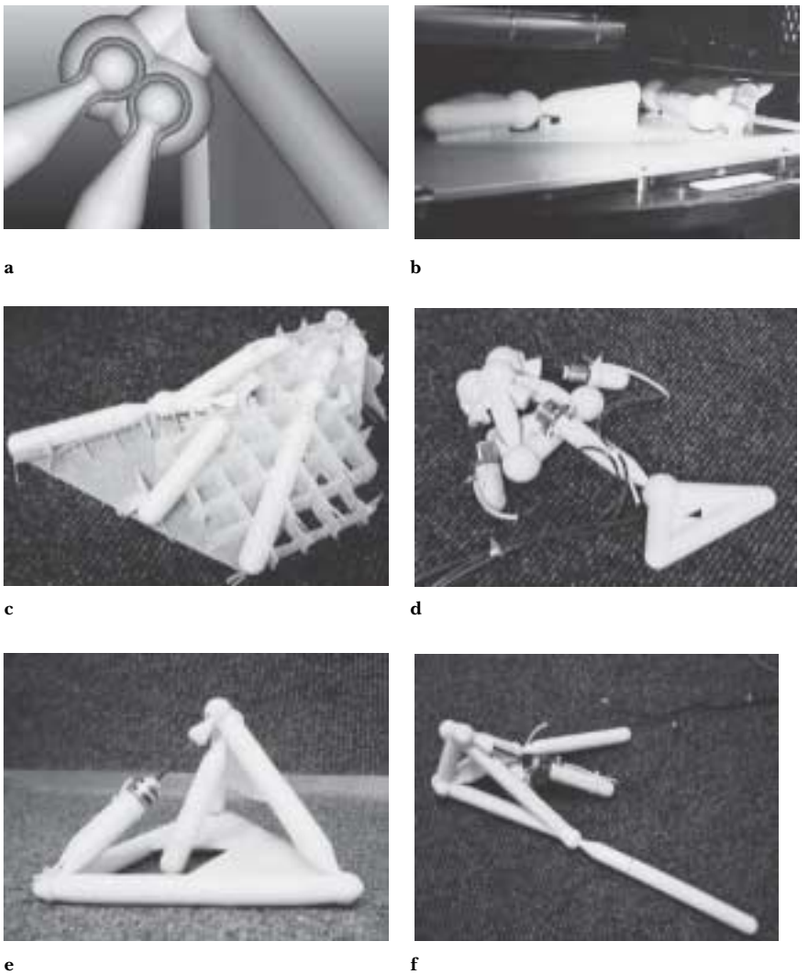


Fig. 3. (a) Fleshed joints; (b) replication progress; (c) pre-assembled robot (see Fig. 2f); (d,e,f) final robots with assembled motors. (© Hod Lipson and Jordan Pollack)

Table 1. A Simple L-System.

rule head	condition		successor
a(n):	$(n > 2)$	$\rightarrow$	$a(n-2)b(n)$
a(n):	$(n > 0)$	$\rightarrow$	$c\ a(n-1)$
b(n):	$(n > 2)$	$\rightarrow$	$d\ a(n-1)$
b(n):	$(n > 0)$	$\rightarrow$	$c\ d\ b(n-1)$

Table 2. Several iterations of rewriting according to the L-system of Table 1.

a(4)
a(2)b(4)
ca(1)da(3)
ca(0)da(1)b(3)
ca(0)dca(0)da(2)
ca(0)dca(0)dcca(0)

motion is broken down into a series of statically stable frames solved independently. While quasi-static motion cannot describe high-momentum behavior such as jumping, it can accurately and rapidly simulate low-momentum motion. This kind of motion is sufficiently rich for the purpose of the experiment and, moreover, is easily induced in reality, because all real-time control issues are eliminated. We carried out several evolution runs for the task of locomotion. Fitness was awarded to machines according to the absolute average distance traveled over a specified period of neural activation. The evolved robots exhibited various methods of locomotion, including crawling, ratcheting and some forms of pedalism (Fig. 2). These forms and mechanisms often appear to have design elements from a common engineering vocabulary; however, these elements are not in a database; they emerge from the interaction of evolution and the simulation of potential robotic bodies and their brains.

Selected robots are then replicated in reality: their bodies are first fleshed to accommodate motors and joints and then copied into material using rapid prototyping technology. A temperature-controlled print head extrudes thermoplastic material layer by layer, so that the arbitrarily evolved morphology emerges pre-assembled as a solid three-dimensional structure without a secondary assembly process requiring human intervention. Motors are then snapped in (manually), and the evolved neural network is activated (Fig. 3). The robots then perform in reality much as they did in simulation.

Generation 3: Generative Representations

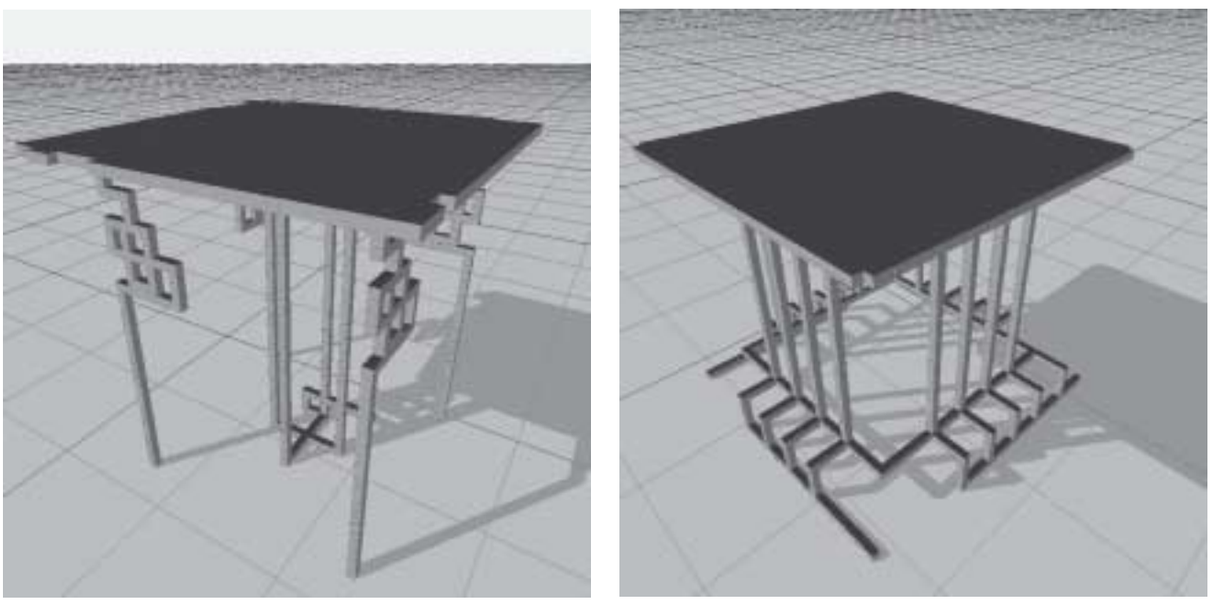
While the GOLEM project validated our approach to automatic design and manufacture, the machines that were produced are obviously fairly simple compared with the kinds of robots buildable by teams of human engineers. In fact most work in automatic design of engineering products, using techniques inspired by biological evolution [31–33], bears the same criticism.

Our third generation starts to address the issue of whether evolutionary automatic design techniques can attain the higher level of complexity necessary for practical engineering projects. Ideally an automated design system would start with a library of basic parts and would iteratively create new, more complex components from those already in the library. Once a component is specified, the system would be able to use the component throughout the design, as well as to create even more complex components, in the same way as the initial starting set of basic parts. To achieve this, we have developed a kind of generative representation, which allows for the reuse of elements in a design. A generative representation involves an encoding of candidate designs not in a direct and primitive form, but more abstractly, using something like a computer program or a grammar [34,35].

Again we found inspiration in natural systems and chose Lindenmayer systems (L-systems) as the basis for the generative representation in which we encode designs. L-systems are a grammatical rewriting system introduced by Lindenmayer in 1968 [36] to model the biological development of multicellular organisms. An L-system consists of a set of rules for rewriting characters in strings, with rules applied in parallel to all characters in the string just as cell divisions occur in parallel in multicellular organisms. Complex strings are created from simpler strings by iteratively rewriting symbols in the string with other symbols according to the rewriting rules. Rules are of the form shown in Table 1.

Rewriting consists of matching symbols in the string being processed and then substituting a sequence of symbols for each one based on given rules. A rule applies if it matches the symbol in the string

Fig. 4. Two tables evolved using a generative representation. (© Gregory Hornby and Jordan Pollack)



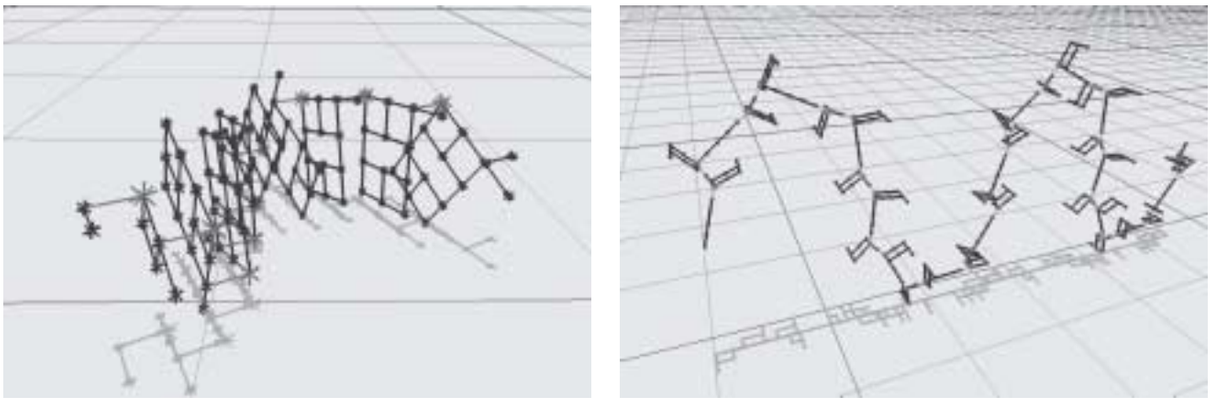


Fig. 5. Two examples of evolved genobots. (© Gregory Hornby and Jordan Pollack)

while the condition part of the rule is satisfied. Symbols that do not satisfy any rewrite rule are not replaced and are copied directly to the next string. Starting from a predetermined symbol, production rules from the L-system are used either until no more apply or for a fixed number of rewriting iterations. For example, using the above set of rules and starting with an initial string consisting of the single symbol, the sequence of strings shown in Table 2 is produced.

The final string of characters is interpreted as an assembly procedure for constructing an object, with each symbol representing a different construction command. Thus, a generative encoding is analogous to a computer language, with production rules a type of sub-procedure for specifying complex components. The end result is an evolutionary algorithm optimizing a population of L-systems, each of which is not a blueprint for a design, but an algorithm for creating the blueprint.

A graphical example of an evolved L-system is shown in Color Plate B No. 1(a), along with the sequence of strings it generates in Color Plate B No. 1(b). In our system we distinguish between symbols that can be rewritten (represented by cubes) and symbols that are later used for constructing the object (represented by colored spheres). In Color Plate B No. 1(a), a rule is graphically represented by a cube connected (by a black dot) to a sequence of symbols. The column of cubes on the left represents rule heads, and the black dots to which they are connected represent different conditions. The symbols following each black dot in sequence are what the sphere will be replaced by. In addition to cubes and spheres, pyramids are used to represent a repeat operator, which indicates that the symbols following it are to be repeated a given number of times. This L-system is started with the first production rule, and the sequences of longer and longer strings shown in Color Plate B No. 1(b) are the symbols generated after each iteration of parallel replacement. The final string of spheres drives the construction of the object, much like a paper tape or the cards for a Jacquard loom.

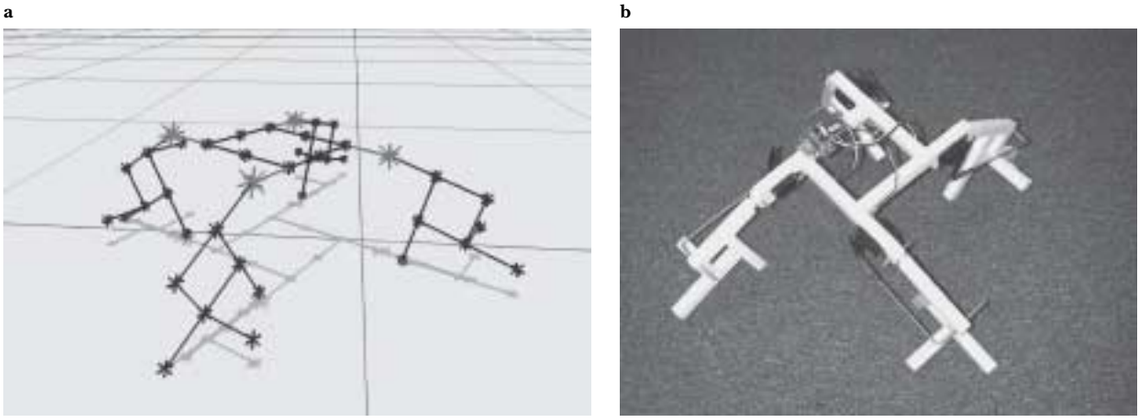
Using this system for evolving L-systems, different design types can be generated by replacing one set of construction commands with another. So far we have evolved 3D static structures [37] (Fig. 4), and locomoting mechanisms called genobots [38–40]. In Fig. 5 we show two simulated genobots, the first of which (Fig. 5, left) rolls along by twisting the connections between each pair of rectangle-shaped pieces, and the second (Fig. 5, right) moves by undulating like a sea serpent. A third genobot, shown in both simulation and reality in Fig. 6, moves by using its four legs to walk.

Discussion and Conclusion

Can evolutionary and coevolutionary techniques be used in the design of real robots as "artificial life-forms"? In this

Fig. 6. A 4-legged evolved genobot shown (a) in simulation and (b) in reality. (© Gregory Hornby and Jordan Pollack)

a **b**



paper we have presented three generations of our work, each of which addresses one or more dimensions of the problem. We have summarized research in the use of simulations for handling high-part-count static structures that are buildable, dynamic electromechanical systems with complex morphology that can be built automatically *and* generative encodings as a means for scaling to complex structures.

The limitations of the work are clearly apparent: these machines do not yet have sensors and are not really interacting with their environments. The simulations do not automatically receive feedback about how robots perform in the real world; rather, humans need to refine the simulations and constraints on the design system. Finally, there is the question of how complex a simulated system can be before the errors generated by transfer to reality are overwhelming.

We cannot claim immediate solutions to these problems. Some work in evolutionary robotics (e.g. by Jakobi [41]) integrates sensors into the evolutionary mix, and we have already demonstrated fixed-morphology robots with sensors that learn in simulation [42] and through interactions with the real environment [43]. In future generations of our evolved creatures we expect to see some sensor integration into our system, allowing us to coevolve the morphology and controllers of reactive robots.

The problem of complexity overwhelming the process has not yet arisen, because the complexity of what can be evolved is still low. This is a primary research topic for us, and we have been studying how coevolution can lead to complex performance in domains such as game-playing [44] and in the design of complex algorithms such as sorting networks [45] and cellular automata [46].

The issue of whether or not this kind of artificial life work will ever be practical and scalable is best related to the history of computer chess. The theory that machines could play a game like chess arose in the 1920s. This was followed by the first chess-playing computer in the mid-1950s, which played by making random legal moves. While early proponents of funding for the new field of AI were overoptimistic, by the end of the century Deep Blue was able to win a tournament against the leading human player, using almost unlimited CPU time and 80-year-old theory [47].

Perhaps over time, with continued development and increases in computer speed and simulation fidelity, coupled to advances in basic theory of coevolutionary dynamics and representation schemes, the small demonstrations of automatic design will lead to a point where fully automatic design is taken for granted, much as computer-aided design is taken for granted in manufacturing industries today.

Our current research moves towards the overall goal via the multiple interacting paths of simulation, theory, building and testing in the real world, and applications. It is a broad, multidisciplinary, long-term endeavor, where what we learn in one path aids the others. We believe that such a broad endeavor is ultimately the only way to construct complex autonomous machines that can justify their own existence in economic terms.

Acknowledgments

This research was supported over the years in part by the National Science Foundation (NSF), the office of Naval Research (ONR) and the Defense Advanced Research Projects Agency (DARPA). Publication of this paper was supported in part by the Rockefeller Foundation.

References

1. H.P. Moravec, "Rise of the Robots," *Scientific American* (December 1999) pp. 124–135.

2. J. Morrison and T. Nguyen, "On-Board Software for the Mars Pathfinder Microrover," *Proceedings of the Second IAA International Conference on Low-Cost Planetary Missions*, Johns Hopkins University Applied Physics Laboratory (April 1996).

3. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants* (New York: Springer-Verlag, 1990).

4. D. Cliff and J. Noble, "Knowledge-Based Vision and Simple Visual Machines," *Philosophical Transactions of the Royal Society of London: Series B* **352,** No. 1165–1175 (1997).

5. W. Lee et al., "A Hybrid GP/GA Approach for Coevolving Controllers and Robot Bodies to Achieve Fitness-Specified Tasks," in *Proceedings of IEEE 3rd International Conference on Evolutionary Computation* (1996) pp. 384–389.

6. H. Lund et al., "Evolving Robot Morphology," *Proceedings of IEEE Fourth International Conference on Evolutionary Computation* (1997) pp. 197–202.

7. P.J. Angeline and J.B. Pollack, "Coevolving High-Level Representations," in *Artificial Life III* (Addison-Wesley, 1994) pp. 55—71.

8. G.S. Hornby and J.B. Pollack, "Evolving L-Systems to Generate Virtual Creatures," *Computers and Graphics* **25**, No. 6, 1041–1048 (2001).

9. H. Juillé and J.B. Pollack, "Dynamics of Co-Evolutionary Learning," in *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (Cambridge, MA: MIT Press, 1996) pp. 526–534.

10. S. Ficici and J.B. Pollack, "Challenges in Coevolutionary Learning: Arms-Race Dynamics, Open-Endedness and Mediocre Stable States," in C. Adami et al., eds., *Proceedings of the Sixth International Conference on Artificial Life* (Cambridge, MA: MIT Press, 1998).

11. J. Holland, *Adaptation in Natural and Artificial Systems* (Ann Arbor: University of Michigan Press, 1975).

12. K. Sims, "Artificial Evolution for Computer Graphics," in *Computer Graphics (SIGGRAPH '91 Proceedings)* (Anaheim, CA: ACM-SIGGRAPH, 1991) pp. 319–328.

13. R. Dawkins, *The Blind Watchmaker* (New York: Norton, 1987).

14. P.J. Bentley, "Generic Evolutionary Design of Solid Objects Using a Genetic Algorithm," Ph.D. thesis, Division of Computing and Control Systems, School of Engineering, University of Huddersfield, U.K. (1996).

15. M. van de Panne and E. Fiume, "Sensor-Actuator Networks," in *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series (Anaheim, CA: ACM SIGGRAPH, 1993) pp. 335–342.

16. L. Gritz and J.K. Hahn, "Genetic Programming for Articulated Figure Motion," *Journal of Visualization and Computer Animation* **6**, No. 3, 129–142 (July 1995).

17. R. Grzeszczuk and D. Terzopoulos, "Automated Learning of Muscle-Actuated Locomotion through Control Abstraction," *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pp. 63–70, Los Angeles, California, August 1995, in *Computer Graphics* Annual Conference Series (1995).

18. K. Sims, "Evolving Virtual Creatures," in *Computer Graphics* Annual Conference Series (1994).

19. K. Sims, "Evolving 3D Morphology and Behavior by Competition," in R. Brooks and P. Maes, eds., *Proceedings 4th Artificial Life Conference* (Cambridge, MA: MIT Press, 1994) pp. 28–39.

20. M. Komosinski and S. Ulatowski, "Framsticks: Towards a Simulation of a Nature-Like World, Creatures and Evolution," in Jean-Daniel Nicoud et al., eds., *Proceedings of 5th European Conference on Artificial Life (ECAL99)*, Vol. 1674 of *Lecture Notes in Artificial Intelligence* (New York: Springer-Verlag, 1999) pp. 261–265.

21. D. Floreano and F. Mondada, "Evolution of Homing Navigation in a Real Mobile Robot," *IEEE Transactions on Systems, Man, and Cybernetics* (IEEE Press, 1996).

22. D. Cliff et al., "Evolution of Visual Control Systems for Robots," in M. Srinivisan and S. Venkatesh, eds., *From Living Eyes to Seeing Machines* (Oxford, U.K.: Oxford Univ. Press, 1996).

23. H. Lund, "Evolving Robot Control Systems," in J.T. Alexander, ed., *Proceedings of 1NWGA* (Vaasa, Finland: University of Vaasa, 1995).

24. J.C. Gallagher et al., "Application of Evolved Locomotion Controllers to a Hexapod Robot," *Robotics and Autonomous Systems*, **19** (1996) pp. 95–103.

25. Y. Kawauchi et al., "Genetic Evolution and Self-organization of Cellular Robotic System," *JSME Int. J. Series C. (Dynamics, Control, Robotics, Design & Manufacturing)* **38**, No. 3, 501–509 (1999).

26. P. Funes and J.B. Pollack, "Computer Evolution of Buildable Objects," in Phil Husbands and Inman Harvey, eds., *Fourth European Conference on Artificial Life* (Cambridge, MA: MIT Press, 1997) pp. 358–367.

27. P. Funes and J.B. Pollack, "Evolutionary Body Building: Adaptive Physical Designs for Robots," *Artificial Life* **4** (1998) pp. 337–357).

28. P. Funes and J.B. Pollack, "Computer Evolution of Buildable Objects," in P. Bentley, ed., *Evolutionary Design by Computers* (San Francisco: Morgan Kaufmann, 1999) pp. 387–403.

29. We use a Stratasys machine that "prints" layer after layer of plastic in response to a computer-generated 3D model. The largest buildable piece fits inside an 8×8×12-in volume.

30. H. Lipson and J.B. Pollack, "Automatic Design and Manufacture of Robotic Lifeforms," *Nature* **406**, No. 6799, 974–978 (2000).

31. C. Kane and M. Schoenauer, "Genetic Operators for Two-Dimensional Shape Optimization," in J.-M. Alliot et al., eds., *Artificial Evolution EA95* (New York: Springer-Verlag, 1995).

32. P. Husbands et al., eds., "Two Applications of Genetic Algorithms to Component Design," in T. Fogarty, ed., *Evolutionary Computing: LNCS 1143* (New York: Springer-Verlag, 1996) pp. 50–61.

33. P. Bentley, ed., *Evolutionary Design by Computers* (San Francisco: Morgan Kaufmann, 1999).

34. J. Koza, *Genetic Programming* (Cambridge, MA: MIT Press, 1992).

35. F. Gruau, "Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm," Ph.D. thesis, Ecole Normale Supérieure de Lyon, 1994.

36. A. Lindenmayer, "Mathematical Models for Cellular Interaction in Development," Parts I and II, *Journal of Theoretical Biology* **18** (1968) pp. 280–299, 300—315.

37. G.S. Hornby and J.B. Pollack, "The Advantages of Generative Grammatical Encodings for Physical Design," in *Congress on Evolutionary Computation* (Piscataway, NJ: IEEE Press, 2001) pp. 600—607.

38. G.S. Hornby et al., eds., "Evolution of Generative Design Systems for Modular Physical Robots," in *Proceedings IEEE International Conference on Robotics and Automation* (2001) pp. 4146—4151.

39. G.S. Hornby and J.B. Pollack, "Body-Brain Coevolution Using L-Systems as a Generative Encoding," in *Proceedings Genetic and Evolutionary Computation Conference* (2001) pp. 868—875.

40. G.S. Hornby and J.B. Pollack, "Creating High-Level Components with a Generative Representation for Body-Brain Evolution," *Artificial Life*, forthcoming.

41. N. Jakobi, "Minimal Simulations for Evolutionary Robotics," Ph.D. thesis, School of Cognitive and Computing Sciences, University of Sussex, U.K. (May 1998).

42. G.S. Hornby, et al., "Evolution of Controllers from a High-Level Simulator to a High DOF Robot," in J. Miller, ed., *Evolvable Systems: From Biology to Hardware; Proceedings of the Third International Conference on Evolutionary Systems (ICES 2000)*, Lecture Notes in Computer Science, Vol. 1801 (New York: Springer, 2000) pp. 80—89.

43. R. Watson et al., "Embodied Evolution: Embodying an Evolutionary Algorithm in a Population of Robots," in P. Angeline et al., eds., *1999 Congress on Evolutionary Computation* (1999).

44. J.B. Pollack and A.D. Blair, "Coevolution in the Successful Learning of Backgammon Strategy," *Machine Learning* **32** (1998) pp. 225–240.

45. H. Juillé and J.B. Pollack, "Co-Evolving Intertwined Spirals," *Proceedings of the Sixth International Conference on Genetic Algorithms* (1996) pp. 351—358.

46. H. Juillé and J.B. Pollack, "Coevolving the 'Ideal Trainer': Discovery of Cellular Automata Rules," in J. Koza, ed., *Proceedings of the Third Annual Genetic Programming Conference* (July 1998).

47. M. Newborn, *Kasparov vs. Deep Blue: Computer Chess Comes of Age* (New York: Springer-Verlag, 1996).

Manuscript received 8 February 2002.

Jordan B. Pollack is a computer-science professor at Brandeis University. He received his Ph.D. from University of Illinois in 1987. He directs the DEMO laboratory, which is known for work in evolution, robotics, learning and complex systems.

Gregory S. Hornby received his Ph.D. in computer science from Brandeis University in 2002 and now works at NASA Ames Research Center.

Hod Lipson is a mechanical engineering professor at Cornell University. He received his Ph.D. from Technion in 1998, and was a postdoctoral researcher at Brandeis University through 2001.

Pablo Funes received his Ph.D. from Brandeis University in 2000, and is now a staff member at Icosystem.