Incremental Evolution of Neural Controllers for Robust Obstacle-Avoidance in Khepera.

J. Chavas, Christophe Corne, P. Horvai
Jérôme Kodjabachian, Jean-Arcady Meyer

November 10, 1999

Abstract

An incremental approach is used to simulate the evolution of neural controllers for robust obstacle-avoidance in a Khepera robot and proves to be more efficient than a direct approach. During a first evolutionary stage, obstacle-avoidance controllers in medium-light conditions are generated. During a second evolutionary stage, controllers avoiding strongly-lighted regions, where the previously acquired obstacle-avoidance capacities would be impaired, are obtained. The best controllers thus evolved are successfully downloaded on a Khepera robot. The SGOCE paradigm that is used in these experiments is described in the text. Future research will target at furthering the incremental evolutionary process and evolving more intricate behaviors.

1 Introduction

According to a recent review [26] of evolutionary approaches to neural control in mobile robots, it appears that the corresponding research efforts usually call upon a direct encoding scheme, where the phenotype of a given robot — *i.e.*, its neural controller and, occasionally, its body plan — is directly encoded into its genotype. However, it has often been argued (e.g., [13, 18]) that indirect encoding schemes — where the genotype actually specifies developmental rules according to which complex neural networks and morphologies can be derived from simple programs — are more likely to scale up with the complexity of the control problems to be solved, if only because the size of the genotypic space to be explored may be much smaller than that of the space of the resultant phenotypes.

The feasibility of such indirect approaches, which combine the processes of evolution and development, has been demonstrated through several simulations [2, ?, 5, 10, 19, 20, 32, 36, 38, 39] and a few applications involving real robots [6, 15, 16, 11, 28, 29]. However, the fact that the great majority of controllers and behaviors that have thus been generated are very simple, together with the difficulties encountered when more complex controllers and behaviors were sought [11, 20], led us to suspect that so-called incremental approaches [4, 12, 23] should necessarily be used in conjunction with indirect encoding schemes in more realistic applications. In other words, according to such a strategy, appropriate controllers and behaviors should be evolved and developed through successive stages in which good solutions to a simpler version of a given problem are used iteratively to seed the initial population of solutions likely to solve a harder version of the same problem.

In [20] such an incremental strategy has been used to evolve and develop neural controllers that permitted a simulated insect to successively walk, follow an odor gradient, and avoid obstacles. In this paper, it is used within the context of an evolutionary robotics application, where neural controllers for robust obstacle-avoidance in a Khepera robot are automatically generated. This work calls upon a two-stage approach, in which controllers for obstacle-avoidance in medium-light conditions are first evolved, and then improved to operate also in more challenging strong-light conditions, when a lighted lamp is added into the environment. Comparisons with results obtained under the alternative one-shot strategy are provided and support the above-mentioned intuition about the usefulness of an incremental approach.

2 Material and methods

This section will describe the task to be accomplished, and the SGOCE[1] paradigm that underlies our methodology. This task derives from the characteristics and limitations of the sensory motor apparatus of Khepera, which will be briefly summarized hereafter. Likewise, a short description will be provided of how this sensory motor apparatus has been simulated in this work. As for the description of the SGOCE methodology, it will deal successively with the developmental code that links the genotype of the robot to its phenotype, the syntactic constraints that limit the complexity of the phenotypes generated, the evolutionary algorithm inspired from genetic programming [21, 22] that generates developmental programs, and the incremental strategy that helps produce neural control architectures likely to exhibit increasingly adaptive capacities.

2.1 The obstacle-avoidance task

2.1.1 The real Khepera.

Khepera [31] is a circular-shaped miniature mobile robot — with a diameter of 55 mm, a height of 30 mm, and a weight of 70 g — that is mounted on two wheels and two small Teflon balls. In its basic configuration, it is equipped with eight proximity sensors — six on the front, two on the back — that may also act as visible-light detectors. The wheels are controlled by two DC motors with incremental encoders that move in both directions.

[1]This name is the acronym for the expression "Simple Geometry Oriented Cellular Encoding".

In each proximity sensor of Khepera, an infra-red light emitter and receiver are embedded. This hardware allows two things to be measured: the normal ambient light — through receivers only — and the light reflected by the obstacles — using both emitters and receivers. In medium-light conditions, this hardware makes it possible to detect an obstacle a short distance away — not more than about 5 cm. However, under strong light conditions, the corresponding receptors tend to saturate : the light emitted by the robot and reflected by obstacles cannot be distinguished from ambient light and, thus, cannot be detected (Figure 1). Therefore, this work aims at automatically evolving a robust obstacle-avoidance controller likely to differentiate between the two light conditions and to take appropriate motor decisions.

[Figure 1 comes about here]

In the present work, such a controller has been generated through simulations performed under the SGOCE paradigm. Then the corresponding network has been downloaded onto a Khepera robot and its ability to generate the requested behavior has been checked.

2.1.2 The simulated Khepera.

A cylindrical robot like Khepera is easier to model than a robot of arbitrary shape and with many degrees of freedom. Still, some phenomena, like friction, cannot be simulated with precision. Also, each sensor or motor has a unique behavior that can only be approximated in a simulation.

Our simulator is based on, and improves, an already existing simulator [27] and makes it possible to execute the same control program, either on the simulated robot or on the real one. It has four important features.

Firstly, it can be controlled by another independant program, making it easier to interface it with an already existing evolutionary algorithm software. This is important from a practical point of view, because the code can be reused more easily.

Secondly, it contains a set of functions specifically designed for artificial neural network evolution. One such function makes it possible for the evolutionary software to send to the simulator the description of a dynamic neural network, which will be connected in a specific way with the sensors and motors of the robot, whether real or simulated. Another function makes it possible to simulate the dynamics of that network during a given period of time, in order to control the robot. This function returns a fitness value, which is computed on the basis of information normally available to the robot, and which, when used with the real robot, is run entirely on board.

A third important feature of our simulator is its speed. Integer calculations are used to update the state of the neural network when computations are performed on board. Moreover, the sensor simulation method used by Michel has been replaced by a tabulation technique, according to which, prior to evolution, the values returned by a sensor in a given environment are recorded in a look-up-table for a number of different positions and orientations. Note that unlike in [30], where the values stored were measured on the real robot, here, we synthetize these values to make it easier to change the environmental conditions. At evaluation time, the sensor values are computed by interpolation from the values stored in the table.

Finally, another important feature to mention is the way in which sensor behavior is modelled. As already stated, Khepera IR sensors can work in either of two different modes. In passive mode, they return a measure m of the ambient light intensity I. In active mode, they return m^+, a measure of the intensity I^+, *i.e.*, the sum of the ambient intensity I and of the intensity dI of the light possibly reflected off an obstacle (Figure 1).

If the robot is at a specific position relative to a given configuration of obstacles, then the value dI will be the same whatever the level I of the ambient light. The proximity measure $p = K \cdot (m - m^+)$ can thus be used to caracterize the presence of an obstacle. However, because the function relating intensity I to measure m is non-linear, the same dI value will not yield the same difference $(m - m^+)$ for different levels of I. For this reason, p is not simply a function of the intensity dI reflected from the IR-ray, but also depends on I.

We have modified Michel's IR sensor model in order to take into account the possible effect of the ambient light level I on p. At tabulation time, we sum the intensities conveyed by rays emitted by punctual light sources placed in the environment and possibly by the robot, which are received at the position of the sensor. Only then is the value of the measure returned by the sensor computed, using the response curve of Figure 1.

2.2 The SGOCE evolutionary paradigm

This paradigm is used to encode, into a robot's genotype, the developmental rules that will generate its phenotype. In the present application, this phenotype is instantiated as a general recurrent neural network controlling the behavior of the robot that is grown from a few initial cells provided by the experimenter. This neural network is made up of individual neurons each behaving as a leaky integrator [35] — *i.e.*, it is a universal dynamics approximator, liable to approximate the trajectory of any smooth dynamic system [1].

2.2.1 The developmental code.

Our encoding scheme is a simple geometric variation of Gruau's cellular encoding [10]. It implements developmental rules that are encoded into artificial tree-like chromosomes that contain two categories of instructions. Some specify morphological transformations applying to specific cells, while others are used to generate structured developmental programs.

[Figure 2 comes about here]

This scheme also employs a two-dimensional substrate within which the experimenter initially arranges a set of sensory cells that may be connected to the robot's sensors, a set of motoneurons that may be connected to the robot's actuators, and a set of precursor cells from which the developmental process will be initiated (Figure 2). Each precursor cell is given a copy of the robot's genotype and, as it executes the corresponding program, divides, grows connections to other cells, differentiates into a functional neuron, or dies (Figures 3 and 4).

[Figure 3 comes about here]

[Figure 4 comes about here]

At the end of such a process, a complete neural controller is obtained, whose architecture reflects the geometry and symmetries initially imposed by the experimenter, to a degree that depends on the side-effects of the developmental instructions that have been carried out. This controller is connected to the sensors and actuators of the robot through connections to the sensory cells and motoneurons incorporated into its architecture. This, together with the use of an appropriate fitness function (to be described later) makes it possible to assess the controller's capacity to generate the specific behavior sought by the experimenter.

2.2.2 Syntactic constraints.

[Figure 5 comes about here]

In order to reduce the size of the genotypic search-space and the complexity of the networks generated, we restrict the structure of the corresponding developmental programs by requiring that all evolving subprograms be well-formed trees according to a given context-free tree-grammar (Figure 5). Furthermore, such a grammar makes it possible to control the nature and size of the program modifications that occur between two successive generations through use of genetic operators like mutation or crossover. Thus, a mutation operator makes it possible to replace a sub-tree by another randomly generated compatible[2] sub-tree. Likewise, a crossover operator makes it possible to exchange a sub-tree in

[2]Two sub-trees are said to be compatible if they are derived from the same grammatical variable. For instance, if grammar GRAM-A (Figure 5) is used to define the constraints on

one developmental program for a compatible sub-tree in another developmental program.

2.2.3 Evolutionary algorithm.

To slow down convergence by favoring the creation of ecological niches, we use a steady-state evolutionary algorithm that involves a population of N randomly-generated well-formed programs distributed over a circle and whose mode of operation is outlined in Figure 6.

[Figure 6 comes about here]

The following procedure is repeated until a given number of individuals have been generated and tested:

1. A position P is chosen on the circle.
2. A two-tournament selection scheme is applied in which the better of two programs randomly selected from the neighborhood of P is retained[3].
3. The program selected is allowed to reproduce, and three genetic operators may modify it. The recombination operator is applied with probability p_c. It exchanges two sub-trees between the program to be modified and another program randomly selected from the neighborhood of P. Two types of mutation are used. The first mutation operator is applied with probability p_m. It changes one randomly selected sub-tree into another randomly generated one. The second mutation operator is applied with probability 1 and modifies the values of a random number of parameters, implementing what Spencer called a *constant perturbation strategy* [37]. Firstly, the number n_{mut} of parameters to be modified is drawn from a binomial distribution $B(n, p)$, and n_{mut} parameters are then selected randomly — all parameters having the same probability of being chosen — to be mutated.
4. The fitness of the new program is assessed by collecting statistics while the behavior of the animat controlled by the corresponding artificial neural network is simulated over a given period of time.
5. A two-tournament anti-selection scheme, in which the worst of two randomly chosen programs is selected, is used to decide which individual (in the neighborhood of P) will be replaced by the modified program.

a given sub-program, sub-tree SIMULT3(SETBIAS, DEFTAU, SIMULT4(GROW, DRAW, GROW, NOLINK)) in that sub-program may be replaced by sub-tree DIE, because both sub-trees are derivations of the *Neuron* variable in GRAM-A.

[3]A program's probability p_s of being selected decreases with the distance d to P: $p_s = max(R - d, 0)/R^2$, with R=4. Programs for which d is greater than or equal to R cannot be selected ($p_s = 0$).

In all the experiments reported in this paper, $N = 100$, $p_c = 0.6$, $p_m = 0.2$, $n = 6$ and $p = 0.5$.

2.2.4 Incremental approach.

The artificial evolution of robust controllers for obstacle-avoidance was carried out using the Khepera simulator to solve successively two problems of increasing difficulty. Basically, this entailed evolving a first neural controller that used its sensors in active mode to measure the proximity value p, in order to avoid obstacles successfully in medium-light conditions. Then, a second neural controller was evolved that operated in passive mode in strong-light conditions and used measures of the ambient light level m to modulate the normal function of the first controller. In other words, such an incremental approach relied upon the hypothesis that the second controller would be able to evaluate the local intensity of ambient light so as to change nothing in the correct obstacle-avoidance behavior secured by the first controller in medium-lighted regions, but to alter it — in whatever adapted manner evolution would discover — when the robot travelled through strong-lighted regions likely to impair the proper operation of the first controller.

During Stage 1, to evolve the first controller and generate a classical obstacle-avoidance behavior in medium-light conditions, the following fitness function was used:

$$f_1 = \sum_t \left(0.5 + \frac{v_l(t) + v_r(t)}{4 \cdot V_{max}}\right) \cdot \left(1 - \frac{|v_l(t) - v_r(t)|}{2 \cdot V_{max}}\right) \cdot \left(1 - \frac{\sum_{front} p_i(t)}{4 \cdot P_{max}}\right) \quad (1)$$

where $v_l(t)$ and $v_r(t)$ were the velocities of the left and right wheels, respectively; V_{max} was the maximum absolute velocity; $p_i(t)$ was the proximity measure returned by each sensor i among the four front sensors; P_{max} was the largest measured value that can be returned.

In the righthand part of this equation, the first factor rewarded fast controllers, the second factor encouraged straight locomotion, and the third factor punished the robot each time it sensed an obstacle in front of it.

Using the substrate of Figure 2 and the grammar of Figure 5, controllers likely to include eight different sensory cells and four motoneurons were evolved after a random initialization of the population. The fitness of these controllers was assessed by letting them control the simulated robot over 500 time-steps, in a square environment containing an obstacle (Figure 7-a).

For this purpose, the sensory cells $D0$ to $D7$ in Figure 2 were connected to the robot's sensors such that the instantaneous activation value each cell propagated throught the neural network to the motoneurons was set to the proximity measure p returned by the corresponding sensor. Likewise, the motoneurons were connected to the robot's actuators such that a pair of motoneurons was associated with each wheel, the difference between their inputs determining the speed and direction of rotation of the corresponding wheel.

[Figure 7 comes about here]

Each controller was evaluated five times in the environment of Figure 7-a, starting in the same position, but with five different orientations, its final fitness being the mean of these five evaluations.

After 10,000 reproduction events, the controller with the highest fitness (called AVOID1 hereafter) has been used to seed an initial population that was subjected to a second evolutionary stage involving strong-light conditions. During Stage 2, the corresponding fitness function became:

$$f_2 = \sum_t \left(0.5 + \frac{v_l(t) + v_r(t)}{4 \cdot V_{max}}\right) \cdot \left(1 - \frac{|v_l(t) - v_r(t)|}{2 \cdot V_{max}}\right) \tag{2}$$

In this equation, the third term that was included in the righthand part of equation 1 was eliminated because it referred to active-mode sensory inputs that could not be trusted in strong-light conditions. However, fast motion and straight locomotion were still encouraged.

[Figure 8 comes about here]

Using the substrate of Figure 8 and the grammar of Figure 9, controllers likely to include 16 different sensors and four motoneurons were evolved during 10,000 additional reproduction events.

This time, the activation values that the new sensory cells $L0$ to $L7$ in Figure 8 propagated through a given controller were each set to the ambient light measure m returned by the robot's corresponding sensor. The fitness of the corresponding controller was assessed in the same square environment as the one used in Stage 1, but with a lighted lamp positioned in one of its corners (Figure 7-b).

Again, the final fitness was the mean of five evaluations that corresponded to five trials of 500 time-steps, each starting in the same position, but with different orientations and light intensities. In particular, one such trial was performed in medium-light conditions when the lamp was switched off, two others were performed when the lamp was contributing a small amount of additional light, and the last two were performed in strong light conditions, when the lamp contributed its maximum light intensity.

At the end of Stage 2, the best neural network thus obtained was downloaded and tested on a Khepera for 50 seconds.

[Figure 9 comes about here]

3 Experimental results

The evolutionary run just described has been replicated ten times, the fitnesses of the best controllers obtained at the end of Stage 1 in medium-light conditions, on the one side, and at the end of Stage 2 in strong-light conditions, on the other side, being respectively given in columns 1 and 4 of Table 1.

[Table 1 comes about here]

Column 2 of Table 1 provides fitnesses that have been obtained when, at the end of Stage 1, each controller selected in medium-light conditions was transferred and tested in strong-light conditions. As for column 3 of Table 1, it provides fitnesses that were obtained when the best controllers selected at the end of Stage 2 were tested in medium-light conditions again.

The comparison of strong-light fitnesses indicates that controllers selected at the end of Stage 1 are less efficient than those that are obtained at the end of Stage 2 when the lighted lamp is added to the environment. Thus, this second evolutionary stage helped improving the behavior of the robot in strong-light conditions. Likewise, comparison of medium-light fitnesses indicates that the controllers selected at the end of Stage 2, according to their capacities at coping with strong-light conditions, didn't loss the essential of their abilities to generate appropriate behavior in medium-light conditions. Indeed, although their fitnesses tend to be slightly lower than those of the best controllers of Stage 1, they still are of the same order of magnitude.

To assess the usefulness of the incremental approach advocated here, ten additional control runs have been performed, each involving 10,000 reproduction events and the same number of evaluations (100,000) that were done in the incremental runs. Each such run directly started with the substrate of Figure 8 and called upon both GRAM-A and GRAM-B grammars, thus permitting the simultaneous evolution of both neural modules. The fitness of each individual was the mean of 10 evaluations: five in the conditions of Stage 1 described above, and five in the conditions of Stage 2. Columns 5 and 6 of Table 1 provide the fitnesses of the best individuals thus selected, these fitnesses having been assessed in both medium-light and strong-light conditions. A quantitative comparison of incremental and control runs indicates that the means of the medium-light and strong-light fitnesses obtained at the end of the incremental runs are statistically higher (Mann-Whitney test, significancy level = 0.05) than the corresponding means obtained at the end of the control runs. Moreover, a qualitative comparison of the behaviors generated by these controllers indicate that the behaviors of the control runs are far less satisfactory than those of their incremental competitors. In fact, in every control run, but Run 4, the robot alternated moving forward and backward and never turned. As for the controller of Run 4, its fitness in medium-light conditions suddenly increased in the last generations and led to an obstacle-avoidance behavior as good as those of the best controllers evolved in the incremental runs, but at the detriment of its abilities to deal with strong-light conditions, which were severely impaired.

To understand how the controllers obtained during the incremental runs succeeded to generate satisfactory behaviors, the internal organization of the neural networks obtained at the end of Stage 1 (Figure 10) and Stage 2 (Figure 11) in one of these runs has been scrutinized. The corresponding simulated behaviors are respectively shown in Figures 12A-D. It thus appears that the single-module controller uses the four front sensors only. It drives the robot at maximal speed in open areas, and makes it possible to avoid obstacles in two different ways. If the obstacle is detected on one given side, then the opposite wheel slows down, allowing for direct avoidance. When the detection is as strong on both sides, then the robot slows down, reverses its direction, and turn slightly while recoiling. After a short period of time, forward locomotion resumes. However, when placed in strong-light conditions, this controller is unable to detect obstacles and thus keeps bumping into walls. The two-module controller corrects this default by avoiding strongly lighted regions in the following way. In medium-light conditions, all L-sensors return high m values. Excitatory links connect each of the two frontal L-sensors — $L0$ and $L1$ — to one interneuron of module 1 apiece. Each of these interneurons, in turn, sends a forward motion command to the motor on the opposite side. As a consequence, whenever the value m returned by one of these two sensors decreases — an event that corresponds to the detection of a high light intensity — the corresponding interneuron in Module 1 becomes less activated and the wheel on the opposite side slows down. This results in a light avoidance behavior.

[Figure 10 comes about here]

[Figure 11 comes about here]

[Figure 12 comes about here]

[Figure 13 comes about here]

Figures 13A-D show the behavior exhibited by Khepera when the networks described above are downloaded onto the robot and are allowed to control it for 50 seconds in a square arena of size 60x60 cm designed to scale the simulated environment. Such figures were obtained through the on-line record of the successive orders sent to the robot's motors and through the off-line reconstruction of the corresponding trajectories. They demonstrate that the behavior actually exhibited by Khepera is qualitatively similar to the behavior obtained through simulation — in terms of the robot's ability to avoid obstacles and to quickly move along straight trajectories — and that it fits the experimenter's specifications. The main behavioral difference occurs when, at the end of the medium-light stage, controllers are tested in the presence of the additional lamp: the actual behavior of Khepera is more disrupted than the simulated behavior, probably because light that is reflected by the ground in the experimental arena is not adequately taken into account by the simulator (Figures 12.B and 13.B). Such discrepancies do not occur at the end of the strong-light stage because the robot then avoids the region where the lamp is situated and where such disturbing light reflections are the strongest.

4 Discussion

Although obstacle-avoidance would appear a behavior easy to evolve in Khepera, as demonstrated by the successful results already obtained by numerous researchers [6, 9, 8, 17, 24, 28, 29, 30, 34], results presented herein indicate that such a behavior is easily disrupted when the ambient light is high. These results also indicate that evolving a robust obstacle-avoidance behavior, although not trivial, is nevertheless possible. The solution that has been automatically discovered here consists in avoiding situations were the sensory capacities of the robot become too limited to secure a still adapted behavior. Finally, these results do not contradict the intuition that such non trivial behaviors are easier to evolve using a divide-and-conquer incremental approach. This intuition is further supported by the observation that nature itself seems to resort to such an incremental approach, if one admits that, in an ever changing environment, selection pressures never remain constant, and if one observes that, since the appearance of life on Earth, the adaptive capacities of man clearly originate in the simpler adaptive capacities of numerous intermediate species.

Be that as it may, the results obtained herein are preliminary, and numerous additional experiments should be performed to assess the usefulness of a variety of implementation details. It seems *a priori* possible, for example, that individual neurons, behaving as traditional threshold units [25, 33], (McClelland and Rumelhart, 1986; Rumelhart and McClelland, 1986), might be used to produce similar results to those obtained here with leaky integrators, although the recoiling behavior reported in Section 3 might have been harder to produce with non dynamic neurons. We nevertheless used such neurons because their dynamic properties might prove to be mandatory in future extensions of this work.

Likewise, it is presently unclear whether the local interactions and the genetic operators that were used in our evolutionary algorithm are truly relevant and whether they might have been replaced by other options. Finally, one may wonder how integral each detail of the initial setup chosen by the experimenter — e.g., the grammars, the substrate's layout, the fitness functions — was for evolutionary success. However, it is interesting to note that the evolutionary parameters used were the same here and in all other applications of the SGOCE paradigm [19, 20, 14, 7]. The different developmental substrates and grammars used in all applications were also chosen to be as similar as possible and satisfactory results were always found without tuning.

To further assess the potentialities of incremental evolution, future research efforts will aim at carrying on the evolutionary process one step further, resorting to a third evolutionary stage and a third fitness function. This might entail incorporating into Khepera's control architecture a rudimentary motivational system, according to which the robot — while still being able to avoid encountered obstacles — would seek the light when a simulated internal energy sensor detected low energy conditions, and would avoid light in normal or high energy conditions. Comparisons with a similar, but simpler, experiment carried on by [8] are likely to be enlightening because, in the latter approach, evolution was directly performed on the physical robot, i.e., without human intervention, and with a direct encoding scheme.

5 Conclusion

Preliminary results presented herein support the hypothesis that complex behaviors in real robots are more likely to be generated through an incremental evolutionary process than through direct evolution. They also suggest that realistic simulators may be devised, which permit neural controllers evolved in simulation to be successfully downloaded onto the corresponding robot, at least for simple robots. A two-stage incremental strategy made it possible to evolve a robust obstacle-avoidance behavior in a Khepera robot, although additional experiments are required to assess the relevance of each detail of the corresponding implementation. Future research will aim at elaborating the behavior thus far obtained through additional evolutionary stages that will manage a rudimentary motivational system.

References

[1] R. D. Beer, On the dynamics of small continuous-time recurrent neural networks, *Adaptive Behavior* **3(4)** (1995) 469–510.

[2] E. Boers and H. Kuiper, *Biological Metaphors and the Design of Modular Artificial Neural Networks*, Master's thesis, Dept. of Computer Science and Experimental and Theoritical Psychology, Leiden University, (August 1992)

[3] A. Cangelosi, D. Parisi and S. Nolfi, Cell division and migration in a 'genotype' for neural networks, *Network* **5** (1994) 497–515.

[4] H. de Garis, *Genetic Programming: GenNets, Artificial Nervous Systems, Artificial Embryos*, Ph.D. thesis, Université Libre de Bruxelles, Belgium, (1991)

[5] F. Dellaert and R. Beer, Toward an evolvable model of development for autonomous agent synthesis, in *Proceedings of the Fourth International Workshop on Artificial Life*, R. A. Brooks and P. Maes, eds., The MIT Press/Bradford Books, Cambridge, MA, (1994)

[6] P. Eggenberger, Cell-cell interactions as a control tool of developmental processes for evolutionary robotics, in *From Animals to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack and S. W. Wilson, eds.,The MIT Press/Bradford Books, Cambridge, MA, (1996)

[7] D. Filliat, *Evolution de réseaux de neurones pour le contrôle d'un robot hexapode*, Technical Report, AnimatLab, University Paris 6, (Septembre 1998)

[8] D. Floreano and F. Mondada, Evolution of homing behavior in a real mobile robot, *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics* **26** (1996) 396–407

[9] D. Floreano and F. Mondada, Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot, in *From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson, eds., The MIT Press/Bradford Books, Cambridge, MA, (1994)

[10] F. Gruau, Automatic definition of modular neural networks, *Adaptive Behavior* **3(2)** (1994) 151–184

[11] F. Gruau and K. Quatramaran, Cellular encoding for interactive evolutionary robotics, Tech. rep., University of Sussex, School of Cognitive Sciences, EASY Group, Brighton, UK, (1996)

[12] I. Harvey, P. Husbands and D. Cliff, Seeing the light: Artificial evolution, real vision, in *From Animals to Animats 3. Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, D. Cliff, P. Husbands, J.-A. Meyer and S. W. Wilson, eds., 392–401. The MIT Press/Bradford Books, Cambridge, MA, (1994)

[13] P. Husbands, I. Harvey, D. Cliff and G. Miller, The use of genetic algorithms for the development of sensorimotor control systems, in *From Perception to Action. Proceedings of the PerAc'94 Conference*, P. Gaussier and J. Nicoud, eds., 110–121. IEEE Computer Society Press, Los Alamitos, CA, (1994)

[14] A.J. Ijspeert and J. Kodjabachian, *Evolution and Development of a Central Pattern Generator*, Technical Report, Dep. of AI, University of Edimburgh, (September 1998)

[15] N. Jakobi, Evolutionary Robotics and the radical envelope of noise hypothesis, *Adaptive Behavior* **6(1)** (1997) 131–174

[16] N. Jakobi, Half-baked, ad-hoc and noisy: minimal simulation for Evolutionary Robotics, in *Fourth European Conference on Artificial Life*, Husbands and Harvey, eds., The MIT Press / Bradford Books, (1997)

[17] N. Jakobi, P. Husbands and I. Harvey, Noise and the reality gap: The use of simulation in Evolutionary Robotics, in *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, Moran, Moreno, Merelo and Chacon, eds., Springer Verlag, (1995)

[18] J. Kodjabachian and J.-A. Meyer, Evolution and development of control architectures in animats, *Robotics and Autonomous Systems* **16(2–4)** (December 1995) 161–182

[19] J. Kodjabachian and J.-A. Meyer Evolution and development of modular control architectures for 1D locomotion in six-legged animats, *Connection Science* **10(3–4)** (1998) 211–237

[20] J. Kodjabachian and J.-A. Meyer, Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects, *IEEE Transaction on Neural Networks* **9(5)** (September 1998)

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

[22] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Subprograms*, The MIT Press, (1994)

[23] W. Lee, J. Hallam and H. Lund, Learning complex robot behaviours by evolutionary approaches, in *Proceedings of the 6th european workshop on learning robots*, Brighton, (1997)

[24] H. Lund and O. Miglino, From simulated to real robots, in *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, IEEE Computer Society Press, (1996)

[25] J. L. McClelland and D. E. Rumelhart, eds., *Parallel Distributed Processing*, vol. 1, The MIT Press/Bradford Books, Cambridge, MA, (1986)

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

[27] O. Michel, Une approche inspirée de la vie artificielle pour la synthèse d'agents autonomes, in EA'95, Brest, France, (1995)

[28] O. Michel, An artificial life approach for the synthesis of autonomous agents, in *Artificial Evolution*, Alliot, E. Lutton, Ronald, M. Schoenauer and D. Snyers, eds., Springer Verlag, (1996)

[29] O. Michel and P. Collard, Artificial neurogenesis: An application to autonomous robotics, in *Proceedings of the 8th. International Conference on Tools in Artificial Intelligence*, Radle, ed., IEEE Computer Society Press, (1996)

[30] O. Miglino, H. Lund and S. Nolfi, Evolving mobile robots in simulated and real environments, *Artificial Life* **2** (1995) 417–434

[31] F. Mondada, E. Franzi and P. Ienne, Mobile robot miniaturization: A tool for investigation in control algorithms, in *Proceedings of the Third International Symposium on Experimental Robotics*, T. Yoshikawa, F. Miyazaki, eds., 501–513, Springer Verlag, Tokyo, (1993)

[32] S. Nolfi, O. Miglino and D. Parisi, Phenotypic plasticity in evolving neural networks, in *From Perception to Action. Proceedings of the PerAc'94 Conference*, P. Gaussier and J. Nicoud, eds., IEEE Computer Society Press, Los Alamitos, CA, (1994)

[33] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing*, vol. 2., The MIT Press/Bradford Books, Cambridge, MA, (1986)

[34] R. Salomon, Increasing adaptativity through Evolutionary Strategies. in *From Animals to Animats 4. Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack and S. W. Wilson, eds., The MIT Press/Bradford Books, Cambridge, MA, (1996)

[35] I. Segev, Simple neuron models: Oversimple, complex and reduced, *Trends in Neurosciences* **15(11)** (1992) 414–421

[36] K. Sims, Evolving 3D morphology and behavior by competition, in *Proceedings of the Fourth International Workshop on Artificial Life*, R. A. Brooks and P. Maes, eds., The MIT Press/Bradford Books, Cambridge, MA, (1994)

[37] G. Spencer, Automatic generation of programs for crawling and walking, in *Advances in Genetic Programming*, K. E. Kinnear Jr., ed., 335–353. The MIT Press / Bradford Books, Cambridge, MA, (1994)

[38] J. Vaario, *An Emergent Modeling Method for Artificial Neurol Networks*, Ph.D. thesis, University of Tokyo, (August 1993)

[39] J. Vaario, A. Onitsuka and K. Shimohara, Formation of neural structures, in *Proceedings of the Fourth European Conference on Artificial Life, ECAL97*, 214–223, The MIT Press, (1997)

Tables

Table 1/1:

	Stage 1		Stage 2		Control	
run	medium-light	strong-light	medium-light	strong-light	medium-light	strong-light
1	323.253	097.104	345.746	359.786	245.934	258.740
2	326.283	102.576	296.758	309.380	240.169	252.095
3	339.701	159.331	171.105	224.239	241.174	252.125
4	302.774	105.761	280.408	314.011	397.838	183.006
5	321.416	101.187	301.514	345.597	254.706	226.549
6	396.320	107.621	258.273	239.041	228.097	230.399
7	204.426	230.999	193.847	235.449	245.495	246.534
8	312.598	185.041	296.922	262.145	203.891	232.949
9	310.051	219.790	255.324	304.069	170.998	224.549
10	398.471	181.863	346.630	409.280	221.416	231.129

Figures

Figure 1/13:

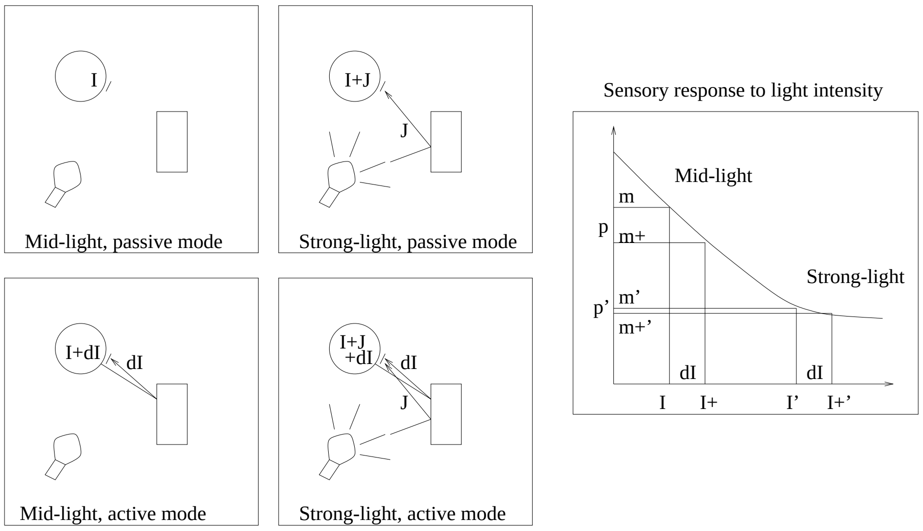


Figure 2/13:

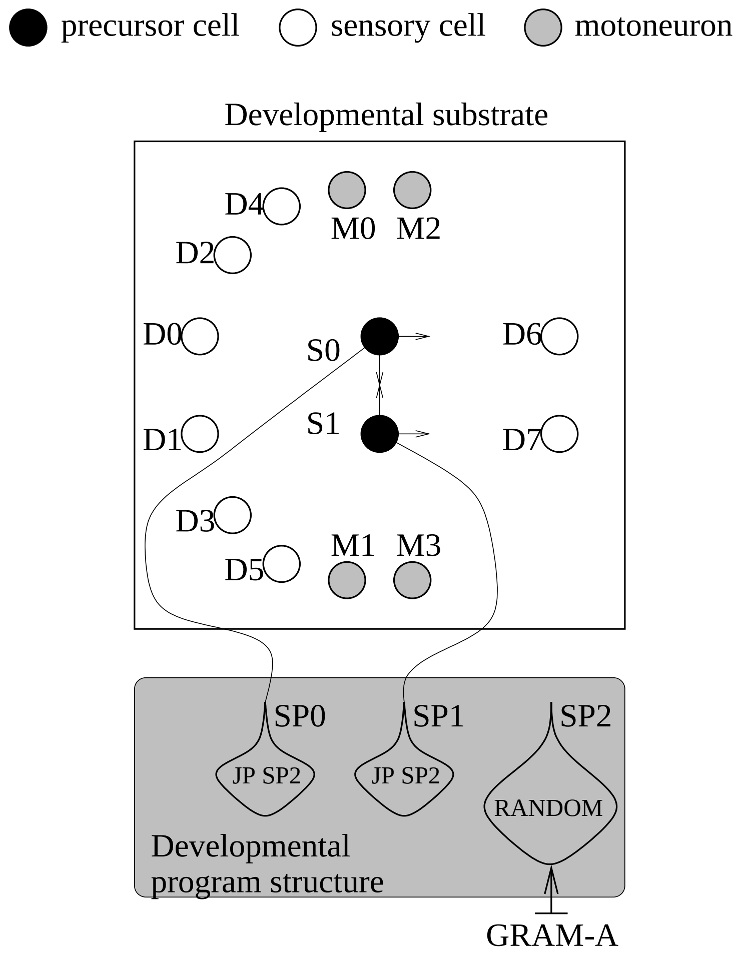

precursor cell
sensory cell
motoneuron
Developmental substrate
D4
M0
M2
D2
D0
S0
D6
S1
D1
D7
D3
M1
M3
D5
SP0
SP1
SP2
JP SP2
JP SP2
RANDOM
Developmental program structure
GRAM-A

Figure 3/13:

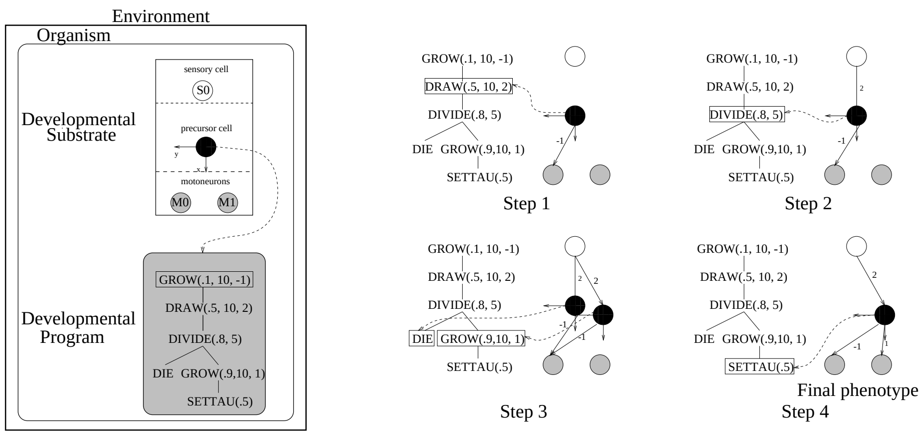

Environment
Organism
Developmental Substrate
sensory cell
S0
precursor cell
y
x
motoneurons
M0
M1
Developmental Program
GROW(.1, 10, -1)
DRAW(.5, 10, 2)
DIVIDE(.8, 5)
DIE
GROW(.9,10, 1)
SETTAU(.5)
Step 1
Step 2
Step 3
Step 4
Final phenotype
2
-1
1

Figure 4/13:

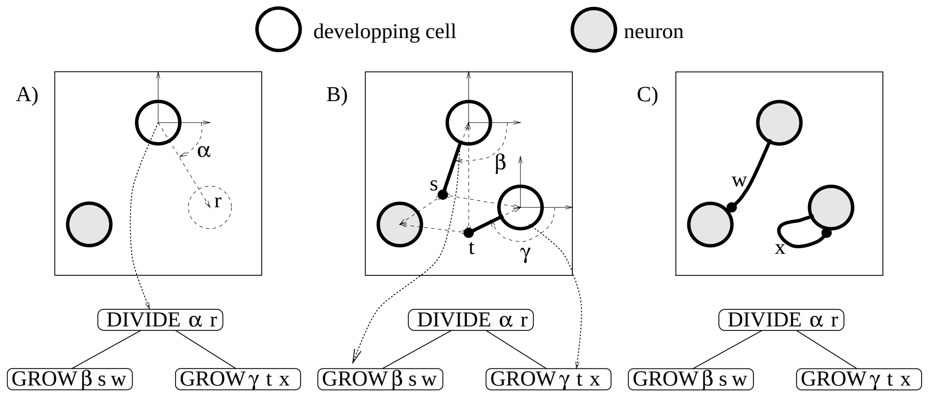

developping cell
neuron
A)
α
r
DIVIDE α r
GROW β s w
GROW γ t x
B)
β
s
t
γ
DIVIDE α r
GROW β s w
GROW γ t x
C)
w
x
DIVIDE α r
GROW β s w
GROW γ t x

Figure 5/13:

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

Figure 6/13:

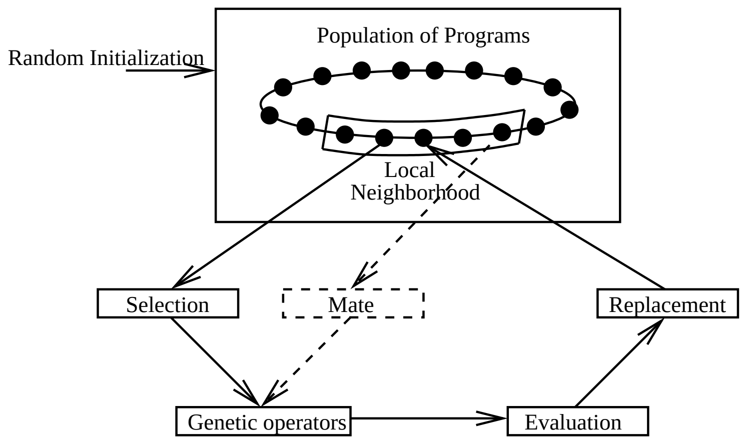
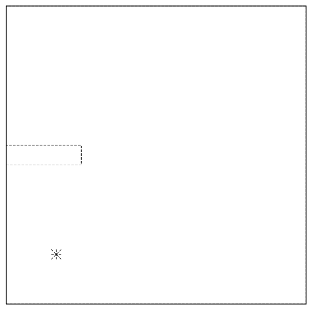

Population of Programs
Random Initialization
Local
Neighborhood
Selection
Mate
Replacement
Genetic operators
Evaluation

Figure 7/13:


(a) Medium-light environment.

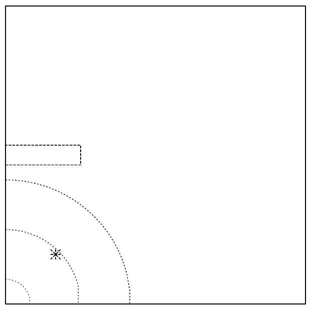

(b) Strong-light environment.

Figure 8/13:

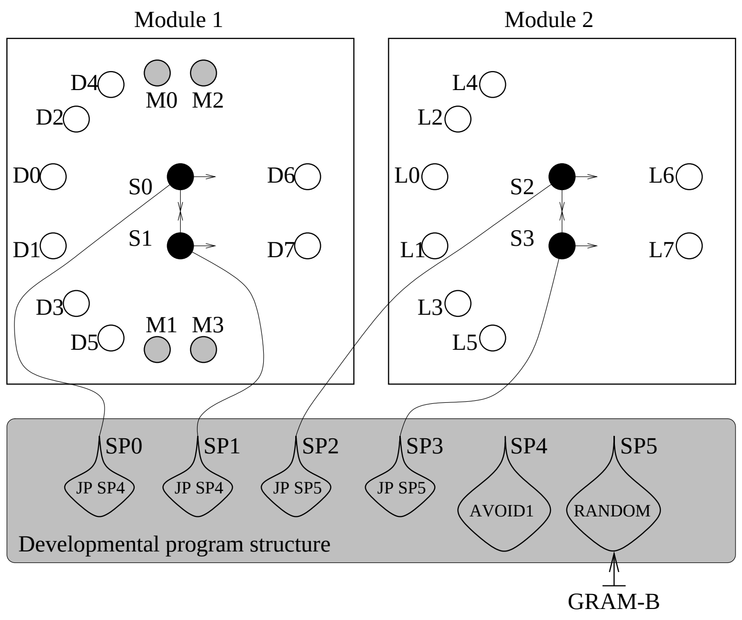

Module 1
Module 2
D4
M0
M2
D2
D0
S0
D6
D1
S1
D7
D3
D5
M1
M3
L4
L2
L0
S2
L6
L1
S3
L7
L3
L5
SP0
SP1
SP2
SP3
SP4
SP5
JP SP4
JP SP4
JP SP5
JP SP5
AVOID1
RANDOM
Developmental program structure
GRAM-B

Figure 9/13:

Terminal symbols
DIVIDE, GROW, DRAW, GROW2, SETBIAS, SETTAU, DIE, NOLINK, DEFBIAS, DEFTAU, SIMULT3, SIMULT4.
Variables
Start1, Level1, Neuron, Bias, Tau, Connex, Link.
Production rules
Start1$\longrightarrow$DIVIDE(Level1, Level1)
Level1$\longrightarrow$DIVIDE(Neuron, Neuron)
Neuron$\longrightarrow$SIMULT3(Bias, Tau, Connex) | DIE
Bias$\longrightarrow$SETBIAS | DEFBIAS
Tau$\longrightarrow$SETTAU | DEFTAU
Connex$\longrightarrow$SIMULT4(Link, Link, Link, Link)
Link$\longrightarrow$GROW | DRAW | GROW2 | NOLINK
Starting symbol
Start1.

Figure 10/13:

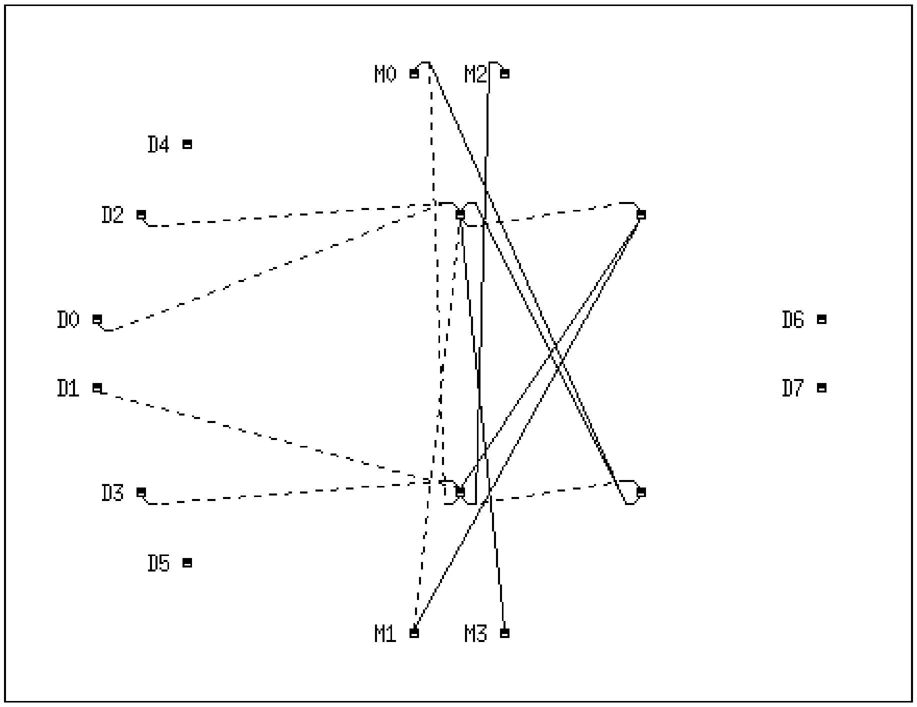


Figure 11/13:

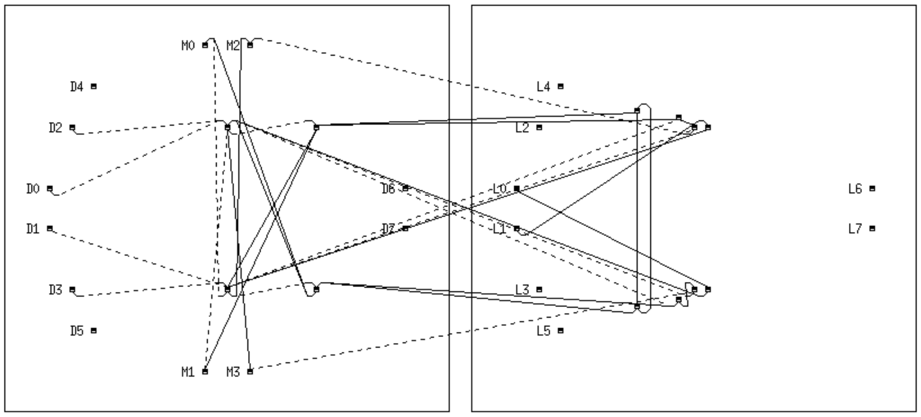
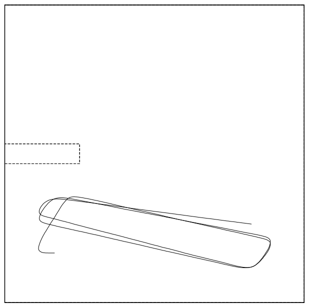
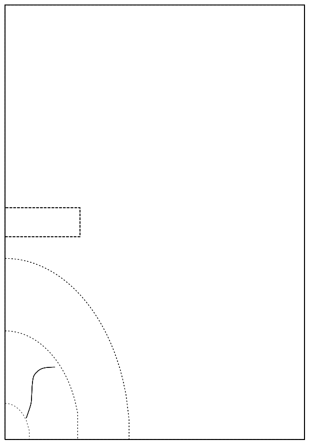
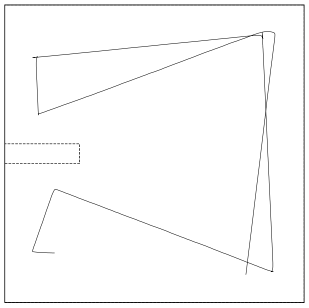


Figure 12/13:


(A) Best of Stage 1, medium-light (sim.)


(B) Best of Stage 1, strong-light (sim.)


(C) Best of Stage 2, medium-light (sim.)

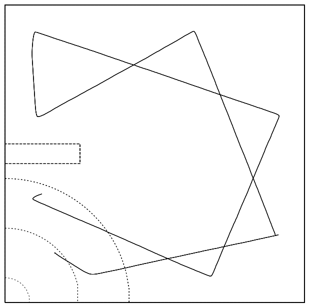
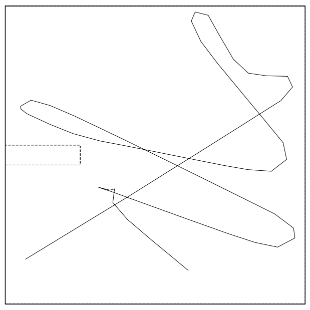
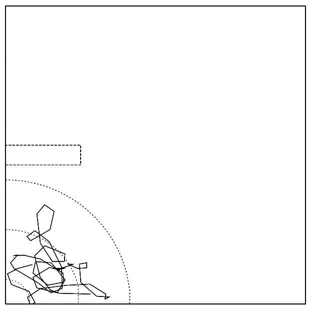

(D) Best of Stage 2, strong-light (sim.)

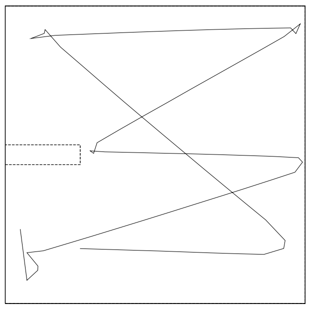
Figure 13/13:



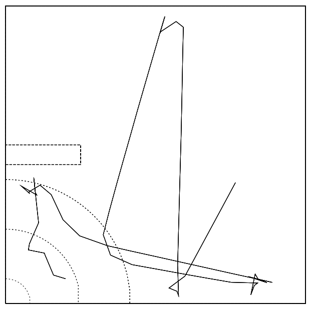
(A) Best of Stage 1, medium-light (real)



(B) Best of Stage 1, strong-light (real)



(C) Best of Stage 2, medium-light (real)



(D) Best of Stage 2, strong-light (real)

Table and Figure captions

Table 1/1:

The performance of the best individuals of each run, when evaluated either in medium-light or in strong-light conditions. The incremental approach results in the performance values shown in columns 3 and 4 (Stage 2), while the control experiments results are shown in columns 5 and 6 (Control). The first two columns (Stage 1) provide results obtained after Stage 1 during the incremental approach.

Figure 1/13:

(Left) This part of the figure shows the different sources that contribute to the intensity received by a sensor. The geometrical configuration considered is the same in all four situations. The ambient light contributes an intensity I. When the lamp is lighted, the sensor receives an additional contribution J, through direct and/or reflected rays. Finally, in active mode, the reflected IR-ray yields an intensity dI at the level of the sensor. The value of dI depends only on the geometrical configuration considered. Intensities are summed, but the sensor response is non-linear. (Right) In strong-light conditions, the response of a sensor can saturate. In such a case, where $I' = I + J$, the same increase in intensity dI caused by the reflection of an IR-ray emitted by the robot causes a smaller decrease of the value measured than it does in the linear region of the response curve (medium-light conditions). In other words, although dI intensity increases can be sensed in medium-light conditions — and the corresponding obstacle configurations can thus be detected —, such is not the case in strong-light conditions.

Figure 2/13:

An example of the initial state of the developmental substrate and of the structure of the developmental program. This structure is determined by the experimenter. Here, each of the two precursor cells carries out a developmental program that starts by a jump instruction to the evolving sub-program SP2. To solve other control problems, additional precursor cells might have been included in the substrate, each of which would execute a different sub-program before, occasionally, jumping to SP2 or to other evolving sub-programs. During evolution the composition of sub-program SP2 can be modified within the limits defined by the constraints encoded in grammar GRAM-A (see explanations on syntactic constraints in the text). $D0$ to $D7$ are sensory cells and $M0$ to $M3$ are motoneurons, which have been placed by the experimenter in specific positions within the substrate.

Figure 3/13:

While reading the genotype, a precursor cell generates a neural network after several developmental steps. This network may involve the sensory cells and the motoneurons made available by the experimenter. Each precursor cell is associated with a frame of reference that is inherited by its daughter cell when division occurs.

Figure 4/13:

Depending on the values of the arguments of some developmental instructions, targets for connections are sought in a given direction and at a given distance, in the local framework associated with the acting cell. These connections link two different neurons or correspond to self-connections. They can also regress and die when their targets lie outside the developmental substrate.

Figure 5/13:

The GRAM-A grammar. This grammar defines a set of sub-programs — those that can be generated from it, starting with the $Start1$ symbol. When GRAM-A is used, a cell that executes such a sub-program undergoes two division cycles, yielding four daughter cells, which can either die or modify internal parameters (time-constant and bias) that will influence their future behavior as neurons. Finally, each surviving cell establishes a limited number of connections, either with another cell, or with the sensory cells and motoneurons that have been positioned by the experimenter in the developmental substrate.

Figure 6/13:

The evolutionary algorithm (See text for explanation).

Figure 7/13:

The environment that was used to evaluate the fitness of each controller. (a) Medium-light conditions of Stage 1. The star indicates the starting position of each run. (b) Strong-light conditions of Stage 2. A lighted lamp is positioned in the lower-left corner of the environment. Concentric arcs illustrate the corresponding intensity gradient.

Figure 8/13:

The initial configuration of the developmental substrate and the program structure used for Stage 2. The same substrate is used for control experiments, in which both modules are evolved simultaneously. In these latter experiments, both SP4 and SP5 are initialized randomly and are submitted to evolution under constraints given by GRAM-A and GRAM-B, respectively.

Figure 9/13:

The GRAM-B grammar. It is identical to GRAM-A except for the addition of one instruction (GROW2) that makes it possible to create a connection from the second to the first module.

Figure 10/13:

The best controller obtained after Stage 1 for the particular run that resulted in the controller of Figure 11. The outputs of the motoneurons $M2$ and $M3$ are interpreted as forward motion commands for the right and left wheels, respectively, while the output of the motoneurons $M0$ and $M1$ correspond to backward motion commands. Solid lines correspond to excitatory connections, while dotted lines indicate inhibitory links. This network contains four interneurons.

Figure 11/13:

The best controller obtained after Stage 2. This networks contains four interneurons in Module 1 and eight interneurons in Module 2.

Figure 12/13:

(Top) The simulated behavior of the single-module controller of Figure 10. When starting in front of the lamp, the corresponding robot gets stuck in the corner with the lamp. (Bottom) Simulated behavior of the two-module controller of Figure 11. Now, the robot avoids the lighted region of the environment.

Figure 13/13:

The paths actually travelled by the Khepera robot, which have been reconstructed off-line using motor orders recorded during the trial. (Top) Real behavior of the single-module controller of Figure 10. Due to reflections present in the real world, but not in the simulation, the behavior under strong-light conditions is different from that of Figure 12. (Bottom) Real behavior of the two-module controller of Figure 11. Now, the real behavior is qualitatively similar to the simulated behavior shown in Figure 12.