# Evolving a behavior-based control architecture - From simulations to the real world

**Marc Ebner and Andreas Zell**
Eberhard-Karls-Universität Tübingen, Wilhelm-Schickard-Institut für Informatik
Arbeitsbereich Rechnerarchitektur, Köstlinstraße 6, 72074 Tübingen, Germany
{ebner,zell}@informatik.uni-tuebingen.de, Tel.: +49 7071 / 29 76455

## Abstract

Genetic programming makes it possible to automatically search the space of possible programs. First we evolved a behavior-based control architecture using computer simulations. Then we replicated one of the experiments with a service robot, showing that Koza's classic experiment of evolving a control architecture can be transfered to the real world with a change of representation. The use of a service robot necessitates safety measures which are also explained. Results are reported for the experiments using computer simulations and with the real robot.

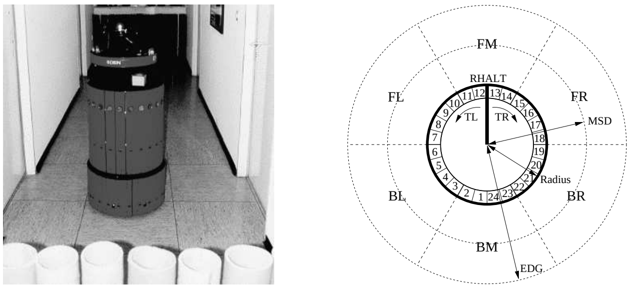


Figure 1: The real robot, a RWI B21, in its environment (left). The robot's radius is approximately 26.7cm. A drawing of the robot seen from above (right). The front of the robot is marked with a vertical line.

## 1 MOTIVATION

Many mobile robots are carefully programmed by hand. Apart from repetitive tasks which may simply be recorded, programming robots to complete tasks in arbitrary environments is a difficult process which usually takes a long time. The difficulty of the task is due in part to unforeseeable interactions between the robot and the environment. In addition, a handwritten program may behave very different in reality from the way it was intended to perform. Using Darwinian evolution this difficult process may be automated (Braitenberg 1984). We are investigating the question if genetic programming (Koza 1992; Koza 1994; Banzhaf et al. 1998) can be used to evolve a behavior-based control architecture for a service robot, a Real World Interface B21 (Fig. 1), using sonar sensors to navigate inside a corridor.

Much research has already been done in evolutionary robotics. Issues in evolutionary robotics are discussed by Harvey et al. (1993). An overview about the field is given by Matarić and Cliff (1996) and by Meyer et al. (1998). The evolution may be performed in simulation, in the real world or in a mixture between the two (Nolfi et al. 1994). The importance of using real robots as opposed to simulated ones is emphasized by Brooks (1992). Provided that reality is simulated accurately enough, such that results are transferable, computer simulations may be used (Miglino et al. 1996). Simulations may be used to speed up the search for a suitable representation as well as parameters for the evolution. Most work in evolutionary robotics focused on the use of neural nets as a control architecture, e.g. (Floreano and Mondada 1994; Mondada and Floreano 1995; Nolfi 1997). With a predefined neural architecture one still has to search for the one best suited for the task. Harvey et al. (1997; 1993) evolved the architecture of a neural net using a variable length genotype.

## 2 BACKGROUND

In contrast to these approaches we are exploring the use of genetic programming for evolutionary robotics. Genetic programming offers the advantage that arbitrary elementary functions may be used which operate on any level of abstraction from low-level machine language to high level action primitives. Some of our results were reported in (Ebner 1998).

Koza (1994) evolved a program using computer simulations to control a robot to mop a $8 \times 8$ grid which also contained some obstacles. In a more realistic setting but also with discrete movements, Koza (1992) evolved a behavior-based control architecture (Brooks 1986) realizing a wall-following behavior for a mobile robot using computer simulations. Ross et al. (1996) considered using the task of evolving a wall-following robot as a benchmark problem for the application of genetic programming to emergent robotic behavior and analyzed the search space. Reynolds (1994d) evolved an obstacle avoidance behavior for a simulated robot. In other experiments, Reynolds used noise to promote the evolution of robust controllers (Reynolds 1994b; Reynolds 1994c). He also investigated the influence the representation had on the difficulty of the problem (Reynolds 1994a). Nordin and Banzhaf (1995, 1997a, 1997b) were the first to use genetic programming to evolve a control architecture for a real miniature mobile robot (Khepera) using infrared sensors. Olmer et al. (1996) evolved real-time behavioral modules for a miniature mobile robot using the same representation. An overview is given by Banzhaf et al. (1997). Wilson et al. (1997) evolved hierarchical behaviors for a Lego robot. Lee et al. (1997) evolved behavior primitives and behavior arbitrators for a Khepera robot using genetic programming.

Our work differs from the above in that we are evolving a hierarchical behavior-based control architecture for a service robot. Koza and Reynolds only used computer simulations. Banzhaf et al. worked with a miniature mobile robot and used a different type of genetic programming. They evolved linear genotypes of machine code instructions. Wilson et al. did not use a conditional statement in their set of primitive functions. In addition, to our knowledge, it is the first time that a control architecture is evolved which uses sonar sensors to control a service robot. A service robot is very different from a miniature mobile robot in many respects. Experiments with a miniature mobile robot may be continued even if the robot bumps into a wall. The robot either comes to a halt or slides along the wall. A service robot is able to exert a large force and may produce a considerable damage if it crashes into an obstacle. Therefore safety measures have to be taken. We now start by describing the representation and the safety measures which we used to evolve a control architecture for this mobile robot.

# 3 REPRESENTATION

Koza (1992) used a population size of 1000 and was able to evolve an individual with 145 nodes capable of solving the task in generation 57. We had to limit the size of the population to 75 and the number of generations to 50 because we wanted to perform one of the experiments also with the real mobile robot. Increasing either number would have prolonged the experiment. With this setting the experiment with the real robot took two months to perform. After preliminary experiments with Koza's original representation yielded no results we tried to reduce the search space. This is most easily achieved by reducing the number of sonar sensors. In addition, we continuously controlled the heading of the robot which moved at constant speed as opposed to discrete movements which were used in Koza's representation. The following representation was used for all experiments which are described below.

## 3.1 TERMINALS

The 24 sonar sensors of the robot are combined into 6 virtual sensors as shown in Fig. 1: `FL` (front left), `FM` (front middle), `FR` (front right), `BL` (back left), `BM` (back left), `BR` (back right). Each of the 6 terminals returns the minimum value measured by the corresponding 4 physical sensors. Because of reflections sometimes a larger value may be measured by a physical sensor. Thus taking the minimum increases robustness. In addition to the 6 virtual sensors we also use the terminal `SS` which returns the minimum value of all 24 physical sensors. The two constant terminals `EDG` (desired edging distance) which returns 50cm and `MSD` (minimum safe distance) which returns 70cm are also included in the set of terminals.

Instead of using discrete robot movements our robot moves at a constant speed of 0.1 meters per second. The rotational speed may be set to $-40$, 0 or $+40$ degrees per second. The following terminals control the rotational velocity of the robot `TL` (turn left), `RHALT` (halt rotation) and `TR` (turn right). `TL` returns the average value of the physical sensors 11 and 12, `RHALT` returns the average value of the physical sensors 12 and 13 and `TR` returns the average value of the physical sensors 13 and 14.

## 3.2 PRIMITIVE FUNCTIONS

As primitive functions we used the four argument conditional `IFLTE` and the two argument function `PROGN2`. `IFLTE` evaluates the first two arguments. If the return value of the first argument is less than or equal to the second the third argument is evaluated otherwise the fourth argument is evaluated. The primitive function `PROGN2` evaluates its two arguments in sequence and returns the value of the second argument.

### 3.3 FITNESS CALCULATION

To evaluate the fitness of the individuals each control algorithm was tested for a fixed number of fitness cases. Raw fitness was calculated as the mean over all fitness cases. Raw fitness was to be minimized. For each fitness case the individual was evaluated for a maximum of 5 minutes. The evaluation was aborted if at any time during the 5 minutes one of the physical sonar sensors reported a range closer than 40cm or one of the robot's tactile sensors was activated. The tactile sensors were activated if a part of the robot's enclosure was pressed against the robot. Due to reflections of the sonar beams of the real robot it could occur that the robot bumped into a wall while none of the sonar sensors reported a value that would have aborted the evaluation. After each evaluation we calculated the fitness for this fitness case. After the fitness was calculated, the robot was automatically turned in a direction facing away from any obstacles and then a 15cm move was made in this direction. This was done to give all individuals a fair chance of survival.

For the fitness calculations we used a mixture between survival time and amount of rotatory movements done during that time. The robot tried to maximize survival time and minimize the number of rotations. This was done to prevent evolution from finding solutions where the robot only runs around in circles and thereby avoids bumping into any walls. In principle, pareto optimization should be used to optimize a function according to different criteria (Fonseca and Fleming 1995). However, the problem of choosing a suitable architecture among the evolved solutions remains. This would have to be done by visual inspection of the behavior. For our experiments we have used fitness functions which are similar to fitness functions that have previously been used in evolutionary robotics. Below we only specify how the fitness for one fitness case was calculated. To simplify the notation we are using the following variables. Let $t$ be the time until the robot bumped into a wall and $T$ the total time available to the robot during one fitness case. Then $D = \frac{t}{T}$ specifies the normalized survival time of the robot. Let $r_s$ be the sum of all signed rotations the robot made during the fitness case and let $\omega$ be the rotational velocity of the robot. Then $R_s = \frac{|r_s|}{\omega t}$ measures the amount of unbalanced turning of the robot during one fitness case.

## 4 EXPERIMENTS

A number of experiments were performed using computer simulations. Two processes were used, one modelled the robot in its environment and the other executed the control algorithm. Movement commands were sent from the control algorithm to the robot simulator. Sensor variables of the simulator were read by the control algorithm. Because two separate processes were used, small delays might be introduced into the system due to the scheduling mechanism of the operating system. Results achieved in simulation do not necessarily carry over into the real world. Therefore, one of the experiments was repeated with the real robot. For all experiments the following main parameters were used. A population size of 75 individuals was evaluated using tournament selection with a size of 7 for 50 generations with crossover, reproduction and mutation probabilities set to 85%, 10% and 5% respectively.

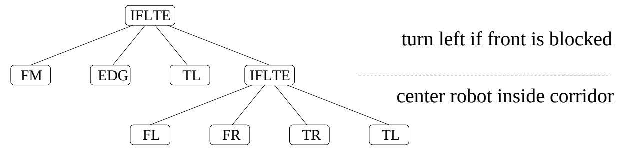


Figure 2: Control architecture for centering behavior. The conditional which centers the robot inside a corridor was found in one of the runs using genetic programming. The top layer was added manually.

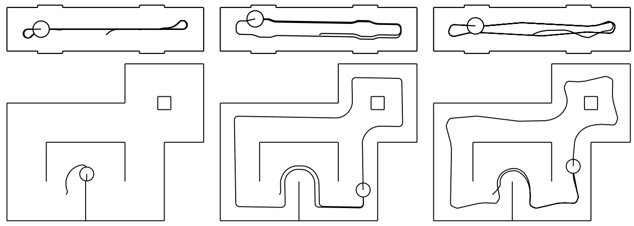

Figure 3: Paths of three different individuals for two environments. The paths on the left, middle, and right were produced by the individuals shown in Fig. 2, 4, and 5 respectively. The robot is shown in the position at the end of the fitness test.

### 4.1 BEHAVIOR OF MANUALLY CONSTRUCTED INDIVIDUALS IN SIMULATION

We constructed several different individuals manually to see how they perform. We discuss three individuals, two manually constructed and one which evolved in an experiment described below which we extended manually. This individual uses the front left sensor and the front right sensor to keep the robot centered inside a corridor (Fig. 2). The strategy does not work for thin obstacles if they are approached from the thin side (Fig. 3). It is an individual with a trivial structure showing a seemingly complex behavior (see Braitenberg (1984) for a number of different individu-

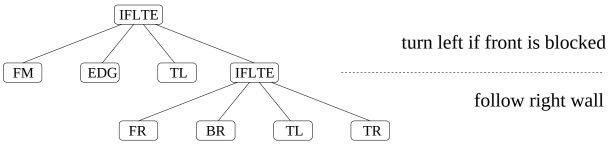


Figure 4: Control architecture for wall following behavior (manually constructed).

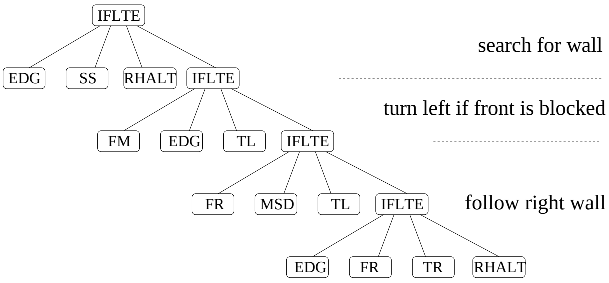


Figure 5: Control architecture for wall following behavior (manually constructed).

als showing complex behaviors which can be realized with a simple neural control architecture).

The second individual performs wall following behavior (Fig. 4). Provided that the front is not blocked, the individual uses the front right and back right sensors to follow the right wall. This individual works fine in the large environment. Unfortunately, it does not manage to survive in the small environment (Fig. 3). The robot bumps into the door which can be seen in the upper left. The third individual also performs wall following behavior (Fig. 5). First, it searches for a wall. It turns left if the front is blocked. Otherwise it performs wall following in the third layer. It performs successfully in both environments (Fig. 3).

## 4.2 EXPERIMENTS USING COMPUTER SIMULATIONS

Experiments were performed in different environments. Three fitness cases were used and total fitness was computed as the average over the fitness trials. In one run where we used the raw fitness function $1 - D(1 - R_s)^3$ and the large environment, which was already introduced above, a very successful individual emerged. We call an individual successful if it survived 5 minutes of a fitness test. It consists of 131 nodes. The path followed by this individual and the adjusted fitness values can be seen in Fig. 6. Notice that the individual keeps on turning to the right and then to the left and back again. This strategy allows the individual to avoid the thin obstacles which might otherwise be almost invisible to the individual.

For the remaining experiments we used the fitness function $1 - D(1 - \sqrt{R_s})$. A similar fitness function was

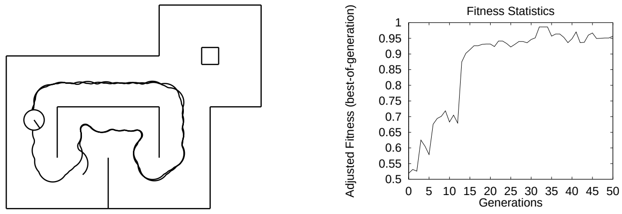


Figure 6: The individual which is shown on the left evolved with the fitness function $1 - D(1 - R_s)^3$. The adjusted fitness values are shown on the right.

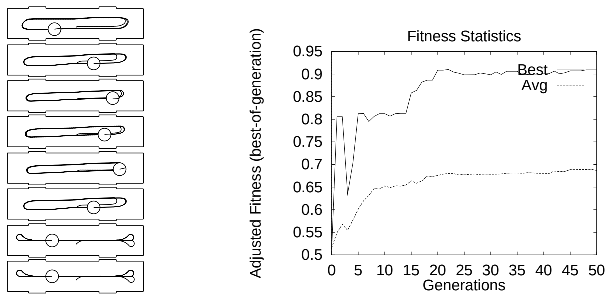


Figure 7: Eight successful individuals evolved with 3 fitness cases (left). The best adjusted fitness for the best individual is shown on the right together with the average adjusted fitness over all 20 experiments.

previously used by Floreano and Mondada (1994) and by Mondada and Floreano (1995) to evolve weights for a neural net based architecture. We tried to model the environment which was available for the experiments with the real robot. The environment for the real robot can be seen in Fig. 1. Due to space limitations in our lab, the environment is rather small, approximately 1.56m $\times$ 6.29m. We experimented with one, two, and three fitness cases. For each we performed 20 runs. Only 2 successful individuals evolved when we used 1 fitness case. 6 successful individuals evolved with 2 fitness cases and 8 with 3 fitness cases. Simulation results are presented for 3 fitness cases. The paths of the 8 successful individuals and the adjusted fitness values are shown in Fig. 7. A very small individual achieved the best fitness of all 20 runs: `(IFLTE FR FL TL TR)`.

## 4.3 EXPERIMENT WITH THE REAL ROBOT

Next we replicated the experiment with the real robot. To limit the time needed for the experiment with the real robot we used only 2 fitness cases. The experiment was performed over a period of 2 months. A segment of the corridor in our lab was blocked using several tubes as a barrier (Fig. 1). The time required for the evolution was 197 hours. Due to the limited battery capacity of the robot we had to exchange bat-

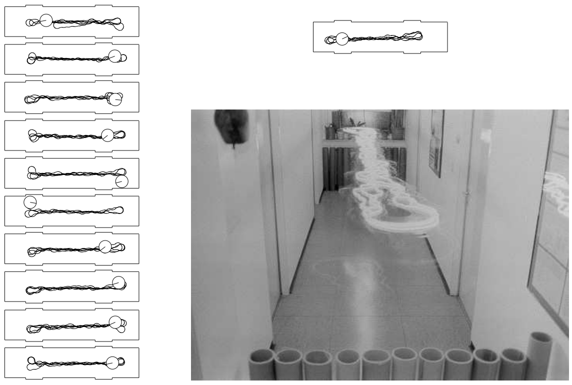

Figure 8: Ten tests of the best individual which was evolved with the real robot (left). The paths were recorded using the robot's odometry. Additional test (right) where the path was recorded using a standard camera set to long term exposure and a small light bulb which was mounted to the top of the robot. The photograph on the right and the path on the top right were recorded at the same time.

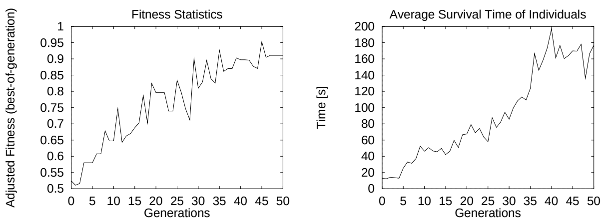


Figure 9: Adjusted fitness of the best individual for each generation for the experiment with the real mobile robot (left). Average survival time of the individuals for every generation (right).

teries periodically. Initially batteries were exchanged between generations. Later in the run batteries were also exchanged during fitness testing of a generation. The evolution was temporarily halted to perform the exchange.

The best fitness of the run was achieved by an individual with 457 nodes in generation 45. The individual is not shown here due to its large size. The individual was tested in a series of 10 experiments. It managed to survive 8 of the 10 runs (Fig. 8). Graphs of the adjusted fitness values and the average survival time are shown in Fig. 9. The individuals of the first generation either bumped very soon into a wall or went around in tight circles. As evolution proceeded the quality of the individuals improved. Therefore, more time was required to complete one generation. Initially, average survival time of the individuals was 12.9s which rose to a maximum of 196.7s in generation 40.

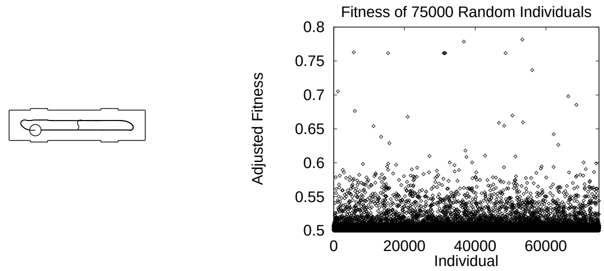


Figure 10: Scatter plot of 75000 random individuals (right). The best individual is shown on the left.

### 4.4 SAMPLING OF RANDOM INDIVIDUALS

In 50 generations with a population size of 75 a total number of 3750 individuals are evaluated. The search range with depths from 2 to 6 was used to initialize the first generation for all runs using ramped half and half initialization. We evaluated 3750 random individuals using ramped half and half initialization with depths between 2 and 6 and three fitness cases. No successful individual was found during this random search. Continuing the search with different random seeds yielding a total of 75000 random individuals produced the individual shown in Fig. 10.

## 5 CONCLUSION

A behavior-based control architecture has been evolved for a mobile robot using genetic programming. Experiments have been performed both in computer simulations and with the real robot. The experiment with the real robot lasted two months. Due to the length of the experiment it could only be done once. However, high speed computer simulations could be used instead, provided that the results are transferable to the real world. Our results show that it is indeed possible to replicate the results achieved in computer simulations with a real mobile robot, that is Koza's classic experiment of evolving a control architecture can be transfered to the real world.

We used an appropriate representation which made a replication of the experiment in the real world possible. Results were reported both for the experiments using computer simulations and for the experiment with the real robot. The experiment was carried out with a service robot which differs in many respects from other small sized robots which are usually used in evolutionary robotics.


**Acknowledgments**

This work was supported in part by a scholarship according to the Landesgraduiertenförderungsgesetz to

the first author. For our experiments we used the lil-gp Programming System (Zongker and Punch 1996). We thank William B. Langdon from the Centrum voor Wiskunde en Informatica, Amsterdam, who gave helpful comments on an earlier version of this paper.

## References


W. Banzhaf, P. Nordin, and M. Olmer (1997). Generating adaptive behavior using function regression within genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, eds., *Genetic Programming 1997: Proc. of the 2nd Int. Conf. on Genetic Programming*, pp. 35–43, San Francisco, CA. Morgan Kaufmann Publishers.

W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone (1998). *Genetic Programming - An Introduction: On The Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann Publishers, San Francisco, CA.

V. Braitenberg (1984). *Vehicles: Experiments in Synthetic Psychology.* The MIT Press, Cambridge, MA.

R. A. Brooks (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23.

R. A. Brooks (1992). Artifical life and real robots. In F. J. Varela and P. Bourgine, eds., *Toward a practice of autonomous systems: Proc. of the 1st Europ. Conf. on Artificial Life*, pp. 3–10, Cambridge, MA. The MIT Press.

M. Ebner (1998). Evolution of a control architecture for a mobile robot. In M. Sipper, D. Mange, and A. Pérez-Uribe, eds., *Proc. of the 2nd Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES 98), Lausanne, Switzerland*, pp. 303–310, Berlin. Springer-Verlag.

D. Floreano and F. Mondada (1994). Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, eds., *From animals to animats 3: Proc. of the 3rd Int. Conf. on Simulation of Adaptive Behavior, Brighton, England*, pp. 421–430. The MIT Press.

C. M. Fonseca and P. J. Fleming (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16.

I. Harvey, P. Husbands, and D. Cliff (1993). Issues in evolutionary robotics. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, eds., *From animals to animats 2: Proc. of the 2nd Int. Conf. on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pp. 364–373. The MIT Press.

I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20:205–224.

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

J. R. Koza (1994). *Genetic Programming II, Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, MA.

W.-P. Lee, J. Hallam, and H. H. Lund (1997). Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *Proc. of the IEEE 4th Int. Conf. on Evolutionary Computation*, pp. 501–506. IEEE Press.

M. Matarić and D. Cliff (1996). Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19:67–83.

J.-A. Meyer, P. Husbands, and I. Harvey (1998). Evolutionary robotics: a survey of applications and problems. In P. Husbands and J.-A. Meyer, eds., *Proc. of the 1st Europ. Workshop on Evolutionary Robotics, Paris*, pp. 1–21.

O. Miglino, H. H. Lund, and S. Nolfi (1996). Evolving mobile robots in simulated and real environments. *Artificial Life*, 2:417–434.

F. Mondada and D. Floreano (1995). Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems*, 16:183–195.

S. Nolfi (1997). Evolving non-trivial behaviors on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 22:187–198.

S. Nolfi, D. Floreano, O. Miglino, and F. Mondada (1994). How to evolve autonomous robots: different approaches in evolutionary robotics. In R. A. Brooks and P. Maes, eds., *Artificial Life IV: Proc. of the 4th Int. Workshop on the Synthesis and Simulation of Living Systems*, pp. 190–197. The MIT Press.

P. Nordin and W. Banzhaf (1995). Genetic programming controlling a miniature robot. In *Working Notes of the AAAI-95 Fall Symposium Series, Symposium on Genetic Programming, MIT, Cambridge, MA*, pp. 61–67.

P. Nordin and W. Banzhaf (1997a). An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behaviour*, 5(2):107–140.

P. Nordin and W. Banzhaf (1997b). Real time control of a khepera robot using genetic programming. *Cybernetics and Control.*

M. Olmer, P. Nordin, and W. Banzhaf (1996). Evolving real-time behavioral modules for a robot with GP. In M. Jamshidi, F. Pin, and P. Danchez, eds., *Robotics and Manufacturing, Proc. 6th Int. Symposium on Robotics And Manufacturing (ISRAM-96), Montpellier, France*, pp. 675–680, New York. Asme Press.

C. W. Reynolds (1994a). The difficulty of roving eyes. In *Proc. of the 1st IEEE Conf. on Evolutionary Computation*, pp. 262–267. IEEE.

C. W. Reynolds (1994b). Evolution of corridor following behavior in a noisy world. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, eds., *From animals to animats 3: Proc. of the 3rd Int. Conf. on Simulation of Adaptive Behavior, Brighton, England*, pp. 402–410. The MIT Press.

C. W. Reynolds (1994c). Evolution of obstacle avoidance bahavior: Using noise to promote robust solutions. In K. E. Kinnear, J., ed., *Advances in Genetic Programming*, pp. 221–241, Cambridge, MA. The MIT Press.

C. W. Reynolds (1994d). An evolved, vision-based model of obstacle avoidance behavior. In C. G. Langton, ed., *Proc. of the Workshop of Artificial Life III SFI Studies in the Sciences of Complexity*, pp. 327–346. Addison-Wesley.

S. J. Ross, J. M. Daida, C. M. Doan, T. F. Bersano-Begey, and J. J. McClain (1996). Variations in evolution of subsumption architectures using genetic programming: The wall following robot revisited. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, eds., *Genetic Programming 1996, Proc. of the 1st Annual Conf., Stanford University*, pp. 191–199, Cambridge, MA. The MIT Press.

M. S. Wilson, C. M. King, and J. E. Hunt (1997). Evolving hierarchical robot behaviours. *Robotics and Autonomous Systems*, 22:215–230.

D. Zongker and B. Punch (1996). *lil-gp 1.01 User's Manual (support and enhancements B. Rand).* Michigan State University.