Evolution of neural controllers for locomotion and obstacle-avoidance in a 6-legged robot

David Filliat, Jérôme Kodjabachian, Jean-Arcady Meyer

Evolution of Neural Controllers for Locomotion and Obstacle-Avoidance in a 6-legged Robot

D. Filliat†* J. Kodjabachian* ** J.-A. Meyer* **

*AnimatLab / OASIS - LIP6
Case 169. 4, place Jussieu. Paris. France.
75252 PARIS CEDEX 05 France.
†phone : (+33 1 44 27 88 04)

**MASA
24 Boulevard de l'Hôpital.
75005 Paris. France.

Keywords

Evolutionary robotics, neural controllers, locomotion, obstacle avoidance, walking robot.

Abstract

This article describes how the SGOCE paradigm has been used within the context of a "minimal simulation" strategy to evolve neural networks controlling locomotion and obstacle-avoidance in a 6-legged robot. A standard genetic algorithm has been used to evolve developmental programs according to which recurrent networks of leaky-integrator neurons were grown in a user-provided developmental substrate and were connected to the robot's sensors and actuators. Specific grammars have been used to limit the complexity of the developmental programs and of the corresponding neural controllers. Such controllers have been first evolved through simulation and then successfully downloaded on the real robot.

1 Introduction

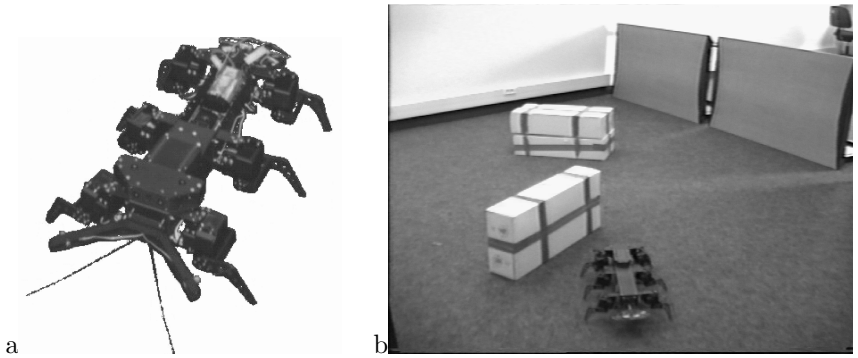
To overcome the difficulties of designing the control architecture of a truly autonomous robot - i.e., a robot that has to survive and fulfill its mission in a variety of challenging circumstances that are impossible to predict - numerous researchers advocate the use of evolutionary procedures that bypass human intervention insofar as possible and that more or less automatically adapt a robot's controller and morphology to the specific problems it has to solve. The basic idea underlying such approaches to evolutionary robotics (Gomi, 1997, 1998; Higuchi et al., 1997; Husbands and Meyer, 1998), is to draw inspiration from biology and to implement a process of artificial selection that exploits the coding of a robot's phenotype into its genotype.

In three previous articles (Kodjabachian and Meyer, 1995, 1998a, 1998b) the advantages of inserting a developmental process between the genotype and the phenotype have been emphasized and a methodology implementing such a process has been developed. In particular, it has been shown how the so-called SGOCE evolutionary paradigm could be used to generate neural networks capable of controlling the behavior of a simulated artificial insect. More specifi-

cally, developmental programs generating controllers for locomotion, obstacle-avoidance and gradient-following have been automatically generated, thus endowing the insect with navigation abilities through a simple guidance strategy (Trullier and Meyer 1997, Trullier et al. 1997).

In this article, we show how the SGOCE paradigm has been used to generate neural controllers for locomotion and obstacle-avoidance in a real 6-legged SECT robot manufactured by Applied AI Systems (figure 1).

The article first outlines the main characteristics of the SGOCE paradigm. Then, it describes how neural controllers for locomotion and obstacle-avoidance has been evolved in the SECT robot. Finally, it discusses the advantages and limitations of SGOCE.



a b

Figure 1: a : The SECT robot. It is equipped with infra-red and light sensors. Each leg is controlled by two servo-motors that react to angular-position commands, e.g., one for horizontal moves and one for vertical moves. b : The robot and the environment that has been used to check the evolved behaviors.

2 The SGOCE evolutionary paradigm

The SGOCE (Simple Geometry-Oriented Cellular Encoding) evolutionary paradigm is characterized by an encoding scheme, by an evolutionary algorithm, by an incremental strategy, and by a fitness evaluation procedure that will be sketched in turn. More detailed descriptions can be found in Chavas et al. (1998), Filliat (1998), Kodjabachian and Meyer (1998a, 1998b).

2.1 Encoding scheme

The encoding scheme of SGOCE (figure 2) is a geometry-oriented variation of Gruau's cellular encoding (Gruau 1994). The developmental programs that are evolved have a tree-like structure and call upon developmental instructions that cause a set of precursor cells positioned by the experimenter in a 2D metric substrate to divide, die, or grow efferent or afferent connections. In particular, such

cells can get connected to each other, or to sensory cells and motoneurons that have also been positioned in the substrate. Thus, a possibly short and compact genotype may ultimately produce a complex phenotype, i.e., a fully recurrent neural network made of individual leaky-integrator neurons and able to control the behavior of the robot through its sensors and actuators. Within such networks, the mean membrane potential m_i of a neuron N_i evolves according to the following equation:

$$\tau_i \frac{dm_i}{dt} = -m_i + \sum_j \omega_{i,j} x_j + I_i \tag{1}$$

where $x_i = (1 + e^{-(m_i + B_i)})^{-1}$ is the neuron short-term average firing frequency, B_i is a uniform random variable whose mean b_j is the neuron's bias, and τ_i is the time constant associated with the passive properties of N_i's membrane. I_i is the input that neuron N_i may receive from a given sensor, and $w_{i,j}$ is the synaptic weight of a connection from neuron N_j to neuron N_i.

A leaky-integrator model has been chosen a priori because neurons with internal dynamics were considered as suitable for the implementation of biomimetic central pattern generators (Delcomyn 1980) and because this kind of model is known to be an universal dynamics approximator (Beer 1995).

2.2 Evolutionary algorithm

The evolutionary algorithm of SGOCE is a steady-state genetic algorithm that involves a population of well-formed developmental programs whose structure is constrained by a grammar provided by the experimenter (figure 3). The use of such a grammar makes it possible to reduce the size of the genotypic space explored by the algorithm and to limit the complexity of the neural networks that are evolved.

2.3 Incremental strategy

Finally, the SGOCE paradigm calls upon an incremental strategy that takes advantage of the geometrical nature of the developmental process. In particular, it makes it possible to automatically generate appropriate controllers and behaviors through successive stages, in which good solutions to a simpler version of a given problem are iteratively used to seed the initial population of solutions likely to solve a harder version of the same problem.

Thus, in a first stage of the present work, the SGOCE paradigm was used to generate a recurrent neural network controlling straight locomotion in the SECT robot. At the end of this stage, this network was frozen, in the sense that the number of its neurons, their individual parameters, and their intra-modular connections were not allowed to evolve anymore. However, during a second evolutionary stage, an additional recurrent neural network was evolved and its neurons were allowed to grow, not only intra-modular connections between themselves, but also inter-modular connections to neurons in the loco-

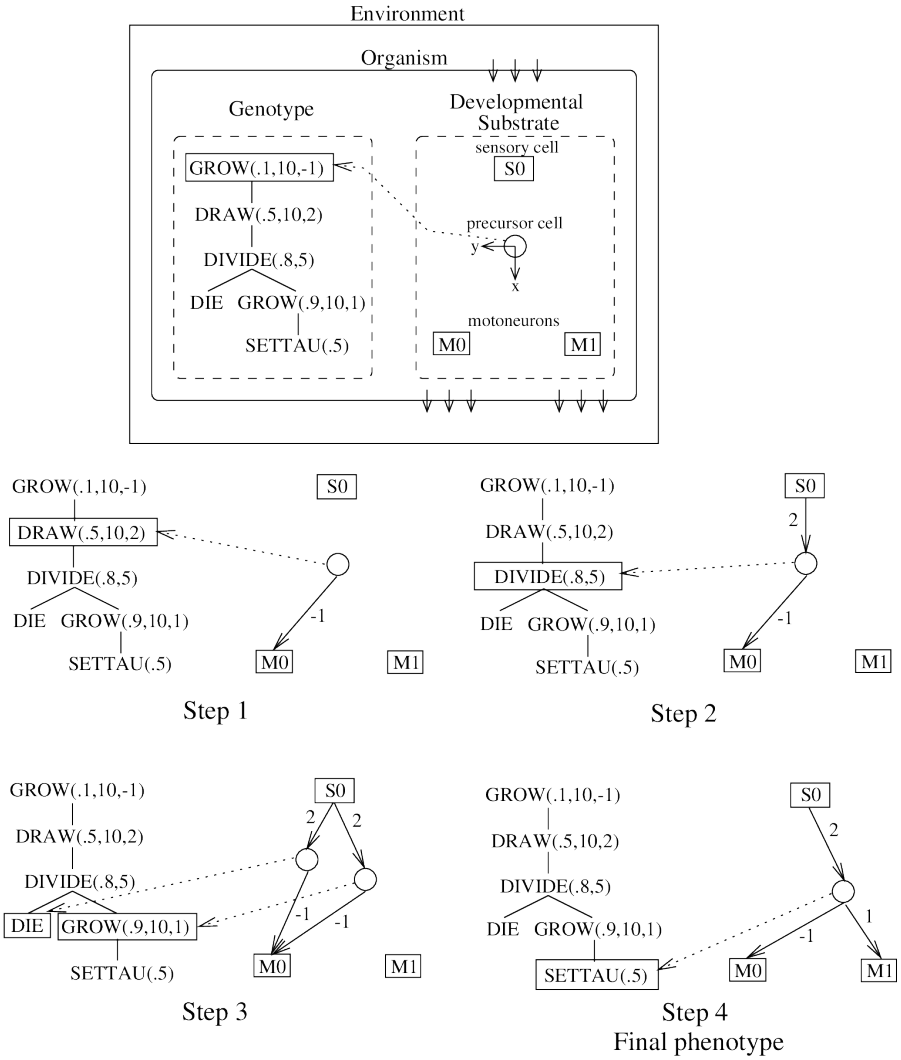


Figure 2: The developmental encoding scheme of SGOCE. The genotype that specifies the robot's nervous system is encoded as a program whose nodes are specific developmental instructions. This developmental program is read at a different position by each cell in the substrate. The precursor cell first makes connections with the motoneuron M0 and the sensory cell S0 (Steps 1 and 2). Then it divides, giving birth to a new cell that gets the same connections than those of the mother cell (Step 3). Finally, the mother cell dies, while the daughter cell makes a connection with the motoneuron M1 and changes the value of the time constant τ in equation 1 (Step 4).

Terminal symbols
DIVIDE, GROW, DRAW, SETBIAS, SETTAU, DIE,
NOLINK, DEFBIAS, DEFTAU, SIMULT3, SIMULT4.
Variables
Start1, Level1, Level2, Neuron, Bias, Tau, Connex, Link.
Production rules
Start1⟶DIVIDE(Level1, Level1)
Level1⟶DIVIDE(Level2, Level2)
Level2⟶DIVIDE(Neuron, Neuron)
Neuron⟶SIMULT3(Bias, Tau, Connex) | DIE
Bias⟶SETBIAS | DEFBIAS
Tau⟶SETTAU | DEFTAU
Connex⟶SIMULT4(Link, Link, Link, Link)
Link⟶GROW | DRAW | NOLINK
Starting symbol
Start1.

Figure 3: This grammar defines a set of developmental programs, i.e., those that can be generated from it, starting with the Start1 symbol. When this grammar is used, a cell that executes such a program undergoes two division cycles, yielding four daughter cells, which can either die or modify internal parameters (e.g., time constant τ or bias B in equation 1) that will influence their behavior within the final neural controller. Finally, each surviving cell establishes a number of connections, either with another cell, or with the sensory cells or motoneurons that have been positioned by the experimenter in the developmental substrate. According to this grammar, no more than three successive divisions can occur and the number of connections created by any cell is limited to four. Thus, the final number of interneurons and connections created by a program well-formed according to this grammar cannot be greater than 8 and 32, respectively.

motion controller. This additional controller was expected to modulate the leg movements secured by the first controller, so as to make it possible for the robot to turn in the presence of an obstacle in order to avoid it.

Figure 4 describes the two substrates that have been used to generate the two modules of the present application.

2.4 Fitness evaluation

Fitness evaluation is one of the main difficulties of the evolutionary design of controllers in real robots (Mataric and Cliff, 1996; Meyer et al., 1998). Such difficulties are enhanced in the case of legged robots, because these robots tend to be more brittle than their wheeled counterparts, and because fitness evaluations require a lot of time and cannot be easily automated. We therefore chose to use simulations to assess the fitness of our controllers, taking advantage of an argument put forth by Jakobi (Jakobi, 1997; 1998), namely that what really matters is to accurately simulate the efficient behaviors that will be used by the real robot. Less efficient behaviors - i.e., behaviors that an efficient robot will not exhibit in reality - do not need to be minutely simulated, as long as one is sure that their fitnesses will be lower than the fitnesses of the behaviors that are sought. Pushing such reasoning to the extreme, Jakobi evolved neural controllers for an octopod robot capable of walking, of avoiding obstacles using its infra-red sensors, and backing away from objects that were hit with

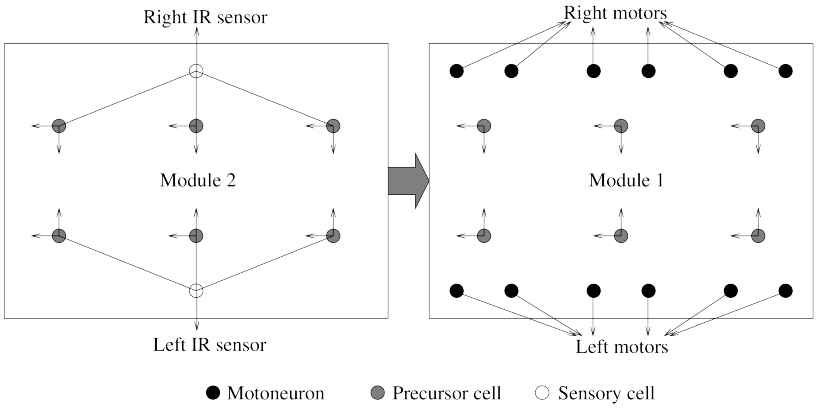


Figure 4: The substrates and the modular approach that have been used in the present work. Module 1 produces straight walking, while Module 2 modifies the behavior of Module 1 in order to avoid obstacles.

its bumpers. Jakobi's approach did not resort to any simulation of the robot's behavior in its environment, and only relied on the specification that legs on the floor should move backwards as fast as possible, and that legs in the air should move forward as fast as possible. Given such specification, the simulation only rewarded controllers that did generate these movements. However, despite the practical success of Jakobi's approach, our aim was to provide less constraints on the target behavior. Therefore, we only specified that the robot should go ahead as far as possible while avoiding obstacles, and we did not provide any hints about leg movements. To this end, we had to design a simulation of the behavior of the robot in its environment.

2.5 Simulation

As Jakobi points out, the difficulty in devising a legged robot simulation is to manage the cases when some leg slippage occurs. However, because such events are only involved in poorly efficient behaviors, they are not expected to occur with a fast-walking gait. As a consequence, controllers producing leg slippage will never be used by an efficient real robot, and therefore leg slippage does not need to be accurately simulated, thus tremendously simplifying the simulation problem. [1]

According to such considerations, our simulation assumed that all the legs

[1] Naturally, this argument holds for the kind of intrinsically non-slippery surface on which the robots that are evolved here are implicitly supposed to walk. It is clear that even the most efficient controller thus evolved will not be able to avoid slippage on a slippery surface. If we were seeking mechanisms to cope with such circumstances, we should either implement additional learning capacities to the evolved controllers or explicitly confront successive generations to the relevant surfaces.

were characterized by the same constant slippage factor, and simply calculated the movement of the robot body that minimized the slippage of any leg touching the floor between two time steps. As a consequence, if the real movement did not involve any slippage, the calculated movement was exact and, conversely, if the real movement did involve slippage, the calculated movement was a good approximation of the real one.

The algorithm used for the simulation is given in figure 5.

- Set initial robot leg and body positions.
- Iterate N times :
    1. Update sensor values using robot position.
    2. Update neural network outputs M times using sensor values.
    3. Update leg positions relative to robot body using network outputs.
    4. Find a set S of 3 legs Li, Lj, Lk, the extremities xi, xj, xk of which form a support triangle.
    5. Add into S all legs lying within a distance d of the plane P generated by xi, xj and xk.
    6. Find the elementary translation and rotation of the body that minimize the displacement of the extremities of the legs in S.
    7. Update body position, applying the optimal rotation and translation.

Figure 5: The algorithm used for the simulation of the 6-legged robot

The simulation of the sensors of the robot (step 1 in figure 5) was straightforward, as only 2 binary obstacle detectors, placed on both sides of the head, were used. It entailed simply calculating the distance of each robot sensor to the closest obstacles it could detect and, if the distance fell under a certain threshold, setting the activity of the corresponding sensory neuron to 1. Otherwise, the activity was set to 0.

At every iteration of the algorithm, the neural network was updated M times (M=1 in the present work) using the updated sensory activities to yield new motoneuron outputs (step 2 in figure 5).

The position of each leg relative to the robot body (step 3 in figure 5) was obtained through a simple geometric computation because leg positions were determined by neural network outputs only.

The determination of the legs in contact with the ground first involved finding three legs whose extremities formed a support triangle (step 4 in figure 5). Such a triangle had to be such that all other leg extremities were above its plane, and that the vertical projection of the robot's center of mass fell inside such a triangle.

Then, all leg extremities ending at a sufficiently short distance from the plane of the triangle were also considered as being in contact with the ground (step 5 in figure 5). The aim of this step was to smoothen the effect of differences in leg lengths and positions in real and simulated robots. This warranted that more

legs were in contact with the ground in the simulation than in reality, which implied that the fitnesses of the controllers were not overestimated because controllers that did not lift the legs above a given threshold were penalized in the simulation.

The computation of the movements of the robot body was based on the assumption that the actual slippage of the legs on the ground minimized the energy dissipated by friction. In the simulation, it was assumed that the slippage factor was constant and identical for all legs. As a consequence, the energy dissipated by friction was proportional to the sum of the distances covered by all leg extremities in contact with the ground. Hence, to determine the body movement between two time steps, the elementary horizontal rotation and translation of the robot that minimized the displacement of the legs on the ground were sought (step 6 in figure 5). To achieve this, the distance covered by the legs in contact with the ground was computed as a function of three parameters : the two coordinates of the body translation and the angle of the body rotation. Minimizing this function, with the assumption that the body rotation remained small, lead to a simple linear system with 3 equations and 3 unknowns, which was computationally easy to solve.

Such simulations have been used to assess the fitness of each developmental program produced by the evolutionary process (figure 6). Once an efficient controller was thus obtained, it was downloaded on the SECT robot, where its ability to generate the target behaviors in reality was assessed again.

3 Experimental results

3.1 Locomotion

The 2D substrate that was used in this experiment corresponds to Module 1 shown in figure 4. It contained 12 motoneurons that were connected to the 12 motors of the robot, in the sense that the activity level of a given motoneuron determined the target angular position that was sent to the corresponding servo-motor. The six precursor cells executed the same evolved developmental program in order to impose symmetrical constraints to the growing neural network. Developmental programs were generated according to the grammar shown in figure 3. As for each controller's fitness, it was estimated by the mean distance covered in 3 obstacle-free environments during a fixed amount of time, increased by a slight bonus encouraging any leg motion (Kodjabachian and Meyer, 1998a; 1998b). It should also be noted that the simulation implicitly rewarded controllers that raised legs above a small threshold as mentioned in section 2.5. Finally, the size of the population was of 100 individuals that evolved during 500 generations (taking 24 hours on a SUN Ultra 1).

All the evolved controllers could be classified into two categories: those generating tripod gaits and those generating symmetrical gaits (i.e., moving the corresponding legs on both sides of the robot simultaneously). Such controllers were as efficient in reality as they were in simulation (figure 7). The main

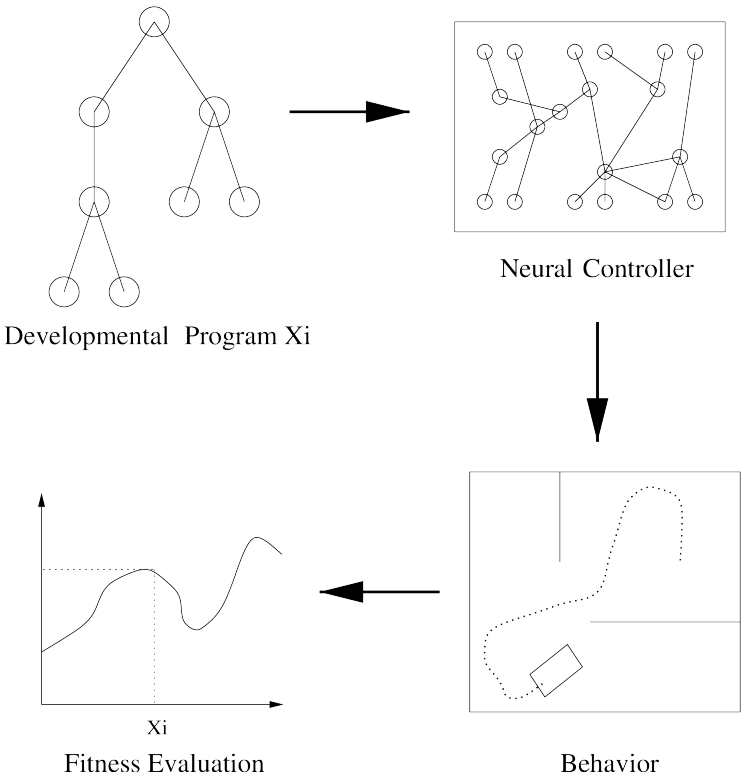


Figure 6: The three stages of the fitness evaluation procedure. An evolved developmental program is executed to yield an artificial neural network. Then, the neural network is used to control the behavior of a simulated robot. Finally, the fitness of the program is assessed according to the result of the simulation.

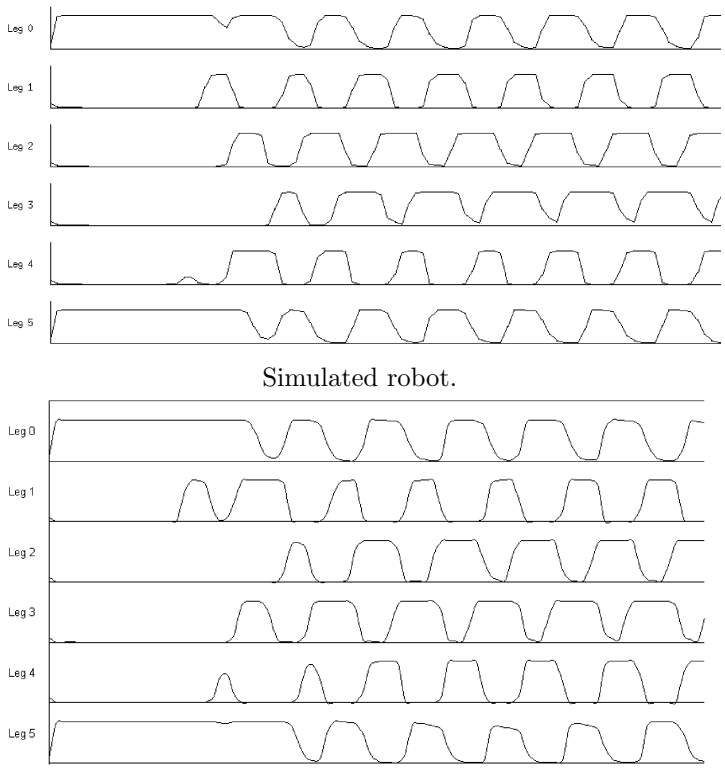


Figure 7: Comparison of leg movements in the simulated and the real robot. Both graphs show commands sent to leg swing motors. Although the controller's outputs might have been altered by the procedure preventing over currents in the real robot, the commands actually sent to the motors, and hence the robot's overall behaviors, are qualitatively the same in simulation and in reality.

differences between both situations were due to the fact that, on the real robot, a continuous monitoring of the motor currents might entail modifying the motor commands independently of the neural network when such current were too high. This security procedure, which was implemented to avoid motor breaks in leg-blocking situations, was triggered in a few occasions when symmetric gaits were used, because such gaits occasionally provoked jumps producing very high motor currents.

Figure 8 shows the best tripod-gait controller that has been obtained. We analyzed its behavior in order to understand how this gait was generated. The mechanism responsible for leg lift is straightforward. It calls upon 6 central pattern generators, one for each leg, made up of only 3 neurons (figure 9). These pattern generators are synchronized by two connections between the legs of the same side of the body, and by two connections linking 2 symmetric legs on each side of the body - thus closely approximating the biologically plausible model of Cruse et al. (1995). The mechanism producing leg swing is far more intricate because it is extremely distributed. In fact, the activation of a given leg's swing neuron depends on neurons involved in the control of all the other legs, and it cannot be decomposed in six similar units as is the case for the lift neurons.

3.2 Obstacle-avoidance

Obstacle-avoidance was evolved using a second module whose neurons could be connected to those of the tripod-gait controller of figure 8. The substrate of this second module is shown on figure 4. The grammar was identical to that of figure 3, with two exceptions. The first one concerns the replacement of rule Start1$\longrightarrow$DIVIDE(Level1, Level1) by Start1$\longrightarrow$DRAWI & DIVIDE(Level1, Level1), where DRAWI is a terminal symbol creating a connection from an input cell, and where character & means that the two symbols it connects should be executed in sequence. This rule enforces connections between input neurons and precursor cells. The second exception concerns the addition of the instruction GROWD to the rule Link$\longrightarrow$GROW | DRAW | NOLINK. This instruction allows the precursor cell to grow an outgoing connection towards neurons in the first module. Each IR sensor was binary, i.e., it returned 0 when it detected no obstacle, and it returned 1 when an obstacle was detected within a 30 cm-range. The robot evolved in 3 different closed environments containing some obstacles (figure 10). Instead of specifying what leg movements should be favored in each possible sensory circumstance, as Jakobi (1998) did, the fitness was the distance covered by the robot until it touched an obstacle, or until the simulation time was over, averaged over the 3 environments. The population was made of 100 individuals and evolution lasted 200 generations.

The robot's simulated and actual behaviors were very similar and simple: legs on the opposite side of a detected obstacle were blocked, thus causing the robot to change its direction and to avoid the obstacle. Analyzing the inner working of the corresponding controller (figure 9), it turned out that such behavior was possible due to a strong inhibitory connection between a given

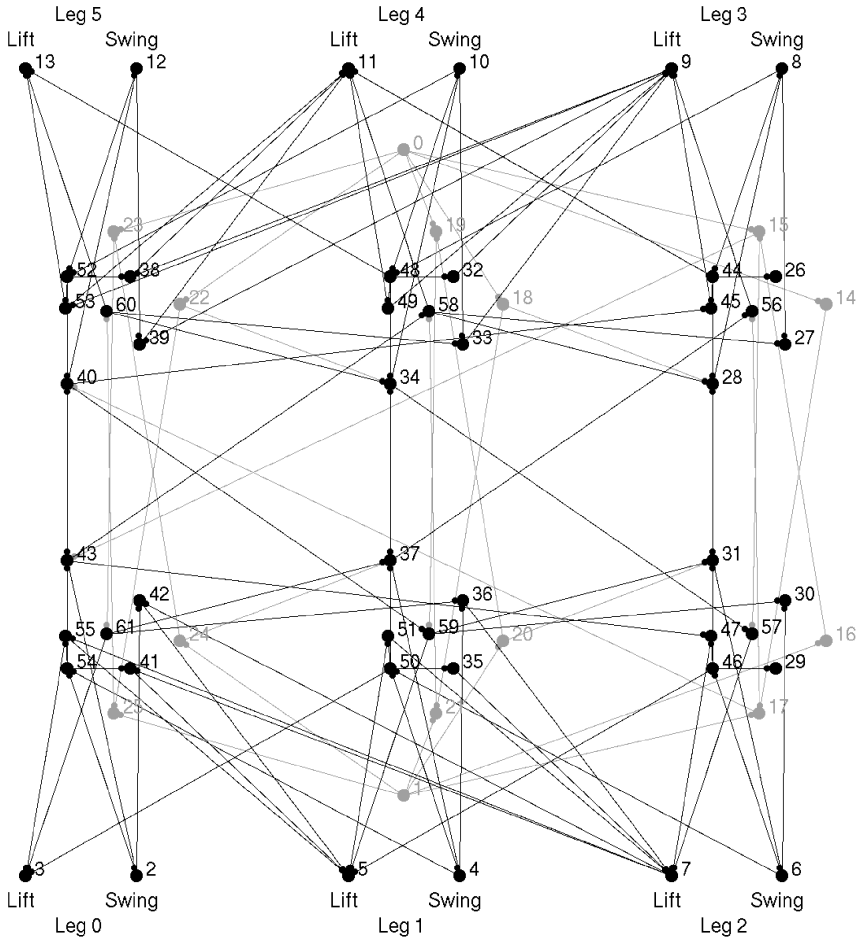


Figure 8: An example of an evolved controller. Black neurons belong to the module controlling locomotion, gray neurons belong to the module controlling obstacle-avoidance. Neurons 0 and 1 are input neurons connected to the IR sensors, neurons 2 to 13 are output neurons connected to the robot motors. The first controller contains 48 neurons and 82 connections; the second contains 14 neurons and 36 connections.

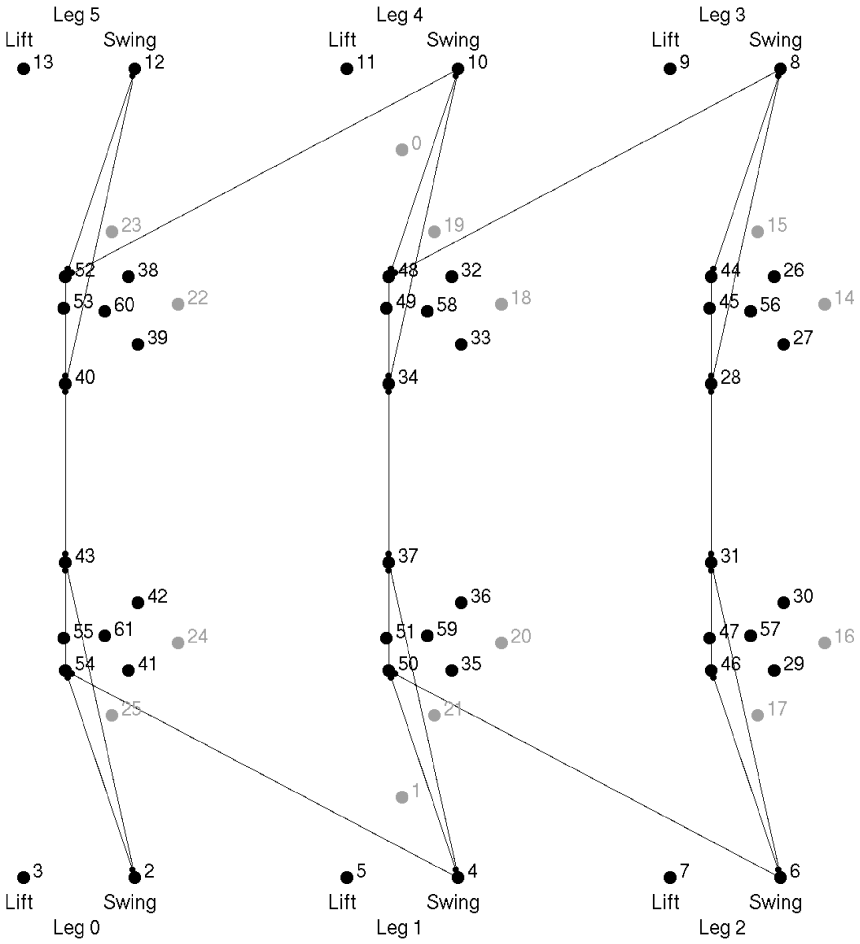


Figure 9: The central pattern generator that controls the swing motors of the robot. A group of 3 neurons (2, 43, 54 and their 5 symmetrical counterparts) participates in six identical and independent oscillators whose phase differences are controlled by the connections which link them (54-4, 40-43 and their symmetrical counterparts). The obstacle-avoidance module gets connected to this subnetwork only. Its inner-working is simple : whenever a IR neuron is activated, the connections between neurons 18-28,20-31,22-34,24-37, 17-40 and 15-43 (not shown here) stop the oscillators on the opposite side of the detected obstacle, hence blocking the leg swing movement and making the robot turn to avoid the obstacle.

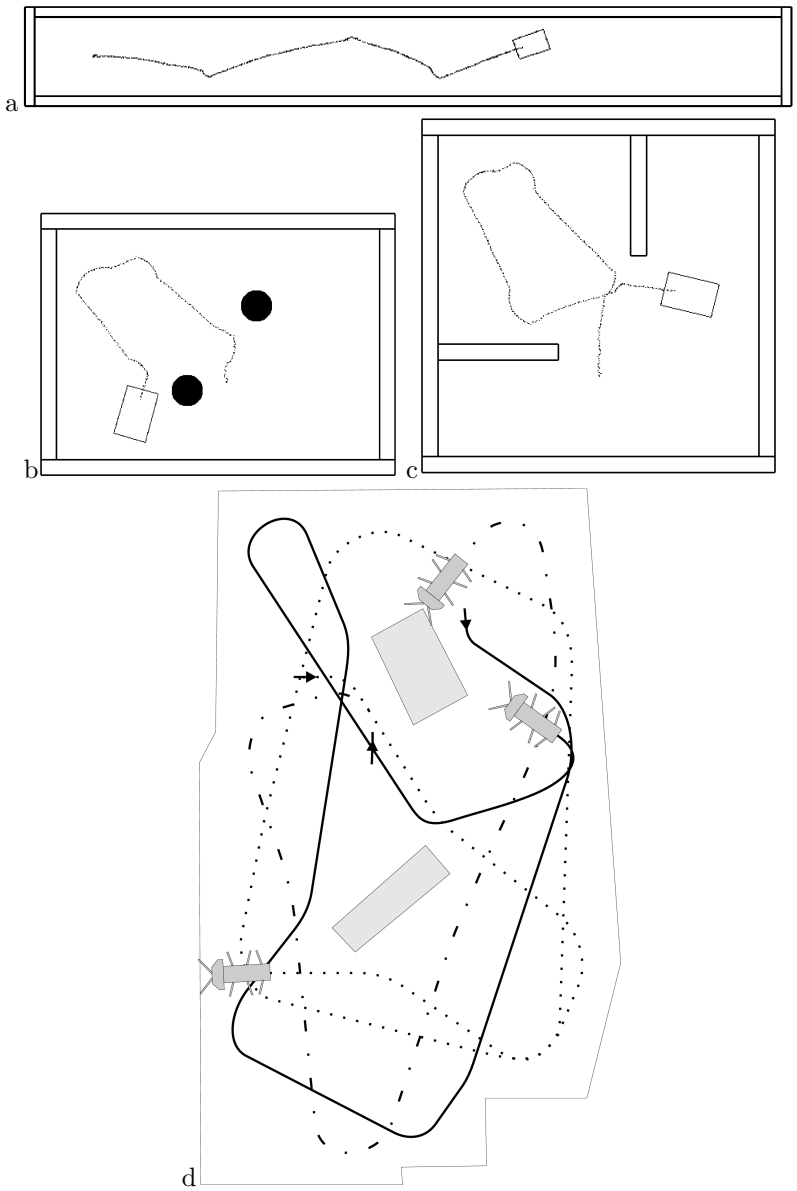


Figure 10: Robot behavior in simulated (a,b,c) and real (d) environments. The 3 trajectories shown in d have been reconstructed from video recordings of actual experiments. They illustrate the fact that the robot actually avoids obstacles most of the time, but that it may occasionally fail because of the dead-angle between its front IR sensors.

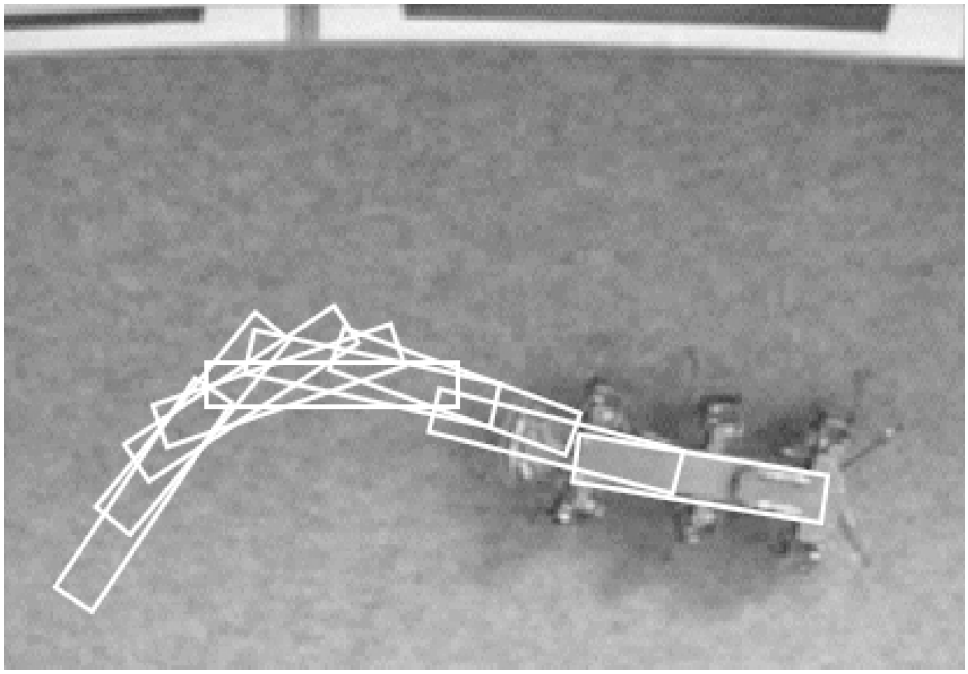


Figure 11: Close-up on the obstacle-avoidance behavior.

sensor and the swing neurons of the legs on the opposite side of the robot.

Figures 11 illustrates a specific avoidance behavior when the robot encounters a given obstacle. Figure 10 shows the robot's trajectories over longer periods of time, in 3 simulated environments and in reality.

4 Discussion

Results that have been shown here demonstrate that the "minimal simulation" methodology advocated by Jakobi (Jakobi, 1997; 1998) may be effectively used to evolve simple behaviors in a real robot. However, our implementation of this methodology makes it possible to avoid specifying as many details about the dynamics of each effector involved in the production of the sought behavior as Jakobi was committed to do. Actually, we succeeded to evolve locomotion and obstacle-avoidance in a legged robot simply rewarding movement and punishing obstacle-hitting, and such high-level specification is more in the spirit of the automatic generation of behavioral controllers that evolutionary approaches afford than Jakobi's solution is. Nevertheless, it must be emphasized that a cost is associated to such a benefit, namely that of entailing a more detailed simulation than that of Jakobi. In other words, what is gained at the level of the simulation, may be lost at the level of the fitness evaluation and, for obvious lack of hindsight reasons, it is definitely unclear where it is worth devoting more resources in order to evolve any given behavior on a real robot.

Results that have been shown here also demonstrate the effectiveness of the SGOCE evolutionary paradigm. Its potential to evolve locomotion and higher-level behaviors in a simulated insect have already been exemplified elsewhere (Kodjabachian and Meyer, 1998a; 1998b). In the present work such potentialities have been extended to a real 6-legged robot. However, for the same lack of hindsight reasons as above, it is still unclear whether each aspect of this paradigm is mandatory, useful, or without any effect at all. Although many identical implementation details have been successfully used in the above-mentioned applications and in others (Chavas et al., 1998; Ijspeert and Kodjabachian, 1999), one may wonder, for instance, if another evolutionary procedure than a genetic algorithm - e.g., an evolution strategy (Schwefel, 1995) or a genetic programming approach (Koza, 1992) - would not lead to better results. Likewise, it is unclear if the developmental instructions or the constraining grammars that have been used here might not have been replaced by others, and if freezing one controller and letting a second one evolve is a better strategy than evolving both controllers at the same time. Concerning the latter point, however, it has been shown on a specific application involving obstacle-avoidance in a Khepera robot that this was not the case (Chavas et al. 1998) but, again, any generalization to other behaviors and other robots would be definitely premature. Finally, it must be acknowledged that there is at the moment definitely no hint about the convergence properties of the algorithm that has been used here.

Current research efforts are aiming at endowing the robot with additional behavioral capacities. In particular, neural modules controlling backward lo-

comotion and light-seeking are sought. Such efforts obviously raise strategic issues like that of deciding in which order such modules should be evolved, according to which fitness functions and which grammars. Intuition suggests that it is wiser to evolve straight-locomotion first, then backward-locomotion, and then strategies for turning. Likewise, it suggests that modules responsible for obstacle-avoidance and light-seeking, could be rather independent from each other, but should both get connected to the modules that control forward and backward locomotion. However, experience shows that nothing can be more counter-intuitive than the results of an artificial evolution process and success in the above-mentioned research efforts will possibly derive from entirely different options. In this respect, current research efforts involving modular architectures (Nolfi, 1997a, 1997b; Tani and Nolfi, 1998; Urzelai et al., 1998) could provide valuable insights.

5 Conclusions

Assuming that an on-board evolution of neural controllers would not have been feasible on a SECT robot because of the excessively high demands that would have been placed on its motors, we called upon a "minimal simulation" approach and upon the SGOCE evolutionary paradigm to evolve tripod-gait locomotion and obstacle-avoidance. Successful neural controllers have been obtained, both in simulation and in reality, and their inner workings have been deciphered. However, it is yet unclear whether every implementation detail that has been used here was mandatory to the present success, nor whether it would be useful in any other application.

Acknowledgements

The authors express their gratitude to Dr. Takashi Gomi and to Applied AI Systems for having kindly lent them the SECT robot.

References

Beer, R.D. (1995) On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior.* 3(4), 469-509.

Chavas, J., Corne, C., Horvai, P., Kodjabachian, J. and Meyer, J.A. (1998) Incremental Evolution of Neural Controllers for Robust Obstacle-Avoidance in Khepera. In Husbands, P. and Meyer, J.A. (Eds.). *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98.* Springer Verlag.

Cruse, H., Brunn, D.E., Bartling, Ch., Dean, J., Dreifert, M., Kindermann, T. and Schmitz, J. (1995) Walking: A Complex Behavior Controlled by Simple Networks. *Adaptive Behavior.* 3(4), 385-418.

Delcomyn, F. (1980) Neural basis for rhythmic behavior in animals. *Science.* 210, 1980.

Filliat, D. (1998) *Evolution de réseaux de neurones pour le contrôle d'un robot hexapode.* Lip6 Technical Report. University Paris VI.

Gomi, T. (1997) *Evolutionary Robotics - Vol I.* AAI Books.

Gomi, T. (1998) *Evolutionary Robotics - Vol II.* AAI Books.

Gruau, F. (1994) Automatic definition of modular neural networks. *Adaptive Behavior.* 3(2), 151-183.

Higuchi, T., Iwata, M. and Liu, W. (Eds.) (1997) *Evolvable Systems : From Biology to Hardware.* Springer Verlag.

Husbands, P. and Meyer, J.A. (Eds.). (1998)*Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98.* Springer Verlag.

Ijspeert, A.J. and Kodjabachian, J. (1999) Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life.* In press.

Jakobi, N. (1997): Evolutionary Robotics and the Radical Envelope of Noise Hypothesis. *Adaptive Behavior.* 6(2), 325-367.

Jakobi, N. (1998) Running across the reality gap : octopod locomotion evolved in minimal simulation. In Husbands, P. and Meyer, J.A. (Eds.), *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98.* Springer Verlag.

Kodjabachian, J. and Meyer, J.A. (1995) Evolution and development of control architectures in animats. *Robotics and Autonomous Systems.* 16, 161-182.

Kodjabachian, J. and Meyer, J.A. (1998a) Evolution and Development of Modular Control Architectures for 1-D Locomotion in Six-legged Animats. *Connection Science.* 10, 211-237.

Kodjabachian, J. and Meyer, J.A. (1998b) Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects. *IEEE Transactions on Neural Networks.* 9(5), 796-812.

Koza, J.R. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press.

Mataric, M. and Cliff, D. (1996) Challenges in evolving controllers for physical robots.*Robotics and Autonomous Systems.* 19(1), 67-83.

Meyer, J.A. (1998) Evolutionary approaches to neural control in mobile robots. In *Proceedings of the IEEE International Conference on Systems and Cybernetics.* San Diego.

Meyer, J.A., Husbands, P. and Harvey, I. (1998) Evolutionary Robotics: a Survey of Applications and Problems. In Husbands, P. and Meyer, J.A. (Eds.), *Proceedings of The First European Workshop on Evolutionary Robotics - EvoRobot'98.* Springer Verlag.

Nolfi, S. (1997a) Evolving non-trivial behavior on autonomous robots : adaptation is more powerful than decomposition and integration. In Gomi, T. (Ed.) *Evolutionary robotics. From intelligent robots to artificial life.* AAI Books.

Nolfi, S. (1997b) Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior* 5, 343-364.

Schwefel, H.P. (1995) *Evolution and Optimum Seeking.* Wiley.

Tani, J. and Nolfi, S. (1998) Learn to perceive world as articulated : an approach for hierarchical learning. In Pfeiffer, R. et al. (Eds.) *From Animals to Animats 5 : Proceeding of the Fifth International Conference on Adaptive Behavior.* The MIT Press.

Trullier, O. and Meyer, J.A. (1997) Biomimetic Navigation Models and Strategies in Animats. *AI Communications.* 10, 79-92.

Trullier, O., Wiener, S., Berthoz, A. and Meyer, J.A. (1997) Biologically-based artificial navigation systems: Review and Prospects. *Progress in Neurobiology.* 51, 483-544.

Urzelai, J., Floreano, D., Dorigo, M. and Colombetti, M. (1998) Incremental Robot Shaping. *Connection Science.* 10, 341-360.