A Dynamic Net for Robot Control

Bridget Hallam, John Hallam and Gillian Hayes
<bridget@aifh.ed.ac.uk> <john@aifh.ed.ac.uk> <gmh@aifh.ed.ac.uk>

Department of Artificial Intelligence,
The University of Edinburgh,
5 Forrest Hill,
Edinburgh EH1 2QL,
SCOTLAND

Fax: 031-650-6899
Phone: 031-650-1000
or direct dial: 031-650-3080
Telex: 727442 (UNIVED G)

Keywords: learning, mobile robot, behavioural model, temporal net.

Abstract

This paper describes ongoing work to assess the appropriateness of using Halperin's Neuro-Connector model, originally devised to explain fish behaviour, to control a robot. The model is described in mathematical detail and suggestions for improving computational efficiency are made. The operation of the model and the information necessary for initialisation are also discussed. Preliminary experiments on a mobile robot platform are described and some simulation results presented.

The Neuro-Connector model is computationally expensive and requires careful initialisation, so is not suitable for standard robot tasks. However, it can tolerate a designer's imprecise understanding of sensory states, refining the stimuli used to trigger each behaviour until only specific and reliable ones are considered, and can tolerate changes in the environment and in its own sensor system. It can be trained as much as programmed and can be instructed whilst operational. This is feasible because it learns in only four stimulus presentations. Therefore this model warrants further investigation with particular reference to controlling sensor-rich robots in complex and changing environments, especially where the task requires fine sensory discriminations to be made.

There has always been an interest in building humanoid robots, both in science fiction and amongst serious scientists. In 1950 Turing said, "It can [also] be maintained that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child." Since attempts to build such a machine have so far met with rather limited success there has been a move away from modelling humans and towards modelling less complicated animals, often insects (Brooks, 1985) (Brooks, 1989) (Beer, 1990) (Beer et al., 1992) (Webb, 1993) (or, for a toad simulation, (Arbib, 1987)). Success in this field has led researchers to become more ambitious again — Brooks now plans to attempt a humanoid robot with human-like intelligence and perhaps even consciousness (Brooks and Stein, 1993).

We believe that there are several intermediate stages to go through before a humanoid robot can become a reality. There are problems in dealing with a rich environment, because the salient information exists within a profusion of sensory data. There are problems in dealing with a constantly changing environment in real time because there is sometimes little time to think, and new events often require an appropriate response even though there have been few training examples. And there are problems in programming a robot which doesn't see things the way that we do, because it is hard to imagine what will turn out to be the salient features once the robot has enough sensors to be interesting. All these problems are being tackled, but we feel it would be more appropriate to attempt to build some other vertebrate as an intermediate stage before moving on to anything quite as complicated as humans.

In this paper we concentrate on the problem of controlling a physical, autonomous, sensor-rich, agent in an unstructured, changing, environment. Most animals have already solved this problem with great success, therefore we chose to investigate a model of animal behaviour. Halperin's Neuro-Connector model of animal learning and motivation is unusually well specified for an ethological model. It was proposed as a possible mechanism for various aggressive behaviour effects of Siamese fighting fish, and has been shown to have good explanatory and predictive powers in that context (Halperin et al., 1992) (Halperin, 1990).

This paper describes the model chosen in some detail and reports some preliminary experimentation. Section 1 sets the model in context, and talks about the types of tasks we are envisaging. Section 2 is the model description. Section 3 describes a Neuro-Connector net in operation and talks about the information required for initialising such a net. Section 4 describes experiments done on a real robot in order to ensure that there were no essential underlying biological processes which were necessary for the correct operation of the model. Section 5 describes simulation experiments exploring the limits of parameter settings. All these results are pulled together in Section 6, where the implications of using the Neuro-Connector model for controlling a mobile robot or artificial animal are considered. A glossary is provided at the end of this paper to explain some of the more specialist terms.

1 Background

Each type of robot controller architecture has its own strengths and weaknesses, making it better for some types of task and worse at others. In this section we talk about the type of task we are trying to tackle, and give a brief overview of the main approaches to robot control and why they are not appropriate for these tasks.

1.1 Tasks

The Neuro-Connector model is being proposed as a controller for tasks requiring a sensor-rich robot to make fine discrimination between similar sensory states where irrelevant details can have any value. It is assumed that it would be extremely hard for a human designer to envisage the details of the relevant sensory discriminations required. We expect versatile behaviour: the ability to learn causal relationships, the ability to disambiguate similar items, the ability to choose between different actions, the ability to decide that an action is not successful and to attempt to replace it with another. We also expect the robot to survive for long periods of time (at least weeks) without qualified human intervention.

Examples of tasks which contain the requirement for different actions to be performed in similar circumstances are:

- survival. One difficult discriminatory task involved in survival is the ability to distinguish friendly humans from unfriendly ones.
- a robot guide in a museum, where the robot needs to notice when people are ready before beginning its next speech or moving on.
- a litter collector in a playground or car park, where the difference between litter, painted lines, and flowers is not always clear, and glass may be embedded in the tarmac.
- a fire watcher in a forest, which needs to avoid boggy places and to decide what degree of assistance is needed when a fire is found.
- a species counter for some ethological project, where similar species must be disambiguated despite movement and non-standard poses.

1.2 Robot Controller Architectures

Control engineers normally concentrate on achieving accurate, stable, reliable, and repeatable position control without damage to robot or surroundings. None of the tasks described above involves co-ordinate position control, which makes standard control engineering techniques inapplicable to our problem.

Traditionally people in Artificial Intelligence have developed robot controllers in computer simulation using a model of the world to provide simulated sensor input. While this can work well if the robot and its interactions with the environment are simple, it is very hard to model realistic robots in this way. The problem of maintaining an accurate world model is intellectually hard due to the need for accurate modelling of real sensors, the need to combine sensor information which has different error characteristics, and the need for accurate simulation of the robot's working environment. The problem is also computationally expensive, since the world continues at its own pace, and that is without any effort spent on the problem of how the robot should be controlled! Solutions which do not preclude real robots altogether mostly involve changing the representation of the world (*e.g.* Chapman and Agre's deictic representation (1991)),, focussing attention so that only relevant pieces of world model are constructed (*e.g.* (Ballard, 1991), (Arkin, 1991)), or removing the world model altogether.

The subsumption architecture proposed by Brooks (1986) is most extreme about not using representations of the world. Many interesting robots have since been built using this approach (for example, see the review of MIT robots in (Brooks, 1990)), but experience shows that programming for emergent functionality is difficult. The subsumption architecture does not, in any case, allow for the robot to learn as a result of experience. This is an essential component of all the tasks outlined above.

Reinforcement learning is capable of dealing with sensor-rich robots in real environments, but the reinforcement algorithm is often non-intuitive and sufficiently difficult to generate that some reinforcement algorithms are generated by genetic algorithms (see, for example, Dorigo and Schnepf (1993)). Human supervised learning is impractical for robots with a long survival time; unsupervised learning is too slow, requiring too many presentations of the stimulus (see Sutton (1990)). The Neuro-Connector model uses semi-supervised reinforcement learning, where feedback comes from the environment.

Many roboticists interested in the control of real mobile robots have used some sort of neural net for the robot 'brain'. Some of these nets were also developed by genetic algorithm, for example the Hopfield net used by Beer (1990) ; others such as the 'Really Useful Robot' controller (Nehmzow and Smithers, 1990) used a self-organising net. Nets produced by genetic algorithm techniques are again evolved in simulation, running into some of the problems described for the world model. This has the advantage that genetic algorithm researchers are having to start building real robots themselves in order to check that the results of their techniques are still realistic *e.g.* (Husbands et al., 1993). Some of the nets used to control real robots use nets of sufficient complexity that they need a supercomputer to do even simple tasks. Others have no concept of real time, relying on program iterations to give them some idea of time and sequence. One of the few neural networks which are reasonable candidates for the task of controlling sensor-rich mobile robots for the type of task outlined above include Grossberg's ART networks (papers collected in (Carpenter and Grossberg, 1991)). ART-1 (Carpenter and Grossberg, 1987) has been used by Baloch and Waxman (1991) to control their visually guided mobile robot MAVIN.

The Neuro-Connector model incorporates real time, is intelligible enough to be designed rather than evolved, and requires sufficiently small amounts of processing power that a mobile robot doing simple tasks can run this net while carrying its own power supply. It was developed to explain the behaviour of sensor-rich agents operating in a real and changing environment, concentrating on management of the correlation between sensory state and required behaviour. As such it is very relevant as a potential solution to the problems of controlling sensor-rich mobile robots in complex discriminatory tasks of the type outlined above.

2 The Neuro-Connector Model of Learning and Motivation

The Neuro-Connector model was proposed (Halperin, 1990) in order to explain various learning phenomena from animal literature. It succeeds in explaining several, including stimulus-response conditioning and the controversial 'backward' conditioning. Extensions to the model, also described in Halperin's thesis, explain the more classical stimulus-stimulus conditioning and allow for motivational input, the effect of hormones and more deliberate behaviour. Some animal learning phenomena such as blocking are not explained by this model.

Any model of the control of animal behaviour is a potentially interesting candidate for the control of sensor-rich robots, but many such models and theories are mathematically poorly specified and thus unimplementable without major revision. Halperin's proposal for the mathematical basis of the Neuro-Connector model is well specified and is given in this section.

The Neuro-Connector model is a connectionist net with unidirectional synapses. Nodes have memory of recent input and recent activity. Connections have plastic weights which adjust according to the temporal firing patterns in the connected nodes. Although nodes are called neurons and the connections synapses, the level of modelling is in fact far above biophysical reality.

The model assumes that the agent has a predefined set of low-level behaviours available to it, *i.e.* that our robot has a set of preprogrammed behaviours which are the actions we want it to display. These are not invariant since they may depend on sensor readings (*e.g.* 'orient towards the noise', 'go to nearest remembered dark place' rather than just 'turn right 90°'). These are in a similar class to animal fixed action patterns (FAPs), except that FAPs are rarely interrupted.

The model also assumes plentiful sensory data and that this data is organised into meaningful and overlapping units. This means that our robot must have at least sufficient sensory states to reliably distinguish whatever the robot needs to disambiguate, and that sufficient sensor preprocessing must be done that, for each behaviour, at least one 'neuron' will fire *only* when that behaviour is appropriate. We do not assume that an external observer such as ourselves knows which neuron is associated with which behaviour.

This section describes the model topology, the components, and the weight-change functions. Some discussion of simplifications leading to better computational efficiency follows. In Section 3 the operation of a Neuro-Connector net and the knowledge necessary for initialisation are described.

2.1 Topology

Three main types of neurons are distinguished in the Neuro-Connector model — sensory (S), releaser (R), and behaviour (B). These neurons are organised into three layers (one for each neuron type) with adjustable-weight connections between the S and R layers, fixed-weight feedback connections between the R and B layers, and fixed-weight inhibitory connections between B neurons, as illustrated in Figure 1.

The structure illustrated in Figure 1 is only part of the net. For most implementations we expect at least several tens of B neurons, each with its own R connected to hundreds of S neurons. A Neuro-Connector net is large, with an S neuron for every sensory state that the robot might possibly be required to disambiguate.

Sensor input is pre-processed to various degrees and is presented (as a sensor value or, more commonly, as logical combinations of sensor or perceptual propositions *e.g.* 'front infra-red on and bumper off' or 'I can see a cup but not a bottle') to S neurons. The amount and temporal sequence of each type of input is used to determine which S neurons should fire. Output from S neurons is combined with information encoded in the current synapse weights, reflecting the previous experience of the agent, and passed to the R neurons. It is these S–R synapses which have adjustable weights and so allow the net to learn.

When a releaser neuron accumulates sufficient input to fire, its behaviour neuron is stimulated, causing some code to run and the robot to act. The B neuron will only fire if the stimulation it receives is sufficient to overcome inhibition from the B neuron currently active. B neurons inhibit each other so that only one behaviour happens at a time; thus it is here that behaviour selection occurs.

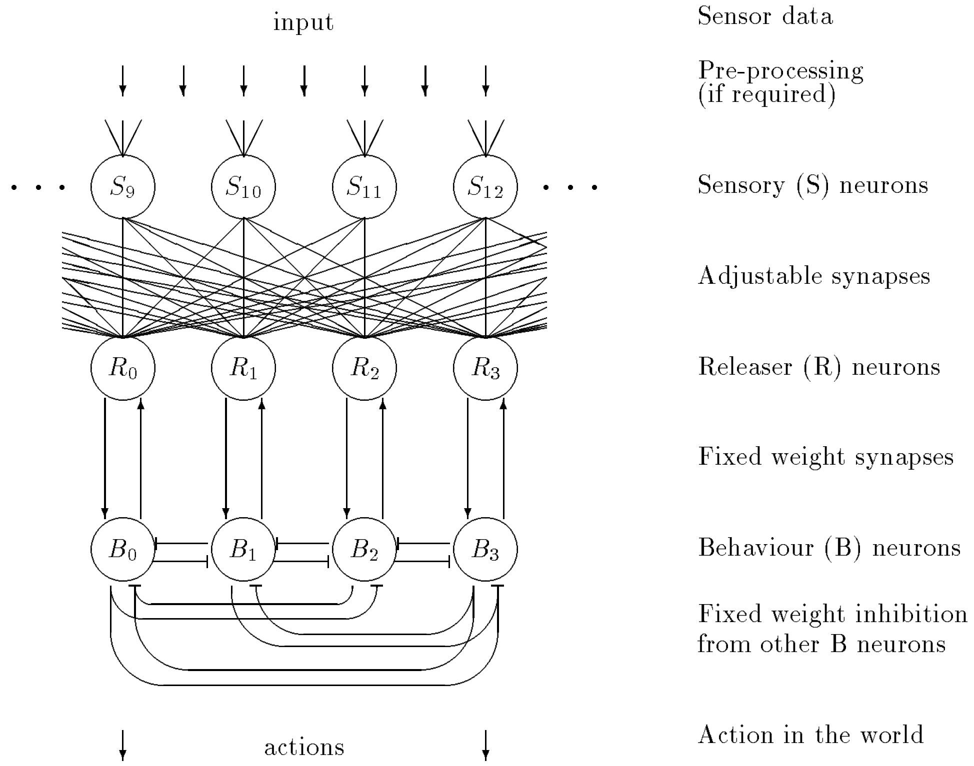

Figure 1: Part of a Neuro-Connector Net

2.2 Neurons

The neurons used in the model, although functionally different, have much the same internal structure, shown diagrammatically in Figure 2. Each neuron computes the sum of the inputs received from its predecessors, weighted by the interconnecting synapses, and compares that value with an input threshold. The difference between the two values becomes the target value of a leaky integrator which computes the value of the neuron's input register $i(t)$ using the differential equation

$$\mathcal{T}_{in}\frac{di}{dt} + i \;=\; \text{target}. \tag{1}$$

(These leaky integrators are also known as decaying average processes, first order lag systems, or relative average smoothing units.) The time constant $\mathcal{T}_{in}$ of the process determines the length of time over which recent input is remembered. Two values are normally specified: one for the situation where $i(t)$ is decreasing and one for the case where it is increasing. This allows variation in the attack-decay times of neurons to be modelled. These time constants are often referred to as 'up gain' and 'down gain', respectively.

The current value in the input register is compared with a time-varying threshold to determine the status of the neuron. In Halperin's proposal, the neuron is binary so its current output is one if the input register exceeds the threshold and zero if not. Outputs are passed to other neurons through the interconnecting synapses, or signal (in B neurons) the behaviour-generating processes in the robot controller to cause particular behaviour to occur.

The neuron also maintains a record of its recent activity in the form of a decaying average of recent outputs. This value, held in the activity register, varies between zero and one — it is zero when the neuron has been off for a significant period of time relative to the activity register integrator time constant, and one when the neuron has been on for an equivalent long time. The time constant $\mathcal{T}_{act}$ is a constant parameter of the neuron; separate values for increasing and for decreasing activity may be set.

The threshold determines the stimulus level necessary for the neuron to become active. In general it is variable, falling when the neuron has recently been quiescent and rising when the

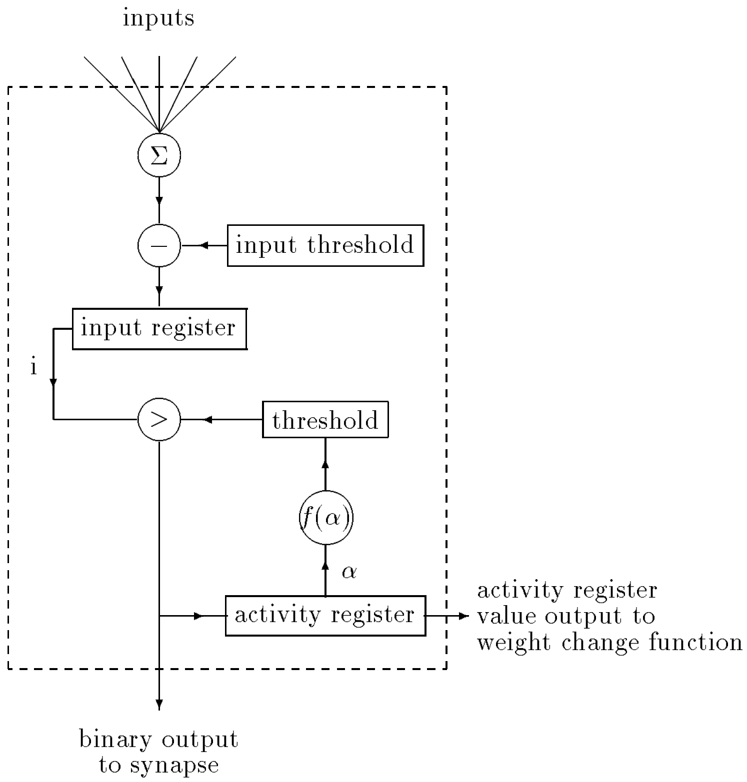

Figure 2: Inside a Neuron

neuron is active. It is thus related to the activity register value, though Halperin (1990) does not propose a candidate function. Note, however, that for binary neurons the relationship cannot be a continuous function as the closed loop within the neuron would then generate oscillations — the threshold generation function must incorporate hysteresis to prevent this.

For the experiments described here the threshold is switched to a constant high value if the neuron is on continuously for a certain period of time, thus switching the neuron off and implementing the 'adaptation' time. The threshold reverts to a constant low value a fixed period after the neuron adapts out, implementing refractory behaviour.

2.3 Synapses

Synapses consist of a weight, a note of whether this weight should be adjustable, and a record of which is their pre- and post- synaptic neuron. Each S–R synapse also has as a parameter the expected time difference, t_{exp}, between its S neuron finishing firing due to it adapting out or to the stimulus disappearing, and its R neuron finishing firing due to the successful completion of the behaviour released by that R. They also keep track of the actual difference in firing, t_{obs}, between their S and R neurons by monitoring the current activity register value in their R neuron and the activity register value of the S neuron t_{exp} seconds ago. The expected time t_{exp} is set such that when a behaviour is successful there should be excellent correlation between observed and expected differences. When a behaviour is unsuccessful the B neuron and therefore its R neuron generally stay on too long, so correlation is not so good. Correlation is also bad if either neuron fails to fire.

Most excitatory S–R synapses have adjustable or plastic weights which are initialised to low values; however, some S–R synapse weights are initialised high or even fixed during setting up in order to provide default behaviours under given circumstances. Fixed-weight synapses are normally initialised to high values and are used for inhibitory connections, the R–B feedback loop and some essential reflexes.

Adjustable weights can be fixed later if it is desirable for the 'learning phase' to finish, but

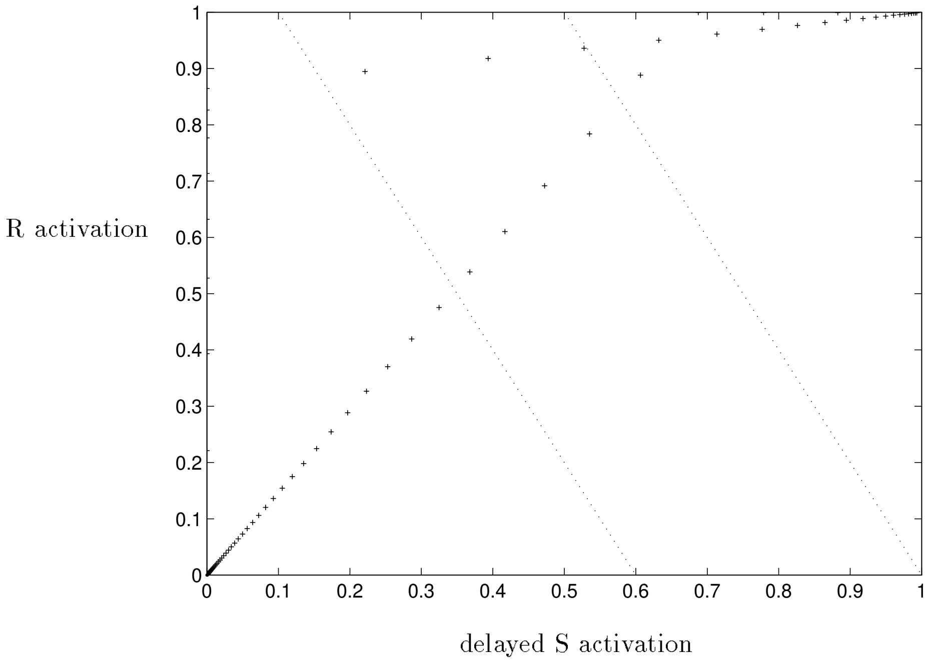

Figure 3: S and R Activations Plotted on a Plan of the Coactivation Surface of Figure 4 showing the Command Trajectory for Figure 6. The trajectory goes up the S axis while only R is on, then across to (1,1) when the delayed S activation increases, starts back along R=1 after S turns off and then heads straight to the origin once both neurons are off. The dotted lines mark the edges of the ditch on the surface of Figure 4 caused by the function changes in Equation 3. The left-hand line is where the minimum value of $z = -2.08$ is returned, and the right-hand line returns zero.

normally the agent learns continuously.

2.4 Weight Adjustment

The weights of the plastic S–R synapses are adjusted continuously by the synaptic dynamics described below as a function of the current activity of the post-synaptic (R) neuron and the activity of the pre-synaptic (S) neuron t_{exp} ago. In addition to the synaptic weight, each plastic synapse maintains an additional register, the *weight change command* register (or simply the command register), which controls the direction and extent of weight changes. The synapse dynamics ensures that significant weight changes take place only when both neurons have been inactive for a certain period, implementing Sinclair's 'Rest Principle' (1981).

The input to the weight change process is the pair of activity register values, $\alpha_S(t - t_{exp})$ and $\alpha_R(t)$, which together define a point in a unit square. As the network operates, the point moves around in response to the patterns of activation of the pre- and post-synaptic neurons, as can be seen in Figure 3. The neural dynamics are such that the point always moves toward one corner of the square — the corner specified by the outputs of the two neurons (the S neuron output being taken from t_{exp} ago).

Command register values are computed by the synapse using a coactivation function. This function assigns to each point on the input square a carefully chosen value designed to express the correlation between pre- and post-synaptic activity summarised by that point. As the input

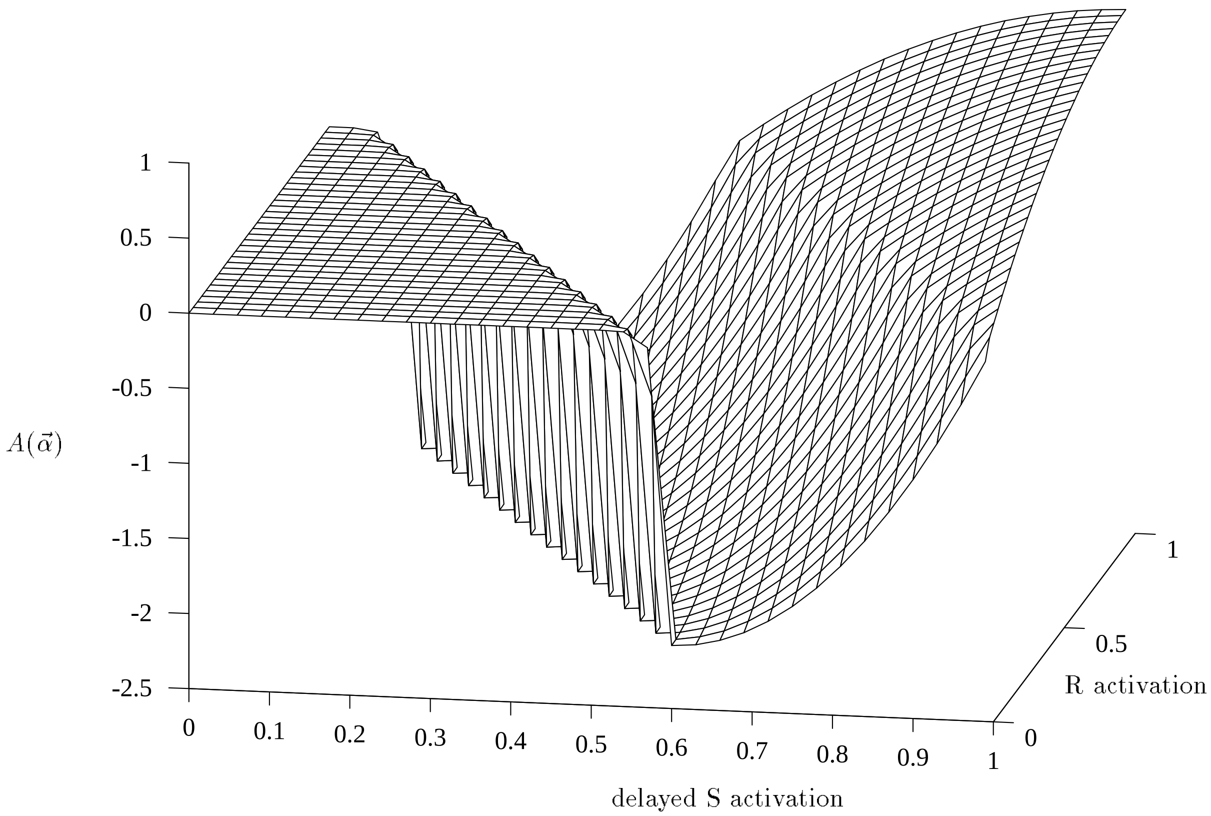

Figure 4: The Control Surface used for Synapse Weight Adjustment $A(\vec{\alpha})$

point moves, describing a trajectory known as the command trajectory in the input space, a leaky integrator process computes the command register value from the varying coactivations corresponding to the moving input point. After both neurons have turned off this trajectory heads towards the origin of the square as both activity register values decay towards zero. Consolidation of learning, in the form of significant weight change, takes place only when the point is within a short distance of the square origin. The coactivation function proposed by Halperin (1990) and illustrated in Figure 4 is designed to realise the following intuitions.

1. If S and R are well correlated then the coactivation function should be positive sufficiently long that the command register value becomes strongly positive. Since the degree of correlation is measured entirely by the values in the activity registers it is biased heavily towards recent activity. Thus when the neurons have been off long enough for the synapse weight to be affected it is the ends of firing in the two neurons which is most significant, and the down gains in the activity registers.
2. If the R goes on without the S the command register value should be near zero. This is so that the same behaviour, *e.g.* 'flee', can be triggered by different stimuli.
3. If the S goes on without the R the command register value should be negative, since it this would indicate that the current situation is unrelated to this R's behaviour.
4. If both neurons go on but are not well correlated then the command register value should be negative.

In formal terms, suppose the input point $(\alpha_S(t - t_{exp}), \alpha_R(t))$ is denoted by $\vec{\alpha}(t)$. The coacti-

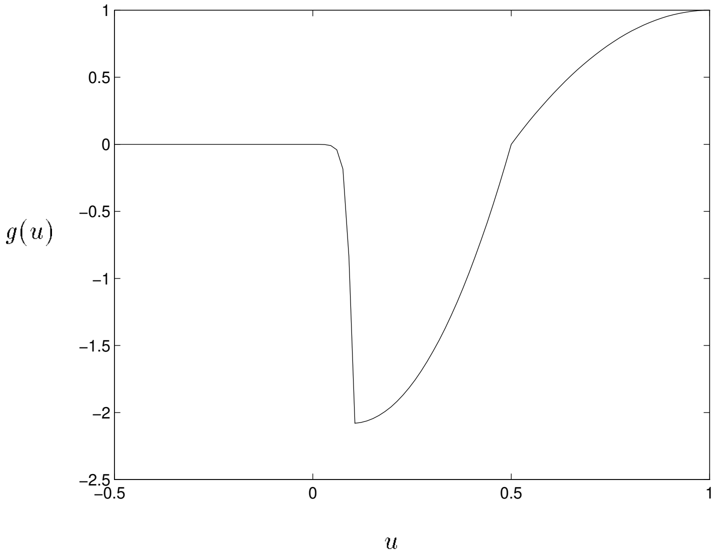

Figure 5: The Profile $g(u)$ which Generates the Coactivation Surface of Figure 4

vation surface $A(\vec{\alpha})$, depicted in Figure 4, is given by

$$A(\vec{\alpha}) \;=\; g\left(\frac{1}{2}(\vec{\alpha}\cdot\vec{u}-1)\right) \tag{2}$$

where $\vec{u}$ is $(2,1)$ and $g(\cdot)$ is the function

$$g(u) \;=\; \begin{cases} -2.08e^{100(u-0.1)} & \text{if } \; u < 0.1 \\ 13(u+0.3)(u-0.5) & \text{if } \; 0.1 <= u <= 0.5 \\ -4(u-0.5)(u-1.5) & \text{if } u > 0.5 \end{cases} \tag{3}$$

The coactivation surface is thus generated by sliding the profile of $g(\cdot)$, shown in Figure 5, along the lines of constant $\vec{\alpha}\cdot\vec{u}$. The command register value $c(t)$ is computed by a leaky integrator process

$$\mathcal{T}_{cmd}\frac{dc}{dt} \;+\; c \;=\; A(\vec{\alpha}(t)) \tag{4}$$

and the synapse weight is updated using the somewhat different process

$$\mathcal{T}_{wt}\frac{dw}{dt} \;=\; \chi(\vec{\alpha})|c|(w_{tgt}-w) \tag{5}$$

in which $\chi(\vec{\alpha})$ implements the consolidation region outwith which weight change is suppressed and w_{tgt} is typically one when c is positive and zero when c is negative. Notice that the command value is used as a rate constant in this integrator process, rather than a target value. In Halperin's proposal the consolidation gating function designed to prevent weight changes whenever neurons have recently been active is given by

$$\chi(\vec{\alpha}) \;=\; (1-\alpha_S)^{30}(1-\alpha_R)^{30} \tag{6}$$

and is effectively zero when either activity register exceeds about 0.1.

In summary, then, the operation of the net induces variation in the activity of the neurons that can be visualised as a collection of trajectories on the unit square, one trajectory per synapse. Each synapse integrates the instantaneous coactivation value along the activity inputs' trajectory and, whenever the input passes into the consolidation region, adjusts its weight according to the update rule above. This process is illustrated in Figures 6 which depicts graphically the values of the neuron activities (Figure 6a), the instantaneous coactivation (Figure 6b), the command register value and the consolidation function (Figure 6c), and the weight itself (Figure 6d). It is the command trajectory for this experiment which was illustrated in Figure 3.

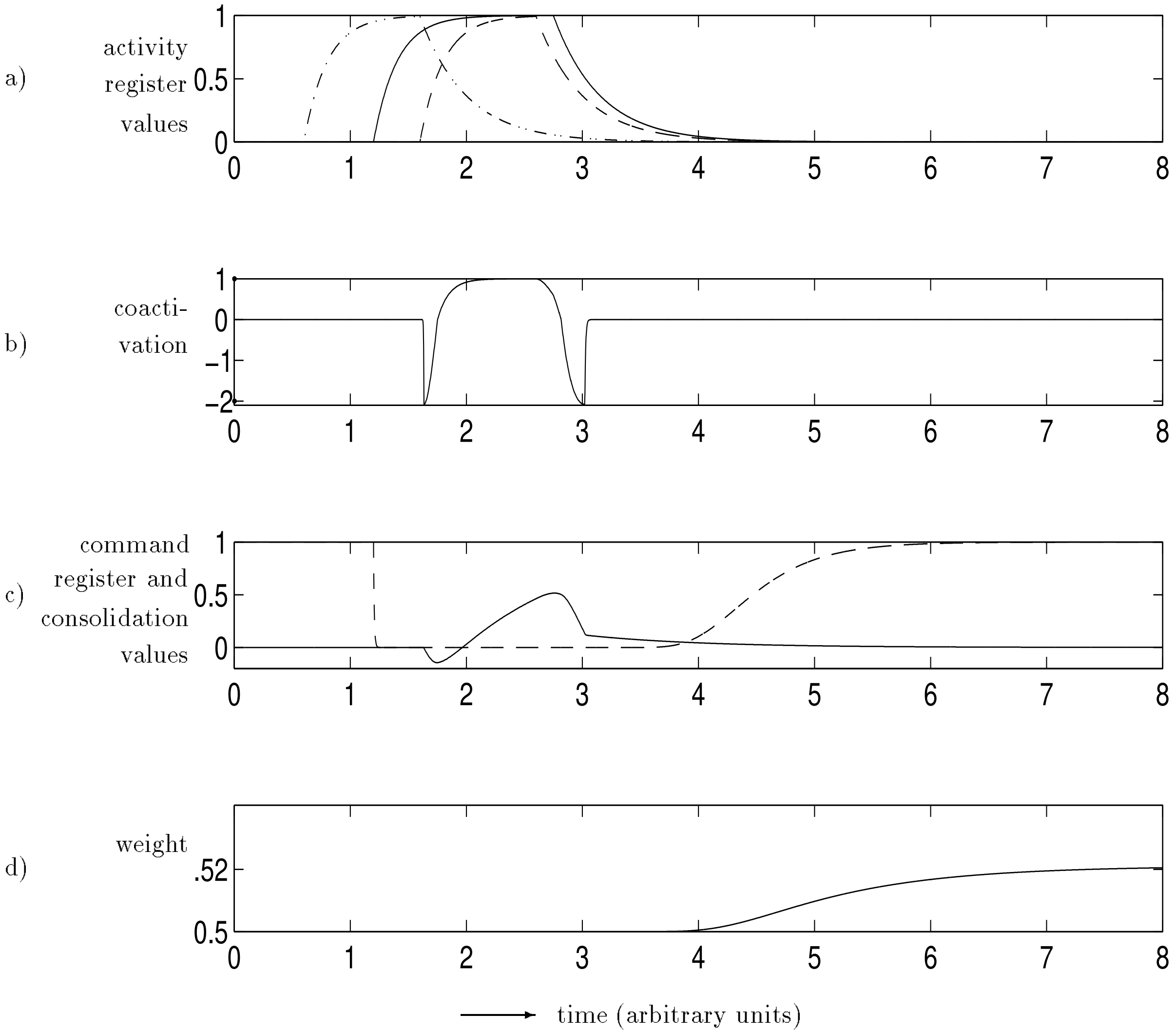

Figure 6: Changes in the Registers

a) Activation Register Values versus Time.
− · − S; —— R; − − − delayed S; $t_{exp} = 1$.

b) Coactivation Value versus Time.

c) Weight Change Rate Constant Factors versus Time.
—— command register value; — — consolidation value returned.

d) Synapse Weight versus Time.

The firing in the S and R neurons is shown by the activity register values in Figure 6a. These values result in the command trajectory shown in Figure 3, which returns the instantaneous coactivation values of Figure 6b. Note the negative values returned as the trajectory crosses the ditch in the surface of Figure 4 on both the outward and the return journey. The command register value follows the coactivation value too slowly to register the second dip so is still positive when the consolidation function starts returning non-zero values. The consolidation function returns zero whenever the neurons are on; the command register value is near zero most of the time that the neurons are off, so there is only a short time during which the weight can change. With the parameters and firing pattern used here, the weight changed from 0.50 to 0.52. Note that the neurons were not on sufficiently long for the command register value to reach maximum in this experiment.

In Figure 6 R is on longer than S but they finish in almost the correct time relationship. Despite the poor overall correlation in firing, this is a strengthening relationship in the Neuro-Connector model. Strengthening occurs for a range of values for t_{obs} around the optimal $t_{obs} = t_{exp}$. This range of values is referred to as the strengthening window as it represents a time window within which synapse weights increase. This window can be considered to be located at a fixed distance t_{exp} after the end of S firing; else at a fixed distance t_{exp} before the end of R firing.

2.5 From an Implementation Perspective ...

If the Neuro-Connector model is to serve as a robot controller, it will be necessary to elucidate the interrelationships between the parameters, to provide conceptual tools for quantifying the net's performance, and to devise efficient implementation strategies. The place to start is with Halperin's proposed mathematicisation, as just described. This section considers the proposal from a robotic designer's viewpoint and makes a start toward analytic and design procedures for the net.

Halperin's proposal as it stands can be implemented on a robot with sufficient processing power: in fact, the robot experiments described in Section 4 were done using just such an implementation. However, the synaptic dynamics are not really suitable for efficient implementation on inexpensive processors as they stand. The command register and weight update equations are not integrable analytically, so numerical techniques are necessary and the evaluation of transcendental functions in computing $A(\vec{\alpha})$ and particularly $\chi(\vec{\alpha})$ require good floating point support if the model is to operate in real time. The situation is aggravated by the fact that synapses are intended to be numerous.

One possible approach is to replace Halperin's continuous $\chi(\cdot)$ with something more computationally efficient, such as a discontinuous step from one, inside the consolidation region, to zero outside. This has the effect of hardening up the boundary of the consolidation region (although Halperin's $\chi(\cdot)$, with its high powers of α_S and α_R, is actually very steep anyway). The legitimacy of this step is made clear by considering the product $A(\vec{\alpha})\chi(\vec{\alpha})$: this determines steady state weight change rate. With Halperin's choice of functions, the product has a minimum value of around -3.5×10^{-12} located close to the trough in the coactivation surface (at about $u = 0$). (There is also a maximum value of the product, of considerably smaller magnitude, in the region where coactivation is positive.) This shows that the direct contribution of coactivation to weight change is negligible, and the sharp consolidation boundary can be placed somewhere convenient in the region of $u < 0$.

The observations above also demonstrate that all of the weight changes induced by the synapse dynamics are generated by the delaying effect of the command register leaky integrator. It is the value in the command register on crossing the boundary of the consolidation region that determines the weight change, and we may approximate $A(\vec{\alpha})$ by making it zero within the consolidation region. Doing so saves some computation and, more interestingly, makes the weight update equation integrable.

Within the consolidation region, the command register value is given in terms of its boundary value c_0 by the relation

$$c(t) = c_0 e^{-t/\mathcal{T}_{cmd}} \tag{7}$$

since the input to the leaky integrator, under our approximation, is zero. The weight update equation becomes

$$\mathcal{T}_{wt}\frac{dw}{dt} = |c_0|e^{-t/\mathcal{T}_{cmd}}(w_{tgt} - w) \tag{8}$$

and, with an initial weight of w_0, this has the solution

$$w(t) = w_{tgt} + (w_0 - w_{tgt})e^{-\frac{|c_0|\mathcal{T}_{cmd}}{\mathcal{T}_{wt}}(1-e^{-t/\mathcal{T}_{cmd}})}. \tag{9}$$

The asymptotic value of the weight, after a stimulus presentation that elicits a command register value c_0, is then

$$w(\infty) = w_{tgt} + (w_0 - w_{tgt})e^{-\frac{|c_0|\mathcal{T}_{cmd}}{\mathcal{T}_{wt}}} \tag{10}$$

— the difference between the initial weight and the target is reduced geometrically by repeated presentations, hence the rapid learning of the Neuro-Connector model.

The final observation to make here is that the last equation affords a way of quantifying the learning rate of the model in terms of the parameters and, therefore, a criterion for selecting the time constants for weight change and command register integration. If we compute the extent to which a single presentation shifts the weight halfway to its target, this will provide an intuitive but quantitative index of learning rate. From the equation above, a single presentation achieving command register value c_0 is equivalent to

$$H_w = \frac{\mathcal{T}_{cmd}|c_0|}{\mathcal{T}_{wt}log_e 2} \tag{11}$$

half-lives of the weight change process, *i.e.* n presentations of the stimulus reduces the weight difference to $2^{-H_w n}$ of its initial value. Thus, if we wish to achieve full weight change in five presentations, say, we should set H_w to be about 1.4 — the weight after five presentations is then within 1% of its target.

3 The Net in Operation

The expected pattern of firing is that at any time various S neurons will be on, reflecting the current sensory state of the agent (both internal and external sensors). Various Rs will be on due to their getting enough input from the Ss. One R will be on because its B is on, whether it currently has much S input or not. One B will be on. The Ss and Rs currently active will keep changing. B firing will tend to be more persistent because of the feedback loop with the R and the inhibition from the current B to the others. Of course the current B will change, either due to another B coming on else due to this one adapting out.

As the robot operates it is constantly learning about the relationship between its sensory state and the success of its behaviours. S neurons responding to sensory states which are reliably associated with the start of a behaviour form high-weight connections with the releasing neuron for this behaviour, but only if they are specific. S neurons responding to more general stimuli have their synapses with each R weakened, unless one behaviour is much more common than the others, since the strengthening which occurs when one R is on in the right time relationship with the S is overshadowed by the weakening which occurs when any other R goes on instead.

Changes in the environment need not prevent the correct operation of the net if enough S neurons are available. Every S neuron responding to a sensory state specific to a behaviour will develop a high-weight synapse with the releaser neurons for that behaviour, so there will often be several releasing stimuli for any behaviour, *e.g.*, releasing connections for a behaviour which needs to recognise people might connect with S neurons representing 'I can see a face', 'Here is a moving heat source', and 'I can hear voices'. Therefore although a change such as the lights being switched off may cause visual sensors to cease being useful, the information from the infra-reds can still be used to trigger the behaviour. When the lights are switched back on then the light-dependent stimuli will become significant again. Weights do not change significantly unless the S neuron fires, so responses to rare events are not forgotten.

If the net is initialised with extra inputs to R or B neurons which connect to external signals *e.g.* from a radio modem, then behaviours can be forced or suggested (depending upon signal strength) from outside the net. The robot's behaviour can then be corrected while the system is operational. Halperin herself proposes such extra inputs coming from hierarchical net levels not described here. In the model, these levels are to provide motivational stimulation.

It should be noted that the net responds to timing coincidences and does not measure cause-and-effect. The assumption is that coincidences do not happen in the correct time relationship very often, so perturbations in synapse weights caused by coincidences are rare and are removed next time the stimulus is seen. This assumption does mean that experimenters can artificially set up coincidences and see the effects, as in certain sorts of animal conditioning experiments.

Stimulus – response connections are easy to condition, in fact the whole operation of a Neuro-Connector net can be seen as S–R conditioning. Halperin also suggests a mechanism for S–S conditioning in her thesis, although this involves extensions to the model not described here.

An unusual capability of this model is its ability to explain postponed conditioning. Postponed conditioning occurs when a previously neutral stimulus becomes a releasing stimulus having been shown after the behaviour started. Since this model considers correlations in the ends of firing to be most important and takes little account of early correlation in firing, whether the stimulus occurred before or after the behaviour started does not affect the weight change in the synapse. Therefore a short stimulus which occurs towards the end of a long behaviour can easily become a releasing stimulus for that behaviour. It should be noted, however, that using this newly-conditioned stimulus as a releasing stimulus causes the effect to disappear, since the stimulus is short, the behaviour long, and the releasing connection has a short expected time. This is in accord with reports on backward or postponed conditioning in the animal literature (Spetch et al., 1981).

3.1 Controlling Timings

It is important that the time between the end of firing in the S neuron and the end of firing in the R neuron is fairly constant when the behaviour operates successfully, as the comparison between this expected time for the behaviour and the actual time taken is the only information available to the learning part of the model.

For many behaviours, *e.g.* 'grasp can', the time taken to complete them successfully is fairly constant. When the 'grasp can' behaviour fires correctly, *i.e.* when the hand is poised with a can between the fingers, the time taken for the hand to grasp the can is roughly constant. When the behaviour fires wrongly, *e.g.* the robot identified a picture of the can as the can itself, then the attempt at grasping continues until the robot gives up, which is typically a rather longer time than a successful grasp would take. By measuring how long this sort of behaviour takes, it would be possible to have a good idea of whether the behaviour was successful or not. In fact, measuring time is the only way that this model has of deciding whether a behaviour fired correctly or not and so whether the synapses involved should be strengthened or weakened.

For behaviours such as this which have an approximately fixed duration it is most useful if potentially releasing S neurons can have an extra condition added to them, one which becomes false at some fixed point in the operation of the behaviour. In our example, S neurons with releasing connections to R-GRASP CAN would not only include some way of recognising 'can between fingers' but would also probably contain a condition such as 'and fingers fully open' which would ensure that these Ss finished firing in a constant time relationship with the R/B neurons, keeping t_{obs} fairly constant. Where this is difficult or would cause the S neurons to be on for too short a time (see Section 5.2) then the S neurons can be made to have a fixed duration of firing by causing them to adapt or time out. Then we can be sure that t_{obs} will be fairly constant, being calculated by

$$t_{obs} \;=\; beh(dur) - S(dur) + R(delay) \tag{12}$$

where $beh(dur)$ is the (roughly constant) duration of the behaviour, $S(dur)$ is the adaptation time of the S neuron, and $R(delay)$ is the time it takes for the R to start firing after S fires. This second method of fixing t_{obs} is not optimal since the S neurons time out whether the behaviour has fired or not. This means that, if the behaviour under consideration cannot be performed immediately because a 'more important' behaviour is operating, it will not be performed later because the releasing S neurons will have turned off so the robot is unaware of the opportunity.

For some behaviours, *e.g.* approaching a moving object, the duration of the behaviour is expected to vary considerably, depending upon the distance at which the object is detected and whether it is moving towards or away from the robot. One solution is to add a sensory condition into the behavioural specification such that the S and the R/B turn off together. A significant sensory state for our example would be something like “moving object more than 50cm away”. The behaviour would be something like “move towards object *until within 50cm*”. Then the behaviour will finish at about the same time as the S neuron goes off. It is always possible to arrange that the behaviour takes an approximately constant time after a particular stimulus disappears, with a little thought. Unfortunately, this form of timing control means that incorrect firing in the B neuron cannot be detected unless one of the neurons adapts out, otherwise neither neuron will finish firing and no weakening will occur. It is best if the B neuron times out as this allows another behaviour to take over earlier and also guarantees weakening — if the S adapts out too early in relation to the R finishing firing then no weakening will occur as it will appear as if the S never fired. A better solution would be to make timing sensitivities less important for some behaviours. The mathematics involved in such a change is under investigation.

3.2 Initialisation

There are two main ways of setting up a Neuro-Connector net, one involving time and patience but little skill; the other involving careful planning. Hybrid schemes are also possible.

Both methods require that the behaviours necessary for each subtask and for general survival have already been programmed in and tested. This is reasonable since robot tasks are normally well specified so the necessary skills for each component are not difficult to determine. They also require that each behaviour has at least one specific ‘releasing stimulus’ to which a specific S neurons responds, *i.e.* that the robot can reliably disambiguate significant circumstances by the firing in single neurons.

The demonstration learning method involves running the robot in its environment controlling the robot by hand and measuring timings. This method is particularly useful where the specifications given were poor, or the robot is to respond to sensory readings which are poorly understood, or the people wanting to use the robot are not programmers. It is feasible because the net learns so quickly, typically in only four or five encounters.

The programming method requires the robot to be initialised sufficiently well that it will perform correctly most of the time from its first release in its working environment. This involves considerable designer knowledge of robot, task, environment and robot sensor readings. The designer knowledge does not have to be completely accurate, but does have to be ‘accurate enough’. This method is useful where the robot will not be accessible to human control in its working environment, *e.g.* for space exploration.

3.2.1 Behaviours

We have assumed that the behaviours necessary have already been designed, implemented and debugged. What remains is for priorities and adaptation times to be set.

When behaviours are designed, it is normally obvious which ones belong together as a sequence. Each sequence will relate to a particular subtask and will end with a behaviour which either achieves the subtask or is the result of a test failing. For example, if our robot is to collect empty drink cans the expected sequence of behaviours once a can is seen might be as in Figure 7. In this case there are two end-of-sequence behaviours, one of which puts an empty can into the bin and one of which replaces a full can where it was found. Note that the sequence is not expected to run as a continuous whole; the ‘goto bin’ behaviour in particular may involve considerable navigation and might well be interrupted by the opportunistic performance of other behaviours from other subtasks.

Each later behaviour in the sequence must be given a higher priority than earlier behaviours, otherwise the earlier behaviours will not be interrupted correctly so will not finish at the right time. Priorities between sequences are less clear and will depend upon the relative importance of

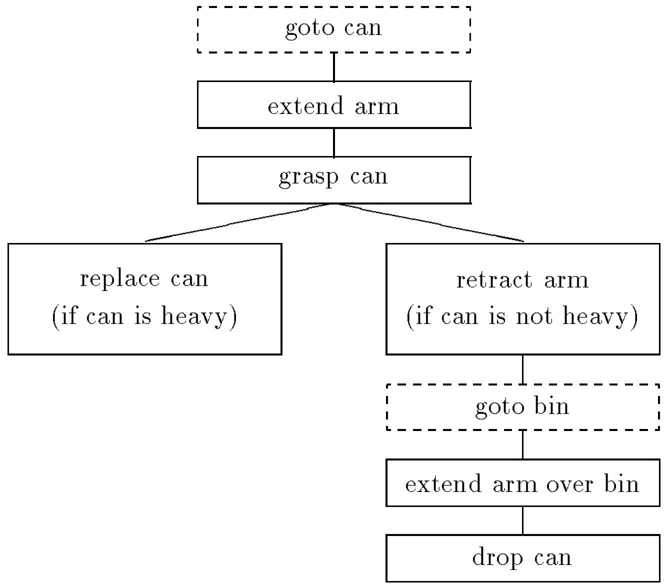

Figure 7: A Behavioural Sequence for Empty Can Collecting.

Dashed boxes indicate behaviours with several components.

the subtasks. There will also be some essential reflexes which are needed to minimise damage to the robot, obviously these need top priority.

B neurons controlling end-of-sequence behaviours and reflexes should be given adaptation times that cause the B to finish firing and turn the R off within the strengthening window (see Section 2.4) for their releasing connections. This ensures that these behaviours are self-reinforcing, acting as a reward for the whole sequence. These behaviours can never be performed incorrectly. Other B neurons should have longer adaptation times such that the system can notice when these behaviours have overrun.

3.2.2 Sensory Neurons

It is important that there exist sensory neurons for each behaviour which *only* fire when that behaviour should be active and go off before or as the behaviour is successfully completed. This means that many very specific S neurons are needed for any sizable task. Accurate knowledge about which S neurons fulfill that requirement for each behaviour is not needed, however, since this gets refined as a result of the normal operation of the net.

Often it is possible to imagine the sensory conditions under which each behaviour is to be performed and the changes in sensory state which will occur as the behaviour operates. Sensory neurons can then be made which respond to 'releasing conditions and not behaviour-active conditions', *e.g.* for the 'extend arm over bin' behaviour in our example the releasing S neuron would respond to something instantiating 'robot is in front of a bin and hand contains a can and arm is retracted'. The previous behaviour will have positioned the robot in front of a bin, this behaviour is only appropriate here if the hand already contains a can, and 'arm is retracted' becomes false as soon as the behaviour starts operating, thus switching off the S neuron at a time expected to be constant *wrt* B finishing firing. Where the particular sensor readings which reliably and specifically indicate such a state are accurately known, then a single S neuron can be made which responds to that set of readings. However, it is more common that the exact sensory state corresponding to something like 'in front of a bin' is not known. In this case a set of S neurons should be made which cover all possible 'in front of a bin' readings as well as the other conditions. Any or several of these may turn out to be suitably specific for this behaviour.

If no servoing is required to position the robot hand correctly over the bin, then the 'extend arm over bin' behaviour may be identical to the 'extend arm' behaviour and they may be implemented together. In this case the behaviour will have two distinct sets of S neurons connected with it.

3.2.3 Synapses

Synapse weights and expected times t_{exp} are both crucial for the correct operation of the net. Both of these are easy to set if the robot can be run through its task in its work environment by hand, but more difficult to set by thought alone.

Demonstration Learning

If the robot can be run through its task several times, then setting up synapses is easy. For small nets or where computing power is limiting, no S–R synapses are initialised. The robot is controlled by hand, perhaps by radio modem commands stimulating the B neurons directly, and is made to record the times at which its neurons switch off. Analysis of these times will show which relationships are constant. High-weight synapses can then be set up with the measured t_{exp} between neurons with a constant time relationship. Since this model is only likely to be used where the environment is expected to change, other synapses are also needed. Low-weight synapses should therefore be set up between each R neuron and any S which might become significant. All synapses connected to the same R are expected to need the same t_{exp}.

Where analysis of neural off times is likely to be tedious and an estimate of t_{exp} can be made, then the net can be set up with suitable synapses already in place. If the estimates of t_{exp} have large variance, then multiple synapses can be set up between each suitable S–R pair, each with a slightly different t_{exp}. It should be remembered that t_{exp} does not have to be entirely accurate anyway, since there is a window around t_{exp} in which strengthening occurs. Weights can all be initialised to low values, and the robot run through its task as above. S neurons will fire in response to environmental cues; R and B neurons will fire in response to the trainer's commands. S neurons responding to specific releasing stimuli with the correct t_{exp} for any R will have their synapses strengthened. After of the order of five constant-environment trials for each behaviour reliable releasing connections will have been made. If the robot is required to operate in several environments (*e.g.*, with room lights on or off) then the trials should be repeated in each environment to be sure of success.

Setting up by Program

Theoretical analysis of the program and running the robot in a simulated work environment can lead to reasonable estimates for both releasing stimuli and for t_{exp}. Multiple synapses can be used where only a rough estimate of t_{exp} can be obtained, each with a slightly different t_{exp}. The synapses connecting each R neuron with the S(s) expected to be releasing stimuli are given a high weight, other synapses being initialised to low weights.

Estimates of releasing stimuli must be sufficiently good that the robot performs a behaviour correctly the first few (approximately, four out of the first five) times. Then the net will have adjusted itself such that the best releasing connections have highest weights. Where multiple synapses were initialised low-weight ones can be pruned. Other synapses can also be pruned but this removal of connections reduces the capacity of the net to learn.

4 Implementation on a Mobile Robot

This section describes the preliminary experiments done on a mobile robot to ensure that the model could be used as a robot controller and that there were no underlying biological processes which were necessary but had not been included in the model. This section describes the robot, some assumptions made in the simplification of the model for this implementation, the experiments performed and the results obtained. Discussion specific to the robot experiments then follows.

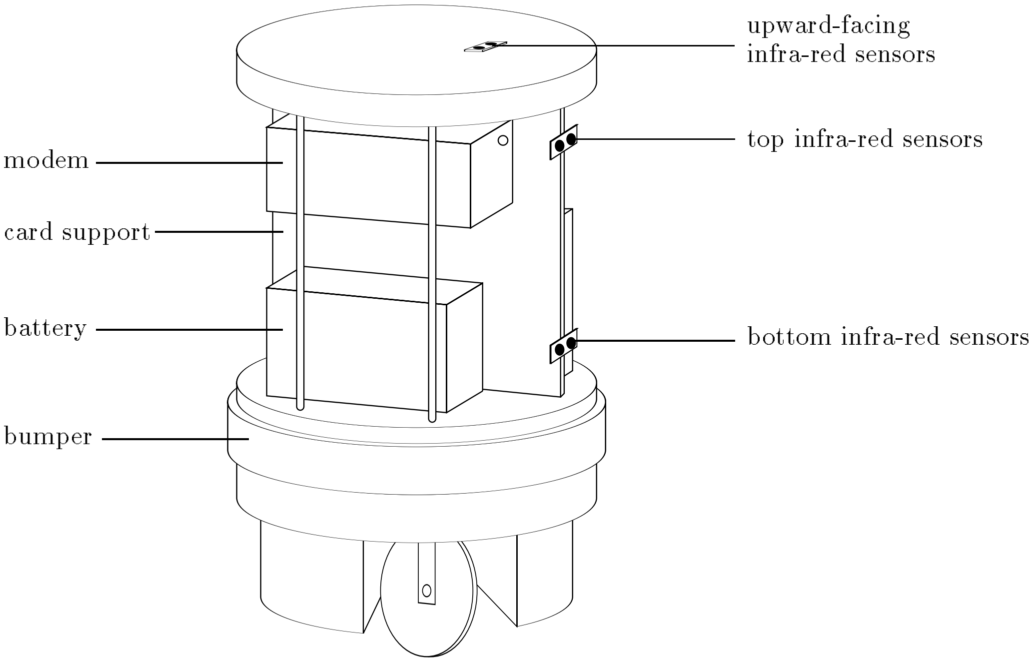

Figure 8: The Mobile Robot Ben Hope

4.1 The Robot

'Ben Hope' is an autonomous transputer-based mobile robot, see Figure 8, developed at the Department of Artificial Intelligence in Edinburgh in the course of another project (Forster, 1991). It is a cylindrical robot with a diameter of 30cm and a height of 50cm. It has sufficient battery power to run the base for about 30 hours before recharging, and has a separate battery for the head which will support about 2.5 hours of typical activity. The head comprises a card support plate with transputer motherboard, four transputer modules, a Flight 68K processor card and a power converter card; also the head battery and a full duplex 9600baud radio modem. Various sensors have been used with Ben Hope, including a CCD camera, a ring of twelve Polaroid sonars, an eight-segment bumper, and active infra-red sensors.

The sensors used here were six infra-red emitter-receiver pairs and the bumper. The infra-red sensors were thresholded and used as binary sensors, arranged in pairs on top and on the front of the robot as in Figure 8. Each pair consisted of one infra-red unit detecting white objects at about 60cm and one detecting white objects at about 20cm.

For the experiments described here the transputer's facility for parallelism was not used as all the code was contained in one Parallel C 'task'. Increasing the degree of parallel implementation would probably be an advantage for more complex tasks as the model is inherently parallel — however the simplified version of the model described here is easily computationally tractable without the need for parallel implementation.

4.2 Robot Experiments

This was necessarily a sequential implementation in which the program cycled, doing each operation in order. Between updates of each function it was assumed that sensor readings and the target values for the exponential decays did not change. While obviously a false assumption, it does give reasonably accurate results for most functions, especially since integrations were done from the equations and not approximated. However, this method may not be so accurate for the more complicated weight change functions.

In the implementation described here R neuron thresholds did not adjust; S and B neuron threshold functions were step functions, flipping from minimum to maximum after an 'adaptation time' described in seconds and flipping back after a 'refractory time' similarly described. Behaviours were prioritised by hand by putting dissimilar weights on the B–B inhibitory links, allowing behaviour switching to occur.

4.2.1 Behavioural Chaining

This experiment was designed to demonstrate that behaviours will tend to occur in sequences determined by their relative priorities despite there being no explicit connection between the behaviours. The robot was to change from using an ADVANCE behaviour (see Table 1) to using a WIGGLE behaviour automatically once the ADVANCE behaviour had got it close enough to a wall for the 'near' infra-red sensor to be stimulated.

The net used comprised twenty-one sensory neurons, four releaser neurons and four behaviour neurons. The sensor propositions that fed into the sensory neurons, the conditions initially set up to release the behaviours, and all the settable neuron and synapse parameters are described in the Appendix. Each R neuron was connected to each S with a low weight of 0.01. Ten releasing connections as in Table 1 were initialised with weight 0.8. All S–R synapses were plastic and all others fixed. The timestep used (program cycle time) was set to be 0.1 seconds; t_{exp} was set to be 1 second for all synapses.

Behaviour	Priority	Actions Involved	Sensor Conditions (Infra-reds)	Sensory Neuron Numbers (see Appendix)
ADVANCE	0	Approach object	Far top and/or far bottom	8, 11, 16, and 17
WIGGLE	1	Back 8cm, wiggle back and forth	Near bottom and near top else near bottom and far top, with or without bumper	5, 6, 13, and 14
AVOID	2	Avoid object — back 15cm, turn 90°	None — bumper only	3
AFTERPAT	3	Back, turn 180°, pause	Upwards-pointing left and right infra-red sensors	0

Table 1: The Behaviours Used and their Releasing Stimuli

The behaviours implemented were as listed in Table 1. ADVANCE was triggered by infra-red reflective objects within 60cm; WIGGLE was triggered by infra-red reflective objects within 20cm. Note that none of these behaviours were active unless specifically triggered by their R neurons.

The down-loading port for the robot was rather too far from any convenient wall for the wall to be detectable by the far infra-red sensors. Therefore the intention of this experiment was that the experimenter would lead the robot to the wall by using an infra-red reflective object which would be kept at a distance of between 25 and 60 cm from the robot while the robot moved, *i.e.*, which would trigger the behaviour ADVANCE and keep it active until the wall was close enough. However it was found that the robot ceased following due to the ADVANCE behaviour timing out, and that attempts to re-interest the robot once ADVANCE had finished being refractory were swiftly ineffective. Thereafter the ADVANCE behaviour could not be restimulated until new releasing connections were set off-line. Therefore a wall was simulated by using a large white box which was placed 50cm from the robot. The robot then succeeded in advancing and wiggling as expected.

4.2.2 Sensory Discrimination

This experiment shows what happens when too many releasing connections are initialised and some are wrong. The robot was given the task of wiggling when it detected a box at the base of a wall, and the net was set up as in the previous experiment. This includes the WIGGLE behaviour having the four releasing conditions in Table 1, causing the robot to respond to both walls with boxes in front and walls without, *i.e.* to respond under both correct and incorrect conditions.

Note that the WIGGLE behaviour starts by causing the robot to back up by 8cm. This was enough to cause the wall to become invisible to the near infra-red sensors, thus the WIGGLE behaviour starts by causing its releasing sensory neurons to switch off. However, WIGGLE continues for 3 seconds and the synapses connecting to R-WIGGLE have the default t_{exp} of one second. Therefore these synapses weaken unless WIGGLE is interrupted at about the right time by another behaviour starting. A suitable behaviour is AFTERPAT which can be triggered by a human trainer stimulating both the upward-facing infra-red sensors. AFTERPAT is self-reinforcing as it adapts out after only 1.3 seconds, *i.e.* about one second after the releasing S neuron goes off (S-0 has an adaptation time of 0.5 seconds), and t_{exp} is one second for this synapse. Normally the sensory conditions triggering reward behaviours are expected to come from the environment and be caused by the successful operation of the previous behaviour, but here a human was used instead.

'Walls' were presented to the robot by a human experimenter moving two white boxes in front of the infra-red sensors, either both level to simulate a wall else offset by one boxwidth to simulate a wall with a box in front. Wiggling in front of walls with boxes was rewarded by the trainer, wiggling in front of walls without boxes was not.

After four unrewarded wiggles for walls alone wall presentation ceased to elicit wiggling, demonstrating that the releasing connection had been unlearned. Wiggling continued for walls with boxes in front if the pat was timed such that AFTERPAT inhibited WIGGLE correctly.

4.2.3 Postponed Conditioning

The ability to explain postponed conditioning (see Section 3) is one of the distinctive and unusual features of this model. Demonstration of postponed conditioning therefore indicates that the implementation captures the essence of the model, and that the model does not rely on underlying and unstated biological processes.

For this experiment the robot was tethered so that neural stimulation could be seen on a monitor. The net was set up as before and the WIGGLE behaviour stimulated. After the robot had started wiggling, the bottom near infra-red sensor was stimulated such that it would adapt one second before the wiggle was due to stop. This caused S-19, which had previously had low-weight connections with all behaviours, to fire in the correct time relationship with R-WIGGLE. This schedule was followed four times and then the bottom near infra-red sensor was stimulated alone. The robot wiggled, demonstrating that a new releasing connection had been formed.

4.3 Discussion of Robot Results

The robot experiments reported here show that the Neuro-Connector model has the features necessary for a robot to learn to adjust the sensory conditions under which behaviours become active, provided enough sensory neurons are available. This means that imperfect designer knowledge can be used for setting up without inaccuracies necessarily causing the robot to fail, and that slow changes in the environment will not cause failure but just cause new releasing stimuli to be learned.

The results of the first experiment show that behavioural chaining can occur despite there being no explicit connections sequencing the behaviours. Behavioural priorities determine the behavioural hierarchy and therefore sequencing can be obtained by choosing these priorities wisely. However, it will not always be clear in advance what all the priorities should be, especially between sequences.

The importance of ensuring that adaptation times are sufficiently long was also demonstrated. Adaptation times that are too short cause the system to fail due to the weakening of perfectly correct responses, as demonstrated by the experiment in Section 4.2.1. Adaptation times that are too long should have a much less dramatic effect. These cause the robot to keep trying an unsuccessful behaviour rather inefficiently long and also cause an increase in the number of times an unsuccessful behaviour is tried (since the amount of weakening which is obtained when R is on 'too long' falls off with greater mismatch until there is virtually no weakening at all if R fires without the S, see Section 2.4). Once a behaviour has no releasing stimuli, it has vanished from the robot's behavioural repertoire and cannot be resurrected unless the higher level inputs mentioned in Section 3 are implemented.

The results of Sections 4.2.2 and 4.2.3 show that the system can compensate for inaccuracies in releasing stimuli, whether these are caused by inaccurate setting up or by subsequent changes in the environment. In Section 4.2.3 an incorrect releasing stimulus was forgotten after only four presentations. In Section 4.2.2 a correctly-timed stimulus that had been missed had become a releaser by presentation number four. Learning in only four presentations of the stimulus is incredibly fast compared with many other robot learning methods, although not as fast as the 'one shot' learning Grossberg considers to be sometimes necessary (Grossberg, 1982). The learning rate here could perhaps be speeded up by adjusting neuronal gains but this is considered unwise since we do not want our robot to learn to respond to coincidences or sensor noise.

Section 4.2.2 is, of course, a special case in that the S neuron which develops a high-weight synapse with the behaviour responds to an event which takes place after the behaviour has started. The event therefore cannot be truely considered a releaser or trigger for the behaviour even though it has become one. Testing this event as a releaser, however, causes incorrect time relationships so the high-weight connection disappears again. Backward conditioning of this type, where found, is notoriously transient (Spetch et al., 1981).

Unfortunately the robot we were using belonged to another project, so only preliminary testing could be done. A similar robot which can be dedicated to this investigation is currently under construction and will be used for further real robot experiments in due course.

5 Simulation experiments

Having shown that the Neuro-Connector model can be used to control a real mobile robot, further theoretical and simulation investigation is being done to explore the variability in possible behaviour. This section describes weight changes occurring in a simulation of a single S – R synapse under various different starting conditions. Note that the various integrations were done continuously in the simulation and not stepwise as in the robot implementation.

5.1 Comparison with Robot Results

Synapse strengthening and weakening were investigated both on the robot and in simulation using the starting conditions described in Section 4.2. Experiments were run where the observed time difference between the end of firing in two S neurons and the end of firing in the R neuron was varied artificially. From starting weights of 0.8 and 0.01, the final weight was recorded after one presentation of the stimulus. The expected time difference t_{exp} for each synapse was set to be 1.0 second; other parameters were as in the Appendix. The results obtained from the simulation were similar to those obtained from the robot, as illustrated in Figure 9.

The results show that the overall shape of the graphs is identical. There is a time window around t_{exp} where the synapse weight has increased during the experiment. Either side of this, where t_{exp} and t_{obs} are just sufficiently different, the synapse weight has weakened. Where t_{exp} and t_{obs} are completely different, *i.e.* where the S and R firing are deemed to be uncorrelated, there is little effect on synapse weight. The differences in the two sets of results are likely to be due mostly to inaccuracies in timing for the robot experiment, especially for the short times; and the difference in integration method.

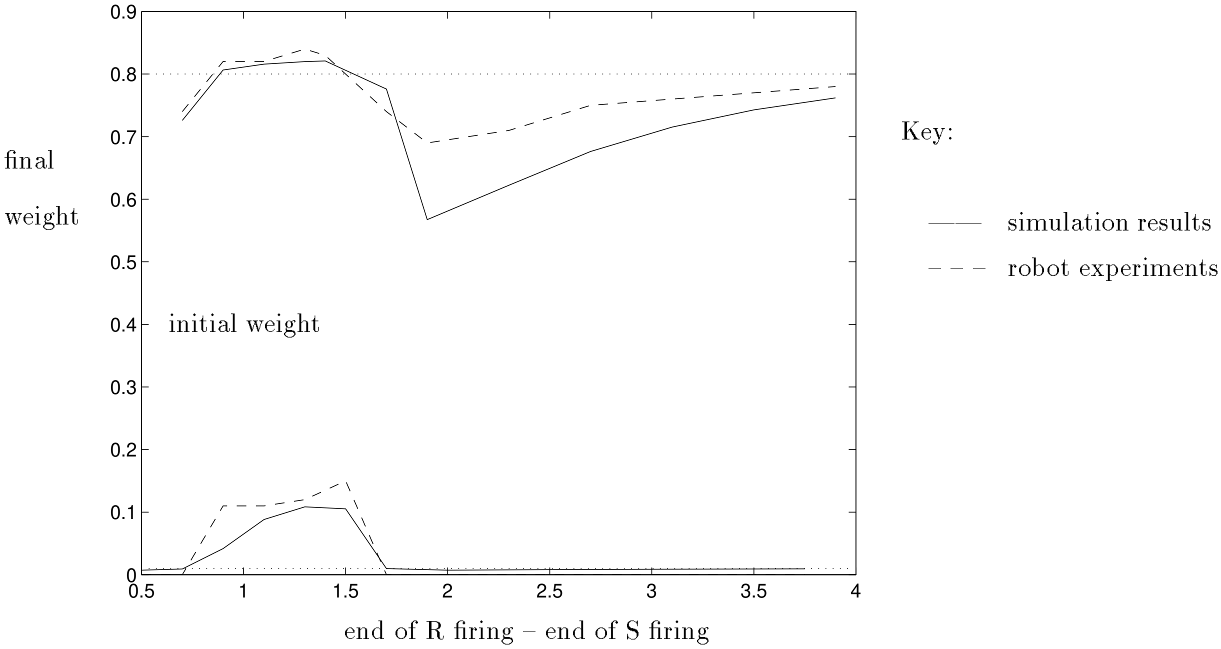

Figure 9: Simulation and Robot Results Compared

5.1.1 Postponed Conditioning

Postponed conditioning, where a neutral stimulus becomes a releasing stimulus despite being shown after the behaviour started, was demonstrated for the robot after four presentations of the training stimulus. Using the same parameters, the simulation showed postponed conditioning after three presentations.

5.2 Effect of Burst Length

Examination of the maths involved in the production of the command register shows that, when the firing burst length is short enough that the command register value never reaches its maximum, even the correct time relationship can cause weakening. This is most easily visualised by a consideration of the command trajectory (see Figure 3). If the activity register value in either neuron does not reach maximum and stay there long enough for the command register value to reach maximum also, then the final weight obtained will be lower than is possible at that time relationship. If either activity register value does not reach 0.68, then the coactivation value returned never becomes positive, the command register cannot gather a positive value, and the weight change has negative sign whatever the temporal relationship between ends of firing bursts.

Figure 10 shows the results of decreasing burst length for a perfectly correlated pair of neurons. It can be seen that once the burst length goes below about 3 time units then the final weight is reduced. Below about 0.7 time units even a perfect correlation in firing causes weakening, the exception being when the S neuron does not come on at all, when there is no weight change.

5.3 Varying Neuronal Gains

The gains in the input register affect the time that the neuron goes on and off and therefore the time relationship between pre- and post- synaptic neuron. They do not affect the weight change that happens with any given time relationship.

The value of the activity register up gains, and even the correlation in these gains between pre- and post-synaptic neuron, affect the final weight only if the burst length is short. If the burst length is sufficiently long then the initial coactivation values are not represented in the command register value achieved.

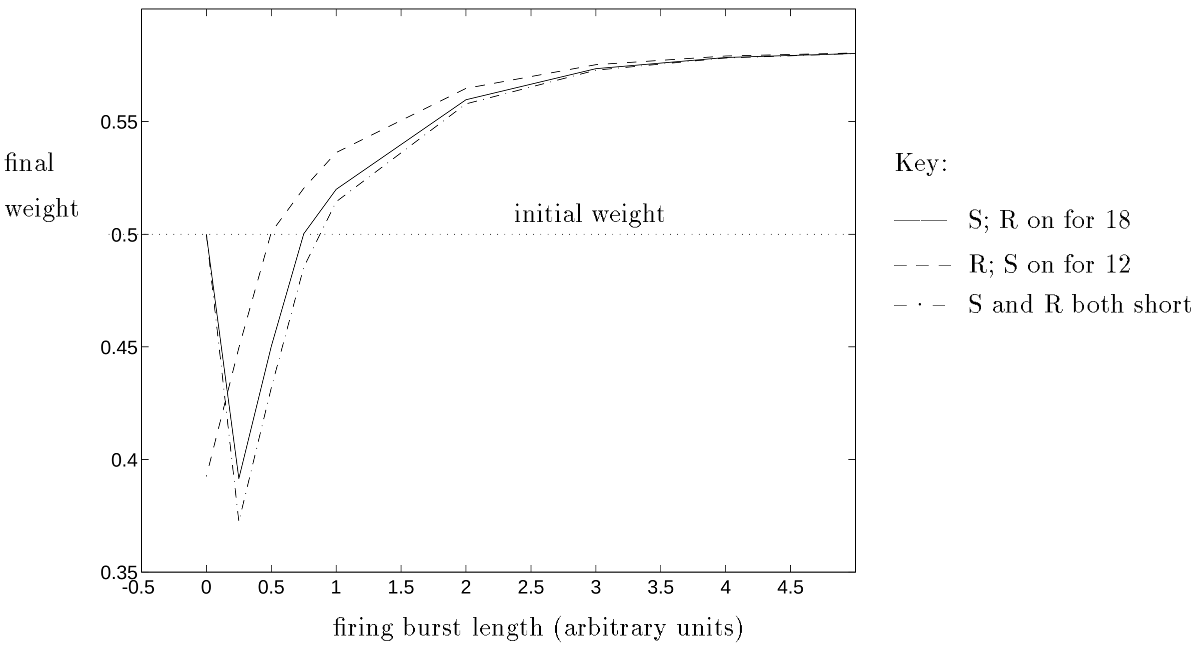

Figure 10: Effect of Varying the Burst Length for a Perfect Correlation in Timing

The neuronal time constants most influential in affecting the weight change are those governing the decay of the activity registers. With command register gains set at 1, the activity register down gain had to be over 1.5 for any strengthening to occur and under 4 if there was to be weakening when R was on 'too long'. Greatest 'R on too long' weakening occurred with an activity register down gain of 2.5. Since strengthening was also strong with this gain, this was the value chosen for subsequent experiments. Results for some of the activity register down gains tried are given in Figure 11. In each case, t_{exp} was 2 time units, synapse weight started at 0.5, S was on for 10 time units from $t(0)$, and R on a variable time from $t(0)$ causing the difference in firing between S and R to be as indicated.

5.4 Varying Synaptic Gains

The only synaptic gains variable are those of the command register. Gains of between 0.5 and 4 were tried, using neuronal activity register down gains of 2.5 and a t_{exp} of zero. The results, given in figure 12, show that gains of between 0.75 and 1.5 give the desired features, although at 1.5 the strengthening peak is very small. With a gain of less than 0.75, the weakening expected when R is on 'too long' disappears. With gains of over 1.5 the strengthening peak disappears.

Since changing the command register gains has a related effect to changing the activity register gains, it was decided that the command register gains could be set at 1 and considered to define the time scale of the net.

5.5 Varying Synaptic Expected Times

Although the model is theoretically independent of the magnitude of the expected time in the synapse, it was felt important to check that the size of t_{exp} does not affect the width of the strengthening window, *i.e.* the accuracy with which t_{exp} has to be known. Various t_{exp} were tried in the range 0 to 6. The results (Figure 13) show that the shape of the graph is indeed independent of the particular t_{exp} used. When the numerical results from the separate runs were compared it was found that the variation was in the fifth significant figure, the last recorded — this being probably due to rounding error.

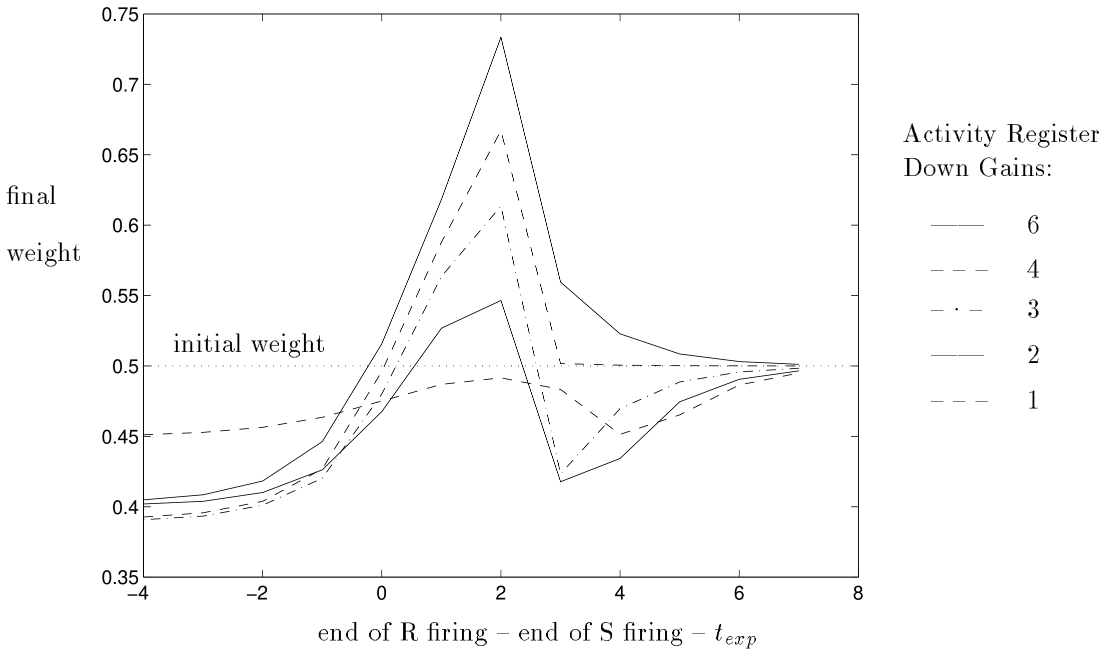

Figure 11: Effect of Varying Activity Register Down Gains at Various Time Relationships

5.6 Effect of Frequency of Presentation

The effect on learning of the frequency with which the stimulus is seen was also investigated, by varying the amount of time between presentations and recording the synapse weight after the second presentation. There was found to be no difference in the final synapse weight for interpresentation intervals between 10 and 600 time units; however intervals of less than 10 time units did cause the final weight to be lower. For intervals of less than 3 time units the final weight was less than its initial value. These effects on the final weight are due to there being insufficient time between the two presentations for the weight change to be completed with the particular time constants used.

5.7 Discussion of Simulation Results

The simulation was first used to show that the results obtained were similar to those obtained from the robot, the main differences probably being due to human timing errors in the robot experiments and the difference in the integration method used. These differences are sufficiently small that conclusions drawn from the results of simulation experiments should prove at least qualitatively valid when transferred back on to the robot.

The effect of burst length on the final weight obtained means that care must be taken when designing experiments to ensure that neurons remain on sufficiently long. This has implications for an agent's response to transient stimuli. Adjusting input register gains such that the S neurons responding to transient stimuli come on quickly and go off slowly would increase the apparent time that the stimulus was present. Increasing the command register gains is not an option, see below.

The model is very sensitive to the down gains in the activity registers, behaving correctly for only a narrow range of values. When activity register down gains are too large, then the command trajectory spends too little time crossing the command surface, arriving at the corners too quickly (see Figure 3). This means that, when R is on 'too long', the command trajectory does not spend enough time in the negative region of the command surface for the command register to acquire a negative value and the weakening which should occur is lost. When activity register down gains are too small, the command trajectory spends too long crossing the negative region of the command surface and any positive value in the command register is removed on the return journey, causing weakening whatever the time relationships between the neurons.

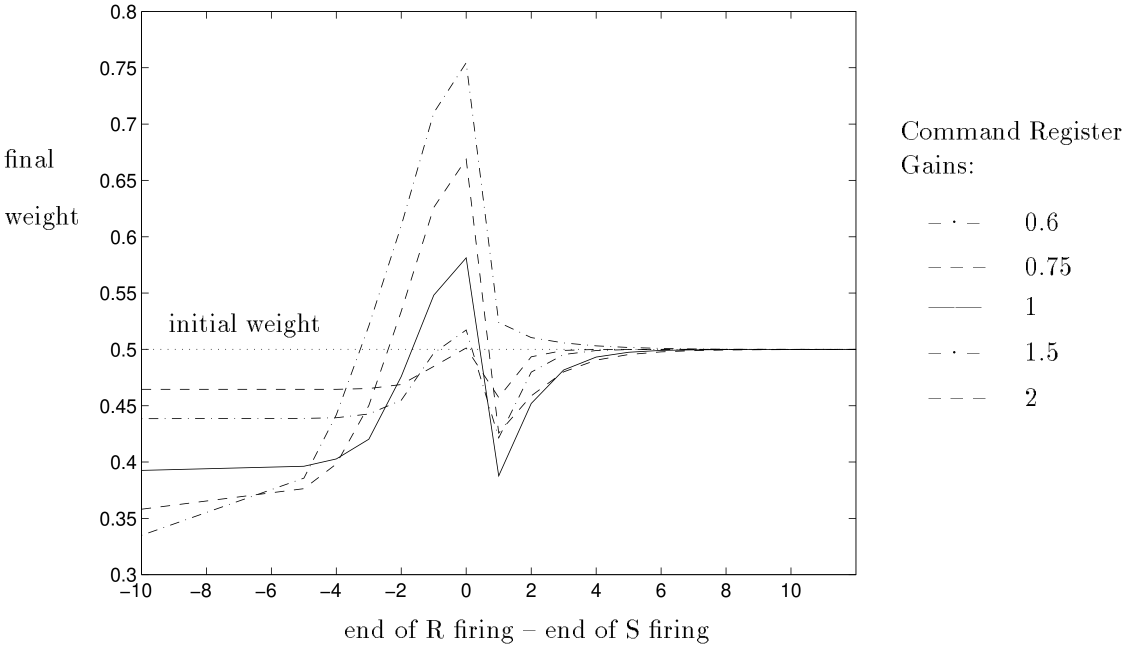

Figure 12: Effect of Varying Command Register Gains at Different Time Relationships

It should perhaps be noted that the S neuron adaptation time used in the pilot robot experiments was 0.5 seconds (equivalent to 0.5 time units with the setup used), which is too short to allow strengthening according to the simulation. However, the activity register down gain used was 4, a value which prevents weakening when R is on 'too long'. It appears that the success of the robot experiments was partly serendipitous, being due to these two effects cancelling out.

Varying the command register gains has a similar but opposite effect to varying neuronal activity register down gains, because both affect the speed at which the command trajectory crosses the command surface but in an opposite sense. The effect is not entirely complementary, however, because varying the command register gains does not affect the consolidation value whereas varying activity register down gains does.

The weight change obtained for any time relationship (R off time − S off time $-t_{exp}$ is independent of the synaptic expected time. It is also independent of the frequency with which the stimulus is seen unless the frequency is very high. Animal experiments have also shown that stimuli shown close together interfere with each other in that expected learning to the first stimulus is reduced (Sinclair, 1981).

6 The Appropriateness of Using the Neuro-Connector Model as a Robot Controller

The Neuro-Connector model as described in Halperin's thesis (1990) was not developed as a robot controller and is not optimised for this task. It is complex and computationally expensive and is therefore not the best system to use for most of the robot tasks which currently exist and for which other control solutions are available. One reason for investigating this model is its potential for controlling a new generation of sensor-rich robots which are intended to perform tasks requiring sophisticated sensory discrimination. This section attempts to be realistic about the difficulties and advantages of the Neuro-Connector model in this context.

It should be pointed out here that the robot does not learn any new behaviours, but can only do behaviours that are already programmed in. What the system learns is the correct circumstances under which each behaviour should be performed. The Neuro-Connector model does not decrease

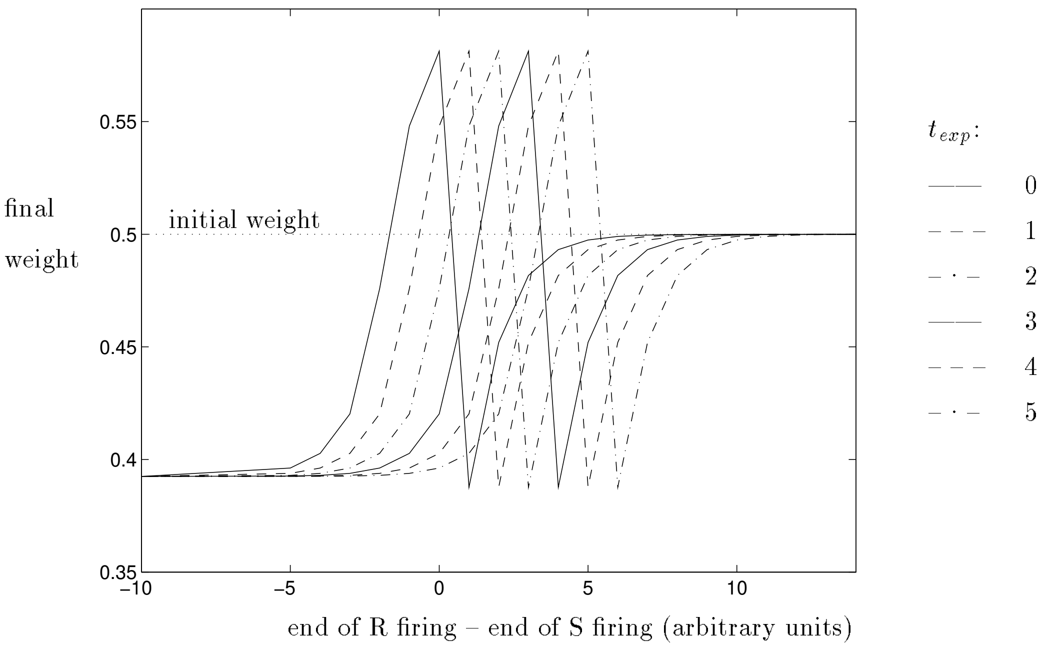

Figure 13: Effect of Varying t_{exp} at Different Time Relationships

the work required to operate a robot, but is intended to make the reliable use of a sensor-rich robot possible.

6.1 Requirements

Behaviours are expected to already exist and to be sufficient for both survival and for the task(s) given. This is felt to be a reasonable requirement, since most robot tasks are easily modularised into suitable component parts and the production of code to cause reliable operation of small pieces of task is not normally too difficult. It should be remembered that the behaviours are expected to have access to relevant sensor information, so servoed behaviours will be common. Movements are expected to take place at constant rate and not depend on power levels. This degree of control is already available on most robots.

Sensor information is expected to be plentiful. It is necessary that each sensory state that the robot is required to disambiguate cause firing in specific S neuron(s) which respond to no other states. It is not necessary that these specific S neurons are known in advance, since they can be learned while the system is in operation. Currently much imagination must be used to decide which particular combinations of sensor readings are likely to be most useful and an S neuron set up for each. This is not considered too difficult a task unless processing capability is severely limited, since extra S neurons are just ignored by the net. In future it may be possible to use a self-organising net for this task.

Critical information needed by the net includes the adaptation times of the neurons, behavioural priorities, t_{exp} for each synapse, and synapse weights. The weights are changed by the operation of the net and can be initialised by running the system as described in Section 3.2.3; the rest need to be set correctly beforehand.

Adaptation times for self-reinforcing behaviours need to be correct, so that the behaviour always finishes in time to strengthen its releasing connection. Other behavioural adaptation times need to be sufficiently long that the net can recognise when a behaviour is being performed incorrectly, but the only upper limit on these times is the patience of the observers. Sensory neuron adaptation times sometimes need to be short so that they finish at a constant time before the behaviours they release finish.

Behavioural priorities are obvious within a behavioural sequence, in that later behaviours should always operate as soon as the situation warrants, so should always have priority over behaviours earlier in the sequence. (Remember that the S neurons with releasing connections to a behaviour are specific for that behaviour so a behaviour won't be triggered unless all its preconditions have been met.) Between sequences it would be useful to be able to use variable priorities which depend partly on sensor readings, so that behaviours such as 'recharge' can have a sliding scale of importance dependent upon battery power level. Currently a series of S neurons is needed which respond to 'power level less than x and recharge point less than y away' to enable opportunistic use of recharging points; else the robot will ignore recharging points until its power level is 'low' and then it will stop everything else to find one.

The expected time for each synapse has to be correct to within 0.8 time units with the gains used for these experiments. Ways of forcing expected times to be constant were discussed in Section 3.1, but it will not be possible to find a constant t_{exp} for all synapses. One solution would be to define our time units differently for different synapses, but this is felt to be inelegant. A better solution would be to widen the time window in which strengthening occurs. Preliminary mathematical investigations indicate that this is not a trivial adjustment, and may be impossible.

6.2 Abilities

The Neuro-Connector model allows behaviours to follow one another in sensible ways. The robot will tend to perform a sequence of behaviours pertaining to a single goal, but will also be able to make use of opportunities which arise and to be interrupted whenever something urgent requires action (including attentional actions, if these have been programmed). Chaining behaviours and responding to interrupts are obviously basic requirements for all robot controllers. The ability to make use of opportunities allows for greater efficiency of operation, especially when the robot has multiple goals.

The model does not require all the information given to it to be correct, although it does require the information to be 'nearly right', enough that behaviours are triggered correctly. The system can therefore be given as much information as is available without fear of failure if some of this information is subsequently discovered to be wrong. This means that it is not necessary to waste information. Any system incorporating sensor noise can accept information which has only a certain probability of being correct, but few check and adjust this information. By contrast, the Neuro-Connector model automatically changes incorrect information about releasing stimuli into accurate information, if the robot is capable of sensing the relevant events. Synaptic expected times do have to be correct, however, and adaptation times have be related to them.

This ability of the net to adjust and refine the releasing stimuli for the behaviours means that the designer does not need accurate imagination about robotic sensory states. As the number of sensory states increases it becomes more difficult for people to understand what each state signifies. It is hard to imagine exactly what state in a sensor-rich robot corresponds to a significant event in the environment and how this state differs from all others, especially once there are several states for each significant event, as is postulated for our system.

This ability to continuously refine sensory data means that changes in the environment are not always fatal either, since the robot will learn to respond to the new sensory states and will forget the old ones if these have become wrong. Many changes in the environment will not affect all the releasing connections for a behaviour anyway, even if these perceived changes are really the result of sensor failure.

Extra input connections to behaviour neurons can be initialised and used not only for the demonstration learning described in Section 3.2.3 but also for instructing the robot while it is in operation. Human-generated input can have sufficient magnitude to force the robot to perform the behaviour instructed, else can be set at an intermediate level such that the input can be considered as a suggestion. This ability to instruct the robot on-line and then have it learn to generalise from the instructions (which is what happens when new releasing connections develop after commands) is extremely useful. Halperin's thesis (1990) describes the model as having three layers, each of which provides extra input to selected R neurons of the layer below. Upper levels can therefore force or suggest the selected behaviours as described above, providing a mechanism for the implementation of motivation.

The appearance of biological learning phenomena such as S–R conditioning and transient postponed conditioning is not surprising given the biological origin of the model. The appearance or otherwise of other animal learning phenomena is discussed elsewhere (Halperin, 1990) (Hallam et al., 1994).

It may be easy to run this model in a parallel implementation, since the model itself is inherently parallel. The speed-up afforded would depend greatly on the degree of overlap between S neurons expected to be significant for different subtasks, and thus the number of R neurons attached to each S.

6.3 Remaining Problems

This model is inherently computationally expensive, both in terms of the weight change functions and in terms of the number of synapses and neurons which are expected.

Simplifications making the weight update functions more computationally efficient are described in Section 2.5. The effects of these simplifications on weight changes under various circumstances are being investigated.

Synapses and neurons can be ruthlessly pruned after a learning phase if the main problem for the designer was understanding robot sensors. As long as the environment is not expected to change too much while the robot is in operation, many non-releasing connections can be removed once the robot has met all situations significant to its task. If the environment is expected to change then any pruning must be done with great care as it reduces the ability of the system to learn new connections. Currently, this pruning must be done by hand.

If a behaviour disappears due to its having no releasing connections, then that behaviour cannot reappear unless forcing inputs are available. This means that there is potential for disaster if either the robot cannot reliably disambiguate significant sensory states or else the initial releasing connections were set up incorrectly. In the first case, each potentially releasing S neuron also fires in relation to other behaviours and has its synapses with each behaviour weakened. In the second case, poor initialisation of releasing connections means that some behaviours are not performed correctly a sufficient number of times for accurate releasing connections to form.

There may be advantages in further adjustment of the mathematical aspects of the model; for instance, adjusting the way that the thresholds change, the way that input from different sources is combined, or the way that the winning behaviour is chosen. For some tasks it may be necessary to develop a version of the model with analogue neural output.

Conclusions

The Neuro-Connector model has many features which make it interesting for those wishing to control sensor-rich robots doing complex tasks in a changing environment. It allows designers to program or entrain responses to specific sensory states without the designer needing accurate knowledge of sensor characteristics; it encourages behaviours to take place in sensible sequences while still allowing opportunistic behaviour; and it allows instruction. It has been tested on a real mobile robot and is now undergoing further investigation in simulation; particularly in relation to improving the computational efficiency of the various functions, and to discovering the range of behaviour that the model can produce.

Acknowledgements

Thanks go to Janet Halperin for designing and carrying out the robot experiments under SERC grant number GR/H/23344. Thanks to Anton Dil for writing the Matlab simulation of a synapse under MRC grant number G9213053. Bridget Hallam acknowledges receipt of SERC grant number 9130847X. Ben Hope was built under SERC grant number GR/F/5852.3.

Thanks go to Edward Jones, Ashley Lotto, and Martin Westhead for reviewing this paper and commenting on its intelligibility.

All authors wish to thank the Department of Artificial Intelligence, University of Edinburgh, for provision of computing and office resources.

Glossary

- Neuro-Connector model – the complete model of learning and motivation as specified in (Halperin, 1990).
- Neuro-Connector net – an implementation of the model, necessarily more concrete and less comprehensive.
- system – a robot with a Neuro-Connector net as its controller, operating in some environment.
- releasing stimulus – a stimulus which causes a behaviour, by whatever means.
- releasing connection – a synapse with a weight sufficiently high that the firing of its pre-synaptic neuron is enough to cause the R neuron to fire and thus potentially to start that R's behaviour.
- adaptation time – period that a neuron under constant stimulation will stay on.
- adapt out – turn off.
- refractory time – period that a neuron will stay off after its adaptation time finishes, despite still having input.
- plastic – changeable, adjustable.

References

Arbib, M. A. (1987). Levels of modelling of mechanisms of visually guided behaviour. *Brain and Behavioral Sciences*, 10:407–465.

Arkin, R. C. (1991). Integrating behavioural, perceptual, and world knowledge in reactive navigation. In Maes, P., editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 105–122. MIT Press. Special Issue of Robotics and Autonomous Systems.

Ballard, D. H. (1991). Animate vision. *Artificial Intelligence*, 48:57–86.

Baloch, A. A. and Waxman, A. M. (1991). Visual learning, adaptive expectations , and behavioral conditioning of the mobile robot MAVIN. *Neural Networks*, 4(3):271 – 302.

Beer, R. D. (1990). *Intelligence as Adaptive Behaviour: an Experiment in Computational Neuroethology.* Perspectives in Artifical Intelligence. Academic Press.

Beer, R. D., Chiel, H., Quinn, R., and Larsson, P. (1992). A distributed neural network architecture for hexapod robot locomotion. *Neural Computation*, 4:356–365.

Brooks, R. A. (1985). A robust layered control system for a mobile robot. A.I. Memo 864, MIT AI lab.

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

Brooks, R. A. (1989). A robot that walks; emergent behaviours from a carefully evolved network. A.I. Memo 1091, MIT AI lab.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.

Brooks, R. A. and Stein, L. A. (1993). Building brains for bodies. AI Memo 1439, MIT AI lab.

Carpenter, G. A. and Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54 – 115. Reprinted as chapter 10 of GA Carpenter and S Grossberg (eds) 1991 *Pattern Recognition by Self-Organizing Neural Networks* MIT Press.

Carpenter, G. A. and Grossberg, S., editors (1991). *Pattern Recognition by Self-Organizing Neural Networks.* MIT Press.

Chapman, D. (1991). *Vision, Instruction and Action.* MIT Press.

Dorigo, M. and Schnepf, U. (1993). Genetic-based machine learning and behavior-based robotics: A new synthesis. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 23, pages 141–154.

Forster, P. (1991). Hardware description of the mobile robot "Ben Hope". RUR working paper 5.1, Dept of Artificial Intelligence, Edinburgh.

Grossberg, S. (1982). Processing of expected and unexpected events during conditioning and attention: A psychophysical theory. *Psychological Review*, 89:529 – 572. Reprinted as chapter 3 of S Grossberg (ed) 1987 *The Adaptive Brain* North-Holland.

Hallam, B. E., Halperin, J. R., and Hallam, J. C. (1994). An ethological model for implementation in mobile robots. *Adaptive Behavior.* In press. Also available as Research paper number 695 from the Department of Artificial Intelligence, University of Edinburgh.

Halperin, J. R. (1990). *A Connectionist Neural Network Model of Aggression.* PhD thesis, Department of Ethology, Toronto University, Canada.

Halperin, J. R., Dunham, D. W., and Ye, S. (1992). Social isolation of *betta splendens* increases primed social display but decreases display readiness. *Behavioural Processes*, 28:13–32.

Husbands, P., Harvey, I., and Cliff, D. (1993). An evolutionary approach to situated AI. In Sloman, A., Hogg, D., Humphreys, G., Ramsay, A., and Partridge, D., editors, *AISB-93: prospects for artificial intelligence: proceedings of the ninth biennial conference of the society for the study of artificial intelligence and the simulation of behaviour.*, pages 61–70. IOS Press, Amsterdam.

Nehmzow, U. and Smithers, T. (1990). Mapbuilding using self-organising networks in "Really Useful Robots". In *Proceedings of conference on "Simulation of Adaptive Behaviour – from Animals to Animats"*, Paris. MIT Press, Cambridge MA. also available as DAI Research Report no. 489.

Sinclair, J. D. (1981). *The Rest Principle: A Neurophysiological Theory of Behaviour.* Erlbaum, Hillsdale, NJ.

Spetch, Wilkie, and Pinel (1981). Backward conditioning: A re-evaluation of the empirical evidence. *Psychological Bulletin*, 89:163–175.

Sutton, R. (1990). Reinforcement learning architectures for animats. In Meyer, J.-A. and Wilson, S., editors, *From Animals to Animats*, pages 288–296.

Turing, A. (1950). Can a machine think? *Mind*, 59:433–460.

Webb, B. H. (1993). *Perception in Real and Artificial Insects: A Robotic Investigation of Cricket Phonotaxis.* PhD thesis, University of Edinburgh.

Appendix

Sensory conditions			
Code	Number	Adaptation time	Location of detected objects
UL-UR	0	0.5	Above on left and right
UL	1	3.2	Above on left side
UR	2	none	Above on right side
C	3	none	Contact, nothing visible
C-FB-FT	4	0.5	Far top and far bottom, also contact
C-NB-FT	5	0.5	Near bottom, far top, also contact
C-NB-NT	6	0.5	Near bottom and near top, also contact
C-FB-NT	7	0.5	Far bottom and near top, also contact
C-FT	8	0.5	Far top, also contact
C-NT	9	0.5	Near top, also contact
C-NB	10	0.5	Near bottom, also contact
C-FB	11	0.5	Far bottom, also contact
FB-NT	12	none	Far bottom and near top
NB-FT	13	none	Near bottom and far top
NB-NT	14	none	Near bottom and top
FB-FT	15	none	Far bottom and top
FB	16	3.2	Far bottom
FT	17	3.2	Far top
NT	18	none	Near top
NB	19	0.5	Near bottom
MS	20	3.2	Motor strain

Behaviours		
Name	Priority	Effect
NULL	-1	Do nothing — no behaviour
ADVANCE	0	Approach object
WIGGLE	1	Back 10cm, wiggle back and forth
AVOID	2	Avoid object — back 15cm, turn 90°
AFTERPAT	3	Back, turn 180°, pause

Sensory neuron parameters		
S-DEF-THRESHOLD	0.5	Default threshold
S-DEF-I-UP-GAIN	9.0	Up gain for input register
S-DEF-I-DN-GAIN	9.0	Down gain for input register
S-DEF-A-UP-GAIN	9.0	Up gain for activation register
S-DEF-A-DN-GAIN	4.0	Down gain for activation register
S-MAX-ON-TIME	0.5	Adaptation time default (seconds)
S-RECOVERY	0.5	Refractory time default (seconds)
S-REFR-TH	1.5	Refractory threshold default

Releaser neuron parameters		
R-DEF-THRESHOLD	0.5	Default threshold
R-DEF-I-UP-GAIN	9.0	Up gain for input register
R-DEF-I-DN-GAIN	9.0	Down gain for input register
R-DEF-A-UP-GAIN	9.0	Up gain for activation register
R-DEF-A-DN-GAIN	4.0	Down gain for activation register

Behaviour neuron parameters		
B-DEF-THRESHOLD	0.5	Default threshold
B-DEF-I-UP-GAIN	9.0	Up gain for input register
B-DEF-I-DN-GAIN	9.0	Down gain for input register
B-DEF-A-UP-GAIN	9.0	Up gain for activation register
B-DEF-A-DN-GAIN	4.0	Down gain for activation register
B-MAX-ON-TIME	1.3	Adaptation time for AFTERPAT and AVOID
	3.0	Adaptation time for WIGGLE
	7.0	Adaptation time for ADVANCE
B-RECOVERY	4.0	Refractory period default (seconds)
B-REFR-TH	1.5	Refractory threshold default

Synaptic parameters		
DEF-SR-K1	1.5	Weight increase integration constant
DEF-SR-K2	3.0	Weight decrease integration constant
RB-DEF-POS-WT	0.8	Default positive weight for R–B connections
RB-FBCK-WEIGHT	0.5	R–B feedback weight
BB-DEF-NEG-WT	-0.01	Default negative weight for B–B connections
BB-DEF-LGNEG-WT	-1.5	Default strong inhibitory weight for B–B
SR-DEF-WEIGHT	0.01	Default weight for S–R connections