# An Evolutionary Controller for Autonomous Multi-Robot Systems

**E. D. V. Simões* & K. R. Dimond****
**Electronic Engineering Laboratories**
**University of Kent at Canterbury**
**Canterbury, Kent, CT2 7NT, United Kingdom**
***edvs1@ukc.ac.uk, ****krd@ukc.ac.uk**

## ABSTRACT

This article describes the implementation of a robot with evolutionary control scheme, and experiments in to examine evolutionary control with a population of five physical autonomous mobile robots. The control for the robots is totally embedded and performs collision-free behaviour. Both morphological features and controller circuit are evolved. The robots constantly adapt to changes of surroundings by modifying their features and the weights of the RAM neural controller. The paper describes the Evolutionary System and the results of experiments. These show that in relatively complex navigation tasks the proposed system evolves very quickly to develop a robust control system.

## 1. INTRODUCTION

The normal approach to studding evolution is to evolve controllers in simulation and then transfer them to the robots [1]. Only a small number of studies have been carried out on real time experiments fully on-board physical robots. The main issues to be addressed in these experiments are: I) to automatically synthesise more complex behaviours than could be produced by hand; II) to exploit all of the available features, considering that some of them may even be opaque to the designer; III) to produce the desired behaviour specifying what the robot should do, but not how the controller works; IV) to show that evolutionary techniques can reduce the human effort required to develop control systems as compared to traditional manual methods.
The main issues addressed in this work lie under II and III. Because of simplicity of the task, issues I and IV do not have significant impact. (i.e. the behaviour produced by our controller is relatively simple, and could have been designed by hand with considerable lesser amount of effort).

The goal of the work is to describe how a continuous process of evolution can take place so that the controller circuit and morphology will respond to changes in the environment. We define the term morphology as the physical, embodied characteristics of the robot, such as its mechanics and sensor organisation [2]. In the experiments described in this paper, the morphological features modified by evolution are the number and position of sensors, as well as the drive motor speed.

When developing genetic algorithms using simulation, to achieve realistic results the simulation must take account of the imperfection of the real world, noise, non-ideal performance of sensors etc. The results of the algorithm optimisation are then loaded into the robot. A better approach is to use the real robot themselves. The problem with this approach is the prohibitively long time taken, as demonstrated by experiments on Khepera robots [4]. We have developed a technique that enables collision free behaviour of the robots to be developed in 17 minutes or 40 generations of 25-second duration.

In our approach the population of robots communicates and exchanges genetic information so as to adapt to solve the targeted problem. The robots “breed” and then “die” so that their bodies can be used for the next generation. The genetic material is stored in RAM within the robots. This material specifies the configuration of the control device and morphological features. The control system will be implemented within the robot microprocessor (a neural network for navigation control) and a set of programmable modules controlling the robot features like speed and physical positioning of photosensors on the robot body.

## 2. ROBOT ARCHITECTURE

The robot architecture consists of a two-wheel differential-drive platform (20cm of diameter), containing a Motorola 68HC11 - 2MHz, 128Kb of RAM, an exchangeable 4-hour battery, bumpers with 8 collision sensors, and 8 peripheral active infra-red proximity sensors (see figure 1). It exchanges information with the other robots at 1.2Kbps by a 418MHz AM radio. Different kinds of actuators can be also attached to the base to allow different tasks in the experiments. The robots form an independent embedded evolutionary system where no host computer is required. Nevertheless, an IBM PC is used to monitor, without interfering, all data exchanged via the radio link, producing a complete record for every generation of the chromosomes, parameters and variables.

## 3. THE NEURAL NETWORK

The process of designing a controller for a physical robot that is able to be evolved by the system is delicate and depends on insights gained through a long trial-end-error process [6]. Unfortunately, most of the inspiration that pointed out a RAM neural network and its configuration cannot be discussed in this paper. It provided a robust architecture, with good stability to mutation and crossover [7]. A binary network is a good solution because it is fast and small enough to cope with the on-board microprocessor speed and memory restrictions. It allows the utilisation of command outputs that can be interpreted as high level routines or employed as low-level incremental commands in every iteration [8].

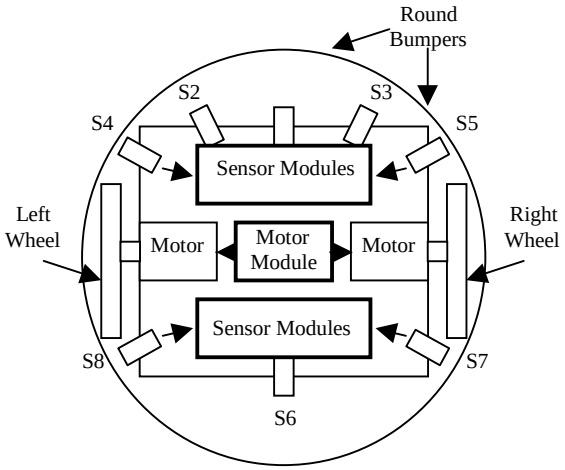


Figure 1 – Robot architecture representing the morphological feature control modules for the sensors and motors. A round bumper with collision sensors surrounds the robot.

Figure 2 shows the RAM neural network chosen to implement the robot programmable control circuit [9]. It consists of a n-tuple classifier that provides commands for the motor control module, containing 64 x 4-input neurons (8 x 8 groups) connected to the 8 sensor control modules (2 bits per sensor). It is a robust controller, suitable to be evolved with the approach selected in the experiment, since the mutation of a fit individual should, on the average, produce an individual of approximately the same fitness; similarly, crossover between two parents of similar fitness should, on the average, produce offspring with similar fitness.

Figure 3 shows how each sensor reading is converted into a 2-bit signal by the sensor control module and connected to the neuron inputs. The network output consists of 8 commands to each motor driver: S - "stop"; FS - "front slow"; FM - "front medium"; FF - "front fast"; TRS - "turn right sharp"; TRL - "turn right long"; TLS - "turn left sharp"; and TLL - "turn left long". The turn sharp command means that the robot will turn with one wheel going forward and the other backwards. The turn long command makes the robot turn by stopping one wheel. A "winner takes all" algorithm selects the proper command according to the sensor inputs for each iteration. The resultant speed value will be determined by the motor control module, that gradually increments or decrements the motor speed in each iteration, until reaching the selected level (i.e. stop; slow; medium; or fast).

For the robot feature selection, controlled by the sensor control modules and the motor control module, a more complex "*dominance approach"* was chosen [10]. Each sensor control module is configured by two genes (2 bits) in the chromosome: I) two genes will determine the presence of a feature (sensor); II) one gene comes from each parent; III) all Features are recessive. The two genes are coded using bits in such a way that the combinations "X,1" and "1,X" disable the sensor, and "0,0" enables it (see figure 3). The motor control module has 10 bits associated to it in the chromosome. Figure 4 shows that the sum of the 10 bits plus 1 defines the fast speed; the sum of 6 bits plus 1 defines the medium speed; and the sum of 3 bits plus 1 defines the slow speed. This approach permits co-adaptation where the chromosome integrates specifications for both controller and morphological features [11], [12]. Evolution can select not only the number of sensors to use, but, if the number of sensors is fixed, select which ones to pick (i.e. the sensor position on the robot).

## 4. THE PROPOSED EVOLUTIONARY SYSTEM

A cyclic process was defined, where the robots perform specific tasks in a "*working season*" for a short period of time (normally 1 to 5 minutes), until an internal timer starts a "*breeding season"*. Then, the individuals are selected to mate according to their fitness and generate new configurations to reprogram all robots, allowing them to be evaluated again, in a new "*working season*".

Each robot is evaluated during the "*working season*", where its fitness function is calculated in order to penalise collisions and lack of movement [13]. For each collision, 8 points are deducted from its fitness value, and for every second it is moving forward, 1 point is added. From gained experience through trial-and-error experimenting, a simple fitness function usually produces the best results, because it does not eliminate the autonomy of evolution. As more complex behaviours are evolved, the designer has a tendency of gradually adding all subgoals to the fitness function, strongly biasing the possible solutions [2].

In the "*breeding season*", each robot transmits its "*mating call*" to the others, where it identifies itself and "shouts" its fitness value. The fittest robot is granted the right to survive to the next generation, while the remaining 4 will mate and reconfigure themselves with their offspring chromosome. To select their partner, the fittest robot (the surviving one) has 80% of chance to be chosen, and a random selection of any other robot will fil in the other 20%. In the crossover phase, a random exchange of the 1050 genes from both parents occurs (64 neurons, 16 bits each, plus 16 bits to select the sensor control modules and 10 to select the motor control module). A small mutation factor is also present, randomly inverting up to 3% of the exchanged bits.

The strategy of allowing the fittest robot to survive ensures that the fitness of the next generation will probably not fall, and permits a high mutation rate. The average of the fitness of all robots, though, is largely affected by a high mutation rate and will make impracticable a system where the robots work in group and a couple of bad adapted robots will cause considerable trouble [14]. A high mutation rate is only efficient to quickly evolve a solution to a problem where the robots will work independently.

## 5. THE EXPERIMENT

An experiment was performed to illustrate the potential of the presented evolutionary system in automatically generating controllers for the five physical mobile robots and keeping their fitness high during many modifications in the environment complexity [15]. The experiment applies evolution directly in a non-trained randomly initialised population to see if it can be evolved to a coherent state by crossover and mutation.
The experiments where performed within a 4m x 4m working domain, containing walls and obstacles of varied sizes, where the robots can perform obstacle avoidance (exploring the environment without colliding with obstacles). Many movable obstacles and internal walls of different sizes were available to constantly change the scope of the workspace where the robots navigate (see figure 5).

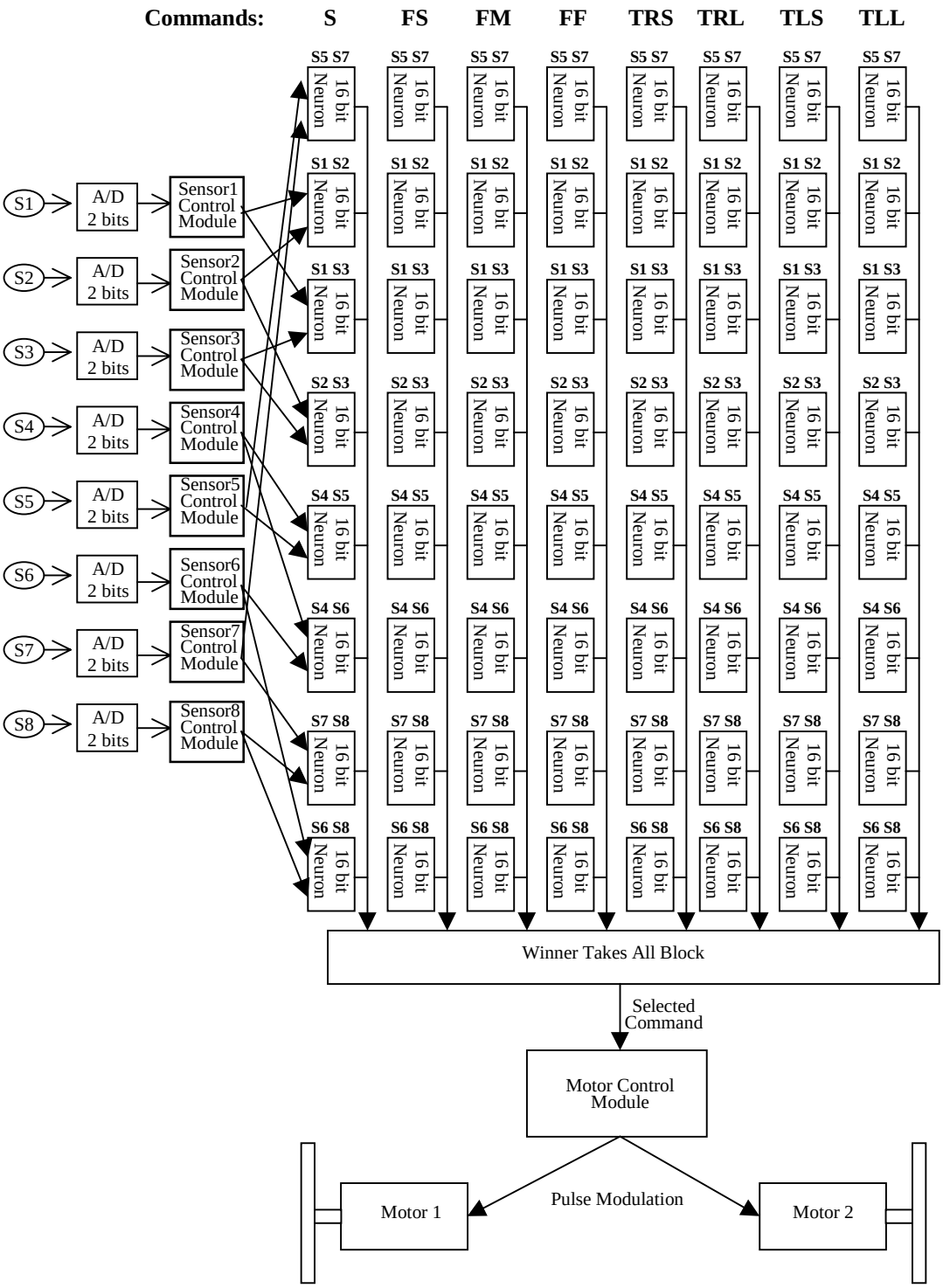


Figure 2 – RAM Neural Network architecture controlling a 2-wheeled Robot: S1 to S8 are infra-red sensors; “S”, “FS”, “FM”, “FF”, “TRS”, “TRL”, “TLS”, and “TLL” represent the groups of neurons of the classes “Stop”, “Front Slow”, “Front Medium”, “Front Fast”, “Turn Right Short”, “Turn Right Long”, “Turn Left Short”, and “Turn Left Long”. Each sensor is connected to an A/D decoder and to the Sensor Control Module. The Sensor Control Modules decide which sensor is connected to the Neural Network. The 4-input Neurons are arranged in groups of 8 for each Command and are connected to two sensors each (under the control of each Module). The winner command is chosen and sent to the Motor Control Module which recognises it and sends the Pulse Modulated Signal to control each Motor.

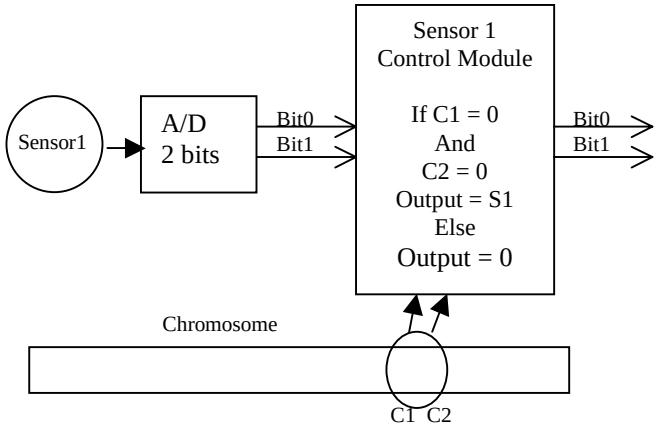


Figure 3 – The Sensor Control Module. Each sensor analogue signal is converted into 2 bits by the A/D and directed to the Control Module. Each Module is configured according to 2 genes from the chromosome. If the two control bits are "0", then the Module Output is connected to the A/D 2-bit signal.

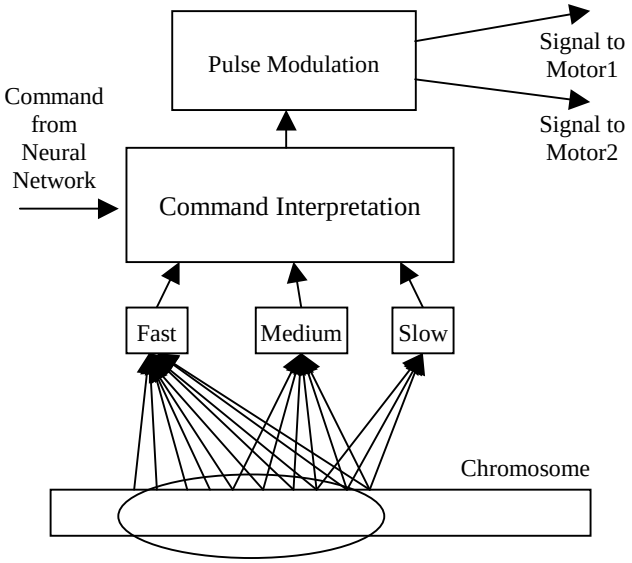


Figure 4 – The Motor Control Module receives a command from the neural net and activates motors 1 and 2 according to the speed levels calculated from the sum of the corresponding genes.

After the neural network weights and the feature selecting genes were assigned random values, the robots are barely able to navigate, but their genetic code starts to be modified by the evolutionary technique in real time. Then, the robot programmable controller and morphology can face a very noisy environment and evolve to perform the chosen behaviour.

Figure 6A shows the result after 120 generations (2 hours run with a working season of 1 minute) in a very simple workspace, constituted of only the external walls and a few large obstacles (see figure 5A). The curve shows the fitness of the fittest robot and the average of the fitness of all five robots. The distance between the two curves depends on the mutation rate. The higher the mutation, the more distant are the curves. The experiment was performed with a mutation rate of 3%. Figure 6B shows the same experiment performed in a much more complex working domain (see figure 5C), where all sorts of obstacles of different sizes and many internal walls were included. It can be seen in the two charts that the fitness curve of the simple environment grows quickly and tends to stay high with considerable oscillation. The complex environment curve seems to climb in three distinct steps. We can analyse these as fallen.

When evolving physical robots it can be notoriously difficult to judge if an expected behaviour has been accomplished. Much of the subsequent analysis is qualitative and based on human judgement and observation.

While the group of robots evolved, it was possible to observe some particular distinctions among the adapting individuals. After an initial period of fuzzy behaviours, where some agents evolved to come to a halt, while others circled around themselves, and still others could not stop shaking. But after about 20 generations, a group of robots that learned to use the front sensor to avoid obstacles and walls in its way were distinctly successful. We call them "species 1". Soon after about 45 generations, another successful solution showed up. It learned to use 2 sensors, a frontal and a lateral one. When it approached an obstacle or wall from its front or working sensor side, it usually avoided it without colliding. The frontal sensor was so efficient that it was very difficult to collide in the simple workspace. It would only collide if approaching an obstacle from its non-working sensor side at a very low angle. These robots were called "species 2". After about 75 generations, a third kind of robot that began to use 3 sensor appeared. It was called "species 3", and used the frontal and two lateral sensors, colliding only with very small obstacles. The three species coexisted peacefully in the later generations of the simple environment, as shown in figure 6A. That coexistence of the three species explains the high oscillation of the fitness, because even the species 2 or 3 can be unlucky enough to collide while a not colliding species 1 can be the winner of the generation, and generate less fitted offspring. Competition was much harder in the complex environment, where species 1 was led to extinction by species 2. And species 3 gradually overcame the other 3, as shown in figure 6B, reducing the oscillation in the curve.

The duration of the working season is also very important in species competition. Figure 6C and 6D show its influence in a simple and complex environment where the same individuals are evaluated for 40 minutes without being changed by evolution. The fitness curves of the three described species are compared to the authors' solution, a manually trained neural network that uses the same 3 sensors of species 3. As the possibility of collision increases with a longer evaluation time, we can observe the curves of the three species distancing. It is more significant in the complex domain, where there are more obstacles and walls to collide with.

The aim of figure 6E is to provide existence proofs that the presented system is robust enough to cope with drastic modifications of the environment and constantly adapt the group of robots to perform well again the desired behaviour. The experiment was performed in 4 periods of 2 hours (480 generations), where in the end of the first 3 periods, the environment was altered to become progressively more complex, as shown in figure 5. It is possible to see the fitness falling every time the workspace is modified, but a constant evolutionary process is able to provide the desired behaviour after some generations. It could be observed the presence of individuals using up to 5 sensors, in the most complex environment.

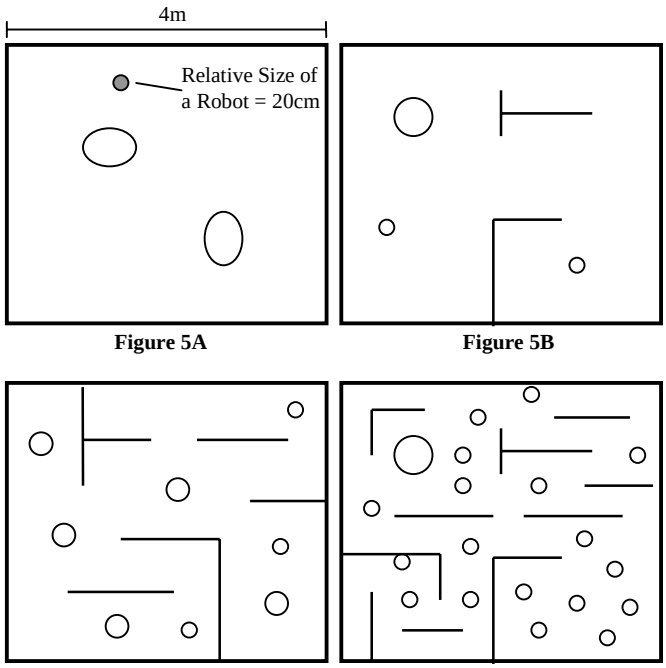


**Figure 5A** **Figure 5B**

**Figure 5C** **Figure 5D**

Figure 5 – Four different configurations of the 4m x 4m workspace showing how the environment was modified in the experiment.

When the last trial was repeated with the environment being gradually and constantly altered with the inclusion or removal of more walls and obstacles, one at a time, evolution could normally keep fitness high, without presenting the drastic falls of figure 6E.

## 6. CONCLUSION

The fear that the robot hardware could not survive the necessary continuous evolutionary process without constant maintenance and repairs proved wrong. But the unavoidable need to exchange the batteries every 4h or so created the need to stop the robots during the experiments. Though, the careful placement of the chromosomes and important variable and parameters in a battery backed-up memory allowed the robots to be turned off for a long period without loosing current information. A pair of exchangeable batteries permitted a constant run, once one can be recharged while the other is being used on the robot.

The strategy of running an initial training phase with examples of pre-defined situations and, then, refining the neuron weights in real time manoeuvring, with gene exchange and mutation, proved to be many times faster than allowing evolution to play with a randomly initialised neural network. Though, it can reduce diversity and cause, in some situations, the population to becoming trapped into a local minimum, until mutation makes it move through the genotype space and climb a fitness slope, reaching a reasonable adapted behaviour. This strategy made the system more sensible to the selection of crossover parameters and mutation rate, but much faster to evolve.

By the end of most trials (e.g. when a threshold level is achieved), we found a few unfit robots and a majority of effective individuals, qualified to work in the environment. The results attested that the selected evolutionary technique succeeded in training the robots to perform the desired tasks, providing the appropriate building blocks for evolution to work with. It allowed the evolutionary process to see the whole robot (body, sensors, motors and "nervous system") as a dynamic system coupled with a dynamic environment. The suggested technique was applied to simple control tasks using low-resolution sensors, but it is the first step towards more complex controllers for physically embedded systems like real-time multi-agent applications in noisy, dynamic environments. Finally, in addition to employing evolution for developing effective solutions and applying them to a specific problem, we hope that so far we promoted the potential of a continuous evolutionary process, where the robots are not just being evaluated, they are performing real tasks.

## 7. REFERENCES:

[1] J-A. Meyer, P. Husbands and I. Harvey. *Evolutionary Robotics: a Survey of Applications and Problems*. Proc. of First European Workshop on Evolutionary Robotics. EVOROBOT'98, April, pp. 16-17, Paris, 1998.

[2] Maja J Mataric and Dave Cliff, *Challenges In Evolving Controllers for Physical Robots*, in "Evolutional Robotics", special issue of Robotics and Autonomous Systems, 19(1), Oct 1996, 67-83.

[3] M. Sipper, D. Mange & A. Péres-Uribe. *On the Automatic Design of Robust Electronics Through Artificial Evolution*. In: Proc. 2nd Int. Conference on Evolvable Systems: From biology to hardware (ICES98), pp.13-24, Springer-Verlag,1998.

[4] Floreano, D. and Nolfi, S. *Adaptive Behavior in Competitive Co-Evolutionary Robotics*. In P. Husbands and I. Harvey, Proceedings of the 4th European Conference on Artificial Life, Cambridge, MA: MIT Press, 1997.

[5] I. Harvey: *Artificial Evolution for Real Problems* 5th Intl. Symposium on Evolutionary Robotics, Tokyo April 1997. Invited paper. In: Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER'97), T. Gomi (ed.). AAI Books, 1997.

[6] D. Keymeulen, M. Iwata, K. Konaka, Y. Kuniyoshi & T. Higuchi. *Evolvable Hardware: A Robot Navigation System Testbed*. New Generation Computing, Vol.16, No.2, pp.97 - 122, 1998.

[7] Floreano, D., and Mondada, F. *Evolutionary Neurocontrollers for Autonomous Mobile Robots*. Neural Networks, 11, 1461-1478, 1998.

[8] N.E. Nawa, M. Korkin, F. Gers & H. de Garis. *ATR's CAM-Brain Project: The Evolution of Large-Scale Recurrent Neural Network Modules*. International Conference On Pararrel Distributed Processing Techniques and Applications (PDPTA'98), Vol.II, pp.1087-1094, 1998.

[9] R. Rohwer & M. Morciniec. *A theoretical and experimental account of n-tuple classifier performance*. Neural Computation, 8, pp.629--642, 1996.

[10] D. Keymeulen, M. Iwata, Y. Kuniyoshi & T. Higuchi. *Comparison between an Off-line Model-free and an On-line Model-based Evolution applied to a Robotics Navigation System using Evolvable Hardware*. Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life, pp. 109-209. MIT Press, 1998.

[11] Floreano, D., Nolfi, S., and Mondada, F. *Co-Evolution and Ontogenetic Change in Competing Robots*. Robotics and Autonomous Systems, To appear, 1999.

[12] Nolfi, S. and Floreano, D. *How co-evolution can enhance the adaptation power of artificial evolution: Implications for evolutionary robotics*. To be published in P. Husbands and J-A.

Meyer, Proceedings of the First European Workshop on Evolutionary Robotics, Berlin: Springer-Verlag, 1998.
[13] Marc Ebner. *Evolution of a control architecture for a mobile robot*. In Moshe Sipper, Daniel Mange and Andrés Pérez-Uribe (editors), Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES 98), Lausanne, Switzerland, © Springer-Verlag, pp. 303-310, 1998.
[14] Francois Michaud and Maja J Mataric, *Representation of behavioral history for learning in nonstationary conditions*, Robotics and Autonomous Systems, 29(2), Nov 30, 1999 (forthcoming)
[15] T. Bäck. *On The Behavior Of Evolutionary Algorithms In Dynamic Environments*. In D. B. Fogel, H.-P. Schwefel, Th. Bäck, and X. Yao, editors, Proc. Second IEEE World Congress on Computational Intelligence (WCCI'98) with Fifth IEEE Conference. Evolutionary Computation (IEEE/ICEC'98), Anchorage AK, May 4-9, IEEE Press, Piscataway NJ., volume 1, pp. 446-451, 1998.

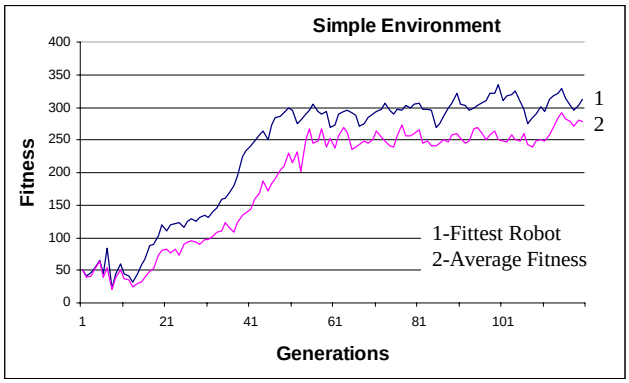


Figure 6A – Fitness curves for environment from figure 5A.

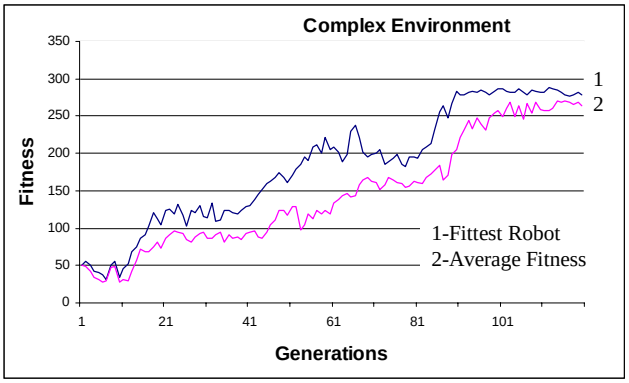


Figure 6B – Fitness curves for environment from figure 5C.


Figure 6C – Fitness curves for environment from figure 5A.

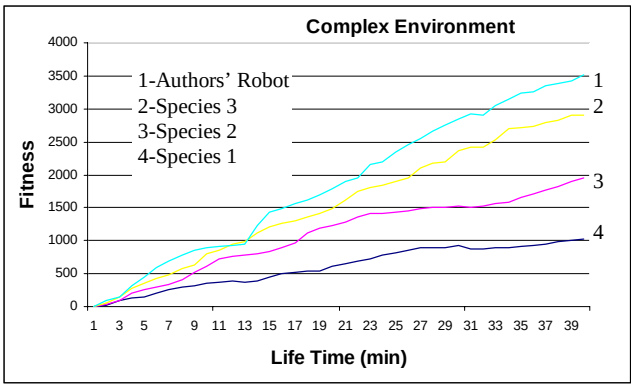


Figure 6D – Fitness curves for environment from figure 5C.

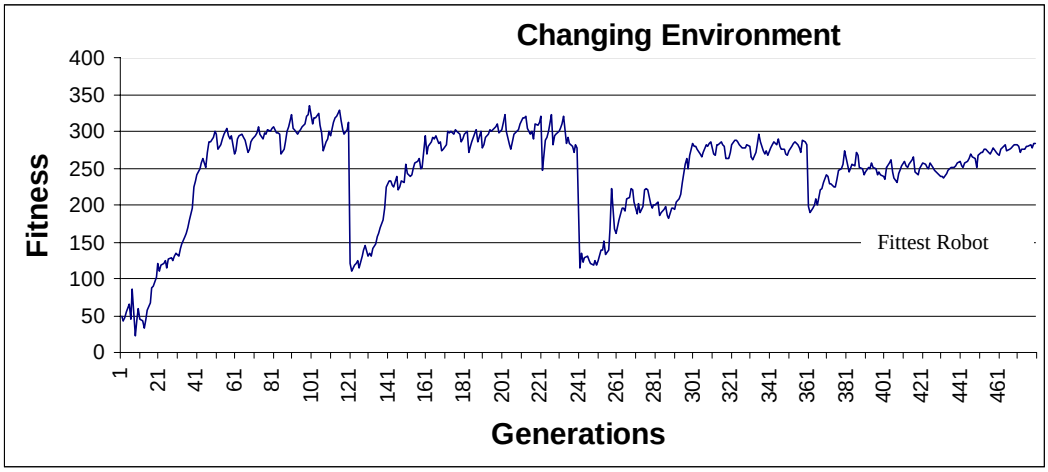


Figure 6E – Fitness curves of all environments from figure 5. The fitness falls in every modification.