Scaling Intelligent Behaviour in the ARBIB Autonomous Robot

R.I. Damper and R.L.B. French

Image, Speech and Intelligent Systems Research Group
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ
UK
[rid|rlbf98r]@ecs.soton.ac.uk

5.4.2001

Abstract

A major concern when building an intelligent robot is: "How can it develop increasingly intelligent behaviours?" This problem is widely recognised, and present research with the ARBIB autonomous robot also addresses this issue. ARBIB consists of a mobile platform that interacts with the world and is controlled by a simulated nervous system of heterogeneous spiking neurons executed on the Hi-NOON neural simulator. Relative to other behaving robots, the neural system is much more biologically-inspired and operates at the level of individual spikes. Rather than using currently-popular reinforcement learning techniques, ARBIB learns from exposure to its environment via low-level mechanisms of habituation, sensitisation and classical conditioning.

In this paper, we describe how ARBIB can scale in complexity in two directions. First, by allowing Hi-NOON to take advantage of distributed computer hardware, ARBIB's nervous system can attain a high degree of complexity aimed at increasing its sensory-motor capabilities. Second, through evolution based on ideas of genetic programming, Hi-NOON is free to develop nervous system architectures whose complexity is no longer governed by initial human design and subsequent intervention. Hence, evolved nervous systems are supported by a simulator architecture that expands to take advantage of additional compute hardware when needed. Client-server architecture schematics of Hi-NOON are given together with an example of a high-bandwidth sensory system running on a server simulator. Nervous system evolution is demonstrated with an obstacle avoidance task which results in emergent wall-following behaviour.

1 Introduction

A major concern when building an intelligent robot is: "How can it develop increasingly intelligent behaviours?" This problem is widely recognised both in robotics (e.g., Brooks 1986; Boden 1994; Nehmzow, Smithers, and McGonigle 1995; Nehmzow 1999, p. 69) and in AI generally (e.g., Boden 1994; Channon and Damper 2000). Present research with the ARBIB autonomous robot (Damper, French, and Scutt 2000) also addresses this issue.

ARBIB consists of a mobile platform that interacts with the world and is controlled by a simulated nervous system of heterogeneous spiking neurons executed on the Hi-NOON neural simulator (Damper, French, and Scutt, forthcoming). Relative to other behaving robots, the neural system is much more biologically-inspired. Rather than using currently-popular reinforcement learning techniques, ARBIB learns from exposure to its environment via low-level mechanisms of habituation, sensitisation and classical conditioning.

Our work is motivated by ideas of 'embodied AI' which hold that possession of a body situated in its environment is a vital part of being a behaving organism (Clark 1987). Further, any such organism

Proc. TIMR 01 - Towards Intelligent Mobile Robots, Manchester 2001.
Technical Report Series, Department of Computer Science, Manchester University, ISSN 1361 - 6161.
Report number UMCS-01-4-1. http://www.cs.man.ac.uk/cstechrep/titles01.html

possesses a nervous system which obviously underlies its intelligent behaviour. Philosophically, then, we align with the *neuron doctrine* of Barlow (1972), according to which behaviour is inextricably linked to neurophysiology. This is the reason for the biological fidelity (spiking neurons and neurobiological learning mechanisms) with which ARBIB's nervous system is modelled and implemented.

In this paper, we describe how ARBIB can scale in complexity in two directions. First, by allowing Hi-NOON to take advantage of distributed computer hardware, ARBIB's nervous system can attain a high degree of complexity aimed at increasing its sensory-motor capabilities. Second, through evolution based on ideas of genetic programming, Hi-NOON is free to develop nervous system architectures whose complexity is no longer governed by initial human design and subsequent intervention.

Hi-NOON was originally written in object-oriented Pascal and used as a tool for studying small systems of spiking neurons (Scutt and Damper 1991, 1992). Since this early work, it has been updated to have a client-server architecture. It is now comprised of two elements: a simulator based on MPI (Message Passing Interface) which distributes computation over client processes; and a non-MPI server simulator intended for sensory and motor processing tasks. Hence, a large nervous system can be broken down into smaller objects that execute concurrently, with the MPI client simulators connecting the server simulators together. This allows the computational hardware that supports ARBIB to scale in line with increasing nervous system complexity.

In previous work, ARBIB's nervous system has been manually-designed and hand-coded before being executed by a Hi-NOON neural simulator during the phase of learning from interaction with its environment. Hence, any scaling of its nervous system towards increasing intelligence is likely to be constrained by its programmer's foresight and prejudices. We address this problem by applying the paradigm of evolutionary computation (Davidor 1991; Koza 1992) to nervous system design.

The remainder of this paper is as follows. In Section 2, we examine the Hi-NOON simulator and its use with large-scale simulated spiking nervous systems, giving an example. Section 3 describes the application of evolutionary computing to nervous system construction and gives an example of a nervous system of spiking neurons evolved to display obstacle avoidance. The evolved nervous system is tested on simulated and real robots. Section 4 concludes with a description of future work.

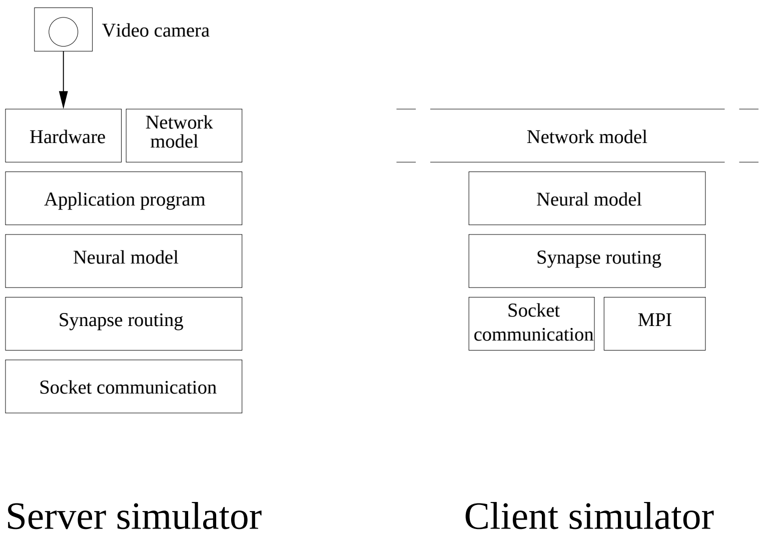
2 Expansion of the Hi-NOON Neural Simulator

In this section, we consider the Hi-NOON neural simulator and how its architecture allows ARBIB to scale-up in sensory-motor complexity. Hi-NOON stands for **Hi**erarchical **N**etwork of **O**bject-**O**riented **N**eurons (Scutt and Damper(1991, 1992);French, Damper, and Scutt 2000; Damper, French, and Scutt, forthcoming). As the name suggests, synapses, neurons and network are represented as dynamically-created objects (equipped with their own data members and member functions) within an object-oriented hierarchy.

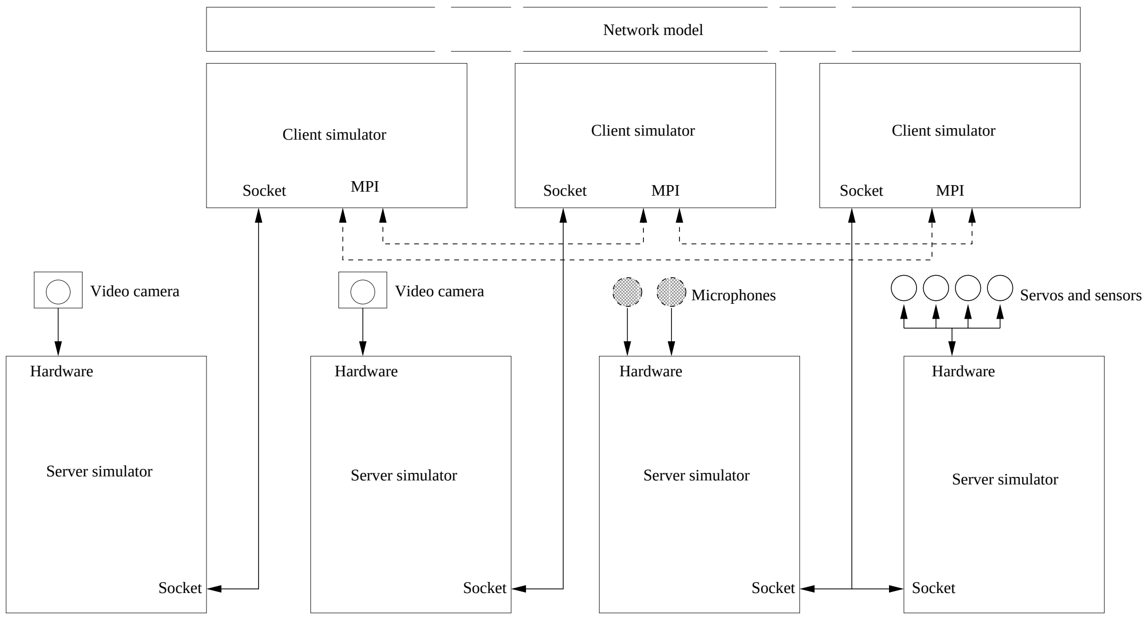
The original program was written in object-oriented Pascal and was used for modelling small systems of biological neurons, notably work with *Aplysia*'s withdrawal reflex (Scutt and Damper 1992). Hi-NOON has subsequently been rewritten in C using the disciplines of object-oriented programming (OOP) (Coad and Yourdon 1991; Eliëns 1994). C was used (rather than C++ with its explicit support of OOP features) to maximise portability among various realisations in different applications. The benefits of the OOP approach are two-fold. First, the ability for objects to inherit properties from other objects means that it is easy to define more physiologically exact neurons in terms of simpler neurons. Thus, the system allows a simple threshold unit as the most basic type of object. More complex objects inherit certain properties from this object (e.g., the fact that it has weighted connections to other objects). The second benefit of OOP is polymorphism. This means that the network may contain many different types of neuron, at many levels of complexity, without the programmer having to be concerned with this.

2.1 Client-server architecture

Hi-NOON comprises two elements (Figure 1): a simulator that uses the Message Passing Interface (MPI) which distributes its simulation over multiple processes, and a non-MPI simulator intended for (but not limited to) sensory and motor processing tasks. By allowing the division of the nervous system model in this way, areas that require special processing resources can be accommodated.



(a)



(b)

Figure 1: The architecture of Hi-NOON. (a) Client and server simulators can be standalone, but are intended to be used together to form a client-server architecture for large neurally-based experimental robot control applications. (b) In such a joint effort, the MPI simulator processes running on a high performance cluster of processors are the clients communicating through sockets to the non-MPI server(s) running on PCs with special purpose input and output (I/O) facilities.

The interconnection between client and server simulators of Hi-NOON is shown in Figure 1. Clients and servers (Cormer and Stevens 1996; Renaud 1996) in this application are very similar, having a common layered structure. The layers in this structure are organised in the following way:

Network model: This describes ARBIB's 'nervous system' that executes on Hi-NOON. The MPI simulator distributes it among all client processes, where each process forms a network object from its portion of the network model. Servers have individual models and share the network layer with hardware specific to their application. It is possible for a server to be "compute-only". That is, it has no hardware interface to the sensory or motor circuits in the robot.

Hardware: This aspect of a server simulator is the interface between the network model and the robot's sensor and motor systems. Examples include serial and parallel communication ports and video digitisers.

Application program: Server simulators that control interface hardware need device driver software. This is provided through the application program. A server that has no hardware interface to robot sensor and motor systems has a very simple application program. In this case, the application program simply calls the simulator code.

Neural model: This contains the algorithms that describe neuron and synapse behaviour.

Synapse routing: This controls the destination of messages that are relayed around the system. For example, a message from a synapse in a server simulator can have a destination that exists within the same simulation, or in the simulation run by the client that the server is connected to (socket-to-socket communication). Communication routing performed by a client is a little more complex. In this case, a message may have a destination that is found in part of the nervous system that exists in the same client process, or in a server (that requires socket-to-socket communication), or indeed on another client process (that requires MPI client-to-client communication.)

Socket communication: Client and server use this to communicate through a socket-to-socket internet connection.

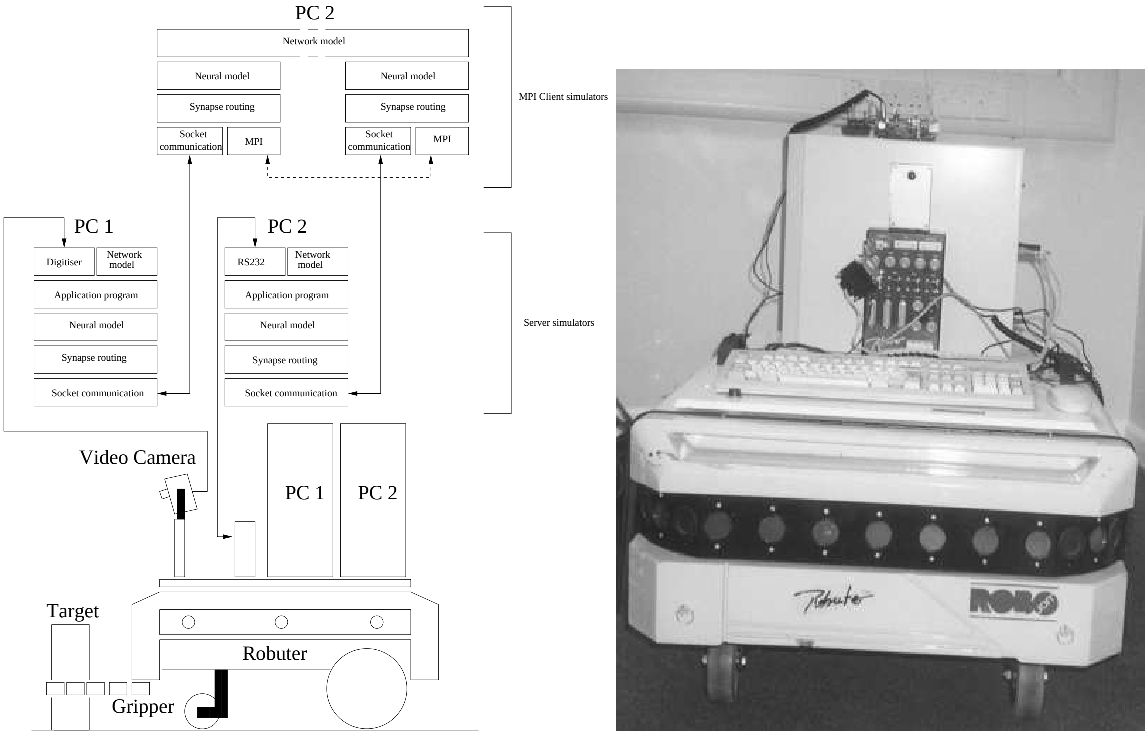
MPI: Client processes use the MPI standard for inter-process communication and synchronisation. Synchronisation is achieved through blocking and is necessary for processes that are running on different CPUs with different loads. Without this safeguard, the different sections of the network model would execute at different rates.

2.2 Example application

This subsection describes related work under development with Hi-NOON and is given as an example of interfacing a complex sense modality to a Hi-NOON server simulator. Figure 2 shows the Hi-NOON client-server architecture applied to a mobile robot (a Robosoft Robuter) equipped with vision and manipulator capabilities. The goal of this application is for the robot to learn the association between the colours of objects in its environment and their electrical conductivity as in the Darwin IV experiment of Verschure, Wray, Sporns, Tononi, and Edelman (1995) (but without the enormous hardware/software complexity of their implementation). Objects that are non-conductive are collected, whereas conductive objects are avoided. Hence, a Darwin IV robot will modify its foraging behaviour based upon its visual perception of objects.

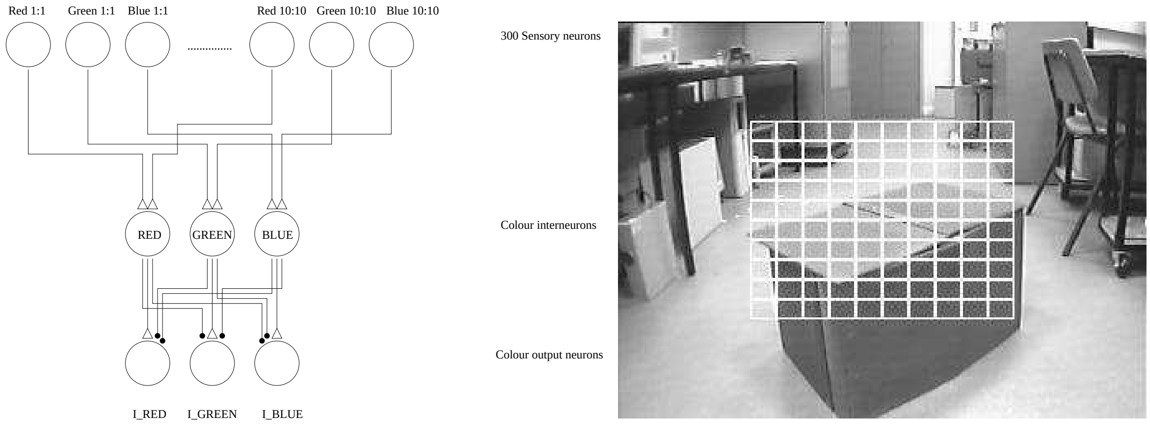
Vision is provided by a server simulator (hosted by `PC 1`) that neurally processes digitised colour images, with robot sonar, motor and manipulator low-bandwidth functions hosted by `PC 2`. MPI simulators are hosted by `PC 2`. Once a frame of video is captured by the frame grabber the image is processed by a simple retina-like structure. The Hi-NOON retina is constructed from an array of (10×10) receptive fields. Each receptive field is inspected by three neurons sensitive to red, green, and blue light. This arrangement results in 300 sensory neurons spread over a grid of 100 receptive fields, shown in Figure 3(a).

Each red sensory neuron makes an excitatory connection with the RED interneuron. The same scheme is followed by the green and blue sensory neurons (which in turn connect to the GREEN and BLUE interneurons, respectively). Inhibitory connections made from the RED interneuron to the I_GREEN and I_BLUE colour output neurons create competition between the colour outputs (with the same reasoning



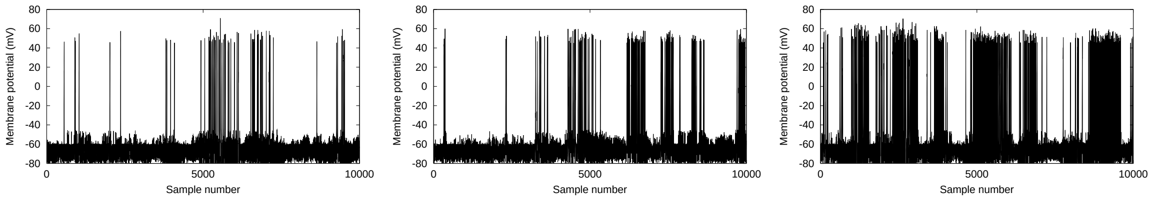
(a) (b)

Figure 2: (a) Replicating the Darwin IV experiment with Hi-NOON. The architecture is to execute on two Linux personal computers: `PC 1` and `PC 2`. Vision is hosted by `PC 1`, whereas `PC 2` hosts MPI and Robuter sensory-motor processing, including control of a compliant gripper used for target-object collection. (b) The Robuter platform in its present stage of development.



(a) (b)

Figure 3: Colour perception of a target-object. (a) A neural network that determines the colour of the image formed on its receptive fields. (b) The first image grabbed by the digitiser of `PC 1` (see Figure 2). The receptive fields have been superimposed to aid presentation.



(a) (b) (c)

Figure 4: Neural activity in the (a) I_RED, (b) I_GREEN, and (c) I_BLUE colour output neurons of Figure 3(a), in response to the image formed on the retina shown in Figure 3(b).

applying to the GREEN and BLUE interneurons). An image taken from the retina of a blue object is shown in Figure 3(b), with an example of circuit activity is shown in Figure 4.

As can be seen in Figure 4(c), the I_BLUE neuron shows the most activity. Hence, this indicates correctly that the image formed on the retina has a greater blue component than red (Figure 4(a)) or green (Figure 4(b)). Bursts of activity from I_RED and I_GREEN neurons are due to other activity taking place over time within the scene observed by the receptive fields.

Because of the concurrent architecture used for this application, only the colour output is projected into the MPI simulators (running on `PC 2`). Hence, high bandwidth processing is restricted to one computer only (`PC 1` with its output projected to `PC 2`), reducing communication bottlenecks between simulators.

3 Evolving a Robot's Nervous System

In this section we consider the evolution of ARBIB's spiking nervous system which is the second thread of our work towards scaling intelligent behaviour in ARBIB. Previously, ARBIB's nervous system was hand-coded and so its architecture was fixed by the imagination and prejudices of its programmer. Because neuroscience has not yet progressed to the state where we understand the relationships between nervous system structure and intelligent behaviour, it seems likely that manual design will limit the potential for the nervous system to scale-up. A possible solution to this problem, which we explore here, is to evolve a nervous system for ARBIB from a population of candidate nervous systems.

3.1 Initial population

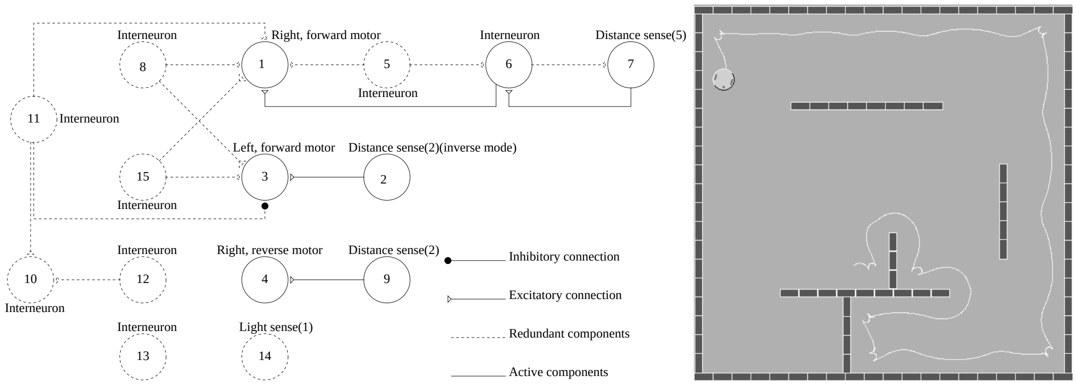
Initially, the evolutionary process must be seeded with a population of randomly-generated individuals (Hi-NOON nervous systems). How are we to do this? The individuals in any new generation (including the initial one) must have a high likelihood of being structurally-viable nervous systems: Thus, it is advantageous to constrain (at least partially) the genetic operators to help achieve this. One way to do this is to ensure that newly-generated neurons of a particular type (e.g., sensory neuron) can only connect to other neurons of a particular type (e.g., interneuron). To maintain diversity, the constraints are sometimes applied and sometimes not, according to a random choice.

Hence, we use the following functions:

add_sensor_to: creates a sensory neuron, at a random sensor position on ARBIB, and connects it to a randomly selected motor or interneuron postsynaptic target.

add_turn: creates an interneuron that carries an identification tag as a 'reflex' interneuron. This interneuron then attempts to connect postsynaptically with motor neurons that connect to the robot's differential drive.

add_reflex: creates an interneuron and connects it with a randomly selected postsynaptic reflex interneuron.



(a) (b)

Figure 5: (a) The evolved nervous system for obstacle avoidance. Note that some components are redundant: they have no sensory input or are disconnected from postsynaptic targets. On removal of redundant components, the circuit divides into three separate sub-assemblies. (b) A screenshot of ARBIB exploring a simulated environment using the evolved nervous system. Wall-following behaviour is clearly present.

add_interneuron: creates an interneuron and connects it to a randomly selected motor, interneuron or reflex postsynaptic target.

add_forward: creates an interneuron and connects it to available forward motor postsynaptic targets.

These functions are used both to generate the initial population and in the creation of new generations during the subsequent evolutionary process itself.

3.2 Selection, recombination and mutation

The selection operator used here is the stochastic universal sampling (SUS) method (Baker 1987; Whitley 1993) with elitism. SUS was selected because it is computationally efficient and maintains diversity among the population.

Recombination is implemented as single-point crossover. Two parents are chosen together with a common neuron (a neuron with a numerical identifier, the 'neuron number', common to both parent nervous systems) which acts as a crossover point. Two children are instantiated who then undergo pruning of any synapses that no longer have valid postsynaptic targets. This method has been chosen in preference to any attempt at analysing the two parents looking for an ideal crossover point in the interests of computational efficiency.

Mutation operates by adding or deleting individual neurons and synapses. A random choice is made as to whether to apply the functions listed in Section 3.1 or not. Because the Hi-NOON data structures are dynamic, and because we manipulate the nervous system representation directly during the evolutionary process, there is no limit on the size of genotype in this application. This removes the limit placed on nervous system complexity with a fixed length genotype typical of genetic algorithms (Koza 1992, p. 18) without neurogenesis. Hence, we see this development as being closer to the paradigm of genetic programming than genetic algorithms, with Hi-NOON viewed as a virtual machine.

3.3 Evolved Behaviour

In this section, we examine results obtained by applying the evolutionary paradigm to nervous system development for the Khepera instantiation of ARBIB in simulated and real environments. The primary task for the robot was obstacle avoidance and was specified in the robot's fitness function by causing a penalty to be incurred by the robot for collisions with obstacles and a reward given for forward motion.

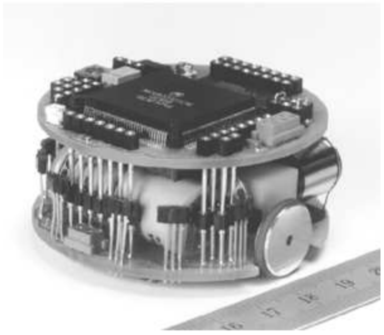


Figure 6: The Khepera robot used for the real-world test.

Figure 5(a) shows the best nervous system found in the first of 10 separate runs. It was evolved from a population of 100 individuals over 100 generations (on a Pentium 300 Linux PC). It has several redundant components that perform no useful function here: They are remnants or 'leftovers' from the evolutionary process. Had evolution continued longer, it is possible that these remnants might eventually have played a part in robot behaviour, i.e., they could serve as preadaptations for some as yet undiscovered functionality.

The nervous system has organised itself into three separate parts:

1. neurons 1, 6 and 7: form a sub-assembly that fires the right forward motor when a wall is detected by distance sensor 5 on Khepera (a wall on the robot's right hand side);
2. neurons 2 and 3: these fire the left forward motor in the absence of an obstacle in front of the robot;
3. neurons 4 and 9: these fire the right reverse motor in the presence of an obstacle in front of the robot.

Figure 5(b) shows the path taken by ARBIB (with the final evolved nervous system from the first run) in a test environment. It is quite clear that it shows wall-following behaviour. Interestingly, 7 out of 10 separate runs also resulted in wall-following behaviour.

How different is this 'nervous system' from one that we might have designed manually? First and foremost, it is considerably simpler. The evolutionary solution has just 7 neurons, whereas our earlier, manually-designed solutions had some 30-50 neurons. Our manual designs have used central pattern generator circuits to give ARBIB a drive to explore its environment, together with a hardwired obstacle-avoidance reflex which was modified by learning (classical conditioning). Here, however, there is no ontogenetic learning so, by focusing crudely on neuron count, we are not strictly comparing like with like. Also, the division of the evolved solution into three separate sub-assemblies was interesting. All our previous designs featured a single, complete neural circuit.

3.4 Real test environment

Results from the simulations were verified using a real Khepera robot shown in Figure 6. No changes to the evolved nervous system of spiking neurons were made.

A Khepera robot in mode 3 (controlled by serial link protocol at 38400 Baud) was placed into a test environment (Figure 7), facing away from the nearest wall. The Khepera graphical user interface (GUI) of Michel (1996) was used as the front-end for Hi-NOON, and was loaded (and executed on) an 80486 Linux PC.

After selection of the real Khepera (through the GUI), ARBIB started to search for a wall. If nothing was in view, it rotated until a wall was seen. It then approached the wall, oriented itself parallel to it, and moved forwards. On encountering a corner, it adjusted its position/orientation and continued along the new wall. On one occasion, a new object (a black coffee jar) was introduced into the environment and directly in ARBIB's path. The robot manoeuvred around the object until its top circuit board hit the jar, whereupon it got stuck. At this point, the second author intervened and unjammed the robot,

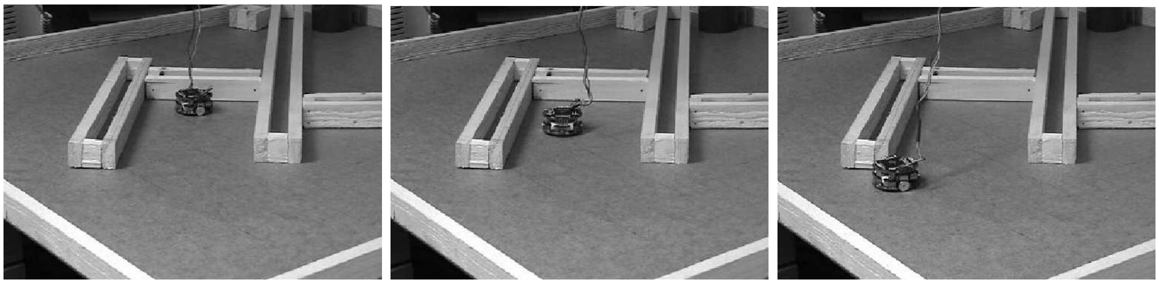


Figure 7: Arbib negotiates a test environment using the evolved spiking nervous system.

which then negotiated around the jar, found a wall and continued its journey around the periphery of the environment.

Hence, a system of spiking neurons evolved for a simulated Khepera transfers very well to the real world. This robustness probably relies largely on the sensory and motor noise which is deliberately introduced as a feature of the Khepera simulator (cf., Chongstitvatana 1999).

4 Conclusion

In this paper, we have outlined two methods by which the Arbib autonomous robot can scale-up in complexity: (1) By an expandable client-server architecture for the Hi-noon neural simulator, and (2) by evolving Arbib's nervous system.

Example applications of the client-server architecture and evolutionary development of a nervous system are given. The client-server architecture is demonstrated with the problem of interfacing a high-bandwidth sense modality to a mobile robot, and the evolutionary development of a nervous system is demonstrated with an obstacle avoidance competence on simulated and real robots.

The final stage of this work is to combine learning in the robot with the evolution of its nervous system. Hence, two levels of adaptation will have been demonstrated with Arbib: ontogeny (the development of an individual during its 'life') and phylogeny (the development over evolutionary time). Future research will concentrate on the addition of more complex sensory-motor systems to Arbib and will start by replicating the Darwin IV experiment of Verschure et al. (1995).

References

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In J. J. Grenfenstett (Ed.), *Genetic Algorithms and their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms*, Hillsdale, NJ, pp. 14–21. Lawrence Erlbaum Associates.

Barlow, H. B. (1972). Single units and sensation: A neuron doctrine for perceptual psychology? *Perception 1*, 377–394.

Boden, M. A. (1994). New breakthroughs or dead-ends? *Philosophical Transactions of the Royal Society of London (Series A) 349*, 1–13.

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

Channon, A. D. and R. I. Damper (2000). Towards the evolutionary emergence of increasingly complex advantageous behaviours. *International Journal of Systems Science 31*(7), 843–860.

Chongstitvatana, P. (1999). Using perturbation to improve robustness of solutions generated by genetic programming for robot learning. *Journal of Circuits, Systems, and Computers 9*(1–2), 133–143.

Clark, A. (1987). Being there: Why implementation matters to cognitive science. *Artificial Intelligence Review 1*, 231–244.

Coad, P. and E. Yourdon (1991). *Object Oriented Analysis*. Englewood Cliffs, NJ: Prentice-Hall. Second Edition.

Cormer, D. E. and D. Stevens (1996). *Internetworking with TCP/IP Volume 3: Client-Server Programming and Applications*. Englewood Cliffs, NJ: Prentice Hall.

Damper, R. I., R. L. B. French, and T. W. Scutt (2000). ARBIB: an autonomous robot based on inspirations from biology. *Robotics and Autonomous Systems 31*(4), 247–274.

Damper, R. I., R. L. B. French, and T. W. Scutt (forthcoming). The Hi-NOON neural simulator and its applications. *The Microelectronics Journal*, in press.

Davidor, Y. (1991). *Genetic Algorithms and Robotics.* Singapore: World Scientific Press.

Eliëns, A. (1994). *Principles of Object-Oriented Software Development.* Wokingham, UK: Addison-Wesley.

French, R. L. B., R. I. Damper, and T. W. Scutt (2000). The Hi-NOON neural simulator and its applications to animal, animat and humanoid studies. *First IEEE-RAS International Conference on Humanoid Robots*, Boston, MA. (No pagination; Proceedings on CD-ROM).

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

Michel, O. (March 1, 1996). Khepera Simulator version 2, User Manual. `http://diwww.epfl.ch/Khepera/`.

Nehmzow, U. (1999). *Mobile Robotics: A Practical Introduction.* Berlin: Springer-Verlag.

Nehmzow, U., T. Smithers, and B. McGonigle (1995). Increasing behavioural repertoire in a mobile robot. In J.-A. Meyer, H. Roitblat, and S. W. Wilson (Eds.), *From Animals to Animats 2: Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, pp. 291–297. Cambridge, MA: Bradford Books/MIT Press.

Renaud, P. (1996). *Introduction to Client/Server Systems.* New York, NY: John Wiley.

Scutt, T. W. and R. I. Damper (1991). Computational modelling of learning and behaviour in small neuronal systems. In *Proceedings of International Joint Conference on Neural Networks*, Singapore, pp. 430–435.

Scutt, T. W. and R. I. Damper (1992). Object-oriented modelling of small neuronal systems. *Artificial Intelligence and Simulation of Behaviour Quarterly 80*, 24–33.

Verschure, P. F. M. J., J. Wray, O. Sporns, G. Tononi, and G. M. Edelman (1995). Multilevel analysis of classical conditioning in a behaving real world artifact. *Robotics and Autonomous Systems 16*, 247–265.

Whitley, D. (1993). A genetic algorithm tutorial. *Statistics and Computing 2*(4), 65–85.