Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 20 (2007) 172–181
Neural Networks
www.elsevier.com/locate/neunet

Learning movement sequences with a delayed reward signal in a hierarchical model of motor function

S.M. Stringer, E.T. Rolls*, P. Taylor

Oxford University, Centre for Computational Neuroscience, Department of Experimental Psychology, South Parks Road, Oxford OX1 3UD, United Kingdom

Received 5 July 2005; accepted 23 January 2006

Abstract

A key problem in reinforcement learning is how an animal is able to learn a sequence of movements when the reward signal only occurs at the end of the sequence. We describe how a hierarchical dynamical model of motor function is able to solve the problem of delayed reward in learning movement sequences using associative (Hebbian) learning. At the lowest level, the motor system encodes simple movements or primitives, while at higher levels the system encodes sequences of primitives. During training, the network is able to learn a high level motor program composed of a specific temporal sequence of motor primitives. The network is able to achieve this despite the fact that the reward signal, which indicates whether or not the desired motor program has been performed correctly, is received only at the end of each trial during learning. Use of a continuous attractor network in the architecture enables the network to generate the motor outputs required to produce the continuous movements necessary to implement the motor sequence.

Keywords: Movement sequence; Delayed reward; Reinforcement learning; Hierarchical motor function; Continuous attractor neural network

1. Introduction

A key problem in reinforcement learning is how an animal can learn a sequence of movements when the reward signal, which indicates whether the sequence was performed correctly, only occurs at the end of the sequence. A standard solution to the problem of delayed rewards is *temporal difference* learning, which is a learning algorithm that learns to predict future rewards (Sutton, 1988; Sutton & Barto, 1998). Temporal difference learning has been applied to the problem of learning sequential movements with a dopamine-like reinforcement signal (Suri & Schultz, 1998). However, temporal difference learning requires that the algorithm makes use of an error term in the synaptic weight updates that is based on the difference between the future rewards predicted by the algorithm at successive timesteps. This may be complicated biologically, for it requires successive predictions of expected reward value to be available during a learning trial, for differences between these predictions to be calculated, and for the resulting error term to be used to influence synaptic weight modifications.

One approach to solving the problem of delayed reward in reinforcement learning with biologically plausible, associative learning rules may be through a hierarchical organization of the motor system. Motor systems in humans have a hierarchical structure, with higher levels specifying increasingly complex motor actions (Ghez & Krakauer, 2000, chap. 3; Hughlings Jackson, 1878). This hierarchy may be divided into three distinct brain regions: the motor areas of the cerebral cortex, the brain stem, and the spinal cord. The highest level in the motor hierarchy is the collection of motor areas in the cortex, which specify complex voluntary motor programs. The motor areas in the cortex send connections to the brain stem and the spinal cord. The brain stem is the next layer in the motor hierarchy, and governs posture and goal directed movements. The brain stem also sends connections to the spinal cord. The spinal cord is the lowest level in the motor hierarchy, and encodes a variety of low level reflexes. The hierarchical structure underlying motor control in humans permits the motor systems to form complex motor programs from elemental motor primitives that have stereotyped spatial and temporal characteristics (Lacquaniti, Terzuolo, & Viviani, 1983). An additional feature of motor control in humans is that each level of the motor hierarchy relies on sensory input which is appropriate to that level, with more

* Corresponding author. Tel.: +44 1865 271348; fax: +44 1865 310447.
E-mail address: Edmund.Rolls@psy.ox.ac.uk (E.T. Rolls).
URL: http://www.cns.ox.ac.uk (E.T. Rolls).

0893-6080/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2006.01.016

complex sensory information extracted at each level from the spinal cord to the cortex (Ghez & Krakauer, 2000, chap. 3). These sensory representations may provide an important basis from which motor sequences at each level are regulated.

In this paper we describe how a hierarchical dynamical model of motor function is able to solve the problem of delayed reward in learning movement sequences using associative (Hebbian) learning. The model uses just the reward information at the end of the trial, and does not need, during learning, to keep track of expected value during a trial, and to calculate errors based on differences between expected values at different times in order to determine how to adjust the synaptic weights and what to do next, as in temporal difference learning. Moreover, the network does not just produce an abstract sequence, but instead implements the full continuously changing set of motor outputs required to perform all the movements throughout the whole sequence.

The hierarchical model of motor function investigated here builds on a model of motor function proposed by Stringer, Rolls, Trappenberg, and De Araujo (2003), and extended to hierarchical motor function by Stringer and Rolls (in preparation). The models proposed by these authors are able to learn arbitrary dynamical motor sequences, execute such motor sequences at arbitrary speed, and perform the motor sequences in the absence of external visual or proprioceptive cues. These are important properties for models of motor control in biological agents (Bizzi & Polit, 1979; Laszlo, 1966, 1967; Polit & Bizzi, 1978; Schmidt, 1987, 1988). The model described in this paper introduces rewards delivered at the end of a trial that enable a set of motor primitives to be performed later in the correct sequence. The model thus shows how reinforcement learning can be used to enable whole sequences of continually evolving motor commands to be learned.

2. Model overview

The model described in this paper utilizes a hierarchical structure. At the lowest level, the model encodes motor primitives that consist of unidirectional movements with a given starting and finishing position. These are implemented in the model by a continuous attractor network that allows continuous movement through the state space. In the model, this is implemented by the postural state continuous attractor x which interacts with the motor output network y shown in Fig. 1 in a way that will be detailed below and was described by Stringer et al. (2003). In the new model described here, this coupled pair of networks that implement the motor primitives is controlled by the movement selector cells r^{MS}, which activate simultaneously all the motor primitives that will be required for the full movement sequence. The order in which the primitives are executed is determined by the state/motor networks x and y, which, given an initial postural starting position, can only evolve through their state space in one way, in that there is only one way that the primitives given their start and end positions can evolve continuously through the state space. In particular, although all of the motor primitives that will be required for the full movement sequence are activated simultaneously, the different motor primitives are only performed when the agent is in the correct part of its state space, because the outputs from the movement selector cells are modulated by the state of the agent before reaching the motor cells themselves. This is one concept that is important to the operation of the system. The second new concept to the system is that reward given at the end of the trial facilitates the learning of connections from the high level movement selector cells in such a way that one high level movement selector command is associated with the set of primitives that are active in a successful trial in the movement selector cells r^{MS}. For the reward signal to operate effectively at the end of the trial, the movement selector cells must still be firing at the end of the trial (perhaps held on during training by the movement selector signal t^{MS}), and the high level movement selector cells r^{HMS} must also still be firing at the end of the trial when the reward is given. This then enables the reward signal to facilitate associative learning at the synapses w^4 in Fig. 1.

Thus, the delayed reward signal is used to teach the high level movement selector command cells r^{HMS} for the high level motor programme only which movement selector cells r^{MS} need to be activated at some point throughout the entire high level motor program sequence. Once the movement selector command cells r^{MS} for the low level motor primitives are activated, the motor primitives themselves encode the mappings from the state space of the agent to the motor output, because the outputs from the movement selector cells r^{MS} onto the motor cells r^{M} are modulated by the postural state r^{S} of the agent. This ensures that the motor primitives are actually performed in their correct temporal order. We emphasize that the model implements the full trajectory through all the motor commands needed within every part of the sequence, so that this is a full motor controller, and is not just a system that can learn abstract sequences, which is what temporal difference learning can implement.

3. The network model

The network is trained in two stages. First, in Section 3.1 we describe how the network learns the low level motor primitives. The motor primitives are learned using associative learning rules, but with no reward signals used during this process. Second, in Section 3.2 we describe how the network is able to learn two high level motor programs, each of which is composed of specific temporal sequences of the motor primitives, using an associative learning rule that uses the delayed reward signal at the end of each trial to learn to perform the desired high level motor program.

3.1. Learning the motor primitives

The network of state cells (with firing rate r_i^{S} for state cell i) represents the current positional (or postural) state of the agent. The state cells are recurrently connected and form a continuous attractor neural network (the generic details of which are described by Amari (1977), Rolls and Deco (2002), Stringer, Rolls, Trappenberg, and De Araujo (2002), Stringer,

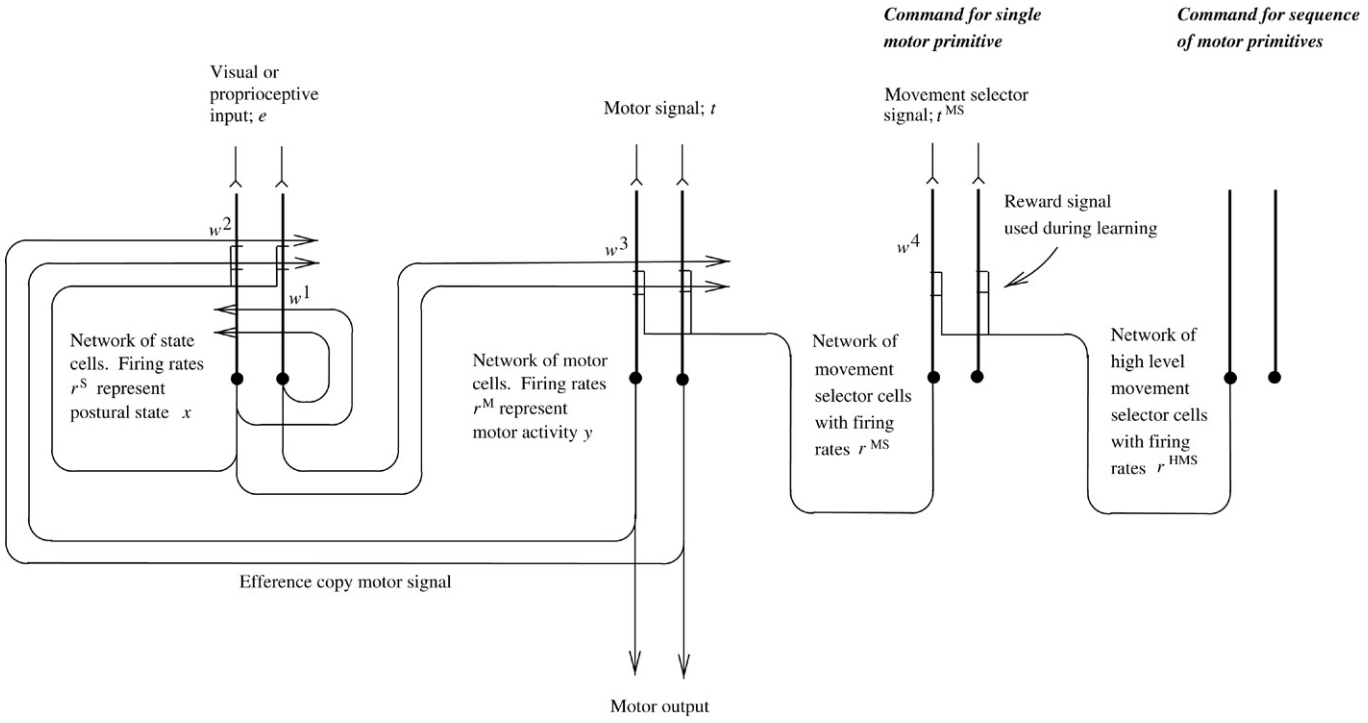

Fig. 1. Network architecture of the hierarchical dynamical model of motor function. There is a network of state (S) cells that represent the postural state of the agent, a network of motor (M) cells that represent the motor activity, a network of movement selector (MS) cells that represent the command to perform a motor primitive, and a network of high level movement selector (HMS) cells that represent the high level motor programs, which are composed of sequences of the motor primitives. The forward model is implemented by the synaptic connections w^2 from Sigma–Pi couplings of state cells and motor cells to the state cells. The connections w^2 are able to update the representation in the network of state cells given the current patterns of firing in both the network of state cells and the network of motor cells. The inverse model is implemented by the synaptic connections w^3 from Sigma–Pi couplings of state cells and movement selector cells to the motor cells. Given a motor primitive represented by the firing of the movement selector cells, the connections w^3 are able to drive the motor cells to perform the appropriate motor actions. The movement selector (MS) cells are driven by the network of high level movement selector (HMS) cells through the synapses w^4. While the movement selector cells represent the motor primitives, the high level movement selector cells represent the high level motor programs which are composed of sequences of the motor primitives.

Trappenberg, Rolls, and De Araujo (2002) and Taylor (1999)). The activation h_i^{S} of state cell i is governed by

$$\tau \frac{\mathrm{d}h_i^{\mathrm{S}}(t)}{\mathrm{d}t} = -h_i^{\mathrm{S}}(t) + \frac{\phi_0}{C^{\mathrm{S}}} \sum_j (w_{ij}^1 - w^{\mathrm{INH}}) r_j^{\mathrm{S}}(t) + e_i + \frac{\phi_1}{C^{\mathrm{S\times M}}} \sum_{j,k} w_{ijk}^2 r_j^{\mathrm{S}} r_k^{\mathrm{M}} \quad (1)$$

where w_{ij}^1 is the excitatory (positive) synaptic weight from state cell j to state cell i, and w^{INH} is a global constant describing the effect of inhibitory interneurons within the layer of state cells.[1] τ is the time constant of the system. e_i represents an external input to state cell i, which may be visual or proprioceptive. The last term in Eq. (1) is a sum of coupled inputs[2] from the state and motor cells $\sum_{j,k} w_{ijk}^2 r_j^{\mathrm{S}} r_k^{\mathrm{M}}$, where r_j^{S} is the firing rate of state cell j, r_k^{M} is the firing rate of motor cell k, and w_{ijk}^2 is the corresponding strength of connection from these cells.[3]

The firing rate r_i^{S} of state cell i is determined from the sigmoid activation function

$$r_i^{\mathrm{S}}(t) = \frac{1}{1 + \mathrm{e}^{-2\beta(h_i^{\mathrm{S}}(t) - \alpha)}}, \quad (2)$$

where α and β are the sigmoid threshold and slope, respectively.

The network of motor cells (with firing rate r_i^{M} for motor cell i) represents the current motor output. The activation h_i^{M} of motor cell i is governed by

$$\tau \frac{\mathrm{d}h_i^{\mathrm{M}}(t)}{\mathrm{d}t} = -h_i^{\mathrm{M}}(t) + t_i + \frac{\phi_2}{C^{\mathrm{S\times MS}}} \sum_{j,k} w_{ijk}^3 r_j^{\mathrm{S}} r_k^{\mathrm{MS}} \quad (3)$$

where t_i is the motor training signal for motor cell i. The last term in Eq. (3) is produced from couplings in w^3 of the state cells (with firing r_j^{S}) and movement selector cells (with firing r_k^{MS}), and w_{ijk}^3 is the corresponding strength of the connection

[1] The scaling factor (ϕ_0/C^{S}) controls the overall strength of the recurrent inputs to the layer of state cells, where ϕ_0 is a constant and C^{S} is the number of presynaptic connections received by each state cell from other state cells.

[2] These are Sigma–Pi synapses, but these and the other Sigma–Pi synapses in the model can, in principle, be replaced by a layer of neurons that competitively learn combinations of the inputs, and then map these using Hebb (two term) associative synapses to the correct output, as shown by Rolls and Stringer (2005).

[3] The scaling factor $\frac{\phi_1}{C^{\mathrm{S\times M}}}$ controls the overall strength of the motor inputs, where ϕ_1 is a constant and $C^{\mathrm{S\times M}}$ is the number of Sigma–Pi connections received by each state cell.

from these cells. This driving term initiates activity in the network of motor neurons when the movement selector cells are activated.[4]

The firing rate r_i^{M} of motor cell i is determined from the sigmoid activation function

$$r_i^{\mathrm{M}}(t) = \frac{1}{1 + \mathrm{e}^{-2\beta(h_i^{\mathrm{M}}(t)-\alpha)}}. \quad (4)$$

During training, the motor training signals t_i received by each motor cell i cause the agent to proceed through a set of positional states. Simultaneously, each positional state cell i is driven by external visual or proprioceptive input e_i carrying information about the state (i.e. position) of the agent. While this happens, three different learning rules are responsible for setting up the synaptic weights w_{ij}^1, w_{ijk}^2 and w_{ijk}^3 within the network.

The recurrent connections w_{ij}^1 are set up to permit the network of state cells to operate as a continuous attractor network and support stable patterns of firing in the absence of external visual or proprioceptive input. The learning rule used to update the recurrent synapses w_{ij}^1 is the Hebb rule

$$\delta w_{ij}^1 = k^1 r_i^{\mathrm{S}} r_j^{\mathrm{S}}. \quad (5)$$

The connections w_{ijk}^2 are responsible for enabling an efference copy of the motor signal to update, using path integration, the positional state CANN, and thus implement a *forward model* of motor function (Miall & Wolpert, 1996). These connections are trained by a traced Sigma–Pi learning rule, which allows a combination of the short-term memory traced activity within the state cell network (which represents the preceding position) and the short-term memory traced activity within the motor cell network (which represents the preceding motor command) to be associated with the current positional state. The learning rule used to update the synapses w_{ijk}^2 is

$$\delta w_{ijk}^2 = k^2 r_i^{\mathrm{S}} \bar{r}_j^{\mathrm{S}} \bar{r}_k^{\mathrm{M}}, \quad (6)$$

where $\bar{r}_j^{\mathrm{S}}$ refers to a memory trace of the firing r_j^{S}, and $\bar{r}_k^{\mathrm{M}}$ refers to a memory trace of the firing of r_k^{M}. The trace value $\bar{r}$ of the firing rate r of a cell is calculated according to

$$\bar{r}(t + \delta t) = (1 - \eta) r(t + \delta t) + \eta \bar{r}(t) \quad (7)$$

where η is a parameter in the interval [0,1] which determines the relative contributions of the current firing and the previous trace. For $\eta = 0$, the trace becomes just the present firing rate, and as η is increased the contribution of preceding firing at times earlier than the current timestep is increased. The short-term memory traces inherent in these operations could be implemented by a number of biophysical processes, including the long time constant of NMDA (N-methyl-D-aspartate) receptors (Stringer, Rolls, et al., 2002; Stringer, Trappenberg, et al., 2002).

The synaptic weights w_{ijk}^3 are set up by a learning rule which associates the co-firing of the movement selector cells and a particular cluster of state cells with the firing of the appropriate cluster of motor cells. The synaptic connections w_{ijk}^3 thus implement an *inverse model* of motor function: given a desired movement primitive represented by the firing of the movement selector cells, the synaptic connections w_{ijk}^3 drive the motor cells to perform the appropriate motor actions. The synaptic weights w_{ijk}^3 are updated according to

$$\delta w_{ijk}^3 = k^3 r_i^{\mathrm{M}} r_j^{\mathrm{S}} r_k^{\mathrm{MS}}. \quad (8)$$

During the initial learning phase in which the network learns the low level motor primitives, the agent performs each motor primitive to be learned. As the agent performs each motor primitive, the state cells are driven by the visual or proprioceptive inputs e_i, the motor cells are driven by the training signal t_i, and the synaptic weights w_{ij}^1, w_{ijk}^2 and w_{ijk}^3 are updated according to the simple learning rules discussed above. During repeated learning cycles of each motor primitive, the synaptic connectivity of the network self-organizes such that, after training, the correct motor primitive may be stimulated solely by stimulating the particular set of movement selector cells.

3.2. Learning high level motor programs with delayed rewards

In this section we describe how the network is able to learn two high level motor programs, each of which is composed of a particular temporal sequence of low level motor primitives using delayed rewards. The command to perform a high level motor program is represented by the pattern of firing in the network of high level movement selector (HMS) cells shown in Fig. 1. We show how the network is able to learn the synaptic connections w^4 from the high level movement selector cells to the movement selector (MS) cells representing the motor primitives such that, after training, a pattern of activity within the network of high level movement selector cells is able to stimulate the correct temporal sequence of motor primitives. The synaptic connections w^4 are learned using a simple, biologically plausible, associative learning rule tha uses the delayed reward signal at the end of each attempt to perform the desired motor program.

For consistency, the movement selector cells are modelled dynamically. The activation of the movement selector cells is governed by

$$\tau \frac{\mathrm{d}h_i^{\mathrm{MS}}(t)}{\mathrm{d}t} = -h_i^{\mathrm{MS}}(t) + t_i^{\mathrm{MS}} + \frac{\phi_3}{C^{\mathrm{HMS}}} \sum_j w_{ij}^4 r_j^{\mathrm{HMS}}. \quad (9)$$

The term t_i^{MS} is a training signal for movement selector cell i, which is present only during training with a new high level motor program. The last term in Eq. (9) is the input from the high level movement selector cells $\sum_j w_{ij}^4 r_j^{\mathrm{HMS}}$, where r_j^{HMS} is the firing rate of high level movement selector cell j, and

[4] The scaling factor $\frac{\phi_2}{C^{\mathrm{S}\times\mathrm{MS}}}$ controls the overall strength of the inputs from couplings of state and movement selector cells, where ϕ_2 is a constant, and $C^{\mathrm{S}\times\mathrm{MS}}$ is the number of connections received by each motor cell from couplings of the state and movement selector cells.

w_{ij}^4 is the corresponding strength of the connection from these cells.[5]

The firing rate r_i^{MS} of movement selector cell i is determined from the sigmoid activation function

$$r_i^{\mathrm{MS}}(t) = \frac{1}{1 + \mathrm{e}^{-2\beta(h_i^{\mathrm{MS}}(t) - \alpha)}}. \tag{10}$$

The network learns to perform a high level motor program as follows. Firstly, a new set of cells in the high level movement selector network (HMS) is activated, and it is this new set of high level movement selector cells that learns to drive the network through the new high level motor program composed of a set of primitives. Then, during training, the agent makes repeated attempts at performing the desired high level motor program. For each attempt at performing the motor program, a new (possibly random) set of motor primitives is chosen. These motor primitives are then stimulated by applying a constant training signal t_i^{MS} to the movement selector cells that encode those primitives. During each attempt at performing the high level motor program, the set of motor primitives stimulated by the training signal t_i^{MS} during the course of that attempt does not alter. However, although the training signal t_i^{MS} stimulates all of the relevant movement selector cells equally throughout the entire attempt at the motor program, the different motor primitives represented by these cells are only performed when the agent is in the part of its state space relevant to each primitive. This is because the outputs from the movement selector cells are modulated by the state of the agent in the Sigma–Pi synapses w_{ijk}^3 before reaching the motor cells. This ensures that, even though all of the movement selector cells are stimulated simultaneously, the motor primitives are still performed by the motor cells in a strict temporal sequence. Whenever the agent completes the high level motor program successfully during training, the agent receives a reward signal and the synapses w_{ij}^4 from the high level movement selector cells to the movement selector cells are updated according to the associative (Hebb) rule

$$\delta w_{ij}^4 = k^4 r_i^{\mathrm{MS}} r_j^{\mathrm{HMS}} r^{\mathrm{R}} \tag{11}$$

where r^{R} is 1 if a reward is obtained and 0 if a reward is not obtained. The effect of this learning rule is to associate the pattern of firing in the network of high level movement selector cells, which represents the command to perform the high level motor program, with the movement selector cells representing all of the motor primitives that were required to perform the high level motor program. That is, the role of the delayed rewards used to train the synapses w_{ij}^4 is to enable the network to associate the pattern of activity in the high level movement selector network with all of the motor primitives that will be required at some point in the high level motor program.

After training the synapses w_{ij}^4 using delayed reward signals, the new set of high level movement selector cells is able to drive the network through the new high level motor program as follows. When the new set of high level movement selector cells is activated, the outputs from these cells mediated by the synapses w_{ij}^4 stimulate all of the movement selector cells that represent the motor primitives required to perform the high level motor program. All of these movement selector cells become equally active at the beginning of the high level motor program, and remain active until the end of the motor program. However, although all of the relevant movement selector cells remain equally active throughout the entire high level motor program, each motor primitive represented by these cells is only performed when the agent is in the correct part of its state space, because the outputs from the movement selector cells are modulated by the state of the agent in the Sigma–Pi synapses w_{ijk}^3 before reaching the motor cells. This ensures that, even though all of the movement selector cells are stimulated simultaneously, the motor primitives are still performed by the motor cells in the correct temporal sequence. The high level motor program involves a mapping from the state space of the agent to the set of motor primitives mediated through the Sigma–Pi synapses w_{ijk}^3, where, for each successive state of the agent, the correct motor primitives must be performed. However, the delayed reward signal is used to teach the high level movement selector cells only which motor primitives (encoded by the movement selector cells) are needed throughout the entire high level motor program sequence, while the motor primitives themselves encode the mappings from the state space of the agent to the motor output.

3.3. Stabilization of activity packets in the continuous attractor network of state cells in the presence of noise

As described by Stringer, Rolls, et al. (2002) and Stringer, Trappenberg, et al. (2002), the recurrent synaptic weights within the continuous attractor network of state cells may be corrupted by a certain amount of noise from the learning regime. This, in turn, can lead to drift of the activity packet within the network of state cells when there is no external visual or proprioceptive input available even when the agent is not moving. Stringer, Trappenberg, et al. (2002) proposed that, in real nervous systems, this problem may be solved by enhancing the firing of neurons that are already firing (cf. Lisman, Fellous, and Wang (1998)). This is achieved in the numerical simulations by resetting the sigmoid threshold α_i at each timestep depending on the firing rate of cell i at the previous timestep. That is, at each timestep $t + \delta t$ we set

$$\alpha_i = \begin{cases} \alpha^{\mathrm{HIGH}} & \text{if } r_i(t) < \gamma \\ \alpha^{\mathrm{LOW}} & \text{if } r_i(t) \geq \gamma \end{cases} \tag{12}$$

where γ is a firing rate threshold. This helps to reinforce the current position of the activity packet within the continuous attractor network of state cells. The sigmoid slopes are set to a constant value, β, for all cells i.

[5] The scaling factor $\frac{\phi_3}{C^{\mathrm{HMS}}}$ controls the overall strength of the inputs from the high level movement selector cells, where ϕ_3 is a constant and C^{HMS} is the number of connections received by each movement selector cell from the high level movement selector cells.

4. Results

We demonstrated the ability of the network model shown in Fig. 1 to learn two high level motor programs, each of which consists of 3 particular motor primitives (chosen from a set of 6) which must be executed in the correct temporal order.[6] First, it is necessary to have primitives in the network so that later learning can incorporate them into a sequence. In Section 4.1 we describe how the network learned a total of 6 low level motor primitives. The motor primitives are learned using associative learning rules, but with no reward signals used during this process. Second, in Section 4.2 we demonstrate the network learning two high level motor programs, each of which is composed of a temporal sequence of 3 particular motor primitives. This is accomplished using an associative learning rule that uses the delayed reward signal at the end of each attempt to perform the desired high level motor program.

4.1. Learning the low level motor primitives

The numerical simulation began with the learning phase in which the synaptic weights, w^1_{ij}, w^2_{ijk} and w^3_{ijk}, were self-organized. During learning, just the state and motor cells were moved together by their inputs e and t through the 6 motor primitives shown in Figs. 2 and 3, while the movement selector cells were in the fixed state shown during execution of each of the primitives. During the training, at any one time the firing in the networks had a gaussian profile (for details, see Stringer et al. (2003)). What is shown in Figs. 2 and 3 is the execution of each primitive after training when no training signals e and t are applied, and just one movement selector signal is held active in the position shown. It is a property of the system that any movement selector signal only produces a movement if the state cells S are in the correct position when a particular movement selector signal is applied.

4.2. Learning to perform a high level motor program

In stage 2 of the simulation, the network was trained to perform two high level motor programs. The high level motor program shown on the left of Fig. 4 was to move the state of the agent from that represented at state cell 20 to that represented at state cell 180, and to do this the system needed to sequence appropriately the primitives 1, 2 and 3 shown in Fig. 2. During each learning trial of the high level motor program shown on the left of Fig. 4, cells 1–10 in the high level movement selector (HMS) network were active to represent the high level movement command. On each of 10 learning trials, 3 of the 6 motor primitives were selected at random and the related movement selector cells were then set firing by applying a constant training signal t^{MS}_i to these cells. The set of movement selector cells MS representing the 3 motor primitives was maintained active throughout each trial by the training signals t^{MS}_i. Inputs e and t were not applied during this phase of the training. Whenever the agent completed the high level motor program successfully during training, the agent received a reward signal and the synapses w^4_{ij} from the high level movement selector cells (HMS) to the movement selector cells (MS) were updated according to Eq. (11). No other synapses were modified during this stage of the training. Similar training was performed for the motor program shown on the right of Fig. 4 which was to move the state of the agent from that represented at state cell 180 to that represented at state cell 20.

For each program, there were 10 training trials, only one of which performed the sequence correctly (e.g. for the program on the left from position 20 to position 180 in the state space x) and received a reward $r^{\mathrm{R}} = 1$. For program 1, the training phase led to strong synaptic weights w^4_{ij} from the high level movement selector cells (HMS) 1–10 to the movement selector cells (MS) representing the 3 correct motor primitives 1, 2 and 3.

After the learning phase for the high level motor programs was completed, the network was tested to see if it could perform the high level motor programs when only the appropriate high level movement selector cells were activated. For program 1, before the start of the experiment, an activity packet was initiated at position 20 in the state space x. In Fig. 4 (left) we show the firing rate profiles within the four networks through time as the agent performs the learned high level motor program. It is shown that applying steady activity to the relevant high level movement selector cells 1–10 during timesteps 81–790 drives the network through the high level motor program composed of a temporal sequence of the 3 motor primitives 1, 2 and 3. It is made clear in the state cell part of the Fig. 4 (left) that the state cells do pass through the sequence defined by the primitives 1, 2 and 3 shown in Fig. 2 and performed in that order. From Fig. 4 (left) it can be seen that the high level movement selector cells stimulate firing in all of the movement selector cells required for the high level motor program. All of these movement selector cells become equally active at the beginning of the motor program, and remain active until the end of the motor program. However, although all of the relevant movement selector cells remain equally active throughout the entire motor program, each motor primitive represented by these cells is only performed when the agent is in the correct part of its state space, because the outputs from the movement selector cells are modulated by the state of the agent in the Sigma–Pi synapses w^3_{ijk} before reaching the motor cells. This ensures that, even though all of the movement selector cells are stimulated simultaneously, the motor primitives are still performed by the motor cells in the correct temporal sequence. As the movement selector cells fire,

[6] The following parameter values were used. The parameters governing the learning were: $\eta = 0.9$, $k^1 = 0.001$, $k^2 = 0.001$, $k^3 = 0.001$ and $k^4 = 0.001$. The parameters governing the 'leaky-integrator' dynamical Eqs. (1), (3) and (9) were: $\tau = 1$, $\phi_0 = 1.5 \times 10^5$, $\phi_1 = 1 \times 10^7$, $\phi_2 = 1.25 \times 10^6$, $\phi_3 = 2.5 \times 10^6$ and $w^{\mathrm{INH}} = 0.011$. The parameters governing the sigmoid activation functions were as follows. For the state cells, we used: $\alpha^{\mathrm{HIGH}} = 0.0$, $\alpha^{\mathrm{LOW}} = -20.0$, $\gamma = 0.5$, and $\beta = 0.1$. For the motor cells and movement selector cells, we used: $\alpha^{\mathrm{HIGH}} = 10.0$, $\alpha^{\mathrm{LOW}} = 10.0$, $\gamma = 0.5$, and $\beta = 0.3$. Since we set $\alpha^{\mathrm{HIGH}} = \alpha^{\mathrm{LOW}}$ for the motor cells and movement selector cells, there was no enhancement of the firing rates of these types of cells with already high firing rates, and the parameter γ was redundant. Finally, for the numerical simulations of the leaky-integrator dynamical Eqs. (1), (3) and (9), we employed a Forward Euler finite difference method with the timestep set to 0.2.

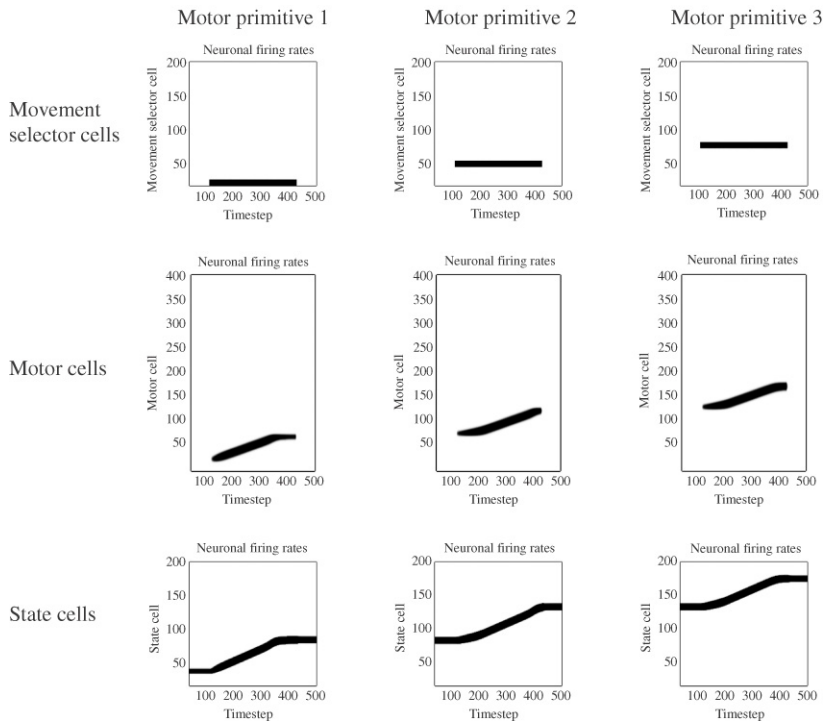

Fig. 2. Numerical results from the first stage of the simulation, in which the network is trained to perform the low level motor primitives. This figure shows the motor primitives 1, 2 and 3, which encode small movements of the state of the agent in the positive x direction. The first column shows the network performing the first low level motor primitive, the second column shows the network performing the second low level motor primitive, and the third column shows the network performing the third low level motor primitive. Within each column, we show the firing rate profiles within the movement selector network, the motor network and the state network through time as the agent performs the learned motor primitive in the absence of the motor training signals t_i, and without the external (visual or proprioceptive) inputs e_i. In all plots, high cell firing rates are indicated by black.

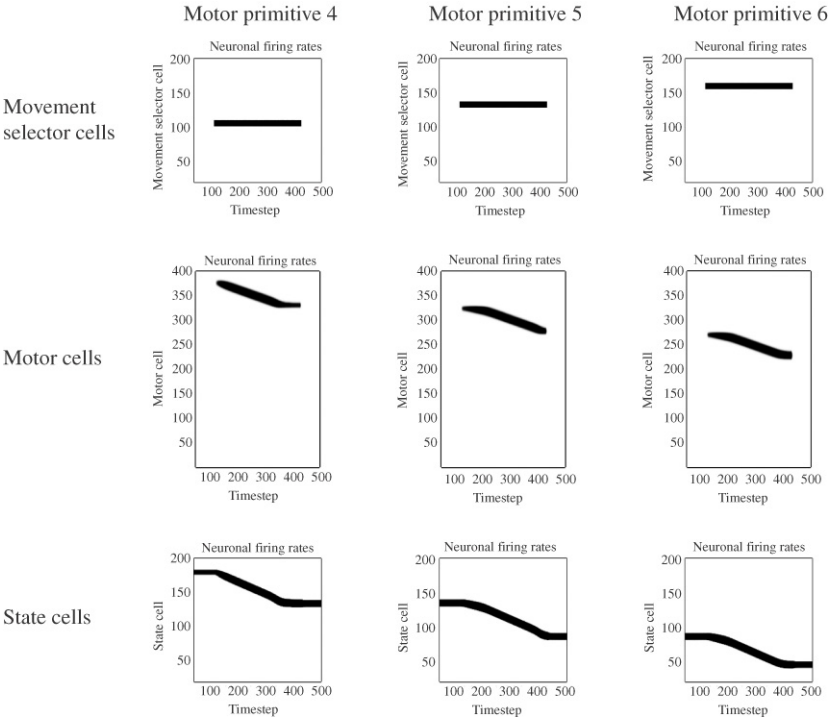

Fig. 3. Numerical results from the first stage of the simulation, in which the network is trained to perform the low level motor primitives. This figure shows the motor primitives 4, 5 and 6, which encode small movements of the state of the agent in the negative x direction. Conventions as in Fig. 2.

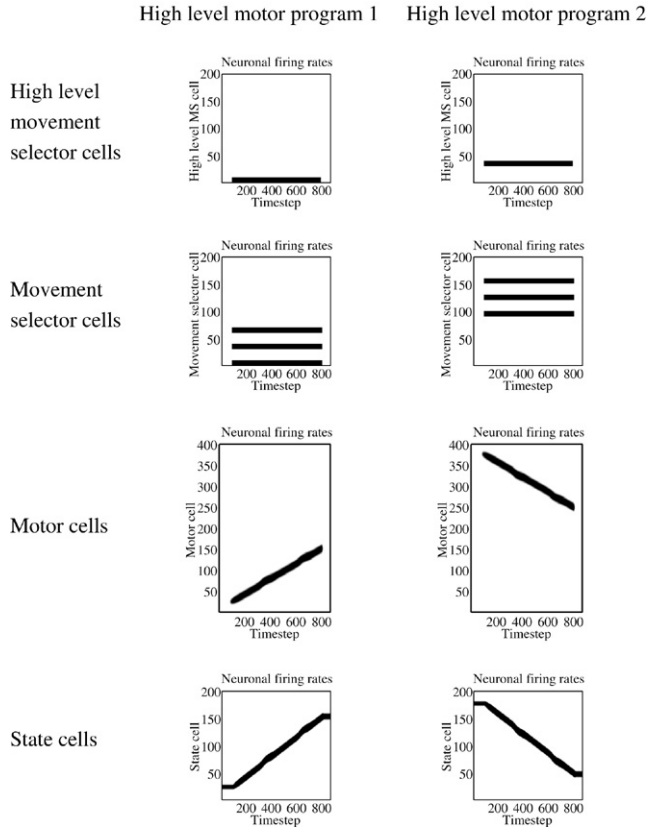

Fig. 4. Numerical results from the second stage of the simulation, after the network has been trained to perform two different high level motor programs. The motor program on the left starts at state cell 20 and moves to state cell 180, as a consequence of external selection of high level movement selector cells 1–10. These high level movement selector cells in turn select the 3 populations of movement selector cells shown. These in turn select the motor primitives 1, 2 and 3 (shown in Fig. 2) in the correct order, because activity in the motor network is gated by the w^3 connections from the state cells (shown in Fig. 1). We show the firing rate profiles within the high level movement selector network, the movement selector network, the motor network and the state network through time as the agent performs the learned high level motor program. The high level motor program is performed without the training signals t_i^{MS} for the movement selector cells, without the motor training signals t_i, and without the external (visual or proprioceptive) inputs e_i. The motor program on the right starts at state cell 180 and moves to state cell 20, as a consequence of external selection of high level movement selector cells 31–40.

the signals from the movement selector cells drive the activity packets in the postural state and motor networks to run through the learned motor primitives in the correct temporal sequence 1, 2, 3. The high level motor program is performed without the training signals t_i^{MS} driving the movement selector cells which represent the selection of the motor primitives, without the motor training signals t_i, and without the external (visual or proprioceptive) inputs e_i.

For motor program 2 shown on the right of Fig. 4 the correct motor sequence is performed in an analogous way. The fact that two different sequences were trained into the network, and each could be initiated by its appropriate high level movement selector command cells, shows that rewards delayed until the end of a whole movement sequence can be used to correctly train different motor sequences in this network architecture.

We performed a further simulation to show that different sequences starting even from the same initial position can be trained into the network. This simulation is illustrated in Fig. 5. On the left, the motor program starts at state cell 100, and moves to state cell 180. The training was analogous to that used for the simulations shown in Fig. 4, except that two motor primitives were learned for each sequence. In this case, high level movement selector cells 61–70 select the movement selector cells shown. These in turn select motor primitives 2 and 3 (shown in Fig. 2) in the correct order, because activity in the motor network is gated by the w^3 connections from the state cells (shown in Fig. 1). On the right, the motor program starts at state cell 100, and moves to state cell 20. In this case, high level movement selector cells 91–100 select the movement selector cells shown. These in turn select motor primitives 5 and 6 (shown in Fig. 3) in the correct order, because activity in the motor network is gated by the w^3 connections from the state cells (shown in Fig. 1). This simulation thus shows that delayed rewards can be used to train the network architecture in such a way that, depending on the high level movement selector command cells, a choice of different sequences can be selected even from the same starting position.

5. Discussion

In this paper, we have demonstrated how a hierarchical dynamical model of motor function is able to provide a possible solution to the problem of reinforcement learning with delayed rewards (or punishments) using associative learning rules.

It is an interesting feature of the system that it uses associative learning rules rather than temporal difference (TD) learning. TD learning requires ongoing representations of predicted reward at each timestep, and calculating errors by a subtraction process at each timestep by which to correct the synaptic weights. In this respect, the model of reinforcement learning described here is simpler, by not requiring these calculations throughout each trial in order to learn the correct sequence. Instead, the model described here uses the reward signal at the end of the trial just to associate the high level movement selector (HMS) command with the set of movement selector output patterns (each one of which activates a motor primitive) that are active throughout the trial.

The actual sequencing between the primitives in the model is not performed by learned pairwise associations between each primitive in the sequence. Instead, the model learns which motor primitives are required, where each motor primitive represents a mapping from the postural state space of the agent to the motor output space. In particular, although all of the motor primitives that will be required for the full movement sequence are activated simultaneously, the different motor primitives are only performed when the agent is in the correct part of its state space, because the outputs from the movement selector cells are modulated by the state of the agent before reaching the motor cells themselves. In the model described, the actual sequence in which the primitives enabled for a trial are executed depends on the starting position within the state space x, and on maintaining continuity of movement in the state

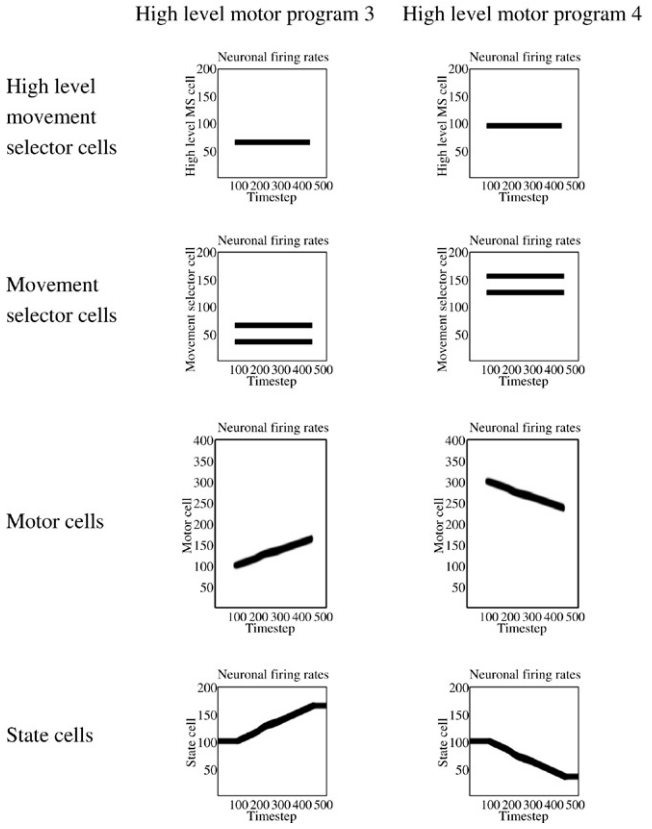

Fig. 5. Simulation of a choice of two previously trained sequences. On the left, the motor program starts at state cell 100, and moves to state cell 180. In this case, high level movement selector cells 61–70 select the movement selector cells shown. These in turn select motor primitives 2 and 3 (shown in Fig. 2) in the correct order, because activity in the motor network is gated by the w^3 connections from the state cells (shown in Fig. 1). On the right, the motor program starts at state cell 100, and moves to state cell 20. In this case, high level movement selector cells 91–100 select the movement selector cells shown. These in turn select motor primitives 5 and 6 (shown in Fig. 3) in the correct order, because activity in the motor network is gated by the w^3 connections from the state cells (shown in Fig. 1). Conventions as in previous Figures. These high level motor programs are performed without the training signals t_i^{MS} for the movement selector cells, without the motor training signals t_i, and without the external (visual or proprioceptive) inputs e_i.

space, which is provided for by the properties of the continuous attractor network x, which naturally moves continuously throughout its space. This constraint, of continuous movement throughout its space, thus determines the order in which the enabled primitives are performed. In contrast, with TD learning of motor sequences, pairwise associations are learned between state and action pairs, and the only sense in which a sequence is performed is that one action leads to the next state, with no order information learned by the network, but instead being inherent in the environment (Suri & Schultz, 1998). In contrast, the system that we describe includes a model of the state space in its continuous attractor, and it is the constraints within this continuous attractor that lead to the selected set of primitives being performed in the correct order. The states in the continuous attractor are updated during the execution of the movement by the motor efference copy using a path integration mechanism (see Stringer, Rolls, et al. (2002) and Stringer, Trappenberg, et al. (2002)).

We note that the architecture as described at present has the following constraints. The movements that are learned are being performed in a continuous space, without breaks. This is, of course, inherent in the physical properties of the agent being modelled. A property of the architecture as simulated is that, for any given high level motor program, each state of the agent (represented in the network of state cells) can be associated with only one action or motor output (represented in the network of motor cells). The same states can be associated with different actions in different high level motor programs. Although this is limiting in one dimension, in higher dimensions the network can learn arbitrary paths, as long as the above conditions hold. We note that complex paths that break these conditions (such as drawing the number '8') could, in any case, be broken into a number of high level motor programs. We further note that, in the current formulation of the model, the state cells represent the location of the agent. However, the state cells could also include additional information about, for example, the current motion of the agent. This would allow the agent to move forwards and backwards in one dimension, or perform a figure 8 in two dimensions. The architecture could thus, in principle, implement tasks such as the Corsi block-tapping task (Kolb & Whishaw, 2003), subject to the above discussion.

Although we do not propose that the framework we describe here operates in exactly this way in the brain, we do believe that the approach that we describe does provide a possible basis for what could be implemented with different implementation details in the brain. The network described above does capture some of what seems to be needed, and which may be difficult to provide with other approaches, namely a continuous unfolding in time of an arbitrary set of motor movements, hierarchical motor control, and a way to learn the sequence with a delayed reward. In addition, and in the direction of biological plausibility, we have described elsewhere how Sigma–Pi neurons can be replaced with neurons that self-organize using competitive learning to represent combinations of the input signals that can then be correctly mapped using pattern association learning (Rolls & Stringer, 2005).

There has been little previous work in applying self-organizing activity packet based continuous attractor networks, in which the activity packet moves round the network, to the problem of learning arbitrary motor programmes. Stringer et al. (2003) provided the design of a network based on the interactions of a continuous attractor network to represent postural state with a motor (M) cell network, which used trace rule learning in order to enable the system to learn to unfold a single motor primitive. In this paper we have shown how, with a hierarchical design, and with a reward signal that is delayed until the end of the trial and which reinforces the set of primitives used during the task, the whole sequence of actions can then be performed as the network moves through its continuous internal model of the state space.

Acknowledgements

This research was supported by the Medical Research Council, grant PG9826105, by the Human Frontier Science

Program, and by the MRC Interdisciplinary Research Centre for Cognitive Neuroscience.

References

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, *27*, 77–87.

Bizzi, E., & Polit, A. (1979). Processes controlling visually evoked movements. *Neuropsychologia*, *17*, 203–213.

Ghez, C., & Krakauer, J. (2000). The organisation of movement. In E. R. Kandel, J. H. Schwartz, & T. M. Jessell (Eds.), *Principles of neural science* (4th ed.) (pp. 653–673). New York: McGraw-Hill.

Hughlings Jackson, J. (1878). In J. Taylor (Ed.), *Selected writings of John Hughlings Jackson 2 1932*. London: Hodder and Staughton.

Kolb, B., & Whishaw, I. Q. (2003). *Fundamentals of human neuropsychology* (5th ed.). New York: Worth.

Lacquaniti, F., Terzuolo, C., & Viviani, P. (1983). The law relating the kinematic and figural aspects of drawing movements. *Acta Psychology*, *54*, 115–130.

Laszlo, J. L. (1966). The performance of a single motor task with kinesthetic sense loss. *Quarterly Journal of Experimental Psychology*, *18*, 1–8.

Laszlo, J. L. (1967). Training of fast tapping with reduction of kinesthetic, tactile, visual, and auditory sensation. *Quarterly Journal of Experimental Psychology*, *19*, 344–349.

Lisman, J. E., Fellous, J. M., & Wang, X. J. (1998). A role for NMDA-receptor channels in working memory. *Nature Neuroscience*, *1*, 273–275.

Miall, R. C., & Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural Networks*, *9*, 1265–1279.

Polit, A., & Bizzi, E. (1978). Processes controlling arm movements in monkeys. *Science*, *201*, 1235–1237.

Rolls, E. T., & Deco, G. (2002). *Computational neuroscience of vision*. Oxford: Oxford University Press.

Rolls, E. T., & Stringer, S. M. (2005). Spatial view cells in the hippocampus, and their idiothetic update based on place and head direction. *Neural Networks*, *18*, 1229–1241.

Schmidt, R. A. (1987). *Motor control and learning: A behavioural emphasis* (2nd ed.). Champaign, IL: Human Kinetics.

Schmidt, R. A. (1988). Motor and action perspectives on motor behaviour. In O. G. Meijer, & K. Roth (Eds.), *Complex motor behaviour: The motor-action controversy* (pp. 3–44). Amsterdam: Elsevier.

Stringer, S. M., & Rolls, E. T. (2006). Hierarchical dynamical models of motor function (in preparation).

Stringer, S. M., Rolls, E. T., Trappenberg, T. P., & De Araujo, I. E. T. (2002). Self-organizing continuous attractor networks and path integration: Two-dimensional models of place cells. *Network: Computation in Neural Systems*, *13*, 429–446.

Stringer, S. M., Rolls, E. T., Trappenberg, T. P., & De Araujo, I. E. T. (2003). Self-organizing continuous attractor networks and motor function. *Neural Networks*, *16*, 161–182.

Stringer, S. M., Trappenberg, T. P., Rolls, E. T., & De Araujo, I. E. T. (2002). Self-organizing continuous attractor networks and path integration: One-dimensional models of head direction cells. *Network: Computation in Neural Systems*, *13*, 217–242.

Suri, R. E., & Schultz, W. (1998). Learning of sequential movements by neural network model with dopamine-like reinforcement signal. *Experimental Brain Research*, *121*, 350–354.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning*. Cambridge, MA: MIT Press.

Taylor, J. G. (1999). Neural 'bubble' dynamics in two dimensions: foundations. *Biological Cybernetics*, *80*, 393–409.