Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Robotics and Autonomous Systems 46 (2004) 159–173

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Maze exploration behaviors using an integrated evolutionary robotics environment

A.L. Nelson, E. Grant*, J.M. Galeotti, S. Rhody

Center for Robotics and Intelligent Machines (CRIM), Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911, USA

Received 8 October 2001; received in revised form 29 October 2003; accepted 17 November 2003

Abstract

This paper presents results generated with a new evolutionary robotics (ER) simulation environment and its complementary real mobile robot colony research test-bed. Neural controllers producing mobile robot maze searching and exploration behaviors using binary tactile sensors as inputs were evolved in a simulated environment and subsequently transferred to and tested on real robots in a physical environment. There has been a considerable amount of proof-of-concept and demonstration research done in the field of ER control in recent years, most of which has focused on elementary behaviors such as object avoidance and homing. Artificial neural networks (ANN) are the most commonly used evolvable controller paradigm found in current ER literature. Much of the research reported to date has been restricted to the implementation of very simple behaviors using small ANN controllers. In order to move beyond the proof-of-concept stage our ER research was designed to train larger more complicated ANN controllers, and to implement those controllers on real robots quickly and efficiently. To achieve this a physical robot test-bed that includes a colony of eight real robots with advanced computing and communication abilities was designed and built. The real robot platform has been coupled to a simulation environment that facilitates the direct wireless transfer of evolved neural controllers from simulation to real robots (and vice versa). We believe that it is the simultaneous development of ER computing systems in both the simulated and the physical worlds that will produce advances in mobile robot colony research. Our simulation and training environment development focuses on the definition and training of our new class of ANNs, networks that include multiple hidden layers, and time-delayed and recurrent connections. Our physical mobile robot design focuses on maximizing computing and communications power while minimizing robot size, weight, and energy usage. The simulation and ANN-evolution environment was developed using MATLAB. To allow for efficient control software portability our physical evolutionary robots (EvBots) are equipped with a PC-104-based computer running a custom distribution of Linux and connected to the Internet via a wireless network connection. In addition to other high-level computing applications, the mobile robots run a condensed version of MATLAB, enabling ANN controllers evolved in simulation to be transferred directly onto physical robots without any alteration to the code. This is the first paper in a series to be published cataloging our results in this field.

Keywords: Evolutionary Robotics; Behavioral Robotics; Evolutionary neural networks; Distributed Robotics; Robot colonies

* Corresponding author.
E-mail address: egrant@eos.ncsu.edu (E. Grant).

0921-8890/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2003.11.002

1. Introduction

1.1. Evolutionary Robotics

Evolutionary Robotics (ER) is a sub-field of Behavioral Robotics [1] and is concerned with the application of evolutionary computing methods to the area of autonomous robotic systems control. One of the central goals of ER is to develop automated methods that can be used to evolve complex behavior-based control strategies. For this reason, mobile robots and robot colonies have been the focus of recent ER research [2]. The main emphasis in ER research continues to be on the development of evolved controllers and intelligent control systems rather than on robot bodies and physical electro-mechanical system design. There has been some disagreement in the field to this view, and in [3,4] the evolution of robot bodies is considered.

The design approach presented here defines a synergy between the simulated robot world and the physical mobile robot world in terms of intelligent controller portability. It is finding the harmonious co-existence between these two worlds that we believe will produce advances in ER research.

In the past few years, a number of reports describing experimental proof-of-concept and demonstrational ER research have been published. These include the evolution of simple behaviors in small mobile robots such as homing and obstacle avoidance [5–7]. Other research groups have evolved controllers for locomotion of legged robots using ER methods [8–10]. These represent a step up in complexity due to the increased number of degree of freedom (DOF) associate with legged robots.

In this paper, a maze searching and navigation task is investigated. Robots were equipped with five simple binary tactile sensors. These sensors supply extremely sparse data to the robot controllers about their environment. In order to overcome this, the controllers must be able retain and respond to information from past sensor readings. Purely reactive controllers generally produce very sub-optimal behaviors when supplied with such simple sensor inputs. The neural network structure in this work was specifically designed to be able to evolve temporal processing abilities. Our results show that controllers do evolve control strategies that rely on both current and past sensor readings to resolve and respond to situations that would be ambiguous to purely reactive controllers.

1.2. Evolutionary Robotics and mobile robot colonies

A crucial element in an ER research platform is the physical robotic systems upon which evolved controllers are to be instantiated. Much of the work to date in ER has focused on computer simulation and software environment development but has fallen short of implementation in real robotic systems. For instance, in [3–12] robot simulation environments are described but the work is not extended to actual physical implementations. There has, however, been a fair amount of ER work implemented in real robots [13–15]. Much of this research involves custom robot platforms with limited computational power. There are various small mobile robot platforms available. Possibly the best known of these is the Khepera mobile robot platform, a system that has been adopted for ER research by various groups worldwide [16–18]. The Alice robots described in [19] have also been used in several mobile robot research studies. These are very small (40 mm) and they have a limited computational power. The Millibots developed at Carnegie Mellon [20] are small modular robots that can be reconfigured to make a heterogeneous population where each robot has limited abilities, but can collaborate with other population members to complete complex tasks.

The research presented in this paper required a small mobile robot platform, with dimensions that could navigate within an in-door purpose built physical maze, but one with high computational and communications capacities that would allow for the transfer of ANN algorithms onto the physical robots. Finding no such robot available, the CRIM team designed and constructed a new evolutionary robotics platform. This design has been fabricated to produce a colony of eight mobile robots and is described briefly in Section 3.

1.3. Artificial neural networks in ER

Artificial neural networks (ANN) have become the computational structure of choice in ER. Recently published ER literature contains numerous exam-

ples of evolutionary neural computing development environments aimed at the production of neural controllers for mobile robots. For instance, in [21,22] simulation environments for evolving multi-layer feed forward networks are used to control mobile robots with two driving wheels. In addition, Watson et al. [23] describe a robot platform in which very simple networks containing several neurons are evolved to perform homing tasks while instantiated in real robots. With the possible exception of the neural controllers developed for legged robot locomotion [8,15], the majority of ER work reported to date makes use of very simple ANN architectures and relatively small networks are actually trained for use in real robots [24]. Most of these networks are capable of little or no temporal processing. Typical examples are the 4-unit network used in [23] and the single layer networks described in [14]. We believe that in order to move the field of ER beyond the proof-of-concept stage, and to develop intelligent robot controllers capable of sophisticated and useful behaviors, mobile platforms need to accommodate larger and more recurrent ANN topologies. Although ER work to date has demonstrated that neural networks can be evolved to perform simple robot behaviors, little has been shown about the possible generalization of such methods to more complicated behaviors that require larger, more sophisticated networks. In fact, larger evolutionary neural networks are qualitatively distinct form their smaller simpler counterparts especially with regard to training using evolutionary methods. A major impetus for the development of our ER research test-bed has been to investigate the evolution of larger, more complicated neural controllers.

The ER research test-bed used in this work supports a generalized class of multi-layer ANN controllers coupled to both simulated and real robots. These networks contain both time-delayed and recurrent connections and are trained in a simulated environment using genetic algorithms. The ER research test-bed consists of several main elements, including (1) a simulated mobile robot environment, (2) a class of simulated mobile robot agents, (3) an ANN-based controller structure and associated evolutionary computation-based training environment, (4) a colony of real mobile robots with advanced computing and communications capabilities, and (5) a physical test-bed (environment) for the real robots to interact with. These elements will be discussed in the following two sections.

A presentation and discussion of the training performance metrics and results generated with this research test-bed will be presented in Sections 4 and 5.

2. The neuro-evolution and simulation environment

2.1. The artificial neural network controllers

Here, we describe the neural network architecture used in both the simulated, and real robots. The networks are a class of multi-layered recurrent and time-delayed neural networks. The delayed and recurrent connections impart the possibility of developing temporal processing. In the case of the behaviors studied in this work, and owing to the very simple sensors used by the robots, controllers that can make use of temporal information have the potential to outperform completely reactive controllers. Inputs from the mobile robots sensors are fed into the neural networks and the resulting outputs are applied as actuator command values to the mobile robot drive-wheel motors. These networks can be considered as fully connected generalizations of the Elman and Jordan networks [25,26] and include recurrent connections from both hidden layer and output neurons. In addition, the networks have time-delayed input, hidden, and recurrent connections. These additional recurrent and time-delayed connections allow for the potential of temporal processing. In this work, the layered structure was specified before training and remained constant during the course of training. Only the weights of the networks were evolved. These layered networks represent a large class of network topologies, but they only very sparsely cover the space of all possible networks. However, setting a connection weight in a fully connected network to zero is equivalent to removing that connection, hence the weight-only search space contains all network architectures of lower dimension. In effect, manual selection of a particular network layered structure and feedback level imposes an upper limit on the complexity of the evolved architectures.

Sigmoid neurons were used in the hidden layers while output layers consisted of either sigmoid or linear neurons. The particular activation functions for

each of these neuron types are given in Eqs. (1) and (2), respectively:

$$y_{\text{sig}}(u) = \frac{1}{1 + \mathrm{e}^{-u}}, \tag{1}$$

$$y_{\text{lin}}(u) = u, \tag{2}$$

where u is the sum of the weighted input connections:

$$u = \sum_i w_i x_i + w_{\text{b}} \tag{3}$$

and w_i, x_i, and w_{b} are the ith weight associated with the ith input, and a bias weighted unity input, respectively. All the weights associated a particular network layer m, time delay level t and recurrence level s, are represented and stored in a separate N by I array. The entire weight set for a given network is then given by the multidimensional matrix:

$$W = [\mathbf{W}_{m,t,s}] | m \in \{1, \ldots, M\}, \quad t \in \{1, \ldots, T\}, s \in \{1, \ldots, S\}, \tag{4}$$

where each sub-matrix of $\mathbf{W}$ is an n by i matrix of weights given by

$$\mathbf{W}_{m,t,s} = [\, \mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_N \,]^{\mathrm{T}}_{m,t,s} \tag{5}$$

and each column vector is an ordered set of weights associated with a set of inputs subscripted by $i \in \{1, \ldots, I\}$ to the nth neuron ($n \in \{1, \ldots, N\}$) of the mth layer, for the tth time delay level, and sth recurrence level:

$$\mathbf{w}_n = [\, w_1 \quad w_2 \quad \cdots \quad w_I \,]. \tag{6}$$

M, T, and S, are the total number of network layers, maximum degree of time-delayed connections, and the maximum feedback depth to previous layers of the recurrent connections, respectively.

A generalized neural network specification and evolutionary training environment for this type of network was developed in MATLAB. Fig. 1 contains two typical network configurations. In both Network A and Network B (Fig. 1(a) and (b)), the thicknesses of the connection lines are proportional to the absolute values of their particular associated weights. These network connection graphs are generated by the evolutionary training environment and show visually the changes occurring in the networks during training.

2.2. The simulation environment

The simulation environments consist of planar grids in which each grid element is either solid/occupied or space/transparent. The material arrangement in each environment is discretized. The space its self is a continuous region in R^2 such that simulated robot agents may exist at any real valued point within the range of the environment.

In each environment, a variable number n of robot agents is maintained. Each robot agent consists of a data structure that stores the robot's current position, orientation, sensor input readings, and actuator output values. In addition a separate unique neural controller structure is associated with each robot.

The mobile robots have two motor-wheel actuators, one on each of the right and left sides of the robot. This is a common and simple drive-wheel arrangement found in many mobile robots and allows robot agents to turn in any direction or move along any diameter arc by varying the inputs to the wheel motors (deferential steering). Fig. 2 shows several simulated worlds and populations of simulated robots agents.

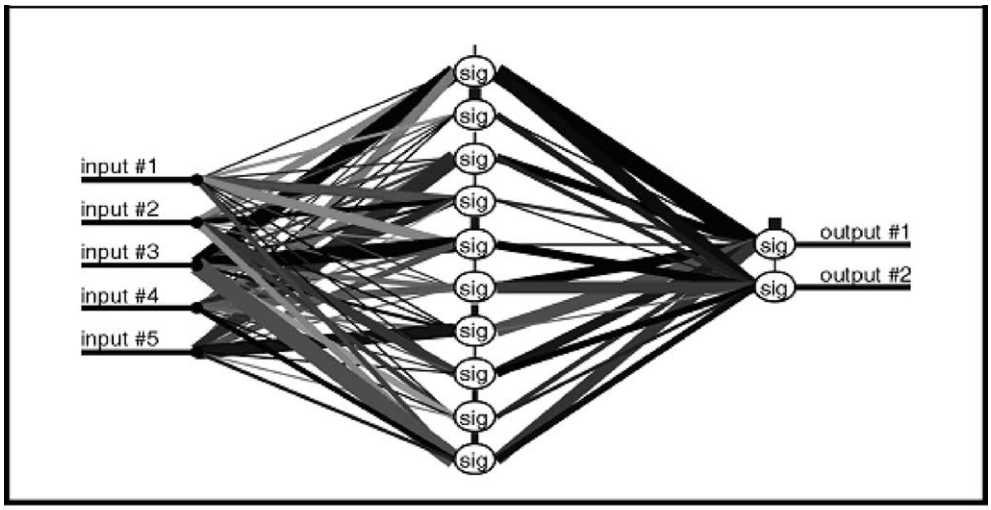
2.3. The evolutionary training algorithm

The neural network controllers were trained using evolutionary computing methods in conjunction with the robot simulation environment. The neural networks were trained using a genetic algorithm to directly adjust connection weights. Selection was based on a performance fitness evaluation function (described later in this section).

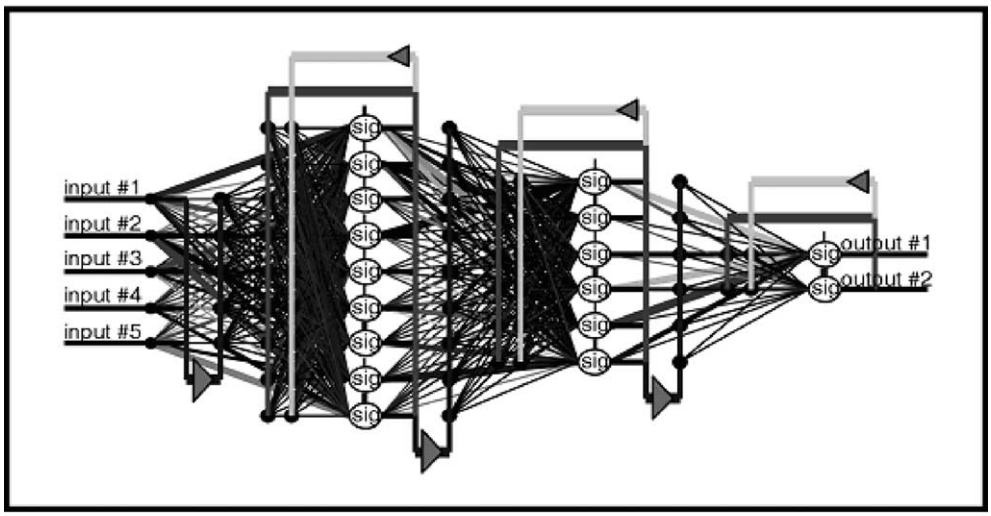
Within the training environment, connection weights can be initialized to small random values from a single random distribution, or from different distributions that depend on the recurrence and degree of time delay of the particular connection. For the general case, weights were initialized using the following equation:

$$w = R\mu(m)\tau(t)\sigma(s), \tag{7}$$

where $\mu(m)$ is linearly decreasing in m, and $\tau(t)$ and $\sigma(s)$ are monotonically decreasing exponentials with maxima of 1. Here m, t, and s represent the layer depth, the degree of temporal delay, and the degree of spatial feedback, respectively, associated with a



(a) Network A



(b) Network B

Fig. 1. Example schematic representations of two neural networks developed by the evolutionary neural computing environment. Network A is a simple feed forward single hidden layer Perceptron. Network B includes two hidden layers and both time-delayed and hidden layer feedback connections.

particular connection/weight. R is a number from a flat random distribution on $\{-1, 1\}$. R is re-sampled for each weight initialization. The effect of the weight initialization equation (7) is that initial weight values closer to the input layer and with a lower degree of recurrence, have larger magnitudes then those farther into the network, and with a greater degree of recurrence. When $\mu(m) = k$ is constant, and t and s are constant unity functions, all weights are initialized to random numbers in the range of $-k$ to $+k$.

The chromosome data structure **C** is a set of real valued scalar numbers where each number corresponds directly to a particular weight w in the neural network weight set **W**, from Eqs. (5) and (6). An individual

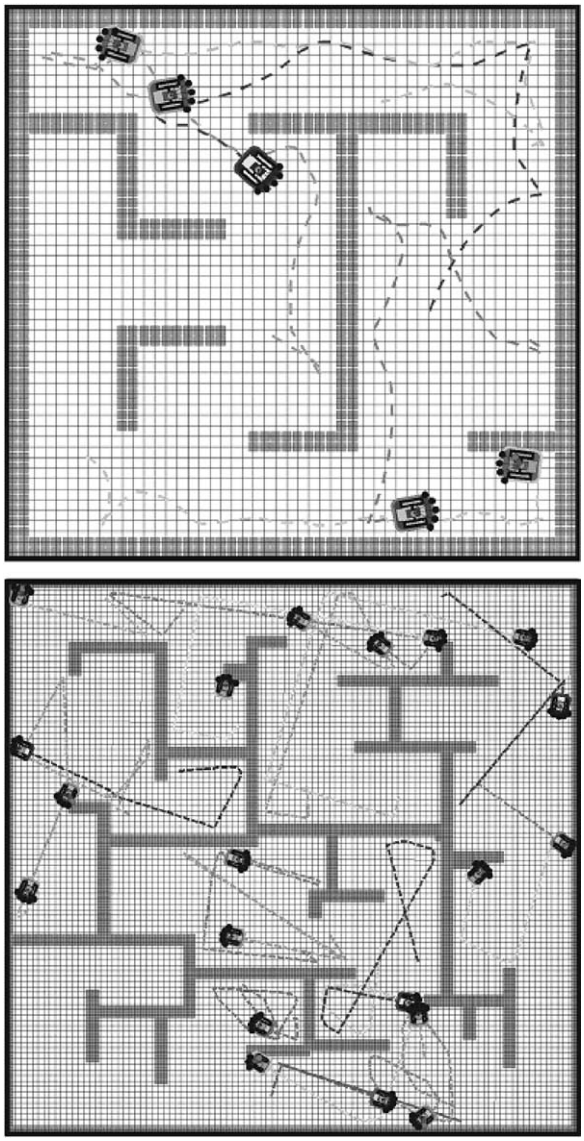


Fig. 2. Two example simulated maze environments including simulated mobile robot agents with trained neural controllers. Dotted lines indicate the paths taken by the robots during the course of the simulation.

chromosome is specified as follows:

$$C = [c_1, c_2, \ldots, c_g] = [w_{1,1,1,1,1}, \ldots, w_{m,t,s,n,i}, \ldots, w_{M,T,S,N,I}]. \quad (8)$$

The rate (probabilistic frequency) of mutation for each weight, w, is dependant on the size of the sub-matrix of $\mathbf{W}$ to which it belongs, and is given by

$$\text{Rate} = \frac{1}{N * I}\text{Base\_rate}, \quad (9)$$

where N and I are the dimensions of the sub-matrix $\mathbf{W}_{m,t,s}$ to which w belongs and Base_rate is a whole number.

Mutation magnitudes, for the weights in an ANN, are scaled in a similar manner to the weight initialization values equation (7). Each weight mutation magnitude depends on the location of the weight within the network structure. Hence, for each member of the robot controller population selected to be mutated, the new chromosome elements c' of $\mathbf{C}$ are given by

$$c' = c + \Delta c = w + \eta RP\mu'(m)\tau'(t)\sigma'(s), \quad (10)$$

where $P \in \{0, 1\}$ is determined by the rate of mutation (Eq. (9)), $\mu'(m)$ linearly decreases with m, $\tau'(t)$ and $\sigma'(s)$ are monotonically decreasing exponentials with maxima of 1, and η a base mutation magnitude or step size. Also, R is a number from a flat random distribution between -1 and 1.

The next generation population, $P(k)$, is constructed from the union following four sets derived from the current population:

$$\begin{aligned} P(k) = {} & \{p_1(k-1) \cdots p_m(k-1)\} \\ & \cup \{p'_1(k-1) \cdots p'_m(k-1)\} \\ & \cup \{p_{2m+1}(k-1) \cdots p_{n-1}(k-1)\} \\ & \cup p_1(k-2), \end{aligned} \quad (11)$$

where $p_n \in P(k)$ is the nth individual of the population at generation k, p'_n a mutated version of p_n, m/n the fraction of the population that is mutated and replaced, and n the total number of individuals in the population. The population P, is ordered from fittest to least fit before Eq. (11) is applied. The result of Eq. (11) is that $1/m$ of the fittest controllers are transferred unchanged to the next generation, this same fraction of the controller population is mutated and added to the next generation, the single fittest member from two generations past is included, and the remainder of the next generation population is made up of the fittest remaining members of the current controller population. The parameters n and m are set at the beginning of each evolutionary run. Values for n and m are selected by the user and reflect a trade-off between evolutionary speed and chaos during training. It was found that

an $m/n = 1/4$ value giving a replacement rate of 25% produces functional controllers for population size of $n = 20$–100. The evolutionary algorithm described in (11) is a form of greedy mutation-only $(\mu + \lambda) - EA$ [27], with the inclusion of the fittest member of the population 2 generations previous to $P(t)$.

2.4. The performance fitness function

Several training fitness evaluation functions (objective functions) were used in this work. These include metrics for controller selection during evolution and metrics for performance of fully evolved controllers operating in the real world.

Navigation and object avoidance behaviors are often used as testing and benchmark behaviors in evolutionary robotics research. In this work, we focus the evolution mobile robot controllers for navigation behaviors in a maze environment. Performance evaluation is based on a weighted sum of several factors, including the net offset between a robot's starting position and its final position, (net_offset), and whether or not the robot becomes stuck on material within the simulated environment, (stuck). The robots are required to make as much progress through the maze as possible. This is measured by the distance a robot travels through a maze in a given number of time steps. Implicitly, robots must learn to negotiate walls to maximize their progress through a maze with many walls. A robot that cannot avoid walls will soon become immobilized when its path is blocked by a wall. In this work, we use the following fitness function to select for navigation behavior in an environment containing walls (a maze):

$$F(p_i) = k_1 * \text{total\_curve\_length} + k_2 * \text{net\_offset} + k_3 * \text{max\_offset} + k_4 * \text{stuck}, \quad (12)$$

where k_1 to k_4 are weighting factors. total_curve_length is the line-integral of the full path followed by the robot and max_offset is the greatest linear distance obtained by the robot and any time during its travel. The weighting factors were derived empirically through trial and error. The desired behavior is represented by the third factor, maximum offset achieved by the robot from its starting position. It was found, however, that inclusion of two other distance measures, and an explicit penalty for becoming stuck, were required to achieve evolution of navigation behaviors in a reasonable amount of time. It should be noted that in all cases of performance evaluation during evolution, run times were limited so that the best possible performance would result in travel from one side of the environment to the other without time for a return trip. After evolution, resulting robot controllers were allowed to operate for much longer periods of time to demonstrate the dynamics of acquired behaviors. Performance fitnesses over several simulation runs were averaged before each generational selection to smooth effects of random robot position initialization. During each generation, the members of the population are initialized to random locations in the simulated environment. Robots do not interact with each other, but only with the environment.

3. The EvBot mobile robot platform

In this section, we will give a brief overview of the design and capabilities to the real mobile robot colony used in this research. A full description of the robot platform has been presented in another forthcoming report [28].

A colony of eight robots has been constructed for this and related ER research. Each robot is 127 mm wide by 165 mm long by 152 mm high and is constructed on a two track treaded wheel base. Each robot is equipped with a PC/104-based onboard computer with the following specifications:

- 32 bit CPU core operating at 133 MHz.
- X86 software compatible.
- Embedded PC BIOS in Fail-safe Boot ROM.
- 8 Mb Disc-On-Chip/64 Mb RAM.
- 96 Mb ATA Flash Memory PC-Card.
- Linksys Wireless Ethernet PC-Card, 11 Mbps capability.

A custom Linux Distribution derived from RedHat Linux 7.1 is used as the operating system and is capable of supporting MATLAB in addition to other high-level software packages. The Linux OS running on the robots includes specific alterations and system commands that facilitate the easy acquisition and transfer of sensor data into high-level controllers. The robots are linked to one another and to the Internet via a Linksys wireless network access point that can

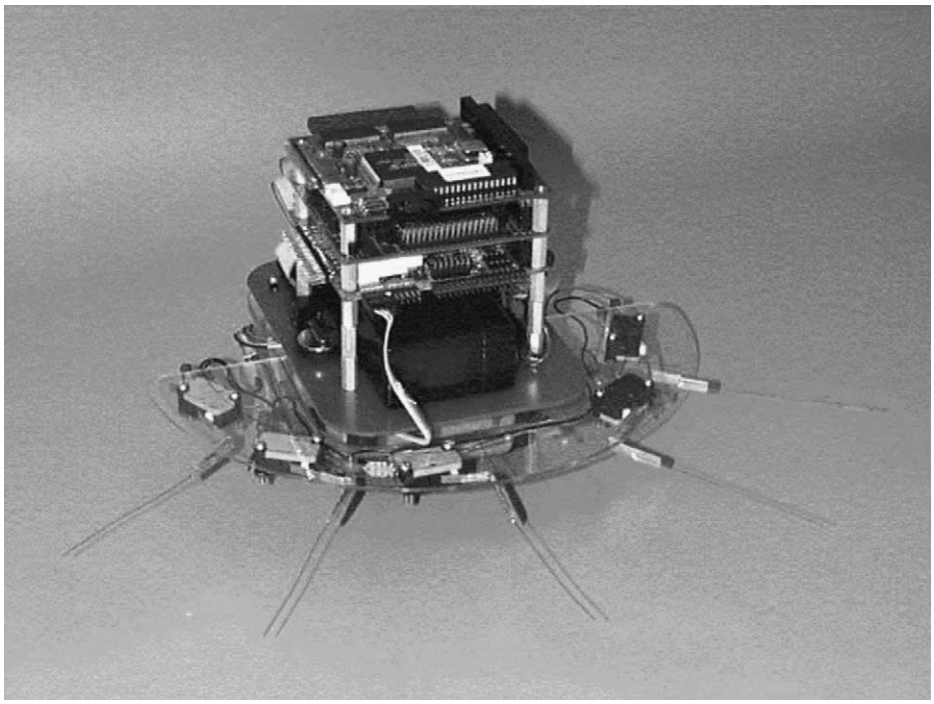


Fig. 3. A fully assembled EvBot mobile robot agent attached with whisker tactile sensor array.

handle up to 21 devices. A photograph of a fully assembled EvBot is shown in Fig. 3.

The EvBots can be fitted with various sensor, including video cameras. In this work, all sensing of the environment was performed using a whisker tactile array of five forward facing binary sensors. A sensor is triggered when its attached wire whisker comes into physical contact with an object. The EvBot in Fig. 3 is shown equipped with a whisker tactile sensor array.

Low-level control of the wheel drive motors and tactile sensors is performed by a BasicX micro-controller. An RS232 serial port interfaces the PC/104 CPU to the Basic-X. The BasicX is programmable with up to 8000 lines of Visual Basic code and is capable of controlling the robot independently of the PC/104, which can be removed to make a very compact mobile robot for applications that require less computing power.

4. Results and discussion

For our preliminary work, and for testing of our evolutionary robotics platform, we chose the development of maze navigation and searching behaviors. Such object avoidance and navigation behaviors are commonly used as test behaviors [29] in ER. As discussed in Section 2, ANN-based controllers were evolved in simulation for maze navigation behaviors and relied on binary tactile sensors for all sensing of the environment. Robot controller selection during evolution was based on performance as measured by Eq. (12). Evolved neural controllers were transferred to real robots and tested in a real maze environment.

A number of neural network architectures were found to be evolvable to perform the navigation task in simulation and to retain ANN controller functionality when transferred to real robots. In related work, it was found that simple single hidden layer feed forward networks could be trained to navigate robot agents in simulation when a model of simulated laser range-finding sensors was used [30]. Those evolved ANN controllers had no capacity for temporal processing. The resulting robot agents were effectively simple Braitenberg vehicles in that they produce motor actuator commands in direct response to current range-finding sensor readings. The level of information provided

by range-finding sensors was sufficient so that purely reactive controllers could perform the task reasonable well in this environment. Simple binary tactile sensors, on the other hand, require controllers to make use of information from the past in order to overcome perceptual aliasing. For example, a robot would receive all zeros (off) from its set of five tactile sensors before it came in contact with a wall, and then again after it had backed away from that wall. Controllers must make use of information from sensor readings from the past in order to distinguish between these two conditions and avoid getting caught in behavioral response loops. That is, a robot must do something different when it is backing away from a wall then when it is approaching a wall, even though it 'sees' the same thing (nothing) in both cases. In this work, we used binary tactical sensors on our robots in part to study an evolutionary system that would benefit significantly from the acquisition temporal processing abilities.

In order to improve the potential functionality of evolved ANN controllers, especially in the context of simple tactile sensor inputs, robots agents with ANN controllers capable of temporal processing were trained in the simulation environments. It was found that networks of moderate complexity produced the best results in the least amount of simulation time. Such networks had 1–3 hidden layers with 5–10 processing units per layer, with all connections duplicated and time delayed for 2–4 time steps. For example, the best-evolved ANN controller tested in the real robots and discussed below has 220 evolvable weights (see Fig. 5). Although our benchmark maze navigation behavior requires some degree of sophistication at the control level, an overly complex neural network structure was not found to be beneficial for this task. The largest networks trained to date contained three hidden layers of 10–20 neurons each, with recurrent connections to each of the three previous layers, and with all inputs and connections duplicated and time delayed for eight time steps. These networks contained approximately 7000 individually evolvable weights. Such networks with a high degree of recurrence were found to be difficult and computationally expensive to train and did not produce results superior to networks of moderate complexity. It is likely that quite simple specially formulated networks could be trained to accomplish the task studied in this work. We specifically focused on networks of greater complexity to show that larger more complicated network architectures can be readily evolved to perform these behaviors.

Fig. 4 shows training performance curves generated during two example evolutionary training runs. At each generation the best, the average, and the worst robot controller behavior performances were recorded (as measured by Eq. (12)). These were plotted against generation (epoch) number over the course of training. As is the case in most evolutionary neural network applications, the course of training is complex and dynamic. Although GA-ANN training dynamics are not the main emphasis of this research, such dynamics must be considered (often empirically) during the parameter selection and initialization phases of evolutionary network training. The progress of training is very dependant on network configuration and on training parameter settings. For example, the training run shown in the first panel of Fig. 4 shows several near-steady state plateaus followed by periods of rapid improvements. On the other hand, training set B shows a slow steady increase of performance level during the course of training.

The evolved controllers were transferred to real robots and tested in a real maze test-bed. The best performing ANN controllers were found to allow the real robots to wander through the real physical maze indefinitely without getting stuck on maze wall, and to allow the robots to make continual progress through the maze, i.e. the robots did not start spinning perpetually in one spot, or bump up against the same wall over and over again.

In this work, the real robots are used mainly to show that evolved controllers transfer to the real world and function qualitatively similarly compared to simulation. The quality of transference from simulation was evaluated in several ways. The responses of the controllers to sensor signals in simulation and in the real robots were compared and found to be identical. This was to be expected because the evolved controllers are identically similar in both cases. This similarity is made possible by the platform architecture, which allows direct transfer of evolved controllers from simulation to real robots without the need for any modification. In addition, the simple binary sensors used provide only logic values of 0 or 1 and inject no noise into either the simulated or real systems. There are however, two differences that cause divergence between real and simulated behaviors.

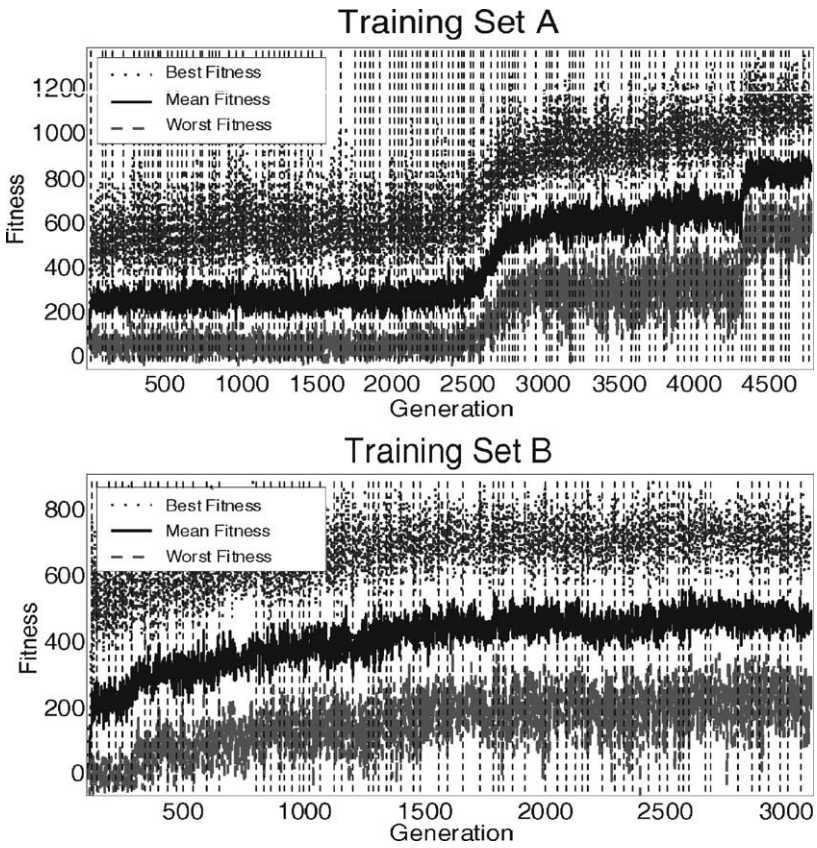


Fig. 4. Results from two evolutionary training sessions. In each graph, performance is plotted against generation epoch. Training set A shows several near-steady state plateaus followed by periods of rapid improvements. Training set B shows a slow steady increase of performance level over the course of training.

These are (1) differences between real and simulated motor/robot-kinetic responses to a given motor command, and (2) differences in sensor triggering when real and simulated robots are in proximity to objects (and simulated objects).

The real and simulated motor outputs have been calibrated to within 15%. The simulated tactile sensors always trigger at a distance of exactly 50 mm from the point at which the sensor whisker would be attached to the robot body. In the real environment, and with the real tactile sensors, the binary switch may trigger when the base of the tactile whisker is anywhere between 25 and 100 mm from an object.

Fig. 5 shows the states of the best network controllers before and after evolution (training). The performance metric given in Eq. (12) was used with the following parameter settings: $k_1 = 0, k_2 = 20, k_3 = 20, k_4 = 50$. Training performance was averaged for each controller for three simulations of 40 time steps before selection and mutation occurred. Three hundred and fifty generations were required to produce the functional controllers tested on the physical robots. The trained network was found to retain functionality when transferred to a real robot operating in a physical maze similar in dimensions to the one used in training. Fig. 6 shows the results of three separate tests of the evolved controller operating in the simulated environment and controlling a simulated robot agent equipped with tactile sensors.

The robot agent in the panels of Fig. 6 shows several different behaviors that allow it to avoid walls and extract itself from corners. The simulated agent is able to make progress through the environment. After encountering a wall and backing out of sensor range, robot

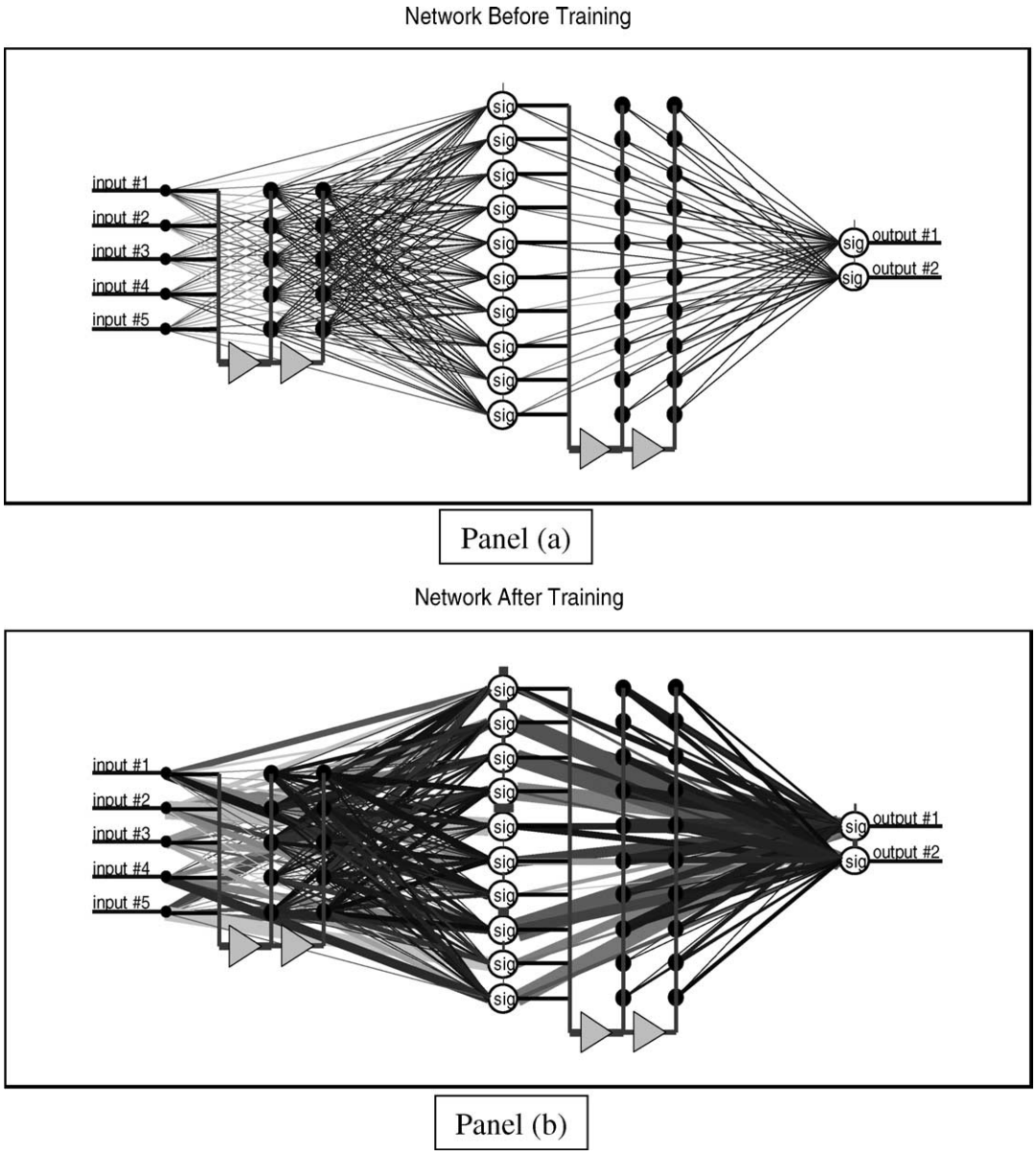


Fig. 5. Two schematic plots of a single hidden layer time-delayed neural network. In panel (a) the network is shown before training. Panel (b) shows the final trained version of the network.

agents displayed sequences of moves of up to five times in duration before stabilizing to a steady-state motor output. This indicates that the neural controllers evolved responses that could make use of, or at least responded to, sensor information receive up to five time steps into the past. Fig. 7 shows a close-up detail of the path of a simulated robot agent in an encounter with a wall. In the figure, the ovoid square shape is the robot, the black bar at the bottom of the figure is the wall, and the small dotted line indicates the path taken by the robot. The light dots indicate the backward moves made by the robot directly after encountering

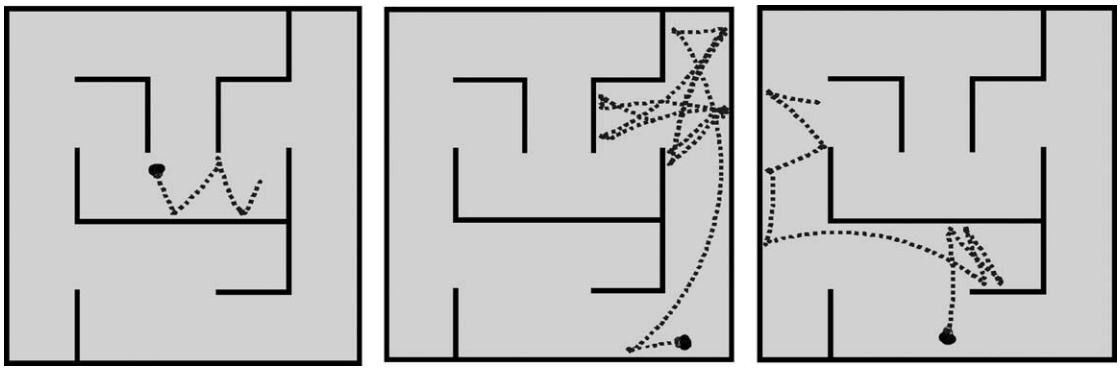


Fig. 6. Three simulation runs using an evolved controller to navigate a simulated agent with simulated tactile sensors through a maze. The robot agent is shown in its final position after each run in the maze. The dotted line indicates the path taken by the robot.

the wall and two subsequent moves the robot made while it was still in sensor contact with the wall. The sequence of dark dots indicates the sequence of moves (all rotating and/or forward) taken by the robot after it was out of sensor contact with the wall, but before its actuator commands had stabilized to a steady state.

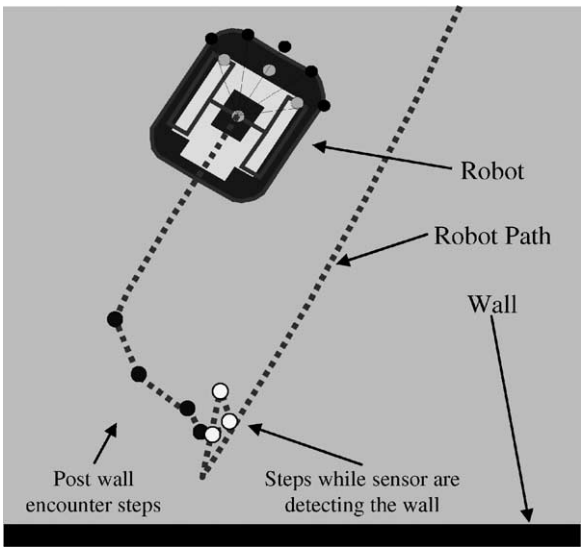


Fig. 7. Close-up of a simulated robot encountering a wall. The small dotted line indicates the path taken by the robot. The light dots indicates the backward move made by the robot directly after encountering the wall and two subsequent moves the robot made while it was still in sensor contact with the wall. The sequence of dark dots indicates the sequence or moves taken by the robot after it was out of sensor contact with the wall, but before its actuator commands had stabilized to a steady state.

The sequence of post-wall-encounter moves indicates the evolved neural controller made use of past sensor information to determine wheel motor commands.

Fig. 8 shows several overhead views of the real maze environment. In each panel of Fig. 8 a real robot controlled by the trained neural network controller displayed in Fig. 5 is shown. The dotted line indicates the path taken by the robot in each case.

The initial positions used for the real robot runs displayed in Fig. 8 correspond to those used in the simulated world shown in Fig. 6. It is clear that the real robot agents in the physical environment deviate from the paths taken by the simulated agents after a few encounters with walls. This is expected since very slight differences in approach angle and order of sensor contact can produce markedly different responses on the part of the evolved controllers. The controllers were found to be quite dynamic and displayed a wide variety of similar but not identical responses to encounters with walls. Qualitatively, robots behaved similarly in the simulation environment as in the real world. It should be noted that when given exactly the same time-sequence of sensor inputs, the real and simulated agents do produce the same motor output command sequences. Sensor input sequences can be quite dynamic in both the simulated and real worlds, however. For instance, a robot may encounter a wall, back up and turn slightly before moving forward into the wall again causing one or more new sensors to be triggered. Very slight differences in the original approach angle can in the end result in different tactile sensor input sequences, which in turn can result

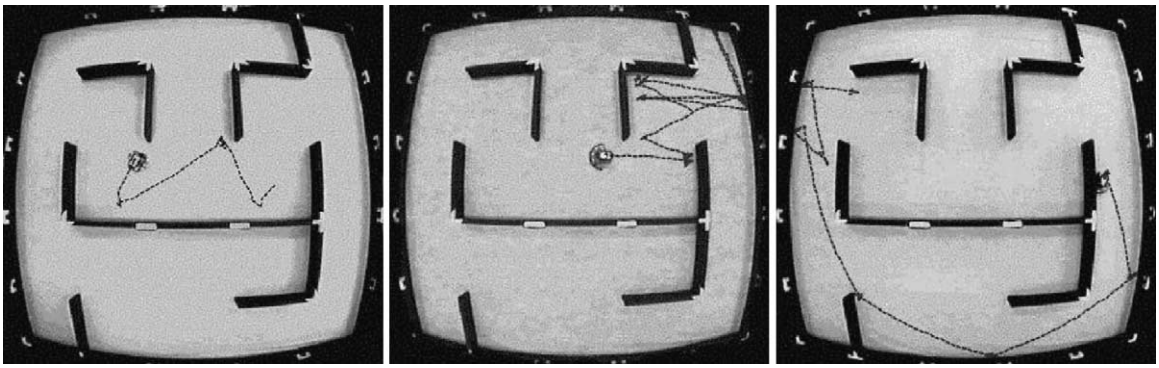


Fig. 8. Three overhead views of the real maze test-bed. In each panel, an EvBot mobile robot with the same evolved neural controller used for the simulation results displayed in Fig. 6 is shown. The robot is shown in its final position after each run in the maze. The dotted line indicates the path taken by the robot. Qualitatively, the sets of behaviors observed are similar to those.

in different behaviors. The exact responses for the controllers evolved for this work were not fully characterized. Assuming a possible sensor history of five time steps, there would be 2^{25} possible sensor input sequences. (Note that a controller with no temporal processing ability would be possible of producing only $2^5 = 32$ responses.)

5. Conclusions and future work

The research presented here developed real robot controllers for robots instantiated in physical maze environments. Populations of controllers were evolved in simulation and the best performing members of the populations were transferred to a real robot and tested in the physical environment. The simulation and evolution environment and physical robot platforms were developed together and coupled so that controllers evolved in simulation could be transferred quickly and easily to real robots. The neural controller architecture supported time-delayed and recurrent connections to allow for the evolution of temporal behaviors. Evolved controllers did evolve to respond to temporal information in order to overcome the inherent perceptual aliasing associated with the simple binary tactile sensors used by the robots in this research.

This work demonstrates the application and use of an integrated evolutionary robotics research platform that will continue to be developed for use in later research. Further developments to the evolutionary training environment include support of additional artificial neural network architectures and systems providing for the co-evolution of network architectures and evolution parameters in addition to network weights.

One area of interest is that of performance metrics for complex tasks, especially the implicit specification of performance through survival in artificial life environments.

Eventual potential applications of this research include search and rescue, terrain mapping, and remote autonomous exploration. In such application, groups of robots may employ populations of evolved controllers that can be selected and exchanged by the robot agents in response to performance in the environment. For example, a robot that has become immobilized could request that another controller from the evolved population be sent to it. Further work involving the use of knowledge-based map information as input to Behavioral Robotics neural network-based controllers is planned for the development of more sophisticated mobile robot colony cooperative navigation behaviors.

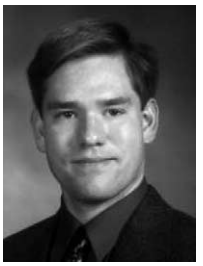
References

[1] R.C. Arkin, Behavior-based Robotics, MIT Press, Cambridge, MA, 1998.
[2] I. Ashiru, C.A. Czarnecki, Evolving communicating controllers for multiple mobile robot systems, in: Proceedings of the 1998 IEEE International Conference on Robotics and Automation, vol. 4, 1998, pp. 3498–3503.
[3] T.E. Revello, R. McCartney, A cost term in an evolutionary robotics fitness function, in: Proceedings of the 2000 Congress on Evolutionary Computation, IEEE, vol. 1, 2000, pp. 125–132.

[4] R.A. Brooks, Artificial life and real robots, in: F.J. Varela, P. Bourgine (Eds.), Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, MIT Press/Bradford Books, Cambridge, MA, 1992, pp. 3–10.
[5] J. Xiao, Z. Mickalewicz, L. Zhang, K. Trojanowski, Adaptive evolutionary planner/navigator for mobile robots, IEEE Transactions on Evolutionary Computing 1 (1) (2000) 18–28.
[6] D. Floreano, F. Mondada, Evolution of plastic neurocontrollers for situated agents. in: P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, S. Wilson (Eds.), From Animals to Animats IV, MIT Press, Cambridge, MA, 1996, 402–410.
[7] H.H. Lund, O. Miglino, L. Pagliarini, A. Billard, A. Ijspeert, Evolutionary Robotics—A Children's Game, Evolutionary Computation Proceedings, 1998, IEEE World Congress on Computational Intelligence, 1998, pp. 154–158.
[8] D. Filliat, J. Kodjabachian, J.A. Meyer, Incremental evolution of neural controllers for navigation in a 6 1egged robot, in: Sugisaka, Tanaka (Eds.), Proceedings of the Fourth International Symposium on Artificial Life and Robotics, Oita University Press, 1999.
[9] N. Jakobi, Running across the reality gap: octopod locomotion evolved in a minimal simulation, in: Proceedings of the First European Workshop on Evolutionary Robotics, EvoRobot'98, 1998.
[10] G.B. Parker, Generating arachnid robot gaits with cyclic genetic algorithms, genetic algorithms 1998, in: J.R. Koza, et al. (Eds.), Proceedings of the Third Annual Conference, Morgan Kaufmann, Madison, WI, 1998, pp. 576–583.
[11] G.S. Hornby, S. Takamura, J. Yokono, O. Hanagata, M. Fujita, J. Pollack, Evolution of controllers from a high-level simulator to a high DOF robot, in: Miller, J. (Ed.), Proceedings of the Third International Conference on Evolvable Systems: From Biology to Hardware, ICES 2000, Lecture Notes in Computer Science, vol. 1801, Springer, Berlin, 2000, pp. 80–89.
[12] F. Gomez, R. Miikkulainen, Incremental evolution of complex general behavior, Adaptive Behavior 5 (1997) 317–342.
[13] S.G. Ficici, R.A. Watson, J.B. Pollack, Embodied evolution: a response to challenges in evolutionary robotics. in: J.L. Wyatt, J. Demiris (Eds.), Proceedings of the Eighth European Workshop on Learning Robots, 1999, p. 1422.
[14] H. Lund, J. Hallman, Evolving sufficient robot controllers, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 1997, pp. 495–499.
[15] N. Jakobi, Running across the reality gap: octopod locomotion evolved in a minimal simulation, in: Proceedings of the First European Workshop on Evolutionary Robotics, EvoRobot'98, 1998.
[16] O. Michel, An artificial life approach for the synthesis of autonomous agents, in: Proceedings of the European Conference on Artificial Evolution, Springer, 1995, pp. 220–231.
[17] S. Nolfi, Adaptation as a more powerful tool than decomposition and integration: experimental evidences from evolutionary robotics, in: Fuzzy Systems Proceedings of the 1998 IEEE Conference on Computational Intelligence, vol. 1, 1998, pp. 141–146.
[18] W. Lee, J. Hallam, H. Lund, Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots, in: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, pp. 495–499.
[19] G. Caprari, K.O. Arras, R. Siegwart, The autonomous miniature robot Alice: from prototypes to applications, in: Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, IROS 2000, October 31–November 5, 2000, pp. 793–798.
[20] R. Grabowski, L.E. Navarro-Serment, C.J.J. Paredis, P. Khosla, Heterogeneous teams of modular robots for mapping and exploration, Autonomous Robots, vol. 8, No. 3, pp. 293–308 (special issue on heterogeneous multirobot systems).
[21] J.A. Driscoll, R.A. Peters II, A development environment for evolutionary robotics, in: Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, 2000, pp. 3841–3845.
[22] F. Southley, F. Karray, Approaching evolutionary robotics through population-based incremental learning, in: Proceedings of the 1999 IEEE Conference on Systems, Man, and Cybernetics, vol. 2, 1999, pp. 710–715.
[23] R.A. Watson, S.G. Ficici, J.B. Pollack, Embodied evolution: embodying an evolutionary algorithm in a population of robots, in: P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala (Eds.), Proceedings of the 1999 Congress on Evolutionary Computation, IEEE Press, 1999, pp. 335–342.
[24] U. Nehmzow, Mobile Robots: A Practical Introduction, Springer-Verlag. ISBN 1-85233-173-9.
[25] K. Swingler, Applying Neural Networks: A Practical Guide, Morgan Kaufmann, San Francisco, CA, 1996. ISBN 0-12-679170-8.
[26] I. Cloete, J.M. Zurada (Eds.), Knowledge-based Neurocomputing, MIT Press, Cambridge, MA, 2000. ISBN 0-262-03274-0.
[27] D.B. Fogel, Evolutionary Computation, Toward a New Philosophy of Machine Intelligence, 2nd Ed., IEEE Press, Piscataway, NJ, 2000.
[28] J. Galeotti, S. Rhody, A. Nelson, E. Grant, G. Lee, EvBots—the design and construction of a mobile robot colony for conducting evolutionary robotic experiments, in: Proceedings of the ISCA 15th International Conference on Computer Applications in Industry and Engineering, CAINE-2002, San Diego, CA, November 7–9, 2002, pp. 86–91.
[29] T. Braunl, Research relevance of mobile robot competitions, IEEE Robotics and Automation Magazine 6 (4) (1999) 32–36.
[30] A. Nelson, E. Grant, G. Lee, Using genetic algorithms to capture behavioral traits exhibited by knowledge based robot agents, in: Proceedings of the ISCA 15th International Conference: Computer Applications in Industry and Engineering, CAINE-2002, San Diego, CA, November 7–9, 2002, pp. 92–97.

Dr. Andrew L. Nelson was born Laramie Wyoming in 1967. He received his B.S. degree from the Evergreen State College in Olympia Washington in 1990. He received his M.S. degree in Electrical Engineering from North Carolina State University in 2000. He received his Ph.D. degree in Electrical Engineering at the Center for Robotics and Intelligent Machines (CRIM) in the department of Computer and Electrical Engineering at North Carolina State University in 2003. His main research interests are in the fields of evolutionary and behavior based robotic control with a particular focus on cooperative multi-robot colonies. Currently, his research is focused on the evolution of artificial neural networks for the control teams of autonomous mobile robots. Before continuing his post baccalaureate education, Andrew Nelson worked as a laboratory technician in Cell and Molecular Biology at the University of North Carolina at chapel and co-authored several research papers in that field. He is currently a Postdoctoral Fellow in CRASAR at the University of South Florida.

Dr. Edward Grant is the Director of the Center for Robotics and Intelligent Machines (CRIM) at North Carolina State University and Associate Professor of Electrical and Computer Engineering. He earned a Bachelor of Science (Hons) in Mechanical Engineering from Dundee College of Technology (now University of Abertay Dundee) in 1969; Master of Engineering in Mechanical Engineering (Fluid Power Control) from the University of Sheffield in 1972; and a Ph.D. in Computer Science from the University of Strathclyde, 2000. Dr. Grant is a Chartered Engineer (C.Eng.), a Fellow of the Institution of Mechanical Engineers (F.I.Mech.E.), and a Senior Member of the Institute of Electrical and Electronics Engineers (S.M.I.E.E.E.). Dr. Grant was the Founding Chairman of the United Kingdom and Republic of Ireland Chapter of the I.E.E.E. Robotics and Automation Society. Dr. Grant is an Associate Editor of the International Journal of Robotics and Autonomous Systems. Dr. Grant's research interests are in the areas of knowledge-based control of robotic systems and dynamic systems, evolutionary robotic systems, machine learning with applications to medical image diagnosis and rehabilitation engineering, e-textiles for smart structures, and search and rescue robots.



John M. Galeotti received his B.S. degree in Electrical and Computer Engineering at North Carolina State University in 2001. He received his M.S. degree in Electrical and Computer Engineering researching in the Center for Robotics and Intelligent Machines (CRIM) in the department of Computer and Electrical Engineering at North Carolina State University in 2002. His research interests during his M.S. Thesis research was in the fields of evolutionary and behavior based robotic control with a particular focus on the design and fabrication of a new class of mobile robot (EvBot) for cooperative multi-robot colonies. Currently, he is pursuing a Ph.D. degree at the Robotics Institute at Carnegie Mellon University. His Ph.D. research is focused on imaging related to biomedical applications.

Stacey W. Rhody received his B.S. degree in Electrical and Computer Engineering from North Carolina State University in 2001. His research in the Center for Robotics and Intelligent Machines (CRIM) was focused on the design and fabrication of a new class of mobile robot (EvBot) for cooperative multi-robot colonies, and on the simulation and visualization of robots in urban scenarios. Currently, he is a serving officer in the United States Marine Corps, Camp Lejune, North Carolina.